Debian Developer's Reference

Developer's Reference Team, Andreas Barth, Adam Di Carlo, Raphaël Hertzog, Lucas Nussbaum, Christian Schwarz, e Ian Jackson

28 ottobre 2017

Debian Developer's Reference
by Developer's Reference Team, Andreas Barth, Adam Di Carlo, Raphaël Hertzog, Lucas Nussbaum, Christian Schwarz, e Ian Jackson

Published 2017-10-28

If you want to print this reference, you should use the pdf version. This page is also available in French, German, Italian, Russian, and Japanese.

Indice

Capitolo 1

Scopo di questo documento

Lo scopo di questo documento è quello di fornire una panoramica delle procedure consigliate e delle risorse disponibili per gli sviluppatori Debian.

Le procedure descritte qui includono come diventare un maintainer (Capitolo 2); come creare nuovi pacchetti (Sezione 5.1) e come caricare i pacchetti (Sezione 5.6); come gestire le segnalazioni di bug (Sezione 5.8); come spostare, rimuovere o rendere orfani i pacchetti (Sezione 5.9); come fare il port dei pacchetti (Sezione 5.10); e come e quando fare versioni provvisorie dei pacchetti di altri maintainer (Sezione 5.11).

Le risorse discusse in questo manuale includono le mailing list (Sezione 4.1) e server (Sezione 4.4); una discussione sulla struttura dell'archivio Debian (Sezione 4.6); spiegazione dei diversi server che accettano caricamenti di pacchetti (Sezione 5.6.1); e una discussione delle risorse che possono aiutare i maintainer con la qualità dei propri pacchetti (Appendice A).

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

Domanda per diventare un maintainer

2.1 Iniziare

So, you've read all the documentation, you've gone through the Debian New Maintainers' Guide (or its sucessor, 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?

In primo luogo, iscriversi alla debian-devel@lists.debian.org se non lo si ha già fatto. Inviare la parola `subscribe` nell'`Oggetto` di una email a debian-devel-REQUEST@lists.debian.org. In caso di problemi, contattare l'amministratore della lista di posta elettronica listmaster@lists.debian.org. Maggiori informazioni sulle mailing list disponibili possono essere trovate in Sezione 4.1. debian-devel-announce@lists.debian.org, è un'altra lista, che è obbligatoria per chiunque desideri seguire lo sviluppo di Debian.

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 Sezione 2.2 per i dettagli. Il canale IRC `#debian` può anche essere utile, si consulti Sezione 4.2.

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 Sezione 7.5.1 e `https://wiki.debian.org/DebianMentorsFaq` per 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.

Un trabocchetto potrebbe essere una parte locale troppo generica nella vostra mailadress: Termini come posta, admin, root, master dovrebbero essere evitati, consultare `https://www.debian.org/MailingLists/` per i dettagli.

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 Sezione 4.1 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.

Inoltre, se si dispone di alcuni pacchetti pronti per l'inclusione in Debian, ma che sono in attesa che si affronti il processo per diventare un nuovo maintainer, si potrebbe trovare uno sponsor per caricare il prorio pacchetto al posto vostro. Gli sponsor sono persone che sono sviluppatori ufficiali di Debian, e che sono disposti a criticare e caricare i pacchetti per voi. Leggere prima le debian-mentors FAQ su `https://wiki.debian.org/DebianMentorsFaq`.

Se volete essere un mentore e/o uno sponsor, maggiori informazioni sono disponibili in Sezione 7.5.

2.3 Registrazione come sviluppatore Debian

Prima di decidere di registrarsi con Debian, dovete leggere tutte le informazioni disponibili nell'Angolo del nuovo maintainer. Esso descrive nel dettaglio i preparativi che dovete fare prima di potervi registrare per diventare uno sviluppatore Debian. Ad esempio, prima di presentare la domanda, si deve leggere il Debian Social Contract. Registrarsi come sviluppatore significa che si è d'accordo e ci si impegna a sostenere il Contratto Sociale Debian, ma è molto importante che i maintainer siano in accordo con le idee essenziali che sono alla base del Debian. Leggere il Manifesto GNU sarebbe anche una buona idea.

Il processo di registrazione come sviluppatore è un processo di verifica della vostra identità e delle vostre intenzioni, e per controllare le vostre capacità tecniche. Poiché il numero di persone che lavorano su Debian, è cresciuto fino a oltre 1000 ed i nostri sistemi sono utilizzati in diversi punti molto importanti, dobbiamo stare attenti sull'essere compromessi. Pertanto, abbiamo bisogno di verificare i nuovi maintainer prima che di dar loro l'account sui nostri server e fargli caricare i pacchetti.

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!

La registrazione richiede che si abbia familiarità con la filosofia di Debian e con la documentazione tecnica. Inoltre, è necessaria una chiave GnuPG che sia stata firmata da un maintainer Debian esistente. Se la chiave GnuPG non è stata ancora firmata, si dovrebbe cercare di incontrare uno sviluppatore Debian di persona per farsi firmare la chiave. C'è una pagina GnuPG Key Signing Coordination, che dovrebbe aiutare a trovare uno sviluppatore Debian vicino a voi. (Se non c'è uno sviluppatore Debian vicino a voi, modi alternativi per passare il controllo ID possono essere autorizzati come un'eccezione assoluta, valutata caso per caso. Si veda la pagina di identificazione per ulteriori informazioni.)

Se non si dispone ancora di una chiave OpenPGP, generatene una. Ogni sviluppatore ha bisogno di una chiave OpenPGP per firmare e verificare i caricamenti dei pacchetti. Si consiglia di leggere il manuale del software in uso, dal momento che dispone di informazioni molto importanti che sono fondamentali per la sua sicurezza. Molti fallimenti sono dovuti ad errori umani che a malfunzionamenti del software o tecniche avanzate di spionaggio. Per maggiori informazioni sulla manutenzione della chiave pubblica si consulti Sezione 3.2.2.

Debian usa il `GNU Privacy Guard` (pacchetto `GnuPG` versione 1 o superiore) come standard di riferimento. È possibile anche utilizzare qualche altra implementazione di OpenPGP. Si noti che OpenPGP è uno standard aperto basato su RFC 2440.

Si ha bisogno di una chiave di versione 4 per utilizzarla nello sviluppo Debian. La lunghezza della chiave deve essere maggiore di 1024 bit; non c'è motivo di usare una chiave più piccola, e così facendo sarebbe molto meno sicuro. [1]

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.

Per proporsi come un nuovo maintainer, è necessario uno sviluppatore Debian esistente per supportare la vostra domanda (un `sostenitore`). Dopo aver contribuito a Debian per un po', e si vuole chiedere di diventare uno

[1] Le chiavi versione 4 sono chiavi conformi allo standard OpenPGP come definito nella RFC 2440. La versione 4 è il tipo di chiave che è creata quando si utilizza GnuPG. Le versioni di PGP dalla 5.x in su potrebbero anche creare chiavi v4, l'altra scelta essendo compatibili le chiavi pgp 2.6.x v3 (anche chiamata eredità RSA da PGP).

Le chiavi (primarie) versione 4 possono utilizzare l'algoritmo RSA o DSA, quindi questo non ha nulla a che fare con la questione di GnuPG su quale tipo di chiave volete: (1) DSA e Elgamal, (2) DSA (solo firma), (5) RSA (solo firma). Se non si hanno esigenze particolari basta scegliere l'opzione predefinita.

Il modo più semplice per dire se una chiave esistente è una chiave v4 o v3 (o v2) è quello di esaminare l'impronta digitale: Le impronte digitali delle chiavi versione 4 sono l'hash SHA-1 di una chiave materiale, in modo che siano 40 cifre esadecimali, solitamente raggruppate in blocchi di 4. Le impronte digitali formate con vecchie versioni utilizzavano l'MD5 e sono generalmente indicate in blocchi di 2 cifre esadecimali. Per esempio, se l'impronta digitale si presenta come `5B00 C96D 5D54 AEE1 206B AF84 DE7A AF6E 94C0 9C7F`, allora è una chiave v4.

Un'altra possibilità è quella di inoltrare la chiave in **PGPdump**, che dirà qualcosa di simile a Public Key Packet - Ver 4.

Si noti inoltre che la chiave deve essere self-signed (cioè deve firmare tutti i propri ID utente; questo impedisce manomissioni dell'ID utente). Tutti i moderni software di OpenPGP lo fanno automaticamente, ma se si dispone di una chiave più vecchia può essere necessario aggiungere manualmente le firme.

sviluppatore registrato, uno sviluppatore esistente con cui si è lavorato negli ultimi mesi deve esprimere la propria convinzione che si possa contribuire a Debian con successo.

Nel momento in cui si sarà trovato un sostenitore, la propria chiave GnuPG firmata e contribuito a Debian per un po', si è pronti a presentare la domanda. Potete semplicemente registrarla sulla nostra pagina applicazioni. Dopo che ci si è registrati, il proprio sostenitore deve confermare la propria domanda. Quando il proprio sostenitore avrà completato questo passaggio verrà assegnato un Application Manager, che accompagnerà attraverso i passaggi necessari del processo di New Maintainer. Si potrà sempre controllare lo stato sulla applications status board.

Per maggiori informazioni, consultare il New Maintainer's Corner sul sito web di Debian. Assicurarsi di avere familiarità con i passi necessari del processo del Nuovo Maintainer prima di presentare la domanda. Se si è ben preparati, si può risparmiare molto di tempo in seguito.

Capitolo 3

Doveri di un Debian Developer

3.1 Doveri di un maintainer di pacchetti

Come un maintainer di un pacchetto, si suppone che si forniranno pacchetti di alta qualità che siano ben integrati nel sistema e che aderiscano alla Policy di Debian.

3.1.1 Lavorare per il prossimo rilascio `stable`

Fornire pacchetti di alta qualità in `unstable` non è abbastanza, la maggior parte degli utenti beneficeranno dei vostri pacchetti solo quando sono rilasciati come parte del prossimo rilascio `stable`. Si è quindi tenuti a collaborare con il gruppo di rilascio per essere sicuri che i vostri pacchetti vengano inclusi.

Più concretamente, si dovrà controllare che i pacchetti stiano migrando verso `testing` (si consulti Sezione 5.13). 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 Sezione 5.8.5). 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 Sezione 5.5.1).

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 Sezione 5.8. 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.

Se, per qualsiasi motivo, non potete risolvere un bug RC in uno dei vostri pacchetti entro 2 settimane (per esempio a causa di vincoli di tempo, o perché è difficile da correggere), si dovrebbe accennarlo chiaramente nel bug report e si dovrebbe contrassegnare il bug come `help` in modo da invitare altri volontari a partecipare alla sua risoluzione. Si sia consapevoli che i bug RC sono spesso bersaglio di Non-Maintainer Uploads (si consulti Sezione 5.11) perché in grado di bloccare la migrazione in `testing` di molti pacchetti.

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 Sezione 7.4).

3.1.4 Coordinamento con gli sviluppatori originali

Grande parte del proprio lavoro come maintainer Debian sarà quello di restare in contatto con gli sviluppatori originali. Gli utenti Debian a volte segnaleranno al nostro sistema di bug-tracking bug che non sono specifici di Debian. Dovrete inoltrare tali segnalazioni di bug agli sviluppatori originali in modo che possano essere corrette in una futura versione originale.

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.

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.

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 Sezione 4.5.

3.2.2 Mantenere la vostra chiave pubblica

Si sia molto attenti con le vostre chiavi private. Non le si metta su qualsiasi server pubblico o macchine multiutente, come ad esempio il server Debian (si consulti Sezione 4.4). Eseguire copie delle chiavi; si conservi una copia offline. Leggere la documentazione fornita con il software; leggere la PGP FAQ.

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.

Se è necessario aggiungere una nuova chiave o rimuovere una vecchia, è necessario che la nuova chiave sia firmata da un altro sviluppatore. Se la vecchia chiave è compromessa o non valida, si deve anche aggiungere il certificato di revoca. Se non vi è alcun motivo reale per una nuova chiave, i Keyring Maintainer potrebbero rifiutare la nuova chiave. Dettagli possono essere trovati presso `http://keyring.debian.org/replacing_keys.html`.

Si applichino le stesse routine di estrazione di chiavi discusse nel Sezione 2.3.

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

Non dovete monitorare le discussioni pre-voto, considerato che il segretario effettuerà diverse chiamate di votazione su debian-devel-announce@lists.debian.org (e tutti gli sviluppatori dovrebbero essere iscritti a questa lista). La democrazia non funziona bene se le persone non prendono parte al voto, è per questo che invitiamo tutti gli sviluppatori a votare. Le votazioni sono condotte attraverso messaggi email GPG-signed/encrypted.

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 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 Sezione 5.11) 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 Debian developers' LDAP database (questa informazione è accessibile solo agli sviluppatori Debian). Non dimenticare di togliere il flag vacanza quando si torna!

Idealmente, si dovrebbe firmare la GPG coordination pages al momento della prenotazione di una vacanza e verificare se qualcuno ci sta cercando per la firma. Questo è particolarmente importante quando la gente va in luoghi esotici dove non abbiamo ancora degli sviluppatori, ma dove ci sono persone che sono interessati a presentare domanda.

3.2.5 Congedarsi

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

1. rendete orfani tutti i pacchetti, come descritto in Sezione 5.9.4.

2. Inviare una email gpg-firmata sul perché si sta lasciando il progetto a

3. Comunicare ai keyring maintainer Debian che si sta lasciando con l'apertura di un ticket in Debian RT e con l'invio di una email all'indirizzo keyring@rt.debian.org con le parole «Debian RT» da qualche parte nella riga dell'oggetto (le maiuscole e minuscole non importano).

4. 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.

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.

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

3.2.6 Ritornare dopo il congedo

l'account di uno sviluppatore congedato è contrassegnato come «emerito» quando il processo in Sezione 3.2.5 è seguito e «disabled» in caso contrario. Gli sviluppatori ritirati con un account «emerito» possono ottenere il loro account riattivato come segue:

- Si contatti da-manager@debian.org.
- Si passi attraverso un processo di NM accorciato (per garantire che il committente tornando sappia ancora parti importanti della P&P and T&S).
- Si dimostri che ancora si controlla la chiave GPG associata all'account, o si fornisca la prova di identificazione su una nuova chiave GPG, con almeno due firme da altri sviluppatori.

Gli sviluppatori congedati con un account «disabilitato» necessitano nuovamente di passare attraverso NM.

Capitolo 4

Risorse per sviluppatori e manutentori Debian

In questo capitolo troverete una breve mappa relativa alle mailing list Debian, delle macchine Debian che possono essere a disposizione degli sviluppatori e di tutte le altre risorse che sono a disposizione per aiutare il maintainer nel suo lavoro.

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

Prima di richiedere una mailing list che si riferisce allo sviluppo di un pacchetto (o di un piccolo gruppo di pacchetti correlati), prendere in considerazione se è più appropriato utilizzare l'utilizzo di un alias (tramite un file a.forward-aliasname su master.debian.org, che si traduce in un indirizzo ragionevolmente bello *`tu-nomealias@debian.org`*) o una mailing list autogestita su Alioth.

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`.

Ci sono altri canali aggiuntivi dedicati a temi specifici. `#debian-bugs` è usato per coordinare le feste di bug squashing. `#debian-boot` è usato per coordinare il lavoro sul debian-installer. `#debian-doc` è usato occasionalmente per parlare di documentazione, come il documento che si sta leggendo. Altri canali sono dedicati ad una architettura o ad un insieme di pacchetti: `#debian-kde`, `#debian-dpkg`, `#debian-jr`, `#debian-edu`, `#debian-oo` (pacchetto OpenOffice.org)...

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

Canali dedicati a Debian esistono anche su altre reti IRC, in particolare sulla rete IRC freenode, a cui puntava l'alias `irc.debian.org` fino al 4 giugno 2006.

Per ottenere un copertura su freenode, si invii a Jörg Jasper <joerg@debian.org> una email firmata dove si indica il proprio nick. Mettere la parola «cloak» da qualche parte nel campo Oggetto:. Il nick deve essere registrato: Pagina sull'impostazione dei nick. La posta deve essere firmata da una chiave nel portachiavi Debian. Si consulti la Freenode documentation per ulteriori informazioni sulla copertura.

4.3 La documentazione

Questo documento contiene molte informazioni utili per gli sviluppatori Debian, ma non può contenere tutto. La maggior parte degli altri documenti interessanti sono indicati nell'Angolo degli sviluppatori. Prendetevi il tempo di sfogliare tutti i collegamenti, imparerete molte più cose.

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 Sezione 7.1.

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

Il server `ftp-master.debian.org` contiene la copia canonica dell'archivio Debian. In generale, il pacchetto caricato su ftp.upload.debian.org finisce su questo server, si consulti Sezione 5.6.

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 Sezione 5.9.

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.

Se si trova un problema con il server web di Debian, si dovrebbe generalmente segnalare un bug nei confronti dello pseudo-pacchetto `www.debian.org`. Ricordarsi di controllare se qualcun altro ha già segnalato il problema al Sistema di tracciamento dei bug.

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 I server VCS

Se è necessario utilizzare un sistema di controllo di versione per qualsiasi tipo di lavoro per Debian, è possibile utilizzare uno dei repository esistenti ospitati su Alioth oppure si può richiedere un nuovo progetto e chiedere il repository VCS che si preferisce. Alioth supporta CVS (cvs.alioth.debian.org/cvs.debian.org), Subversion (svn.debian.org), Arco (tla/baz, sia su arch.debian.org), Bazar (bzr.debian.org), Darcs (darcs.debian.org), Mercurial (hg.debian.org) e Git (git.debian.org). Controllare la pagina web `https://wiki.debian.org/Alioth/PackagingProject` se si prevede di mantenere pacchetti in un repository VCS. Per informazioni sui servizi forniti da Alioth si consulti Sezione 4.12.

4.4.6 chroot per diverse distribuzioni

Su alcune macchine, ci sono chroot disponibili per differenti distribuzioni. Si possono utilizzare in questo modo:

```
vore$ dchroot unstable
Executing shell in chroot: /org/vore.debian.org/chroots/user/unstable
```

In tutti i chroot, sono disponibili le normali cartelle home degli utenti. Si possono scoprire quali chroot sono disponibili tramite `https://db.debian.org/machines.cgi`.

4.5 Il Database degli sviluppatori

Il Database degli svulippatori, su `https://db.debian.org/`, è una cartella LDAP per la gestione di attributi degli sviluppatori Debian. È possibile utilizzare questa risorsa per cercare nell'elenco degli sviluppatori Debian. Parte di queste informazioni sono disponibili anche attraverso il servizio finger sui server Debian, si provi **finger propriologin@db.debian.org** per vedere cosa riporta.

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

- indirizzo a cui inoltrare la posta elettronica debian.org
- 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 15000 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/testing/
dists/testing/main/
     ...
dists/testing/contrib/
     ...
dists/testing/non-free/
     ...

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

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/f/
pool/non-free/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 contiene tre cartelle e cioè main, contrib e non-free.

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.

I pacchetti che non sono conformi alle DFSG sono collocati nella sezione non-free. Questi pacchetti non sono considerati come parte della distribuzione Debian, anche se è consentito il loro uso, e vengono messe a disposizione le infrastrutture (ad esempio, il sistema di tracciamento dei bug e mailing list) per i pacchetti software non-free.

Il Debian Policy Manual contiene una definizione più precisa delle tre sezioni. La discussione di cui sopra è solo un'introduzione.

La separazione delle tre sezioni al livello più alto di archivio è importante per tutte le persone che vogliono distribuire Debian, sia tramite server FTP su Internet sia su CD-ROM: distribuendo solo le sezioni main e contrib, si possono evitare eventuali rischi legali. Alcuni pacchetti nella sezione non-free non permettono la distribuzione commerciale, per esempio.

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.)

Si noti che il termine sezione è usato anche per riferirsi a categorie che semplificano l'organizzazione e la navigazione di pacchetti disponibili, ad esempio admin, net, utils, etc. Tempo fa, queste sezioni (sottosezioni, piuttosto) esistevano in forma di sottocartelle all'interno dell'archivio Debian. Al giorno d'oggi, queste esistono solo nei campi Section delle intestazioni dei pacchetti.

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 1.3 era disponibile solo come i386. Debian 2.0 era distribuita per le architetture i386 e m68k. Debian 2.1 distribuita per i386, m68k, alpha, sparc. Da allora Debian è cresciuta enormemente. Debian 6 supporta un totale di nove architetture Linux (amd64, armel, i386, ia64, mips, mipsel, powerpc, s390, sparc) e due architetture kFreeBSD (kfreebsd-i386 e 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}`.

Se un pacchetto è stato sviluppato appositamente per Debian e non è distribuito al di fuori di esso, c'è solo un file `.tar.{gz,bz2,xz}`, che contiene i sorgenti del programma, è chiamato pacchetto «nativo» dei sorgenti. Se un pacchetto viene distribuito anche altrove, il file `.orig.tar.{gz,bz2,xz}` contiene il cosiddetto `codice sorgente originale`, che è il codice sorgente che viene distribuito dal `maintainer originale` (spesso l'autore del software). In questo caso, il file `.diff.gz` o il file `debian.tar.{gz,bz2,xz}` contiene le modifiche apportate dal maintainer Debian.

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

Il sistema di cartelle descritto nel capitolo precedente è a sua volta contenuto all'interno delle `cartelle di distribuzione`. Ogni distribuzione è di fatto contenuta nella cartella `pool` nel livello più alto dell'archivio Debian stesso.

Per riassumere, l'archivio Debian ha una cartella radice all'interno di un server FTP. Ad esempio, presso il sito mirror, `ftp.us.debian.org`, l'archivio Debian in sé è contenuto in /debian, che è un luogo comune (un altro è `/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 Sezione 5.13.

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 caricamenti per la distribuzione stable per i dettagli.

Notare che lo sviluppo sotto `unstable` continua durante il periodo di congelamento, in quanto la distribuzione `unstable` resta in parallelo con `testing`.

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 informazioni sulla distribuzione di 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://ftp.xy.debian.org/debian/ experimental main
deb-src http://ftp.xy.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

Ogni distribuzione Debian rilasciata ha `un nome in codice`: Debian 1.1 è chiamata `buzz`, Debian 1.2, `rex`, Debian 1.3, `bo`, Debian 2.0, `hamm`, Debian 2.1, `slink`, Debian 2.2, `potato`, Debian 3.0, `woody`, Debian 3.1, `sarge`, Debian 4.0, `etch`, Debian 5.0, `lenny`, Debian 6.0, `squeeze` e la prossima versione si chiamerà `wheezy`. Vi è anche una «pseudo-distribuzione», chiamata `sid`, che è l'attuale distribuzione `unstable`; poiché i pacchetti vengono spostati da `unstable` a `testing` quando si avvicinano alla stabilità, `sid` in sé non è mai rilasciata. Oltre ai soliti contenuti di una distribuzione Debian, `sid` contiene i pacchetti per le architetture che non sono ancora ufficialmente supportate o rilasciate da Debian. È in progetto l'integrazione di queste architetture nella distribuzione tradizionale in una data futura.

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.

Si noti che i mirror sono generalmente gestiti da terzi che sono interessati ad aiutare Debian. Pertanto, gli sviluppatori in genere non hanno account su queste macchine.

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 `http://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.

Il software di manutenzione dell'archivio invierà anche il file `.changes` firmato con OpenPGP/GnuPG che si è inviato alle mailing list appropriate. Se un pacchetto è rilasciato con il campo `Distribution` impostato su `stable`, l'annuncio viene inviato a debian-changes@lists.debian.org. Se un pacchetto è rilasciato con il campo `Distribution` impostato a `unstable` o `experimental`, l'annuncio verrà pubblicato invece su debian-devel-changes@lists.debian.org.

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

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 | 0.1.5-2sarge1 |   oldstable | source, alpha, arm, hppa, i386, ia64, m68k ↩
    , mips, mipsel, powerpc, s390, sparc
evince |  0.4.0-5 |   etch-m68k | source, m68k
evince |  0.4.0-5 |    stable | source, alpha, amd64, arm, hppa, i386, ia64, mips ↩
    , mipsel, powerpc, s390, sparc
evince |  2.20.2-1 |    testing | source
evince | 2.20.2-1+b1 |    testing | alpha, amd64, arm, armel, hppa, i386, ia64,  ↩
    mips, mipsel, powerpc, s390, sparc
evince |  2.22.2-1 |   unstable | source, alpha, amd64, arm, armel, hppa, i386,  ↩
    ia64, m68k, mips, mipsel, powerpc, s390, sparc
```

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.

4.10 L'archivio Debian

Il Package Tracking System (PTS) è uno strumento basato sulla posta elettronica per monitorare l'attività di un pacchetto sorgente. Questo vuol dire che è possibile ottenere gli stessi messaggi di posta elettronica che riceve il maintainer del pacchetto, semplicemente iscrivendosi al pacchetto nel PTS.

Ogni email inviata attraverso il PTS è classificata, secondo una delle parole chiave sotto elencate. Questo permetterà di selezionare i messaggi che si desidera ricevere.

Per impostazione predefinita si otterrà:

`bts` Tutte le segnalazioni di bug e le successive discussioni.

`bts-control` Le notifiche email da control@bugs.debian.org sui cambiamenti di stato delle segnalazioni di bug.

`upload-source` La notifica email da **dak**, quando un pacchetto sorgente caricato è accettato.

`cvs` Altri avvertimenti e messaggi di errore da **dak** (ad esempio una disparità di override per il campo section e/o il campo priority).

`buildd` Le notifiche di errori di compilazione inviate dalla rete dei demoni di compilazione, contengono un riferimento ai log di compilazione per l'analisi.

`default` Qualsiasi email non automatica inviata al PTS da persone che volevano contattare coloro che sono iscritti al pacchetto. Questo può essere fatto mediante l'invio di email a *`pacchettosorgente`*`@packages.qa.debian.org`. Per evitare lo spam, tutti i messaggi inviati a questi indirizzi devono contenere l'intestazione `X-PTS-Approved` con un valore non vuoto.

`contact` Le email inviate al maintainer attraverso gli alias di posta elettronica *@packages.debian.org.

`summary` Email periodiche di riepilogo sullo stato del pacchetto, compreso l'avanzamento in `testing`, notifiche del DEHS su nuove versioni originali, e una notifica se il pacchetto viene rimosso o diventa orfano.

Si può anche decidere di ricevere informazioni supplementari:

`upload-binary` L'email di notifica da **katie**, quando un pacchetto binario caricato è accettato. In altre parole, ogni volta che un demone di compilazione o un autore di port carica il vostro pacchetto per un'altra architettura, è possibile ottenere una email per monitorare come il proprio pacchetto viene ricompilato per tutte le architetture.

`cvs` Le notifiche di commit di VCS, se il pacchetto ha un repository VCS e il maintainer ha impostato l'inoltro di notifiche di commit sul PTS. Il nome «cvs» è storico, nella maggior parte dei casi le notifiche di commit verranno da altri VCS come Subversion o Git.

`bts` Le traduzioni delle descrizioni o i modelli debconf presentati al Debian Description Translation Project.

derivatives Le informazioni sulle modifiche fatte al pacchetto nelle distribuzioni derivate (per esempio Ubuntu).

derivatives-bugs Le segnalazioni di bug e i commenti da distribuzioni derivate (ad esempio Ubuntu).

4.10.1 L'interfaccia email del PTS

È possibile controllare le proprie sottoscrizioni al PTS inviando vari comandi a pts@qa.debian.org.

subscribe <pacchettosorgente> [<email>] Si sottoscriva *email* per comunicazioni inerenti il pacchetto dei sorgenti di *sourcepackage*. L'indirizzo del mittente è utilizzato se il secondo argomento non è presente. Se *sourcepackage* non è un valido pacchetto di sorgenti, si riverà un avvertimento. Tuttavia, se è un valido pacchetto binario, il tracker del pacchetto iscriverà al corrispondente pacchetto sorgente.

unsubscribe <sourcepackage> [<email>] Rimuove una sottoscrizione precedente al pacchetto sorgente *pacchettosorgente* utilizzando l'indirizzo di posta elettronica specificato o l'indirizzo del mittente se il secondo argomento è stato lasciato vuoto.

unsubscribeall [<email>] Rimuove tutte le sottoscrizioni dell'indirizzo di posta elettronica specificato o l'indirizzo del mittente, se il secondo argomento è lasciato vuoto.

which [<email>] Elenca tutte le sottoscrizioni per il mittente o per l'indirizzo di posta elettronica eventualmente specificato.

keyword [<email>] Indica le parole chiave che si stanno accettando. Per una loro spiegazione, si veda sopra. Ecco un breve riassunto:

keyword <pacchetto sorgente> [<email>] Come la voce precedente ma per il dato pacchetto sorgente, dal momento che è possibile selezionare un diverso insieme di parole chiave per ogni pacchetto sorgente.

keyword [<email>] {+|-|=} <elenco di parole chiave> Accetta (+) o rifiuta (-) email classificate sotto le parole chiave. Definisce l'elenco (=) delle parole chiave accettate. Ciò cambia l'insieme predefinito di parole chiave accettate da un utente.

keywordall [<email>] {+|-|=} <elenco di parole chiave> Accetta (+) o rifiuta (-) email classificate sotto le parole chiave specificate. Definisce l'elenco (=) delle parole chiave accettate. Ciò cambia l'insieme di parole chiave accettate di tutte le sottoscrizioni attualmente attive di un utente.

keyword <pacchetto sorgente> [<email>] {+|-|=} <elenco di parole chiave> Come la voce precedente, ma sovrascrive l'elenco di parole chiave per il pacchetto sorgente indicato.

quit | thanks | -- Interrompe l'elaborazione dei comandi. Tutte le righe seguenti vengono ignorate dal bot.

L'utilità da riga di comando **pts-subscribe** (dal pacchetto `devscripts`) può essere utile per iscriversi temporaneamente ad alcuni pacchetti, per esempio dopo aver fatto un caricamento da non-maintainer.

4.10.2 Filtrare mail provenienti dal tracker del pacchetto

Once you are subscribed to a package, you will get mails forwarded by the package tracker. Those mails have special headers appended to let you filter them in a special mailbox (e.g. with **procmail**). The added headers are `X-Loop`, `X-Distro-Tracker-Package`, `X-Distro-Tracker-Keyword`, `X-Debian-Package`, `X-Debian`, `List-Id` and `List-Unsubscribe`.

Ecco un esempio di intestazioni aggiuntive per una notifica di un caricamento dei sorgenti sul pacchetto `dpkg`:

```
X-Loop: dispatch@tracker.debian.org
X-Distro-Tracker-Package: dpkg
X-Distro-Tracker-Keyword: upload-source
X-Debian-Package: dpkg
X-Debian: tracker.debian.org
List-Id: <dpkg.tracker.debian.org>
List-Unsubscribe: <mailto:control@tracker.debian.org?body=unsubscribe%20dpkg>
```

4.10.3 Inoltrare i commit del VCS nel PTS

Se si utilizza un repository VCS accessibile al pubblico per la manutenzione del proprio pacchetto Debian, si consiglia di inoltrare le notifiche di commit al PTS in modo che gli abbonati (ed i possibili co-maintainer) possano seguire da vicino l'evoluzione del pacchetto.

Una volta impostato il repository VCS per generare notifiche di commit, basta assicurarsi che si invii una copia di quelle mail al tracker del pacchetto al `dispatch@tracker.debian.org` o `dispatch+`*`sourcepackage`*`_vcs@tracker.debia`
Nel primo caso, è necessario assicurarsi che il tracker del pacchetto è in grado di identificare il pacchetto sorgente e la parola chiave appropriata ... o aggiungendo le intestazioni `X-Distro-Tracker-Package:` *`sourcepackage`* and `X-Distro-Tracker-Keyword: vcs` o basandosi sul fatto che il tracker del pacchetto rileverà l'header `X-Git-Repo` e supporrà che il nome del repository git corrisponda a un pacchetto sorgente.

Per i repository Subversion, si consiglia l'utilizzo di svnmailer. Si consulti https://wiki.debian.org/Alioth/PackagingProject per un esempio su come farlo.

4.10.4 L'intefaccia web di PTS

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`*.

L'interfaccia web è facile da estendere e si è invitati ad integrare i dati più utili in essa. Se si vuole contribuire, si dia un'occhiata a https://tracker.debian.org/docs/contributing.html.

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

Alioth è un servizio Debian basato su una versione leggermente modificata del software FusionForge (che si è evoluta da SourceForge e GForge). Questo software offre agli sviluppatori l'accesso a strumenti facili da usare, come bug tracker, responsabili delle patch, responsabili di progetto/attività, servizi di file hosting, mailing list, repository VCS ecc. Tutti questi strumenti sono gestiti tramite un'interfaccia web.

Esso è pensato per fornire servizi per progetti di software libero sostenuti o guidati da Debian, facilitare contributi da sviluppatori esterni a progetti avviati da Debian, e contribuire a progetti i cui obiettivi sono la promozione di Debian o le sue derivate. È molto utilizzato da molti team di Debian e fornisce hosting per tutti i tipi di repository VCS.

Tutti gli sviluppatori Debian hanno automaticamente un account su Alioth. Essi possono attivarlo utilizzando la funzionalità di recupero password. Gli sviluppatori esterni possono richiedere gli account ospite su Alioth.

Per ulteriori informazioni visitare i seguenti collegamenti:

- https://wiki.debian.org/Alioth
- https://wiki.debian.org/Alioth/FAQ
- https://wiki.debian.org/Alioth/PackagingProject
- https://alioth.debian.org/

4.13 Benefici per gli sviluppatori e menutentori Debian

Vantaggi a disposizione di sviluppatori e dei maintainer Debian sono documentati su 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 (Sezione 7.1) 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.

Si consiglia di impostare l'oggetto del bug a `ITP: foo -- breve descrizione` , sostituendo il nome del nuovo pacchetto in *foo*. La gravità della segnalazione di bug deve essere impostata su `wishlist`. Inviare una copia a debian-devel@lists.debian.org, utilizzando l'intestazione X-Debbugs-CC (non usare CC:, perché in questo modo il soggetto del messaggio non indicherà il numero di bug). Se si stanno pacchettizzando tanti nuovi pacchetti (> 10) tale che notificare alla mailing list con messaggi separati sia troppo dirompente, si invii invece un riepilogo dopo aver depositato i bug nella lista debian-devel. Questo informerà gli altri sviluppatori sui prossimi pacchetti e consentirà una revisione della vostra descrizione e nome del pacchetto.

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 Sezione 5.8.4).

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.

Consultare http://ftp-master.debian.org/REJECT-FAQ.html sui comuni motivi di rifiuto per un nuovo pacchetto.

5.2 Registrare i cambiamenti nel pacchetto

Le modifiche apportate al pacchetto devono essere registrate nel debian/changelog. Questi cambiamenti dovrebbero fornire una descrizione concisa di ciò che è stato modificato, del perché (se è in dubbio), e annotare se qualche bug è stato chiuso. Devono anche registrare quando il pacchetto è stato completato. Questo file verrà installato in /usr/share/doc/*package*/changelog.Debian.gz, o /usr/share/doc/*package*/changelog.gz per i pacchetti nativi.

Il debian/changelog si adegua ad una certa struttura, con una serie di campi differenti. Un campo di nota, la distribuzione, è descritto in Sezione 5.5. 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 Sezione 5.8.4.

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

```
* New upstream release.
```

Ci sono strumenti che consentono di creare voci e di finalizzare il changelog per la stampa - si consulti Sezione A.6.1 e Sezione A.6.5.

Si consulti anche Sezione 6.3.

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):

- Installare il pacchetto e assicurarsi che il software funzioni, o aggiornare il pacchetto da una versione precedente alla nuova versione, se già esiste un pacchetto Debian.
- Eseguite **lintian** sul pacchetto. È possibile eseguire **lintian** come segue: lintian -v *versione-pacchetto*.changes. Questo controllerà il pacchetto sorgente, così come il pacchetto binario. Se non si comprendono i risultati che **lintian** genera, provare ad aggiungere il commutatore -i, che farà produrre a **lintian** una descrizione molto dettagliata del problema.

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

  Per maggiori informazioni su **lintian**, si consulti Sezione A.2.1.
- Facoltativamente eseguite **debdiff** (si consulti Sezione A.2.2) per analizzare i cambiamenti da una versione precedente, se ne esiste una.
- Degradare il pacchetto alla versione precedente (se esistente): questo prova gli script di postrm e di prerm.
- Rimuovete il pacchetto, quindi reinstallarlo.
- 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.

La prima volta che viene caricata una versione che corrisponde ad una particolare versione originale, il file tar dei sorgenti originali deve essere caricato e incluso nel `.changes`. In seguito, questo stesso file tar deve essere utilizzato per costruire i nuovi file diff e `.dsc`, e non avrà bisogno di essere nuovamente caricato.

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.

Notare che, in pacchetti non nativi, i permessi per i file che non sono presenti nel `*.orig.tar.{gz,bz2,xz}` non saranno conservati, così come diff non memorizza i permessi dei file nella patch. Tuttavia, quando si usa il formato sorgente «3.0 (quilt)», i permessi dei file all'interno della cartella `debian` sono conservati dal momento che sono memorizzati in un archivio tar.

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 Sezione 5.8.5 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.

To ensure that your upload will be accepted, you should discuss the changes with the stable release team before you upload. For that, file a bug against the `release.debian.org` pseudo-package using **reportbug**, including the patch you want to apply to the package version currently in `stable`. The patch should be a source **debdiff** (see Sezione A.2.2) against the current version in `stable`. The changelog entry should be against the `stable` distribution (e.g. jessie') and should be detailed, including `Closes` statements for bugs that are fixed by the upload.

Particolare attenzione dovrebbe essere posta durante il caricamento in `stable`. In sostanza, un pacchetto deve essere caricato solo per `stable` se si verifica una delle seguenti circostanze:

- un problema di funzionalità davvero critica
- il pacchetto diventa non installabile
- una architettura rilasciata necessita del pacchetto

In passato, caricamenti in `stable` sono stati utilizzati per risolvere anche problemi di sicurezza. Tuttavia, questa pratica è sconsigliata, dato che i caricamenti utilizzati per la sicurezza di Debian vengono copiati automaticamente nell'appropriato archivio `proposed-updates` quando l'avviso viene rilasciato. Per informazioni

dettagliate sulla gestione dei problemi di sicurezza si consulti Sezione 5.8.5. Se i team di sicurezza ritengono che il problema sia troppo benevolo per essere risolto attraverso una DSA, i gestori del rilascio stable di solito sono disposti a includere comunque la correzione in un normale caricamento su stable.

Cambiare qualsiasi altra cosa nel pacchetto che non sia importante è sconsigliato, perché anche correzioni banali possono successivamente causare errori.

I pacchetti caricati su stable necessitano di essere compilati su sistemi che eseguono stable, in modo che le loro dipendenze siano limitate alle librerie (ed altri pacchetti) disponibili in stable; per esempio, un pacchetto caricato in stable che dipende da un pacchetto di libreria che esiste solo in unstable sarà respinto. Apportare modifiche alle dipendenze di altri pacchetti (modificando i file Provides o shlibs), eventualmente rendendo quegli altri pacchetti non installabili, è decisamente sconsigliato.

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

5.5.2 Caso particolare: caricamenti su testing/testing-proposed-updates

Consultare le informazioni nella sezione di test per i dettagli.

5.6 Caricare un pacchetto

5.6.1 Caricamento su ftp-master

Per caricare un pacchetto, è necessario caricare i file (incluse le modifiche firmate e i file .dsc) con ftp anonimo su ftp.upload.debian.org nella cartella /pub/UploadQueue/. Per ottenere che i file vengano processati, devono essere firmati con una chiave del portachiavi dei Developer Debian o dei Debian Maintainer (si veda 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.

Per la rimozione di pacchetti, si consulti ftp://ftp.upload.debian.org/pub/UploadQueue/README e il pacchetto Debian dcut.

5.6.2 Caricamenti differiti

A volte è utile caricare immediatamente un pacchetto, ma si desidera che quest'ultimo arrivi nell'archivio solo dopo qualche giorno. Ad esempio, quando si prepara un Non-Maintainer Upload, si potrebbe desiderare di dare al maintainer qualche giorno per reagire.

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.

Con dput, è possibile utilizzare il parametro --delayed *DELAY* per mettere il pacchetto in una delle code.

5.6.3 Caricamenti di sicurezza

NON caricare un pacchetto nella coda dei caricamenti di sicurezza (oldstable-security , stabili-security , etc.) senza la preventiva autorizzazione da parte del team di sicurezza. Se il pacchetto non soddisfa le esigenze del team, causerà molti problemi e ritardi nel gestire il caricamento indesiderato. Per i dettagli, si consulti Sezione 5.8.5.

5.6.4 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.

5.6.5 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 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.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 Sezione 4.6.1. 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 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 Sezione 7.1), 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 .

5.8.1 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 , 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.

5.8.2 Rispondere ai bug

Quando si risponde ad un bug, fare in modo che ogni discussione che si ha sui bug venga inviata sia al mittente originario del bug, e per il bug stesso (ad esempio, *123*@bugs.debian.org). Se si sta scrivendo un nuovo messaggio e non si ricorda l'indirizzo di posta elettronica del mittente, è possibile utilizzare l'email *123*-submitter@bugs.debian.org per contattare il mittente *e* per registrare la propria posta dentro il log del bug (il che significa che non è necessario inviare una copia della email a *123*@bugs.debian.org).

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 Sezione 5.8.4.

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.

5.8.3 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 Sezione 7.1 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.

   Se il mittente del bug non è d'accordo con la vostra decisione di chiuderlo, può riaprirlo fino a quando non si trovi un accordo su come gestirlo. Se non si trova, si consiglia di contrassegnare il bug `wontfix`, per far sapere che il bug esiste ma che non sarà corretto. Se questa situazione è inaccettabile, il maintainer (o il mittente) può richiedere una decisione del comitato tecnico riassegnando il bug a `tech-ctte` (si può usare il comando clone del BTS se si desidera tenerlo riportato sul pacchetto). Prima di procedere, leggere le 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 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. Se il bug è legato alla pacchettizzazione, basta risolvere il problema. Se non si è in grado di risolverlo da soli, allora si contrassegni il bug come `help`. Si può anche chiedere aiuto su debian-devel@lists.debian.org o debian-qa@lists.debian.org. Se è un problema che riguarda il software originale, è necessario inoltrarlo all'autore originale. L'inoltro di un bug non è sufficiente, è necessario controllare ad ogni rilascio se il bug è stato risolto o meno. Se lo è, basta chiuderlo, altrimenti si deve ricordarlo all'autore. Se si hanno le competenze necessarie si può preparare una patch che corregge il bug e inviarla all'autore allo stesso tempo. Assicurarsi di inviare la patch al BTS e di contrassegnare il bug come `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. Una volta che un pacchetto corretto è disponibile in archivio, il bug dovrebbe essere chiuso indicando la versione in cui è stato corretto. Questo può essere fatto automaticamente, si consulti Sezione 5.8.4.

5.8.4 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
```

Noi preferiamo la sintassi `closes: #`*XXX*, in quanto è la voce più concisa e più facile da integrare con il testo del `changelog`. Se non diversamente specificato dal parametro `-v-` di**dpkg-buildpackage**, solo i bug risolti nella

più recente voce del changelog vengono chiusi (in pratica, esattamente i bug menzionati nella parte del changelog nel file `.changes` vengono chiusi).

Storicamente, i caricamenti individuati come non-maintainer upload (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`.

Se capita di sbagliare un numero di bug o dimenticare un bug nel changelog, non esitare ad annullare qualsiasi danno causato dall'errore. Per riaprire bug erroneamente chiusi, inviare un comando `reopen` *XXX* all'indirizzo del sistema di controllo dei bug, control@bugs.debian.org. Per chiudere ogni restante bug che è stato corretto dal proprio caricamento, inviare il file `.changes` per email a *XXX*-done@bugs.debian.org, dove *XXX* è il numero di bug, e mettere Version: *YYY* e una riga vuota come prime due righe del corpo della email, dove *YYY* è la prima versione nella quale è stato corretto il bug.

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 Sezione 6.3.

5.8.5 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 . 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 Sezione 5.8.5.4 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 Sezione 5.8.5.3)

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.

5.8.5.1 The 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.

5.8.5.2 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.

5.8.5.3 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)
- I riferimenti agli avvisi originali, identificatori CVE, e ogni altra informazione utile nel documentare la vulnerabilità

5.8.5.4 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.

Nel momento in cui viene eseguito un aggiornamento alla versione stable, deve essere adottata molta cura per evitare di modificare il comportamento del sistema o di introdurre nuovi bug. Per fare questo, si faccia il minor numero di modifiche possibili per risolvere il bug. Gli utenti e gli amministratori si affidano all'esatto comportamento di un rilascio una volta che viene fatto, quindi qualsiasi modifica apportata potrebbe rompere il sistema di qualcuno. Questo è particolarmente vero per le biblioteche: assicurarsi di non modificare l'API o ABI, non importa quanto piccolo sia la modifica.

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 Sezione A.2.2).

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`!
- Il caricamento deve avere **urgency=high**.
- 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 Sezione 4.4) oppure si configuri un chroot (si consulti Sezione A.4.3 e Sezione A.4.2).

5.8.5.5 Caricamento del pacchetto corretto

NON caricare un pacchetto nella coda di caricamento di sicurezza (`oldstable-security`, `stable-security`, etc.) senza la preventiva autorizzazione da parte del team di sicurezza. Se il pacchetto non soddisfa le esigenze del team, causerà molti problemi e ritardi nel trattare con il caricamento indesiderato.

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.

Dopo aver creato e testato il nuovo pacchetto ed è stato approvato dal team di sicurezza, occorre caricarlo in modo che possa essere installato negli archivi. Per aggiornamenti di sicurezza, il posto giusto per farlo è `ftp://security-master.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.

I caricamenti che sono in attesa di accettazione o di verifica sono accessibili solo da parte del team di sicurezza. Ciò è necessario in quanto ci potrebbero essere correzioni per i problemi di sicurezza che non possono essere ancora rivelati.

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`.

5.9 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.

5.9.1 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 Sezione 5.7.

5.9.2 Rimozione dei pacchetti

Se per qualche motivo si desidera rimuovere completamente un pacchetto (ad esempio, se si tratta di una vecchia libreria di compatibilità che non è più richiesta), è necessario presentare un bug su `ftp.debian.org` chiedendo che il pacchetto sia rimosso; come tutti i bug, questo bug dovrebbe di norma avere gravità normal. Il titolo della segnalazione del bug dovrebbe essere della forma `RM:` *`package[architecture list]--reason`*, dove *`package`* è il pacchetto da rimuovere e *`reason`* è una breve sintesi del motivo per la richiesta di rimozione. *`[architecture list]`* è facoltativo e necessario solo se la richiesta di rimozione vale solo per alcune architetture, non tutte. Si noti che il **reportbug** creerà un titolo conforme a queste regole quando lo si utilizza per segnalare un bug per lo psudo-pacchetto `ftp.debian.org`.

Se si vuole rimuovere un pacchetto che si mantiene, lo si dovrebbe evidenziare nel titolo del bug anteponendo `ROM` (Request Of Maintainer). Ci sono diversi altri acronimi standard utilizzati nel motivare una rimozione di un pacchetto, si consulti per un elenco completo. Quella pagina fornisce anche una comoda visione generale delle richieste di rimozione in sospeso.

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

Si noti che le rimozioni possono essere fatte solo per la distribuzione `unstable`, `experimental` and `stable`. I pacchetti non vengono rimossi da `testing` direttamente. Piuttosto, essi saranno rimossi automaticamente dopo che il pacchetto è stato rimosso da `unstable` e nessun pacchetto in `testing` dipenda da esso. (Rimozioni da `testing` sono possibili presentando una segnalazione di bug di rimozione su `release.debian.org`. Si consulti la sezione Sezione 5.13.2.2.)

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.

Ulteriori informazioni relative a questi ed altri argomenti correlati alla rimozione di un pacchetto possono essere trovate in `http://wiki.debian.org/ftpmaster_Removals` e `https://qa.debian.org/howto-remove.html`.

In caso di dubbio concernente il fatto che un pacchetto sia usa e getta, si mandi una email a debian-devel@lists.debian.org chiedendo pareri. Interessante è anche il programma **apt-cache** dal pacchetto `apt`. Quando invocato come `apt-cache showpkg` *`package`*, il programma mostrerà i dettagli per *`package`*, incluse le dipendenze revocate. Altri programmi utili sono **apt-cache rdepends**, **apt-rdepends**, **build-rdeps** (nel pacchetto `devscripts`) e **grep-dctrl**. La rimozione di pacchetti orfani è discussa su 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`.

5.9.2.1 Rimozione di pacchetti da `Incoming`

In passato, è stato possibile rimuovere i pacchetti da `incoming`. Tuttavia, con l'introduzione del nuovo sistema di gestione dei caricamenti, questo non è più possibile. Invece, è necessario caricare una nuova revisione del pacchetto con una versione superiore rispetto a quello che si desidera sostituire. Entrambe le versioni saranno installate in archivio, ma solo la versione più alta sarà effettivamente disponibile in `unstable` dal momento che la versione precedente verrà immediatamente sostituita dalla più alta. Tuttavia, se si fa una corretta analisi dei pacchetti, la necessità di sostituire un pacchetto non dovrebbe avvenire troppo spesso comunque.

5.9.3 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 Sezione 5.9.2). 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 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.

5.9.4 Pacchetto orfano

Se non è più possibile mantenere un pacchetto, è necessario informare gli altri, e controllare che il pacchetto sia contrassegnato come orfano. È necessario impostare il maintainer del pacchetto a `Debian QA Group <packages@qa.debian` e inviare una segnalazione di bug per lo pseudo pacchetto `wnpp`. La segnalazione deve essere intitolata `O: package --breve descrizione` che indica che il pacchetto è ormai orfano. La gravità del bug dovrebbe essere impostata su `normal`, se il pacchetto ha una priorità di livello standard o superiore, deve essere impostato su important. Se si ritiene necessario, inviare una copia a debian-devel@lists.debian.org, mettendo l'indirizzo nell'intestazione X-Debbugs-CC: del messaggio (no, non usare CC:, perché in questo modo l'oggetto del messaggio non indicherà il numero di bug).

Se proprio si ha intenzione di abbandonare il pacchetto, ma è possibile mantenerlo temporaneamente, allora si dovrebbe invece presentare un bug su `wnpp` ed inserire come titolo `RFA: package -- breve descrizione`. `RFA` è l'acronimo di `Request For Adoption`.

Maggiori informazioni si possono trovare nelle pagine web di WNPP.

5.9.5 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.

Non è appropriato rilevare semplicemente un pacchetto che si immagina sia trascurato: ciò sarebbe un furto di un pacchetto. È possibile, ovviamente, contattare il maintainer attuale e chiedere se si può prendere in consegna il pacchetto. Se si ha motivo di credere che un maintainer abbia abbandonato il lavoro senza avvisare (AWOL), si consulti Sezione 7.4.

In generale, non si può prendere in consegna il pacchetto senza il consenso dell'attuale maintainer. Anche se si è ignorati, ciò non è ancora un motivo sufficiente per adottare un pacchetto. Le segnalazioni riguardanti i maintainer devono essere fatte sulla mailing list degli sviluppatori. Se la discussione non termina con una conclusione positiva, e la questione è di natura tecnica, si consideri di portarla all'attenzione del comitato tecnico (si consulti per maggiori informazioni la pagina web del comitato tecnico).

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 Sezione 4.10 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.

5.9.6 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 (Sezione 5.1).

Si dovrebbe basare il proprio lavoro sull'ultimo pacchetto disponibile. Questo potrebbe essere l'ultima versione da `unstable`, che sarà ancora presente nella archivio snapshot.

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.

5.10 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.

Fare un port significa creare pacchetti Debian per architetture diverse dall'architettura originale del pacchetto binario del maintainer. È un'attività unica ed essenziale. In realtà, gli autori di port effettuano la maggior parte della compilazione dei pacchetti Debian. Per esempio, quando un maintainer carica un pacchetto di sorgenti (portabili) con i binari per l'architettura `i386`, sarà compilato per ciascuna delle altre architetture, pari ad altre 11 compilazioni.

5.10.1 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.

La prima e la più importante cosa è quella di rispondere rapidamente a bug o problemi sollevati dagli autori di port. Trattare gli autori di port con cortesia, come se fossero nei fatti co-maintainer del pacchetto (cosa che, in un certo senso, sono). Si sia tolleranti nei confronti di segnalazioni di bug scarne o addirittura non chiare; fare del prorpio meglio per dare la caccia a qualsiasi problema.

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. Si verifichi che le impostazioni di `Build-Depends` e `Build-Depends-Indip` in `debian/control` siano impostate correttamente. Il modo migliore per verificarlo è usare il pacchetto `debootstrap` per creare un ambiente chroot `unstable` (si consulti Sezione A.4.2). All'interno di tale ambiente chroot, si installi il pacchetto `build-essential` e tutte le dipendenze dei pacchetti citate in `Build-Depends` o in`Build-Depends-Indep`. Infine, si provi a compilare il pacchetto all'interno di tale ambiente chroot. Queste operazioni possono essere automatizzate con l'uso del programma **pbuilder** che è fornito dal pacchetto omonimo (si consulti Sezione A.4.3).

   Se non è possibile impostare una corretta chroot, **dpkg-depcheck** può essere di aiuto (si consulti Sezione A.6.6).

   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. Assicurarsi che il proprio pacchetto dei sorgenti sia corretto. Si invochi `dpkg-source -x` *`pacchetto`*`.dsc` per assicurarsi che il pacchetto dei sorgenti si spacchetti correttamente. Poi, lì dentro, si provi a compilare il pacchetto da zero con **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. Assicuratevi di non basarvi su configurazioni o programmi installati localmente o modificati. Per esempio, non si dovrebbe mai invocare programmi presenti in `/usr/local/bin` o simili. Si cerchi di non fare affidamento su programmi configurati in modo particolare. Si provi a compilare il proprio pacchetto su un'altra macchina, anche se è la stessa architettura.

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**.

5.10.2 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 Sezione 5.11.1.

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`.

Il modo per richiamare **dpkg-buildpackage** è come `dpkg-buildpackage -B -m`*`porter-mail`*. Naturalmente, si indichi in *`porter-mail`* la propria email. Questo farà una compilazione binaria delle sole parti del pacchetto dipendenti dall'architettura, utilizzando il target `binary-arch` in `debian/rules`.

Se si sta lavorando su una macchina Debian per fare il port e si ha bisogno di firmare il proprio caricamento localmente per la sua accettazione nell'archivio, è possibile eseguire **debsign** sul proprio file `.changes` per averlo comodamente firmato, oppure si utilizzi la modalità di firma remota **dpkg-sig**.

5.10.2.1 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.

5.10.2.2 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 Sezione 5.11, 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.

[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.

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.

5.10.3 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.

5.10.3.1 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.

5.10.3.2 Strumenti per gli autori di port

Le descrizioni di diversi strumenti per il port si possono trovare su Sezione A.7.

5.10.3.3 `wanna-build`

Il sistema `wanna-build` viene utilizzato come sistema client-server distribuito per la distribuzione di compilazioni. Di solito è usato in combinazione con demoni per la compilazione che eseguono il programma `buildd`. `I demoni per la compilazione` sono host «slave» che contattano il sistema centrale `wanna-build` per ricevere una lista di pacchetti che hanno bisogno di essere compilati.

`wanna-build` non è ancora disponibile come pacchetto; tuttavia, tutti gli sforzi per creare port di Debian lo utilizzano per la compilazione automatizzata dei pacchetti. Lo strumento utilizzato per fare l'effettiva compilazione dei pacchetti, `sbuild` è disponibile come un pacchetto, si consulti la sua descrizione in Sezione A.4.4. Si noti che la versione pacchettizzata non è la stessa di quella utilizzata dai demoni di compilazioni, ma è abbastanza vicina per riprodurne i problemi.

La maggior parte dei dati prodotti da `wanna-build` che sono generalmente utili per gli autori di port è disponibile sul web all'indirizzo `https://buildd.debian.org/`. Questi dati includono le statistiche aggiornate ogni notte, informazioni sulle code e i log dei tentativi di compilazione.

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`.

5.10.4 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://anonscm.debian.org/cgit/mirror/packages-arch-specific.git/tree/Packages-arch-specific`; consultare la parte iniziale del file per chi contattare per le modifiche.

Si noti che non è sufficiente aggiungere solo il pacchetto in `Packages-arch-specific`, senza far fallire la compilazione sulle architetture non supportate: un autore di port o una qualsiasi altra persona che cerca di compilare il pacchetto potrebbe accidentalmente caricarlo senza notare che non funziona. Se in passato alcuni pacchetti binari sono stati caricati su architetture non supportate, si richieda la loro rimozione segnalando un bug su `ftp.debian.org`.

5.10.5 Marcare i pacchetti non-free come compilabili automaticamente

In modo definitivo i pacchetti della sezione `non-free` non sono compilati dalla rete di compilazione automatica (soprattutto perché la licenza dei pacchetti potrebbe non approvarlo). Per abilitare la compilazione di un pacchetto è necessario eseguire le seguenti operazioni:

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. Si invii una email a nonfree@release.debian.org e si spieghi perché il pacchetto può legittimamente e tecnicamente essere automaticamente compilato.

5.11 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)*.

5.11.1 Quando e come fare un NMU

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

- Il proprio NMU davvero corregge i bug? Risolvere problemi superficiali o cambiare lo stile di pacchettizzazione è scoraggiato in una NMU.
- Si è concesso abbastanza tempo al maintainer? Quando è stato segnalato il bug su BTS? Essere occupato per una settimana o due non è insolito. Il bug è così grave che deve essere risolto in questo momento, o può aspettare ancora qualche giorno?

- 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.

Mentre si prepara la patch, si dovrebbero conoscere bene tutte le pratiche specifiche del pacchetto che il maintainer potrebbe utilizzare. Tenerne conto riduce l'onere di integrare i propri cambiamenti nel normale flusso di lavoro del pacchetto e quindi aumenta la probabilità che l'integrazione avvenga. Un buon posto dove cercare per possibili pratiche specifiche sul pacchetto è `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

- Altri NMU: 10 giorni

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`.

A volte, i gestori dei rilasci decidono di permettere NMU con ritardi più brevi per un sottoinsieme di bug (ad esempio bug critici per il rilascio più vecchi di 7 giorni). Inoltre, alcuni maintainer si inseriscono nell'elenco Low Threshold NMU, e accettano che questi NMU vengano caricati senza ritardo. Ma anche in questi casi, è sempre una buona idea concedere al maintainer un paio di giorni per reagire prima di fare il caricamento, soprattutto se la patch non era prima disponibile nel BTS, o se si sa che il maintainer è generalmente attivo.

Dopo aver caricato un NMU, si è responsabili per i possibili problemi che si potrebbe avere introdotto. È necessario tenere d'occhio il pacchetto (iscriversi al pacchetto sul PTS è un buon modo per raggiungere questo obiettivo).

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.

5.11.2 NMU e `debian/changelog`

Proprio come qualsiasi altro caricamento (di sorgenti), gli NMU deveno aggiungere una voce a `debian/changelog`, descrivendo cosa è cambiato con questo caricamento. La prima riga di questa voce deve esplicitamente menzionare che questo caricamento è un NMU, ad esempio:

```
* 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`).

5.11.3 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 Sezione 5.6.2) 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.

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).

5.11.4 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.

5.11.5 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 Sezione 5.10.2.1. 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`.

5.11.6 NMU e caricamenti di QA

Gli NMU sono caricamneti di pacchetti effettuati da qualcun altro che non è il maintainer assegnato. Vi è un altro tipo di caricamento in cui il pacchetto caricato non è vostro: caricamenti di QA. Questi ultimi sono aggiornamenti di pacchetti orfani.

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`.

Invece di fare un caricamento QA, si può anche prendere in considerazione l'adozione del pacchetto diventando il maintainer. Non è necessario il permesso di nessuno per adottare un pacchetto orfano, basta impostare se stessi come maintainer e caricare la nuova versione (si consulti Sezione 5.9.5).

5.11.7 NMU e caricamenti del team

A volte si sta correggendo o aggiornando un pacchetto perché si è membri di una gruppo di pacchettizzazione (che si avvale di una mailing list come `Maintainer` o `Uploader`, si consulti Sezione 5.12), ma non si desidera aggiungere se stessi agli `Uploader`, perché non si prevede di contribuire regolarmente a questo pacchetto specifico. Se è conforme con la politica del team, è possibile eseguire un caricamento normale senza figurare direttamente come `Maintainer` o `Uploader`. In tal caso, si dovrebbe iniziare la propria voce del changelog con la seguente riga:

```
* Team upload.
```

5.12 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:

- Si imposti il co-maintainer con accesso ai sorgenti dai quali si compila il pacchetto. Generalmente questo significa che si sta utilizzando un sistema di controllo versione con capacità di rete, come `CVS` o `Subversion`. Alioth (si consulti Sezione 4.12) fornisce tali strumenti, tra gli altri.
- 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>
```

- Si utilizzi il PTS (Sezione 4.10), i co-maintainer dovrebbero iscriversi all'appropriato pacchetto di sorgenti.

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. Inserire l'indirizzo della mailing list nel campo `Maintainer`. Nel campo `Uploaders`, inserire i membri del team che si interessano al pacchetto. In questo caso, è necessario assicurarsi che la mailing list accetti le segnalazioni di bug senza alcuna interazione umana (come la moderazione per i non iscritti).

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 Sezione 4.11) 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 Sezione 5.11.7). Al contrario, è una cattiva idea tenere un pacchetto con il solo indirizzo della mailing list come `Maintainer` e senza nessuno `Uploaders`.

5.13 La distribuzione testing

5.13.1 Nozioni di base

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

5.13.2 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:

- Il pacchetto deve essere stato disponibile in `unstable` per 2, 5 o 10 giorni, a seconda dell'urgenza (alta, media o bassa). Notare che l'urgenza è appiccicosa, il che significa che viene presa in considerazione la massima urgenza caricata dalla precedente transizione in `testing`;
- 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`. 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 crontab(5) 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.

Qualche ulteriore analisi delle dipendenze viene visualizzata su `http://release.debian.org/migration/`, ma attenzione, questa pagina mostra anche le dipendenze per la compilazione che non sono considerate da britney.

5.13.2.1 Obsoleti

Per lo script di migrazione di `testing`, obsoleti significare che: ci sono versioni diverse in `unstable` per le architetture di rilascio (ad eccezione per le architetture fuckedarches; fuckedarches è un elenco di architetture che non tengono il passo (in `update_out.py`), ma attualmente, è vuoto). Obsoleto non ha nulla a che fare con le architetture che questo pacchetto ha in `testing`.

Si consideri questo esempio:

	alpha	**arm**
testing	1	-
unstable	1	2

Il pacchetto non è aggiornato in `alpha` in `unstable`, e non andrà in `testing`. La rimozione del pacchetto non aiuterebbe affatto, il pacchetto è ancora obsoleto in `alpha`, e non passerà in `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.)

5.13.2.2 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.

Naturalmente, c'è un altro motivo per rimuovere un pacchetto da `testing`: è troppo pieno di bug (e avere un unico bug RC è sufficiente per essere in questo stato).

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

5.13.2.3 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.

5.13.2.4 Influenza del pacchetto in testing

In generale, non c'è nulla nello stato di un pacchetto in `testing` che influenzi il passaggio della prossima versione da `unstable` a `testing`, con due eccezioni: se il numero di bug RC del pacchetto diminuisce, il pacchetto potrebbe entrare anche se ha ancora bug RC. La seconda eccezione è se la versione del pacchetto in `testing` è fuori sincrono sulle diverse architetture: allora ogni architettura potrebbe semplicemente eseguire l'aggiornamento alla versione dei sorgenti del pacchetto; ma questo può avvenire solo se il pacchetto è stato precedentemente forzato, se l'architettura è in fuckedarches, o se non c'era nessun pacchetto binario di quell'architettura presente in `unstable` durante la migrazione su `testing`.

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.

5.13.2.5 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.

Se si vuole avere maggiori dettagli, è possibile cercare su `http://ftp-master.debian.org/testing/update_output/`.

I suggerimenti sono disponibili tramite `http://ftp-master.debian.org/testing/hints/`, dove è possibile trovare anche la descrizione. Con i suggerimenti, il team di rilascio di Debian può bloccare o sbloccare i pacchetti, spingere o forzare pacchetti in `testing`, rimuovere pacchetti da `testing`, approvare caricamenti al testing-proposed-updates o sovrascrivere l'urgenza.

5.13.3 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`.

Si tenga a mente che i pacchetti che sono inviati lì non vengono elaborati automaticamente, devono passare attraverso le mani del gestore del rilascio. Quindi è meglio avere un buon motivo per caricarli lì. Per sapere qual è un buon motivo agli occhi del gestore del rilascio, si consiglia di leggere le indicazioni che sono fornite regolarmente su debian-devel-announce@lists.debian.org.

Si consiglia di non fare il caricamento su `testing-proposed-updates` quando è possibile aggiornare i pacchetti attraverso `unstable`. Se non è possibile (ad esempio perché si dispone di una versione di sviluppo più recente in `unstable`), è possibile utilizzare questo strumento, ma si consiglia di chiedere prima l'autorizzazione al gestore del rilascio. Anche se un pacchetto è congelato, sono possibili aggiornamenti tramite `unstable`, se il caricamento tramite `unstable` non aggiunge nuove dipendenze.

I numeri della versione sono solitamente individuati concatenando `+debXuY`, 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+deb8u1`.

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;
- Assicurarsi di aver scritto `testing` o `testing-proposed-updates` come distribuzione di destinazione;
- 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`;
- 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.

5.13.4 Domande frequenti

5.13.4.1 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`.

Il conteggio dei bug in `unstable` sono tutti i bug critici per il rilascio che sono contrassegnati come relativi a combinazioni *`pacchetto`*/*`versione`* disponibili in unstable per un'architettura di rilascio. Il conteggio dei bug in `testing` è definito in modo analogo.

5.13.4.2 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.

Tuttavia, la vecchia versione potrebbe aver fornito un pacchetto binario con un vecchio nome di una libreria che è in uso, come `libacme-foo0`. La rimozione del vecchio `acmefoo` rimuoverà `libacme-foo0`, che renderà difettosi tutti i pacchetti che dipendono da essa.

Evidentemente, questo riguarda principalmente i pacchetti che forniscono insiemi diversi di pacchetti binari in versioni differenti (principalmente le librerie). Tuttavia, ne saranno affetti anche i pacchetti verso i quali sono state dichiarate dipendenze con numero di versione (versionate) del tipo ==, <=, or <<

Quando l'insieme di pacchetti binari forniti da un pacchetto sorgente è modificato in questo modo, tutti i pacchetti che dipendevano dai precedenti binari devono essere aggiornati in modo da dipendere dai nuovi. Poiché l'installazione di un pacchetto sorgente in `testing` rende difettosi tutti i pacchetti che ne dipendono in `testing`, in tale momento deve essere posta una certa attenzione: tutti i pacchetti dipendenti devono essere aggiornati e pronti per essere installati in modo da non risultare difettosi, e, una volta che tutto è pronto, è normalmente richiesto un intervento manuale del gestore del rilascio o di un assistente.

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.

Capitolo 6

Buone pratiche per la pacchettizzazione

La qualità della distribuzione Debian è in gran parte dovuta alla Debian Policy, che definisce i requisiti di base espliciti che tutti i pacchetti Debian devono soddisfare. Ma vi è anche una storia condivisa di esperienza che va oltre la policy Debian, un accumulo di anni di esperienza nella pacchettizzazione. Molte persone di grande talento hanno creato grandi strumenti, strumenti che aiutano voi, i maintainer di Debian, a creare e mantenere ottimi pacchetti.

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`

Le seguenti raccomandazioni si applicano al file `debian/rules`. Dal momento che il `debian/rules` controlla il processo di generazione e seleziona i file da inglobare nel pacchetto (direttamente o indirettamente), è normale che i maintainer del file spendano molto tempo su di esso.

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.

Appendice A 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`.

Si può iniziare con `debhelper` leggendo debhelper(1), e guardando gli esempi distribuiti con il pacchetto. **dh_make**, dal pacchetto `dh-make` (si consulti Sezione A.3.2), può essere utilizzato per convertire un pacchetto sorgente normale in un pacchetto `debhelper`izzato. Questa scorciatoia, però, non deve convincere che non è necessario preoccuparsi di capire i singoli script **dh_ ***. Se si ha intenzione di utilizzare uno script di supporto, ci si deve concedere il tempo necessario per imparare ad usare quello script, per imparare le sue previsioni e il suo comportamento.

6.1.2 Separare le proprie patch in più file

Pacchetti grandi e complessi possono avere molti bug con cui ci si deve rapportare. Se si corregge una serie di bug direttamente nel sorgente, e non si sta attenti, può diventare difficile distinguere le varie patch che si sono applicate. Può essere abbastanza caotico quando è necessario aggiornare il pacchetto ad una nuova versione che integra alcune delle correzioni (ma non tutte). Non si può prendere l'intero insieme di diff (ad esempio, da `.diff.gz`) e capire quali patch occorrono per tornare indietro di una unità man mano che i bug vengono corretti nel sorgente originale.

Fortunatamente, con il formato sorgente «3.0 (quilt)» è ora possibile mantenere le patch separate senza dover modificare `debian/rules` per impostare un sistema di patch. Le patch vengono memorizzate in `debian/patches/` e quando il pacchetto sorgente è spacchettato le patch elencate nel `debian/patches/series` vengono applicate automaticamente. Come suggerisce il nome, le patch possono essere gestite con **quilt**.

Quando si utilizza il più anziano sorgente «1.0», è anche possibile separare le patch, ma un sistema di patch dedicato deve essere utilizzato: i file di patch sono distribuiti all'interno del file di patch Debian (`diff.gz`.), di solito nella cartella `debian/`. L'unica differenza è che non vengono applicate immediatamente da **dpkg-source**, ma dalla regola `build` di `debian/rules`, attraverso una dipendenza dalla regola `patch`. Al contrario, essi sono annullati nella regola `clean`, attraverso una dipendenza dalla regola `unpatch`.

quilt è lo strumento consigliato per questo. Fa tutto quanto detto precedentemente e permette anche di gestire le serie di patch. Si veda il pacchetto `quilt` per ulteriori informazioni.

Ci sono altri strumenti per gestire le patch, come **dpatch** e il sistema di patch integrato con `cdbs`.

6.1.3 Pacchetti binari multipli

Un singolo pacchetto sorgente costruirà spesso diversi pacchetti binari, sia per fornire diverse versioni dello stesso software (ad esempio, il pacchetto sorgente `vim`) o per fare diversi piccoli pacchetti invece di uno grande (ad esempio, in modo che l'utente possa installare solo il sottoinsieme necessario e quindi di risparmiare spazio su disco).

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`.

Il primo caso è un po' più difficile in quanto comporta molteplici ricompilazioni dello stesso software, ma con diverse opzioni di configurazione. Il pacchetto sorgente `vim` è un esempio di come gestirlo utilizzando un file `debian/rules` scritto a mano.

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. Sezione 6.2.1 descrive le linee guida comuni ad entrambe le parti della descrizione del pacchetto. In seguito, Sezione 6.2.2 fornisce specifiche linee guida per la sinossi e Sezione 6.2.3 contiene specifiche linee guida per la descrizione.

6.2.1 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.

6.2.2 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: esempio di libreria di supporto
```

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*.

6.2.3 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.

I paragrafi che seguono devono rispondere alle seguenti domande: Perché, come utente, ho bisogno di questo pacchetto? Quali altre caratteristiche ha il pacchetto? Quali le caratteristiche e le carenze ci sono rispetto ad altri pacchetti (per esempio, se si ha bisogno di X, usare Y invece)? Questo pacchetto è correlato ad altri pacchetti in qualche modo che non viene gestito dal gestore di pacchetti (per esempio, questo è il client per il server foo)?

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.
- In che modo questo pacchetto si differenzia da altri? È un'implementazione migliore? più funzioni? caratteristiche diverse? Perché dovrei scegliere questo pacchetto.

6.2.4 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.

6.2.5 La posizione del Version Control System

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

6.2.5.1 Vcs-Browser

Il valore di questo campo dovrebbe essere una URL `http://` che punta a una copia web navigabile del Version Control System utilizzato per mantenere il pacchetto, se disponibile.

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).

6.2.5.2 Vcs-*

Il valore di questo campo deve essere una stringa che identifica in modo inequivocabile la posizione del repository del Version Control System utilizzato per mantenere il pacchetto, se disponibile. `*` individua il Version Control System; attualmente i seguenti sistemi sono supportati dal package tracking system: `arch`, `bzr` (Bazaar), `cvs`, `darcs`, `git`, `hg` (Mercurial), `mtn` (Monotone), `svn` (Subversion). È consentito specificare diversi campi VCS per lo stesso pacchetto: saranno tutti mostrati nell'interfaccia web di PTS.

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.

Nel seguente esempio è mostrata un'istanza del campo per un repository Subversion del pacchetto `vim`. Si noti come l'URL è nello schema `svn://` (invece di `svn+ssh://`) e come si punti al branch `trunk/`. È anche mostrato l'uso dei campi del `Vcs-Browser` e `Homepage` descritti precedentemente.

```
Source: vim
Section: editors
Priority: optional
<snip>
Vcs-Svn: svn://svn.debian.org/svn/pkg-vim/trunk/packages/vim
Vcs-Browser: http://svn.debian.org/wsvn/pkg-vim/trunk/packages/vim
Homepage: http://www.vim.org
```

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 Sezione 5.8.4.

6.3.2 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 Sezione 5.8.4.

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 Sezione 5.8.2.

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.

6.3.3 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 FU w/ ancient CRTs.
```

Troppe sigle e non è troppo chiaro a quale la, uh, fsckup (ops, una parolaccia!) si riferiva, o come è stato sistemato.

```
* 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.
```

Se per qualche motivo non si è citato il numero di bug in una voce precedente del changelog, non è un problema, basta chiudere normalmente il bug nel BTS. Non c'è bisogno di toccare il file changelog, presumendo che la descrizione della correzione sia già indicata (questo vale per le correzioni da parte degli autori/manutentori, non c'è bisogno di tenere traccia dei bug che hanno risolto secoli fa nel proprio 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.

6.3.4 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 Buone pratiche per gli script del mantainer

Gli script del maintainer includono i file `debian/postinst`, `debian/preinst`, `debian/prerm` e `debian/postrm`. Questi script si prendono cura di ogni configurazione di installazione o disinstallazione del pacchetto che non è gestito esclusivamente dalla creazione o dalla rimozione di file e cartelle. Le seguenti istruzioni completano la Debian Policy.

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 [ -x /usr/sbin/install-docs ]; then...
```

Se non si desidera codificare il percorso di un comando nello script del maintainer, la seguente funzione shell che soddisfa POSIX potrebbe essere d'aiuto:

```
pathfind() {
    OLDIFS="$IFS"
    IFS=:
    for p in $PATH; do
        if [ -x "$p/$*" ]; then
            IFS="$OLDIFS"
            return 0
        fi
    done
    IFS="$OLDIFS"
    return 1
}
```

Si può utilizzare questa funzione per cercare il `$PATH` di un nome di un comando, passato come argomento. Restituisce true (zero) se il comando è stato trovato e false in caso contrario. Questo è davvero il modo più portatile, dal momento che `command -v`, **type** e **which** non sono POSIX.

Mentre **which** è una alternativa accettabile, dal momento che fa parte del richiesto pacchetto `debianutils`, non è nella partizione di root. Ovvero, è in `/usr/bin`, piuttosto che `/bin`, quindi non può essere utilizzato in script che vengono eseguiti prima che la partizione `/usr` sia montata. La maggior parte degli script non avranno questo problema, però.

6.5 Gestione della configurazione con `debconf`

`Debconf` è un sistema di gestione della configurazione che può essere utilizzato da tutti i vari script per la pacchettizzazione (principalmente `postinst`) per richiedere indicazioni all'utente riguardo a come configurare il pacchetto. Interazioni dirette degli utenti ora devono essere evitate a favore dell'interazione con `debconf`. Ciò consentirà installazioni non interattive in futuro.

Debconf è un grande strumento, ma è spesso mal utilizzato. Molti errori comuni sono elencati nella pagina man di debconf-devel(7). È qualcosa che si deve leggere se si decide di usare debconf. Inoltre, indichiamo qui alcune buone pratiche.

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).

6.5.1 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.

Riservare le note d'uso a ciò cui appartengono: al file `NEWS.Debian`, o `README.Debian`. Le si utilizzi solamente per le note importanti che possono influenzare direttamente l'usabilità del pacchetto. Ricordare che le note bloccheranno sempre l'installazione fino a quando saranno confermate o abbiano disturbato l'utente tramite email.

Scegliere con cura le priorità delle domande negli script del maintainer. Si consulti debconf-devel(7) per i dettagli sulla priorità. La maggior parte delle domande dovrebbero usare le priorità media e bassa.

6.5.2 Raccomandazioni generali per autori e traduttori

6.5.2.1 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.

6.5.2.2 Sii gentile con i traduttori

I modelli debconf possono essere tradotti. Debconf, insieme al suo pacchetto fratello **po-debconf** offre un framework semplice per ottenere i modelli tradotti dai team di traduzione o anche dai singoli individui.

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).

Evitare di modificare i modelli troppo spesso. Cambiare i modelli di testo produce più lavoro per i traduttori che avranno la loro traduzione confusa. Una traduzione confusa è una frase per la quale l'originale sia stata cambiata da quando è stata tradotta, quindi per essere utilizzabile richiede alcuni aggiornamenti da parte di un traduttore. Quando i cambiamenti sono abbastanza piccoli, la traduzione originale è conservata nel file PO, ma contrassegnata come `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"
```

Questo comando innanzitutto sincronizzerà i files PO e POT in `debian/po` con i file dei modelli elencati in `debian/po/POTFILES.in`. Poi, invierà una richiesta di nuove traduzioni, nella mailing list debian-i18n@lists.debian.org. Infine, invierà una richiesta per aggiornare la traduzione sia al team per il supporto della lingua (menzionato nel campo `Language-Team` di ogni file PO), sia all'ultimo traduttore (menzionato nel campo `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.

6.5.2.3 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
```

6.5.2.4 Non fare ipotesi sulle interfacce

I modelli di testo non dovrebbero fare riferimento ai widget appartenenti ad alcune interfacce di debconf. Frasi come *Se rispondi SI...* non hanno alcun significato per gli utenti di interfacce grafiche che utilizzano checkbox per le domande booleane.

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.

6.5.2.5 Non utilizzare la prima persona

Si dovrebbe evitare l'uso della prima persona (*Farò questo...* o *Consigliamo...*). Il computer non è una persona ed i modelli debconf non parlano a nome degli sviluppatori Debian. Si dovrebbe usare la costruzione impersonale. Per coloro di voi che già hanno scritto pubblicazioni scientifiche, basta scrivere i vostri modelli come quando si scrive un articolo scientifico. Tuttavia, provare a utilizzare la voce attiva, se ancora possibile, come *Abilitate questo se...*, invece di *Questo può essere attivata se...* .

6.5.2.6 Usare il genere neutro

Il mondo è fatto di uomini e donne. Utilizzare le costruzioni di genere neutro nelle vostre scritture.

6.5.3 Definizione dei campi dei modelli

Questa parte fornisce alcune informazioni che sono per lo più prese dal manuale debconf-devel(7).

6.5.3.1 Tipo

6.5.3.1.1 stringa

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

6.5.3.1.2 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.

6.5.3.1.3 booleano

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

6.5.3.1.4 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.

I modelli debconf con sistema a flag offrono molte di queste possibilità. La pagina del manuale di po-debconf(7) elenca tutte queste possibilità.

6.5.3.1.5 multiselect

Come per la selezione del tipo di dato, ad eccezione di quando l'utente può scegliere un qualsiasi numero di elementi dall'elenco delle scelte (o scegliere nessuno di loro).

6.5.3.1.6 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.

6.5.3.1.7 testo

Questo tipo è ora considerato obsoleto: non usarlo.

6.5.3.1.8 errore

Questo tipo è progettato per gestire i messaggi di errore. È molto simile al tipo di nota. Le interfacce possono presentarlo in modo diverso (per esempio, la finestra di cdebconf disegna uno schermo rosso invece del solito blu).

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

6.5.3.2 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.

La descrizione breve dovrebbe essere in grado di stare in piedi da sola. Alcune interfacce non mostrano la descrizione lunga di default, oppure solo se l'utente esplicitamente la chiede o addirittura non la mostrano affatto. Evitare cose come «cosa vuoi fare?»

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.

Non si sia troppo prolissi. Gli utenti tendono a ignorare schermate troppo lunghe. 20 righe sono per esperienza un limite che non si dovrebbe oltrepassare, perché ciò significa che nella finestra di interfaccia classica, le persone avranno bisogno di scorrere e molte persone semplicemente non lo fanno.

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

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

6.5.3.3 Scelte

Questo campo dovrebbe essere utilizzato per la selezione singola e multipla dei tipi. Esso contiene le scelte possibili che verranno presentate agli

6.5.3.4 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.

6.5.4 Guida a specifici stili di modelli di campi

6.5.4.1 Type field

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

6.5.4.2 Il campo Description

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

6.5.4.2.1 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.

6.5.4.2.2 Modelli booleani

- La descrizione breve dovrebbe essere formulata in forma di una domanda che dovrebbe essere mantenuta breve e dovrebbe generalmente terminare con un traduzioni sono spesso più lunghe rispetto alle versioni originali).

- 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.

6.5.4.2.3 Selezione/Multiselezione

- La descrizione breve è un suggerimento e **non** un titolo. **Non** utilizzare inutili costrutti del tipo «Gentilmente scegliete...». Gli utenti sono abbastanza intelligenti da capire che devono scegliere qualcosa... :)

- 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).

6.5.4.2.4 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.
- **Non abusare di debconf.** Le note sono il modo più comune per abusare di debconf. Come scritto nella pagina di manuale di debconf-devel: è meglio usarle solo come avvertimento di problemi molto seri. I files `NEWS.Debian` o `README.Debian` sono il luogo appropriato per molte note. Se, leggendo questo manuale, si sta valutando di convertire i propri modelli di tipo Nota in voci di `NEWS.Debian` o `README.Debian`, si consideri anche di mantenere le traduzioni esistenti per il futuro.

6.5.4.3 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.

6.5.4.4 Campo Default

Se il valore di default, per un modello di selezione, può variare a seconda della lingua dell'utente (per esempio, se la scelta è una scelta della lingua), utilizzare il trucco «_Default».

Questo campo speciale permette ai traduttori di mettere la scelta più appropriata in base alla loro propria lingua. Diventerà la scelta di default quando sarà usato il loro linguaggio mentre la vostra «Scelta di Default» verrà usata quando si utilizza l'inglese.

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:
```

Si noti l'uso dei «cancelletti» che consentano i commenti interni nei campi di debconf. Da notare anche l'uso di commenti che appariranno nel file sui quali i traduttori lavoreranno.

I commenti sono necessari dal momento che il trucco _Default genera un po' di confusione: i traduttori potranno mettere la propria scelta

6.5.4.5 Campo Default

NON usare un campo predefinito vuoto. Se non si desidera utilizzare i valori di Default, non usarli.

Se si utilizza po-debconf (e si **dovrebbe**, si consulti Sezione 6.5.2.2), si consideri di marcare questo campo come traducibile, se si pensa che possa essere tradotto.

Se il valore predefinito può variare a seconda della lingua/paese (ad esempio, il valore predefinito per una scelta della lingua), è possibile utilizzare il tipo speciale _Default documentato in po-debconf(7).

6.6 Internazionalizzazione

Questa sezione contiene le informazioni a livello generale per gli sviluppatori al fine di rendere più facile la vita dei traduttori. Maggiori informazioni per i traduttori e gli sviluppatori interessati alla internazionalizzazione sono disponibili nella documentazione Internazionalizzazione e localizzazione in Debian.

6.6.1 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.

L'obbiettivo di `debconf` era quello di rendere più facile la configurazione di pacchetti per i maintainer e per gli utenti. In origine, la traduzione di modelli debconf è stata gestita con **debconf-mergetemplate**. Tuttavia, tale tecnica è ormai deprecata, il modo migliore per realizzare una internazionalizzazione `debconf` è quello di utilizzare il pacchetto `po-debconf`. Questo metodo è più facile sia per il maintainer e sia per i traduttori; sono forniti script di transizione.

Utilizzando `po-debconf`, la traduzione è memorizzata in file `.po` (prese dalle tecniche di traduzione **gettext**). Speciali file di modello contengono i messaggi originali e marcano quali campi sono traducibili. Quando si modifica il valore di un campo traducibile, chiamando **debconf-updatepo**, la traduzione è contrassegnata come bisognosa di attenzione da parte dei traduttori. Poi, in fase di compilazione, il programma **dh_installdebconf** si prende cura di tutta la magia necessaria per aggiungere il modello con le traduzioni aggiornate nei pacchetti binari. Fare riferimento alla pagina del manuale di po-debconf(7) per i dettagli.

6.6.2 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.

Se si mantiene una documentazione di qualsiasi dimensione, è più facile per i traduttori se hanno accesso ad un sistema di controllo del codice sorgente. Ciò consente ai traduttori di vedere le differenze tra due versioni della documentazione, così, per esempio, si può vedere ciò che deve essere ritradotto. Si raccomanda che la documentazione tradotta mantenga una nota circa la versione del codice sulla quale è basata. Un sistema interessante è fornito dal doc-check nel pacchetto `debian-installer`, che mostra una panoramica dello stato di traduzione per ogni lingua, utilizzando i commenti strutturati per la revisione in corso del file da tradurre e, per un file tradotto, la revisione del file originale della traduzione sulla quale si basa. Si potrebbe desiderare di adattarla e di fornirla nel proprio VCS.

Se si mantiene la documentazione XML o SGML, vi consigliamo di isolare qualsiasi informazione indipendente dal linguaggio e definirla come entità in un file separato che è incluso da tutte le diverse traduzioni. Ciò rende molto più facile, per esempio, mantenere gli URL aggiornati in più file.

Alcuni strumenti (ad es `po4a`, `poxml`, o il `translate-toolkit`) sono specializzati nell'estrazione del materiale traducibile da diversi formati. Producono i file PO, un formato abbastanza comune ai traduttori, che permette di vedere ciò che deve essere ritradotto quando il documento tradotto viene aggiornato.

6.7 Situazioni comuni durante la pacchettizzazione

6.7.1 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.

6.7.2 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.

Delle buone pratiche per la pacchettizzazione delle librerie sono state raggruppate in Guida alla pacchettizzazione delle librerie.

6.7.3 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.

Le pagine man non necessitano di essere scritte direttamente in formato troff. I formati file più diffusi sono DocBook, POD e di reST, che possono essere convertiti usando **xsltproc**, **pod2man** e **rst2man** rispettivamente. In misura minore, il programma **help2man** può anche essere utilizzato per scrivere uno stub.

6.7.4 Specifici tipi di pacchetti

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

- I relativi pacchetti Perl hanno una Perl policy, alcuni esempi di pacchetti che seguono tale policy sono `libdbd-pg-perl` (modulo binario perl) o `libmldbm-perl` (modulo perl indipendente dall'architettura).
- I relativi pacchetti Python hanno la loro policy python; si consulti `/usr/share/doc/python/python-policy.txt.gz` nel pacchetto `python`.
- I relativi pacchetti Emacs hanno la emacs policy.
- I relativi pacchetti Java hanno la loro java policy.
- I relativi pacchetti Ocaml hanno la loro politica, che si trova in `/usr/share/doc/ocaml/ocaml_packaging_policy.gz` del pacchetto `ocaml`. Un buon esempio è il pacchetto dei sorgenti di `camlzip`.
- I pacchetti che forniscono XML o SGML DTD devono essere conformi alle indicazioni fornite nel pacchetto `sgml-base-doc`.
- I pacchetti Lisp si devono registrare con il `common-lisp-controller`, a proposito del quale si veda `/usr/share/doc/common-lisp-controller/README.packaging`.

6.7.5 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.

Tuttavia, se la dimensione dei dati è notevole, si consideri di dividere in un pacchetto separato, indipendente dall'architettura (`_all.deb`). In questo modo, si evitano inutili duplicazioni degli stessi dati in undici o più .debs, uno per ogni architettura. Mentre questo aggiunge un po' di overhead ai file dei `Pacchetti`, fa risparmiare molto spazio su disco sui mirror Debian. Separare i dati indipendenti dall'architettura riduce anche il tempo di elaborazione di **lintian** (si consulti Sezione A.2) quando lanciato sull'intero archivio Debian.

6.7.6 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
```

6.7.7 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.

Ad esempio, con `--guess-dummy`, **deborphan** prova a cercare tutti i pacchetti di transizione che sono stati necessari per l'aggiornamento, ma che ora possono tranquillamente essere rimossi. Per questo, esso cerca la stringa dummy or transitional nella loro descrizione breve.

Quindi, quando si crea un pacchetto di questo tipo, ci si assicuri di aggiungere il testo alla descrizione breve. Se si è alla ricerca di esempi, basta eseguire:**apt-cache search.| grep dummy** o **apt-cache search | grep transitional**.

Inoltre, è raccomandato modificare la sua sezione in `oldlibs` e la sua priorità in `extra` in modo da cancellare il compito di **deborphan**.

6.7.8 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.

6.7.8.1 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.

Non ci sono linee guida universalmente accettate che gli autori originali seguono per quanto riguarda la struttura delle cartelle all'interno del loro tarball, ma **dpkg-source** è tuttavia in grado di affrontare la maggior parte delle tarball originali come sorgente puro. La sua strategia è equivalente alla seguente:

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

   ```
   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)`.

[1] Non possiamo impedire agli autori originali di cambiare il tarball che distribuiscono senza anche incrementare il numero di versione, quindi non ci può essere alcuna garanzia che in qualsiasi momento un tarball puro sia identico a quello di upstream *attualmente* in distribuzione. Tutto ciò che ci si può aspettare è che sia identico a qualcosa che l'autore originale una volta *ha* distribuito. Se una differenza dovesse emergere in seguito (ad esempio, se l'upstream si accorgesse che non stava usando la massima compressione nella distribuzione originale e dunque di comprimerlo nuovamente con **gzip**), semplicemente non è una buona cosa. Poiché non vi è alcun buon modo per caricare un nuovo `.orig.tar.{gz,bz2,xz}` per la stessa versione, non c'è nemmeno alcuna indicazione se trattare questa situazione come un bug.

6.7.8.2 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. **dovrebbe** essere documentata nel pacchetto sorgente risultante. Informazioni dettagliate su come è stato ottenuto il sorgente ripacchettizzato e su come questo può essere riprodotto dovrebbero essere fornite in `debian/copyright`. È anche una buona idea fornire un `get-orig-source` target nel proprio `debian/rules`, che ripete il processo, come descritto nel Policy Manual, Script principale per la compilazione: `debian/rules`.

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

3. **dovrebbe**, salvo che per motivi legali, preservare l'intera infrastruttura per la compilazione e per la portabilità fornita dall'autore originale. Ad esempio, non è una ragione sufficiente per omettere un file che viene utilizzato solo quando si compila su MS-DOS. Allo stesso modo, un `Makefile` fornito dall'autore originale non dovrebbe essere omesso anche se la prima cosa che il proprio `debian/rules` fa è sovrascriverlo eseguendo uno script di configurazione.

   (*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. **dovrebbe** usare *packagename-upstream-version*`.orig` come nome per la più alta cartella nel suo tarball. Ciò rende possibile distinguere tarball puri da quelli ripacchettizzati.

5. **dovrebbe** essere compresso con gzip o bzip con la massima compressione.

6.7.8.3 Modifica dei file binari

A volte è necessario cambiare file binari contenuti nel tarball originale, o per aggiungere file binari che non vi sono contenuti. Questo è completamente supportato quando si usano pacchetti sorgente in formato «3.0 (quilt)», si consulti la pagina di manuale dpkg-source(1) per i dettagli. Quando si utilizza il vecchio formato «1.0», i file binari non possono essere memorizzati nel `.diff.gz` quindi è necessario memorizzare un **uuencode**d (o simile) versione del file e decodificarlo in fase di compilazione in `debian/rules` (e spostarlo nella sua posizione ufficiale).

6.7.9 Buone pratiche per i pacchetti di debug

Un pacchetto di debug è un pacchetto con un nome che termina in -dbg, che contiene ulteriori informazioni che **gdb** può utilizzare. Poiché i binari di Debian vengono separati di default, le informazioni di debug, compresi i nomi delle funzioni e numeri di riga, non sono disponibili quando si esegue **gdb** su binari Debian. I pacchetti di debug consentono di installarli agli utenti che hanno bisogno di queste informazioni aggiuntive, senza appesantire un regolare sistema con queste informazioni.

[2] Come eccezione, se l'omissione di file non liberi porterebbe il sorgente a fallire la compilazione senza assistenza da parte del Debian diff, potrebbe essere opportuno modificare invece i file, omettendo solo le loro parti non-free, o spiegare la situazione in un `README.source` nella radice dell'albero del sorgente. Ma anche in questo caso sollecitare l'autore originale affinché renda i componenti non liberi facilmente separabili dal resto del sorgente.

Spetta al maintainer di un pacchetto se creare un pacchetto di debug o meno. I maintainer sono incoraggiati a creare i pacchetti di debug per i pacchetti di libreria, dal momento che questi possono aiutare nel debug di molti programmi legati ad una libreria. In generale, i pacchetti di debug non devono essere aggiunti per tutti i programmi, facendo così appesantire l'archivio. Ma se un maintainer ritiene che gli utenti spesso hanno bisogno di una versione di debug di un programma, può essere utile fare un pacchetto di debug. I programmi che fanno parte dell'infrastruttura di base, come ad esempio Apache e il server X sono anche dei buoni candidati per i pacchetti di debug.

Alcuni pacchetti di debug possono contenere un completo e particolare versione di debug compilata di una libreria o di altro binario, ma la maggior parte di loro può risparmiare spazio e tempo di compilazione contenendo invece simboli di debug separati che **gdb** è in grado di trovare e caricare al volo quando si effettua il debug di un programma o di una libreria. La convenzione in Debian è quella di mantenere questi simboli in `/usr/lib/debug/`*`path`*, dove *`path`* è il percorso al file eseguibile o alla libreria. Ad esempio, i simboli di debug per `/usr/bin/foo` vanno in `/usr/lib/debug/usr/bin/foo` e simboli di debug per `/usr/lib/libfoo.so.1` vanno in `/usr/lib/debug/usr/lib/libfoo.so.1`.

I simboli di debug possono essere estratti da un file oggetto usando **objcopy - only-keep-debug**. Successivamente il file oggetto può essere spogliato e **objcopy - add-gnu-debuglink** è utilizzato per specificare il percorso del file di simboli di debug. objcopy(1) spiega nel dettaglio come funziona questo processo.

Il comando **dh_strip** in `debhelper` supporta la creazione di pacchetti di debug e può prendersi cura per voi di utilizzare **objcopy** per separare i simboli di debug. Se il pacchetto utilizza `debhelper`, tutto quello che dovete fare è chiamare **dh_strip - DBG-package = libfoo-dbg**, ed aggiungere una voce al `debian/control` per il pacchetto di debug.

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})
```

6.7.10 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`).

La descrizione lunga del metapacchetto deve documentare chiaramente il suo scopo in modo che l'utente sappia che cosa si perderà se rimuovono il pacchetto. È raccomandato essere chiari sulle conseguenze. Ciò è particolarmente importante per i metapacchetti che vengono installati durante l'installazione iniziale e che non sono stati installati esplicitamente dall'utente. Questi tendono ad essere importanti per garantire regolari aggiornamenti del sistema e l'utente dovrebbe essere disincentivato dal disinstallarli in modo da evitare potenziali rotture.

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.

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...
```

Si noti che i tag sono separati da spazi e non possono contenere caratteri di sottolineatura. Se si sta caricando bug per un particolare gruppo o team è consigliato che si imposti il `User` ad una mailing list appropriata dopo aver descritto li i propri intenti.

Per visualizzare bug etichettati con uno specifico usertag, visitare la pagina `http://bugs.debian.org/cgi-bin/pkgrepor`

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 Sezione A.2.1). Se non si riesce a farlo, allora si dovrebbe prendere in considerazione di rendere orfani alcuni dei vostri pacchetti (si consulti Sezione 5.9.4). 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 Sezione 5.12).

7.2.2 Bug squashing parties

Di volta in volta il gruppo QA organizza feste di bug squashing in modo da sbarazzarsi di quanti più problemi possibili. Sono annunciati su debian-devel-announce@lists.debian.org e l'annuncio spiega quale area sarà al centro della festa: di solito si concentrano sui bug critici per il rilascio, ma può accadere che decidano di aiutare a terminare un importante aggiornamento (come una nuova versione **perl** che richiede la ricompilazione di tutti i moduli binari).

Le regole per il caricamento di non-maintainer cambiano durante le feste perché l'annuncio della festa è considerata prioritaria per NMU. Se si dispone di pacchetti che possono essere influenzati dalla festa (perché hanno rilasciato bug critici per esempio), è necessario inviare un aggiornamento per ciascun bug corrispondente per spiegare il loro stato attuale e cosa ci si apetti dalla festa. Se non si vuole un NMU, o se si è solo interessati ad una patch, o se si vuole fare da soli con il bug, spiegarlo nel BTS.

Le persone che partecipano alla festa hanno regole speciali per un NMU, possono fare un NMU senza preavviso se caricano i loro NMU su DELAYED/3-giorni almeno. Tutte le altre regole NMU si applicano come di solito; dovrebbero inviare la patch del NMU al BTS (a uno dei bug aperti fissati dal NMU, o di un nuovo bug, etichettato fisso). Dovrebbero anche rispettare particolari desideri del 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 Sezione 4.10. È possibile farlo utilizzando l'indirizzo email *package*`@packages.qa.debian.org`.

7.4 Rapportarsi con maintainer non attivi e/o non raggiungibili

Se si nota che un pacchetto è carente di manutenzione, è necessario assicurarsi che i maintainer siano attivi e che continueranno a lavorare sui loro pacchetti. È possibile che essi non siano più attivi, ma non si sono deregistrati dal sistema, per così dire. D'altra parte, è anche possibile che abbiano solo bisogno di un promemoria.

C'è un sistema semplice (il database MIA) in cui vengono registrate informazioni inerenti i maintainer etichettati come Missing In Action. Quando un membro del gruppo QA contatta un maintainer inattivo o trova ulteriori informazioni su qualcuno, questo viene registrato nel database di MIA. Questo sistema è disponibile in `/org/qa.debian.org/MIA` sull'host `qa.debian.org` e può essere interrogato con lo strumento **mia-query**. Utilizzare **mia-query - help** per vedere come interrogare il database. Se si scopre che ancora alcuna informazione è stata registrata su un maintainer inattivo, o che è possibile aggiungere ulteriori informazioni, si

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.

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!

D'altra parte, anche se si è volontari, si ha una responsabilità. Così si può sottolineare l'importanza di un bene più grande: se un maintainer non ha più il tempo o interesse, dovrebbe lasciar andare e dare il pacchetto a qualcuno con più tempo.

Se si è interessati a lavorare nel team MIA, si dia un'occhiata al file `README` in `/org/qa.debian.org/mia` su `qa.debian.org` dove i dettagli tecnici e le procedure di MIA sono documentate e contatti mia@qa.debian.org.

7.5 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.

7.5.1 Sponsorizzare pacchetti

Sponsorizzare un pacchetto significa caricare un pacchetto per un maintainer, che non è in grado di farlo da solo. Non è una cosa da poco, lo sponsor deve verificare il package ed assicurarsi che esso soddisfi l'elevato livello di qualità che Debian si sforza di avere.

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

Il processo di sponsorizzazione di un pacchetto è:

1. Il maintainer prepara un pacchetto sorgente (`.dsc`) e lo mette in linea da qualche parte (come su mentors.debian.net) o, meglio ancora, fornisce un collegamento a un repository pubblico di VCS (si consulti Sezione 4.4.5) in cui il pacchetto viene mantenuto.

2. Lo sponsor scarica (o fa il checkout) del sorgente del pacchetto.

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.

Prima di addentrarci nei dettagli su come sponsorizzare un pacchetto, ci si dovrebbe chiedere se l'aggiunta del pacchetto proposto è vantaggiosa per Debian.

Non è semplice rispondere a questa domanda, può dipendere da molti fattori: il codice originale è maturo e privo di falle di sicurezza? Ci sono pacchetti pre-esistenti che possono fare lo stesso compito e come si comparano con questo nuovo pacchetto? Il nuovo pacchetto è stato richiesto dagli utenti e da quanti? Quanto sono attivi gli sviluppatori originali?

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?

È anche una buona idea conoscere la loro posizione rispetto a Debian: sono d'accordo con la filosofia di Debian e hanno intenzione di aderire a Debian? Considerato quanto sia facile diventare un Maintainer Debian, si potrebbe voler sponsorizzare solo le persone che hanno intenzione di aderire. In questo modo fin dall'inizio si è consapevoli che non si dovrà agire in qualità di sponsor a tempo indeterminato.

7.5.1.1 Sponsorizzare un nuovo pacchetto

I nuovi maintainer di solito hanno alcune difficoltà nel creare pacchetti Debian - questo è abbastanza comprensibile. Faranno errori. Ecco perché sponsorizzare un nuovo pacchetto in Debian richiede una profonda conoscenza della pacchettizzazione in Debian. A volte saranno necessarie diverse iterazioni fino a quando il pacchetto sarà abbastanza buono per essere caricato in Debian. Quindi essere uno sponsor implica essere un mentore.

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]

[1] Si possono trovare altri controlli nella wiki nella quale diversi sviluppatori condividono le proprie liste di sponsorizzazione.

- 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).
- Eseguire **lintian** (si consulti Sezione A.2.1). Troverà molti problemi comuni. Assicurarsi di verificare che qualsiasi **lintian** che sovrascriva la configurazione fatta dal maintainer sia pienamente giustificata.
- Eseguire **licensecheck** (parte di Sezione A.6.1) e verificare che `debian/copyright` sia corretto e completo. Cercare i problemi di licenza (come i file con intestazioni del tipo "All rights reserved", o con una licenza non compatibile con DFSG). **grep -ri** è un amico per questo compito.
- Compilare il pacchetto con **pbuilder** (o qualsiasi strumento simile, vedi Sezione A.4.3) per garantire che le dipendenze di compilazione siano soddisfatte.
- Correggere `debian/control`: segue le buone pratiche (si consulti Sezione 6.2)? Sono soddisfatte le dipendenze?
- Correggere `debian/rules`: rispecchia le buone pratiche (si consulti Sezione 6.1)? 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 Capitolo 6)?
- Compilare i pacchetti, installarli e provare il software. Assicurarsi di poter rimuovere ed eliminare i pacchetti. Forse provarli con **piuparts**.

Se il controllo non ha evidenziato alcun problema, si può compilare il pacchetto e caricarlo su Debian. Ricordare che, anche se non si è il maintainer, in qualità di sponsor si è ancora responsabile per ciò che si carica in Debian. Ecco perché si è incoraggiati a tenere il passo con il pacchetto attraverso il Sezione 4.10.

Si noti che non dovrebbe essere necessario modificare il pacchetto sorgente per inserire il proprio nome nella `changelog` o nel file di `control`. Il campo `Maintainer` del file `control` e del file `changelog` dovrebbe elencare la persona che ha fatto la pacchettizzazione, ossia la persona sponsorizzata. In questo modo riceveranno tutta la posta BTS.

Invece di dovrebbe istruire **dpkg-buildpackage** ad usare la propria chiave per la firma. Lo si fa con l'opzione `-k`:

```
dpkg-buildpackage -kKEY-ID
```

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

```
DEBSIGN_KEYID=KEY-ID
```

7.5.1.2 Sponsorizzare un aggiornamento di un pacchetto esistente

Normalmente si presupporrà che il pacchetto sia già passato attraverso una revisione completa. Così, invece di farlo di nuovo, si analizzerà attentamente la differenza tra la versione attuale e la nuova versione preparata dal maintainer. Se non si è fatta da soli la revisione iniziale, si potrebbe anche voler dare uno sguardo più profondo nel caso in cui il revisore iniziale fosse stato sciatto.

Per essere in grado di analizzare la differenza è necessario avere entrambe le versioni. Scaricare la versione attuale del pacchetto sorgente (con **apt-get source**) e ricompilarlo (o scaricare gli attuali pacchetti binari con **aptitude download**). Scaricare il pacchetto sorgente da sponsorizzare (di solito con **dget**).

Leggere la nuova voce del changelog, dovrebbe dire cosa aspettarsi durante la revisione. Il principale strumento che si utilizzerà è **debdiff** (fornito con il pacchetto `devscripts`), lo si può eseguire con due pacchetti sorgente (i file `.dsc`), o due file `.changes` (allora confronterà tutti i pacchetti binari elencati nel file `.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.

Se tutto va bene, compilare il pacchetto e confrontare i pacchetti binari per verificare che le modifiche sul pacchetto sorgente non abbiano conseguenze inattese (come alcuni file cancellati per errore, le dipendenze non soddisfatte, etc).

Si potrebbe controllare il Package Tracking System (si consulti Sezione 4.10) per verificare se il maintainer non abbia perso qualcosa di importante. Magari ci sono aggiornamenti di traduzioni che stanziano nel BTS e che sarebbero potute essere integrate. Forse il pacchetto è stato NMUed e il maintainer ha dimenticato di integrare le modifiche nel loro pacchetto dal NMU. Forse c'è un rilascio per un bug critico che hanno lasciato non gestito e che sta bloccando la migrazione in `testing`. Se si trova qualcosa che avrebbero potuto fare (meglio), è il momento di dirglielo in modo che per la prossima volta possano migliorare e in modo che abbiano una migliore comprensione delle loro responsabilità.

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

7.5.2 Promuovere nuovi sviluppatori

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

7.5.3 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.

Secondo l'Introduzione a i18n di Tomohiro KUBOTA, per i18n (internazionalizzazione) si intende la modifica di un software o di tecnologie correlate in modo che sia potenzialmente in grado di gestire più lingue, costumi e così via in tutto il mondo, mentre L10N (localizzazione) si intende l'attuazione di una determinata lingua per un software già internazionalizzato.

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.

Per messaggi di programma, l'infrastruttura gettext viene usata il più delle volte. La maggior parte del tempo, la traduzione viene gestita originariamente all'interno di progetti come il Free Translation Project, il Gnome translation Project o quello KDE. L'unica risorsa centralizzata di Debian è il Central Debian translation statistics, dove si possono trovare alcune statistiche riguardanti i file di traduzione presenti nei pacchetti attuali, ma nessuna infrastruttura reale per facilitare il processo di traduzione.

Uno sforzo per tradurre le descrizioni dei pacchetti è iniziato molto tempo fa, anche se molto poco sostegno è offerto dagli strumenti che effettivamente li utilizzano (cioè, solo APT possono usare, se configurato correttamente). I maintainer non hanno bisogno di fare nulla di speciale per supportare le descrizioni dei pacchetti tradotti; i traduttori devono usare il Debian Description Translation Project (DDTP).

Per i modelli `debconf`, i maintainer dovrebbero utilizzare il pacchetto `po-debconf` per facilitare il lavoro dei traduttori, che potrebbero usare il DDTP per fare il loro lavoro (ma i team francesi e brasiliani non lo usano). Alcune statistiche possono essere trovate sia sul sito del DDTP (su ciò che è stato effettivamente tradotto) e sul sito Central Debian translation statistics (su ciò che è integrato nei pacchetti).

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.

Per la documentazione di un pacchetto specifico (pagine man, documenti info, altri formati), quasi tutto resta da fare.

Più in particolare, il progetto KDE gestisce la traduzione della sua documentazione nello stesso modo dei suoi messaggi del programma.

C'è uno sforzo per gestire le pagine di manuale specifici di Debian all'interno di uno specifico repository VCS.

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

Per tradurre le descrizioni dei pacchetti o di modelli `debconf`, non dovete fare niente, l'infrastruttura DDTP invierà il materiale da tradurre ai volontari senza necessità di interazione da parte vostra.

Per tutti gli altri materiali (file gettext, pagine man, o altra documentazione), la soluzione migliore è quella di mettere il testo da qualche parte su Internet e chiedere su debian-i18n una traduzione in diverse lingue. Alcuni membri del team di traduzione sono iscritti a questa lista e si prenderanno cura della traduzione e del processo di revisione. Una volta finito, si avrà il proprio documento tradotto da loro nella vostra casella di posta.

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.

La soluzione migliore è quella di aprire un bug normale contenente la traduzione per il pacchetto. Assicurarsi di utilizzare il tag «PATCH» e non usare una severità maggiore di «wishlist», dal momento che la mancanza di traduzione non ha impedito al programma di funzionare.

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**.

Appendice A

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.

Alcune persone preferiscono utilizzare strumenti gli di manutenzione dei pacchetti ad alto livello e altri no. Debian è ufficialmente agnostica su questo problema, qualsiasi strumento che ottiene il lavoro fatto va bene. Pertanto, questa sezione non intende indicare a nessuno quali strumenti deve adottare o come devono condurre le loro funzioni di mantenimento del pacchetto. Né è destinata a promuovere qualsiasi particolare strumento al fine di escludere uno strumento in competizione.

La maggior parte delle descrizioni di questi pacchetti vengono dalle descrizioni effettive dei pacchetti stessi. Ulteriori informazioni possono essere trovate nella documentazione del pacchetto stesso. È anche possibile visualizzare ulteriori informazioni con il comando **apt-cache show *`nome-pacchetto`***.

A.1 Strumeti di base

I seguenti strumenti sono praticamente obbligatori per qualsiasi maintainer.

A.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.

A.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`.

Molti ritengono che questo sistema dovrebbe essere utilizzato per tutti i pacchetti che richiedono una configurazione interattiva, si consulti Sezione 6.5. `debconf` non è attualmente richiesto dalla policy di Debian, ma ciò potrebbe cambiare in futuro.

A.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`.

A.2 Strumenti per la pulizia di pacchetti

Secondo il Dizionario On-line di Informatica (FOLDOC), «lint» è un processore di linguaggio Unix C che effettua verifiche più approfondite sul codice rispetto ai normali compilatori C. Gli strumenti di controllo dei pacchetti aiutano i maintainer trovando nei loro pacchetti automaticamente i problemi più comuni e le violazioni delle policy.

A.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 Sezione 5.3 per maggiori informazioni su come e quando utilizzare Lintian.

È anche possibile visualizzare un riepilogo di tutti i problemi segnalati da Lintian sui vostri pacchetti su `https://lintian.debian.org/`. Questi rapporti contengono gli ultimi output di **lintian** per l'intera distribuzione di sviluppo (`unstable`).

A.2.2 debdiff

debdiff (dal pacchetto `devscripts`, Sezione A.6.1) 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 pacchetto_1-1_arch.deb package_2-1_arch.deb
```

O anche su un paio di file di modifiche:

```
debdiff pacchetto_1-1_arch.changes package_2-1_arch.changes
```

Per ulteriori informazioni consultare debdiff(1).

A.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 Sezione 6.1.1.

A.3.1 `debhelper`

`debhelper` è una raccolta di programmi che possono essere utilizzati in `debian/rules` per automatizzare compiti comuni correlati con la compilazione di pacchetti Debian binari. `debhelper` include programmi per installare vari file all'interno del pacchetto, comprimere i file, correggere i permessi dei file e integrare il pacchetto nel sistema dei menù Debian.

A differenza di altri approcci, `debhelper` è suddiviso in diversi piccoli, semplici comandi che agiscono in modo coerente. Come tale, esso permette un controllo più capillare di alcuni degli altri strumenti per debian/rules.

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-`.

A.3.2 `dh-make`

Il pacchetto `dh-make` contiene **dh_make**, un programma che crea uno scheletro di file necessari per compilare un pacchetto Debian a partire da un albero sorgente. Come suggerisce il nome, **dh_make** è una riscrittura di `debmake` ed i relativi file di template usano i programmi **dh_*** da `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.

A.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.

A.4 Strumenti di compilazione

I seguenti pacchetti aiutano con il processo di compilazione del pacchetto, pilotando dpkg-buildpackage così come gestendo i compiti di supporto.

A.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.

A.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.

A.4.3 pbuilder

`pbuilder` costruisce un sistema chroot, e compila un pacchetto all'interno del chroot. È molto utile per controllare che dipendenze di compilazione di un pacchetto siano corrette e per essere certi che dipendenze non necessarie o sbagliate non esistano nel pacchetto risultante.

Un pacchetto correlato è `pbuilder-uml`, che va anche oltre, facendo la compilazione all'interno di un ambiente User Mode Linux.

A.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 Sezione 5.10.3.3 per maggiori informazioni, e `https://buildd.debian.org/` per vedere il sistema in azione.

A.5 Strumenti per caricare i pacchetti

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

A.5.1 dupload

`dupload` è un pacchetto e uno script per caricare automaticamente i pacchetti Debian nell'archivio Debian, per registrare il caricamento, e per inviare email sul caricamento di un pacchetto. È possibile configurarlo per nuovi repository o metodi per il caricamento.

A.5.2 dput

Il pacchetto e lo script `dput` fanno la stessa cosa di `dupload`, ma in un modo diverso. Ha alcune caratteristiche in più rispetto a `dupload`, come ad esempio la possibilità di verificare la firma GnuPG e i codici di controllo prima di caricare, e la possibilità di lanciare **dinstall** in modalità test dopo il caricamento.

A.5.3 dcut

Lo script **dcut** (parte del pacchetto `dput`, Sezione A.5.2) aiuta a rimuovere i file dalla cartella ftp di caricamento.

A.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`.

A.6.1 devscripts

`devscripts` è un pacchetto contenente wrapper e strumenti che sono molto utili per mantenere i pacchetti Debian. Script di esempio includono **debchange** e **dch**, che manipolano il file `debian/changelog` da riga di comando e **debuild**, che è un wrapper per **dpkg-buildpackage**. L'utilità **bts** è anche molto utile per aggiornare lo stato delle segnalazioni di bug da riga di comando. **uscan** può essere utilizzato per vedere nuove versioni originali fornite dei propri pacchetti. **debrsign** può essere utilizzata per firmare in remoto un pacchetto prima di caricarlo, che è bello (utile) quando la macchina con cui si genera il pacchetto è diversa da dove sono le chiavi GPG.

Si consulti la pagina di manuale devscripts(1) per un elenco completo degli script disponibili.

A.6.2 autotools-dev

`autotools-dev` contiene le buone pratiche per le persone che mantengono i pacchetti e che usano **autoconf** o **automake**. Contiene anche i file canonici `config.sub` e `config.guess` che sono conosciuti per funzionare su tutti i port di Debian.

A.6.3 dpkg-repack

dpkg-repack crea il pacchetto Debian da un pacchetto che è già stato installato. Se sono state apportate modifiche al pacchetto mentre era spacchettato (ad esempio, i file in `/etc` sono stati modificati), il nuovo pacchetto erediterà le modifiche.

Questa utilità può rendere più facile copiare i pacchetti da un computer ad un altro, o ricreare pacchetti che sono installati sul proprio sistema ma non più disponibili, o salvare lo stato attuale di un pacchetto prima di un aggiornamento.

A.6.4 alien

alien converte i pacchetti binari tra vari formati, tra cui Debian, RPM (RedHat), LSB (Linux Standard Base), Solaris e Slackware.

A.6.5 dpkg-dev-el

`dpkg-dev-el` è un pacchetto Emacs che fornisce assistenza durante la modifica di alcuni file nella cartella `debian` del proprio pacchetto. Per esempio, ci sono utili funzioni per l'elencazione dei bug attuali di un pacchetto e per finalizzare l'ultima voce in un file `debian/changelog`.

A.6.6 dpkg-depcheck

dpkg-depcheck (dal pacchetto `devscripts`, Sezione A.6.1) 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).

A.7 Strumenti per i port

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

A.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-shlibdeps**sono state migliorate per supportare la cross-compilazione.

A.8 Documentazione ed informazioni

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

A.8.1 `docbook-xml`

`docbook-xml` fornisce i DTD Docbook XML, che sono comunemente usati per la documentazione Debian (come il vecchio DTD debiandoc SGML). Questo manuale, per esempio, è stato scritto in DocBook XML.

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-*`.

Per produrre un PDF da FO, è necessario un processore FO, come `xmlroff` o `fop`. Un altro strumento per generare PDF da DocBook XML è `dblatex`.

A.8.2 `debiandoc-sgml`

`debiandoc-sgml` fornisce il DTD Debiandoc SGML, che è comunemente usato per la documentazione di Debian, ma ora è deprecato (al suo posto dovrebbe essere usato `docbook-xml`). Fornisce anche gli script per applicare stili e creare dai sorgenti vari formati di output.

La documentazione per la DTD può essere trovata nel pacchetto `debiandoc-sgml-doc`.

A.8.3 `debian-keyring`

Contiene le chiavi pubbliche GPG e PGP degli sviluppatori Debian. Si consulti Sezione 3.2.2 e la documentazione del pacchetto per ulteriori informazioni.

A.8.4 `debview`

`debview` fornisce una modalità Emacs per la visualizzazione dei pacchetti binari Debian. Questo consente di esaminare un pacchetto senza estrarlo.