Packaging System (Sistema Gestione dei Pacchetti): Strumenti e Principi Fondamentali

Capitolo 5

5. Packaging System (Sistema Gestione dei Pacchetti): Strumenti e Principi Fondamentali pag. 71
    1. Struttura di un Pacchetto Binario pag. 72
    2. La Meta-Informazione di un Pacchetto pag. 74
        1. Descrizione: il control File pag. 74
            1. Dipendenze: il Depends Field (il campo "Dipende da ...") pag. 75
            2. Conflitti: il Conflicts Field (il campo "Entra in conflitto con ...") pag. 77
            3. Incompatibilità: il Breaks Field (il campo "Interrompe ...") pag. 77
            4. Provided Items [Gli Oggetti Forniti]: il Provides Field (il campo "Fornisce ...") pag. 77
            5. Sostituzione dei Files: il Replaces Field (il campo "Sostituisce ...") pag. 80
        2. Scripts di Configurazione pag. 80
            1. Installazione e Aggiornamento pag. 81
            2. Rimozione di un Pacchetto pag. 81
        3. Checksum, Elenco dei Files di Configurazione pag. 83
    3. Struttura di un Pacchetto Sorgente pag. 84
        1. Struttura pag. 84
        2. Utilizzo in Debian pag. 87
    4. Manipolazione dei Pacchetti con dpkg pag. 87
        1. Installazione dei Pacchetti pag. 88
        2. Rimozione di un Pacchetto pag. 89
        3. Querying (Richieste) per il Database di dpkg ed ispezione dei Files .deb pag. 90
        4. Il Log File di dpkg pag. 94
        5. Supporto Multi-Arch pag. 94
            1. Abilitare Multi-Arch pag. 94
            2. I cambiamenti dovuti a Multi-Arch pag. 95
    5. Coesistenza con altri Packaging Systems pag. 96

<<Nelle vesti di Amministratore di Sistema, gestirete abitudinariamente i pacchetti .deb, in quanto destinati a relative unità funzionali (applicazioni, documentazione, ecc.), delle quali vi faciliteranno l'installazione e la manutenzione. È bene pertanto conoscere cosa sono e come usufruirne.>>

Questo capitolo descrive la struttura ed i contenuti dei pacchetti binari e dei pacchetti sorgente. I pacchetti binari sono files .deb, che si possono usare direttamente con dpkg, mentre i pacchetti sorgente includono il codice sorgente, ossia le istruzioni necessarie per montare i pacchetti binari.

5.1 Struttura di un Pacchetto Binario

Il formato di un pacchetto Debian è concepito in modo che i suoi contenuti possano essere estratti in qualsiasi sistema Unix, che abbia i classici comandi: ar, tar e gzip (talvolta xz o bzip2). Questa qualità, sebbene possa sembrare futile, garantisce la portabilità ed il disaster recovery.

[Il "porting", in informatica è il processo di adattamento, del software e/o dell'hardware, eseguito allo scopo di renderli compatibili a destinazioni (usi e/o ambienti) differenti rispetto a quelli originari. Di conseguenza in questo caso la "portabilità" di un pacchetto indica "quanto e se" possa essere compatibile ad usi o ad ambienti differenti rispetto a quelli nativi.
Il "Disaster Recovery", in informatica ed in particolare nell'ambito della sicurezza informatica, è l'insieme di contromisure, tecnologiche e logistico-organizzative, impiegate per garantire l'erogazione continua di servizi (di attività commerciali, associazioni, enti, amministrazioni, ecc.), nonostante il sopraggiungere di gravi esternalità, dovute ad errore umano o ad eventi accidentali. Il "Disaster Recovery Plan (DRP)" è solitamente il documento che esplicita tali contromisure ed è incluso nel "Business Continuity Plan (BCP)", ossia nel piano che definisce il livello minimo di qualità dei servizi erogati, da garantire obbligatoriamente anche in contesti deficitari e/o precari].

Immaginate ad esempio, che erroneamente abbiate cancellato il programma dpkg e che non possiate più installare i pacchetti Debian. Essendo lo stesso dpkg un pacchetto Debian, uno sprovveduto istintivamente potrebbe credere che il suo sistema sia stato realizzato appositamente per [i pacchetti Debian]. Diversamente da questi, per fortuna, conoscete le proprietà del formato di un pacchetto, dunque sapete che potete scaricarlo ed installarlo manualmente (leggete al riguardo la casella di testo "dpkg, APT e ar" a pagina 72). Se per qualche esternalità uno o più dei seguenti programmi "ar, tar o gzip/xz/bzip2" dovessero cancellarsi, necessiterete soltanto di copiarli da un altro sistema (in quanto ciascuno dei suddetti programmi funziona in modo completamente autonomo, senza dipendenze e di conseguenza una semplice copia sarà sufficiente). E se per assurdo la fortuna vi avesse girato le spalle al punto che il vostro sistema continui a non funzionare nonostante il soprastante suggerimento (magari a causa della perdita di librerie fondamentali) dovrete fare semplicemente riferimento alla versione "static" di busybox (inclusa nel pacchetto busybox-static), in quanto più indipendente rispetto ai programmi sopracitati e poiché vi consentirà di usufruire dei suoi sotto-comandi fra cui busybox ar, busybox tar e busybox gunzip.

STRUMENTI
TOOLS
dpkg, APT e ar

dpkg è il programma che gestisce i files .deb, specificatamente la loro estrazione, analisi e “spacchettamento”.
APT è un gruppo di programmi che consente di mettere in atto delle rilevanti modifiche (tipiche di un tool di Alto Livello), che hanno influenza sull’intero sistema: effettua l’installazione e la rimozione dei pacchetti (mantenendo nel contempo soddisfatte le dipendenze), aggiorna il sistema, crea una lista di pacchetti al bisogno disponibili, ecc.
Il programma archiver noto come ar consente la gestione dei files richiamandoli semplicemente attraverso il loro stesso nome, ad esempio: `ar t archive` mostra l’elenco dei files contenuti nell’archivio denominato “archive”, presente nella stessa directory da cui avete fatto partire il comando; `ar x archive` estrae i files contenuti nell’archivio denominato “archive”, nella stessa directory da cui avete fatto partire il comando, dove tra l’altro si trova lo stesso archivio; `ar d archive file` cancella il file denominato “file” dall’archivio denominato “archive”, presente nella stessa directory da cui avete fatto partire il comando; e così via ... La sua man page [`ar(1)`] documenta tutte le sue altre funzioni.
Occorre precisare che l’archiver ar è un tool rudimentale, che solitamente un amministratore Unix non utilizza spesso, se non per rare occasioni o necessità, in quanto preferisce utilizzare `tar`, per di più con una certa regolarità, essendo fra i due il programma [di gestione files ed archivi] più moderno ed evoluto. Infatti grazie a tar è possibile ripristinare un file dpkg, cancellato a causa di un evento accidentale. Dovrete soltanto scaricare il pacchetto Debian ed estrarre tramite tar il contenuto dell’archivio `data.tar.gz` nella directory root del sistema (/):

```
# ar x dpkg_1.17.23_amd64.deb
# tar -C / -p -xzf data.tar.gz
```

BASILARE
Man Page: la notazione

La notazione di man page può confondere i neofiti che la incontrano per la prima volta, in particolare quando la manualistica la richiama per citare un programma come ad esempio: "`ar(1)`". In questo caso si riferisce alla pagina man page dedicata ed intitolata ar nella sezione 1.
In realtà la notazione di man page viene comunemente utilizzata per rimuovere le ambiguità, ovvero per distinguere le pagine omonime di sezioni diverse come ad esempio nel caso di `printf(1)`, nella sezione 1 e dedicata al comando printf, e di `printf(3)`, nella sezione 3 e dedicata alla funzione printf in C (linguaggio di programmazione).
Il capitolo 7 "Come risolvere le problematiche e trovare le informazioni adeguate" a pagina 134 tratta le man pages dettagliatamente (andate a vedere il paragrafo 7.1.1, "Manual Pages" a pagina 134).

Adesso verificate il contenuto di un file .deb:

```
$ ar t dpkg_1.17.23_amd64.deb
debian-binary
control.tar.gz
data.tar.gz
$ ar x dpkg_1.17.23_amd64.deb
$ ls
control.tar.gz data.tar.gz debian-binary dpkg_1.17.23_amd64.deb
$ tar tzf data.tar.gz | head -n 15
./
./var/
./var/lib/
./var/lib/dpkg/
./var/lib/dpkg/parts/
./var/lib/dpkg/info/
./var/lib/dpkg/alternatives/
./var/lib/dpkg/updates/
./etc/
./etc/logrotate.d/
./etc/logrotate.d/dpkg
./etc/dpkg/
./etc/dpkg/dpkg.cfg.d/
./etc/dpkg/dpkg.cfg
./etc/alternatives/
$ tar tzf control.tar.gz
./
./conffiles
./postinst
./md5sums
./prerm
./preinst
./control
./postrm
$ cat debian-binary
2.0
```

Come potrete notare, l'archivio `ar` è composto da tre files:

• `debian-binary`. Questo è un file text che semplicemente indica la versione del file .deb utilizzata (ad esempio, nel 2015, la versione 2.0);
• `control.tar.gz`. Questo file d'archivio contiene tutte le meta-informazioni disponibili, tra cui il nome e la versione del pacchetto. Alcune di queste meta-informazioni sono necessarie affinché gli strumenti di gestione dei pacchetti possano valutare se è possibile installare o disinstallare un pacchetto, ad esempio in base all'elenco dei pacchetti già installati sulla macchina;
• `data.tar.gz`. Questo archivio contiene tutti i files che devono essere estratti dal pacchetto; ovvero contiene i files eseguibili, la documentazione, ecc.. Qualche pacchetto può essere in un altro formato di compressione ed in tal caso il nome del file d'archivio che lo riguarderà lo dichiarerà esplicitamente (`data.tar.bz2` per `bzip2`, `data.tar.xz` per `XZ`).

5.2 La Meta-Informazione di un Pacchetto

Il pacchetto Debian non è solo un archivio di files destinato all'installazione. Bensì è parte di un vasto complesso organizzato, tanto che questi definisce anche il suo “rapporto” con gli altri pacchetti Debian (dipendenze, conflitti e suggerimenti). Inoltre i suoi scripts consentono l'esecuzione dei comandi relativi al ciclo di vita dello stesso pacchetto (installazione, rimozione, aggiornamenti). I soprastanti dati sono usati dallo strumento di gestione dei pacchetti, ma non fanno parte del software “impacchettato”; altri non sono che meta-informazione [o metadato, letteralmente "(dato) per mezzo di un (altro) dato"], contenuta all'interno del pacchetto.

5.2.1 Descrizione: il control File

Questo file usa una struttura simile agli headers email (struttura, nel caso degli headers email, stabilita dallo standard RFC 2822). Ad esempio, il file control di apt si presenta in questo modo:

```
$ apt-cache show apt
Package: apt
Version: 1.0.9.6
Installed-Size: 3788
Maintainer: APT Development Team <deity@lists.debian.org>
Architecture: amd64
Replaces: manpages-it (<<2.80-4~), manpages-pl (<<20060617-3~),
openjdk-6-jdk (<< 6 b24-1.11-0ubuntu1~), sun-java5-jdk (>> 0), sun-java6-jdk(>>
0)
Depends: libapt-pkg4.12 (>= 1.0.9.6), libc6 (>= 2.15), libgcc1 (>=
1:4.1.1), libstdc ++6 (>= 4.9), debian-archive-keyring, gnupg
Suggests: aptitude | synaptic | wajig, dpkg-dev (>= 1.17.2), apt-doc,
python-apt
Conflicts: python-apt (<< 0.7.93.2~)
Breaks: manpages-it (<< 2.80-4~), manpages-pl(<< 20060617-3~),
openjdk-6-jdk (<< 6 b24-1.11-0ubuntu1~), sun-java5-jdk (>> 0), sun-java6-jdk (>>
0)
Description-en: commandline package manager
      This package provides commandline tools for searching and managing
      as well as querying information about packages
      as a low-level access to all features of the libapt-pkg library.
.
These include:
```

```
 * apt-get for retrieval of packages and information about them
   from authenticated sources and for installation, upgrade and
   removal of packages together with their dependencies
 * apt-cache for querying available information about installed
   as well as installable packages
 * apt-cdrom to use removable media as a source for packages
 * apt-config as an interface to the configuration settings
 * apt-key as an interface to manage authentication keys
Description-md5: 9fb97a88cb7383934ef963352b53b4a7
Tag: admin::package-management, devel::lang:ruby, hardware::storage,
 hardware::storage:cd, implemented-in::c++, implemented-in::perl,
 implemented-in::ruby, interface::commandline, network::client,
 protocol::ftp, protocol::http, protocol::ipv6, role::program,
 role::shared-lib, scope::application, scope::utility, sound::player,
 suite::debian, use::downloading, use::organizing, use::searching,
 works-with::audio, works-with::software:package, works-with::text
Section: admin
Priority: important
Filename: pool/main/a/apt/apt_1.0.9.6_amd64.deb
Size: 1107560
MD5sum: a325ccb14e69fef2c50da54e035a4df4
SHA1: 635d09fcb600ec12810e3136d51e696bcfa636a6
SHA256: 371a559ce741394b59dbc6460470a9399be5245356a9183bbeea0f89ecaabb03
```

BASILARE RFC - Internet Standards	RFC è l'acronimo di "Request for Comments". Solitamente un documento "RFC" è un documento tecnico il cui contenuto descrive un potenziale internet standard. Prima che questi vengano standardizzati o ibernati sono sottoposti ad una revisione pubblica (modalità da cui ha origine il termine "RFC"). Lo IETF (Internet Engineering Task Force) decide lo status, e quindi la promozione o il declassamento da uno status ad un altro, di questi documenti (proposed standard, draft standard, o standard). Il documento RFC 2026 definisce il processo di standardizzazione dei protocolli internet. ✦http://www.faqs.org/rfcs/rfc2026.html

5.2.1.1 Dipendenze: il Depends Field (il campo "Dipende da...")

Le dipendenze vengono dichiarate attraverso il Depends Field nell'header del pacchetto. Il Depends Field è un elenco di condizioni che devono essere soddisfatte per poter far funzionare il pacchetto correttamente - queste informazioni sono usate dai programmi come apt per installare le librerie necessarie nella loro versione più idonea, ovvero nella versione che al meglio soddisfa le dipendenze. Difatti, attraverso il Depends Field, è possibile soddisfare le dipendenze limitando di fatto il numero di versioni che assecondano l'elenco delle suddette condizioni. Ad esempio, attraverso il Depends Field è possibile dichiarare che il pacchetto ha bisogno del pacchetto `libc6` nella versione maggiore di o uguale alla "2.15" [che si scrive "libc6 (>=2.15)"]. Gli operatori di relazione (o di confronto) [l'operatore, in informatica, è il simbolo che specifica quale legge applicare ad uno o più operandi per generare un risultato] utilizzati per definire le versioni sono i seguenti:

- <<: minore di;
- <=: minore di o uguale a;
- =: uguale a (vi facciamo notare che la versione "2.6.1" non è uguale alla versione "2.6.1-1");

- >=: maggiore di o uguale a;
- >>: maggiore di.

[Un connettivo logico o operatore logico è un elemento grammaticale di collegamento che instaura fra due proposizioni A e B una qualche relazione che dia origine ad una terza proposizione C con un valore vero o falso, in base ai valori delle due proposizioni fattori ed al carattere del connettivo utilizzato.]

In un elenco di condizioni da soddisfare, la virgola viene impiegata per i delimiter-separated values (o DSV) ossia viene utilizzata per delimitare i valori separati. Deve essere interpretata come una congiunzione logica "e" (and). [Ad esempio i file "CSV" o "comma-separated values" devono il loro nome alla loro struttura dove i valori separati sono delimitati da una virgola]. Inoltre nel suddetto elenco di condizioni viene anche utilizzato come delimitatore la "pipe" o "vertical bar" ("|"), che può essere assunta come disgiunzione inclusiva "o" (or) [detta anche "disgiunzione logica"] e non come una "disgiunzione esclusiva" (in latino "aut/aut" — "o/o"). Nelle proposizioni, del summenzionato elenco di condizioni, la disgiunzione inclusiva "o" può essere utilizzata tutte le volte di cui se ne ha bisogno ed ha un ordine di priorità maggiore rispetto ad una congiunzione logica "e". Pertanto la dipendenza "(A o B) e C" verrà espressa nel seguente modo: "A|B, C". Diversamente l'enunciato "A o (B e C)" dovrà essere espresso come "(A o B) e (A o C)" ovvero in "A|B, A|C", dal momento che il Depends Field rispetta l'ordine di priorità degli operatori logici e non consente che le parentesi siano usate per violare tale ordine.

✦ http://www.debian.org/doc/debian-policy/ch-relationships.html

Il sistema delle dipendenze rappresenta un buon metodo organizzativo per garantire il corretto funzionamento dei programmi e grazie ai "meta-packages" ha anche un ulteriore scopo. Difatti questi pacchetti sono vuoti, descrivono solo le dipendenze e sono impiegati singolarmente per facilitare l'installazione di un consistente gruppo di programmi predefinito dal manutentore dello stesso meta-package; di conseguenza, il comando `apt install meta-package` installerà tutti i programmi automaticamente semplicemente soddisfacendo le dipendenze del meta-pacchetto. `gnome`, `kde-full` e `linux-image-amd64` sono esempi di meta-pacchetto.

DEBIAN POLICY	
I fields: Recommends [Le Dipendenze Raccomandate], Suggests [Le Dipendenze Consigliate] e Enhances [Ulteriori Suggerimenti per migliorare le Dipendenze]	I fields Recommends e Suggests descrivono delle dipendenze che in realtà non sono obbligatorie. Le dipendenze “raccomandate” sono le più importanti e migliorano considerevolmente le funzionalità messe a disposizione dal pacchetto, ma non sono indispensabili per il suo funzionamento. Le dipendenze “consigliate” sono al contrario di importanza secondaria e notificano che alcuni pacchetti possono integrare ed incrementare l’utilità del pacchetto in questione, ma è comprensibilmente ragionevole non installare tutti i pacchetti consigliati, se non addirittura limitarsi all’installazione di uno soltanto. Dovreste installare invece sempre i pacchetti “raccomandati”, a meno che non abbiate la certezza che non siano necessari. Diversamente, non è essenziale installare i pacchetti “consigliati” a meno che non abbiate la certezza che siano indispensabili. Anche il field Enhances espone dei suggerimenti, ma che riguardano altro. Il miglioramento proposto di fatto risiede nel pacchetto suggerito e non nel pacchetto che beneficia dei suggerimenti. La sua utilità consiste nella possibilità di poter aggiungere un ulteriore “suggerimento” senza essere obbligati a modificare il pacchetto coinvolto. Pertanto, tutti gli add-ons [i componenti aggiuntivi], i plugins [i programmi non autonomi che interagiscono con un altro programma per ampliarne o estenderne le funzionalità originarie] e le altre estensioni di un programma possono essere elencati fra i suggerimenti pertinenti al software. Quest’ultimo field, pur esistendo da diversi anni è ancora solitamente ignorato dai programmi come apt o synaptic. Inoltre lo scopo del field Enhances è anche proporre all’utente ulteriori suggerimenti, attraverso uno specifico field in aggiunta ai tradizionali suggerimenti che si trovano nel field Suggests.

DEBIAN POLICY Pre-Depends, le Dipendenze più gravose [per apt]	Le “pre-dipendenze”, che sono elencate nel field “Pre-depends” negli headers dei pacchetti, completano le normali dipendenze; la loro sintassi è identica [alle normali dipendenze]. Le dipendenze normali si limitano ad imporre che lo spacchettamento e la configurazione del pacchetto che attenzionano sia eseguito prima della configurazione del pacchetto che ne dichiara la dipendenza. Le pre-dipendenze, invece, stabiliscono che lo spacchettamento e la configurazione del pacchetto che attenzionano sia eseguito prima dello script di pre-installazione del pacchetto che ne dichiara la dipendenza, ovvero prima della sua stessa installazione. Una pre-dipendenza è più “gravosa” per apt, in quanto impone un ulteriore obbligo “intransigibile” nella sua scaletta di pacchetti da installare. Se ne sconsiglia pertanto l’impiego a meno che non sia strettamente necessario. Ad ogni modo è caldamente consigliato consultarsi con altri sviluppatori su debian-devel@lists.debian.org prima del suo impiego. Solitamente è possibile evitarne l’uso ricercando un’altra soluzione come un work-around [in informatica un work-around è una fix transitoria in attesa che si possa trovare una soluzione con il metodo tradizionale].

5.2.1.2. Conflitti: il Conflicts Field (il campo "Entra in conflitto con...")

Il Conflicts Field dichiara se e quando un pacchetto non può essere installato simultaneamente con un altro. Le ragioni scaturenti più consuete sono che: l'altro pacchetto comprende un file con lo stesso nome; l'altro pacchetto fornisce lo stesso servizio sullo stesso port TCP; i pacchetti, se installati simultaneamente, intralciano vicendevolmente le rispettive funzioni. Si precisa che dpkg non installerà mai un nuovo pacchetto che innesca un conflitto con un pacchetto già installato, a meno che il nuovo pacchetto non dichiari esplicitamente di essere in grado di sostituirlo e, si reitera, che solo in tale evenienza dpkg accetterà di effettuare la sostituzione del vecchio pacchetto già installato con quello nuovo. Dal canto suo apt seguirà sempre le vostre istruzioni: se sceglierete di installare un nuovo pacchetto vi offrirà di disinstallare automaticamente i pacchetti che presentano problemi.

5.2.1.3. Incompatibilità: il Breaks Field (il campo "Interrompe ...")

Il Breaks Field produce dei risultati simili al Conflicts Field, ma ha un proposito specifico. Questo campo avvisa l'utente che l'installazione di un pacchetto "interromperà" un altro pacchetto (o delle specifiche versioni di questi). Generalmente l'incompatibilità tra i due pacchetti è transitoria, inoltre la combinazione tra i due pacchetti che ne determina l'incompatibilità è circostanziata a delle specifiche versioni.
Si precisa che dpkg si rifiuterà di installare un pacchetto che interrompe un pacchetto già installato e che
apt, d'altronde, proverà a risolvere il problema aggiornando il pacchetto che potrebbe essere "rotto" ad una versione più recente (che si presume sia stata nel frattempo riparata e, di conseguenza, di nuovo compatibile).
Il soprammenzionato contesto potrebbe verificarsi a causa di aggiornamenti con vizi di retrocompatibilità: ad esempio quando una nuova versione non supporta più alcune funzionalità della versione precedente, innescando un malfunzionamento in un altro programma che si ritroverà scoperto del dovuto sostegno. Il Breaks Field dovrebbe essere in grado teoricamente di tutelare l'utente da tali problematiche.

5.2.1.4. Provided Items [Gli Oggetti Forniti]: il Provides Field (il campo "Fornisce ...")

Il Provides Field introduce un concetto davvero interessante ovvero quello dei "virtual packages". Il virtual package ricopre diversi ruoli, di cui due di notevole influenza. Il primo ruolo consiste

nell'associare il "virtual package" ad un servizio generico (da qui si intuisce le ragioni del nome del campo "Provides Field" ossia "il pacchetto *fornisce* il servizio ..."). Il secondo ruolo invece consiste nell'avvertire l'utente che un pacchetto è in grado di sostituire completamente un altro pacchetto, soddisfacendo tra l'altro anche le dipendenze che il pacchetto replicato è in grado di soddisfare. Ciò rende possibile realizzare un pacchetto sostitutivo senza vincoli sul suo nome.

DIZIONARIO
Meta-package e virtual package

Occorre distinguere i meta-packages dai virtual packages. I primi sono pacchetti reali che includono un file .deb e la loro unica ragione d'esistere è la mera dichiarazione delle dipendenze.
I pacchetti virtuali, diversamente, non esistono fisicamente; sono solo un mezzo per identificare i pacchetti reali filtrandoli in base ad un criterio logico che hanno in comune (il servizio fornito, la compatibilità con un programma standard o con un pacchetto preesistente, ecc.).

Un esempio di un "Servizio fornito". Il caso che qui tratteremo sarà discusso dettagliatamente attraverso un esempio: tutti i servers mail come postfix o sendmail dichiarano di "fornire" il virtual package mail-transport-agent. Pertanto, qualsiasi pacchetto che necessita questo servizio per funzionare (ad esempio un sistema di gestione di mailing list, come smartlist o sympa) dichiarerà semplicemente nelle sue dipendenze di aver bisogno di un mail-transport-agent anziché specificare un'ampia e probabilmente insoddisfacente lista di possibili soluzioni (postfix|sendmail|exim4| ...). È bene ribadire che non ha senso installare due servers mail sulla stessa macchina, per cui ciascuno di questi pacchetti di servers mail dichiarerà a sua volta un conflitto con lo stesso virtual package mail-transport-agent individualmente fornito. Nell'atto pratico durante l'installazione di uno di questi pacchetti di servers mail (premesso che non ci siano già altri servers mail installati sulla stessa macchina) o di uno di questi pacchetti che necessitano di un mail-transport-agent tale conflitto (ovvero il conflitto di un pacchetto con se stesso) sarà ignorato dal sistema e di conseguenza questo metodo impedirà all'utente l'installazione di due servers mail contemporaneamente.

DEBIAN POLICY
L'elenco dei pacchetti virtuali

I nomi dei virtual packages devono essere concordati in modo da garantire la disponibilità. Per tale ragione sono standardizzati nella Debian Policy.
Nell'elenco sono inclusi tra l'altro: mail-transport-agent per i mail servers; c-compiler per i compilatori in linguaggio di programmazione C; www-browser per i web browsers; httpd per i web servers; ftp-server per i FTP servers; x-terminal-emulator per i terminal emulators [gli emulatori di terminale o terminali virtuali] in graphical mode [modalità grafica] (xterm) e x-window-manager per i window managers [i gestori delle finestre].
L'elenco completo può essere trovato sul web.
✦http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt

L'intercambiabilità con un altro pacchetto. Il Provides Field ha la sua importanza anche in quei casi in cui il contenuto di un pacchetto è stato ormai integrato in un pacchetto più grande. Per esempio, in passato il modulo Perl libdigest-md5-perl era inizialmente un modulo opzionale per Perl 5.6, che poi è stato integrato come standard in Perl 5.8 (e nelle sue versioni successive, tra cui la versione 5.20 presente in Jessie). Per questo il pacchetto perl a partire dalla versione 5.8 presenta la seguente dichiarazione <<Provides Field: libdigest-md5-perl>>, per far sì che le dipendenze che necessitano di questo pacchetto siano soddisfatte dalla mera presenza di Perl 5.8 o delle sue versioni successive. Quindi il pacchetto libdigest-md5-perl, modulo opzionale delle vecchie versioni di Perl ormai eliminate, può oggi a sua volta essere eventualmente eliminato, non avendo più ragione di esistere.

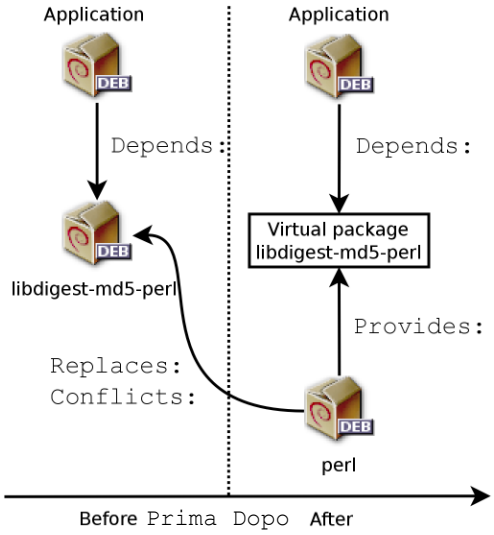

Figura 5.1 Come viene impiegato il Provides field in modo da non "interrompere" le dipendenze.

Questa funzione è molto utile in quanto, pur non essendo possibile prevedere tutte le evoluzioni ad libitum dello sviluppo, metterà in condizioni di far fronte alla ridenominazione e ad altri rimpiazzi automatici del software obsoleto.

BASILARE Perl, un linguaggio di programmazione	Perl (Practical Extraction and Report Language) è un linguaggio di programmazione molto famoso. Presenta molti moduli pronti all'uso che possono essere impiegati da una vasta gamma di applicazioni e che sono distribuiti attraverso i servers CPAN (Comprehensive Perl Archive Network), un'esaustiva rete di pacchetti in Perl. ✦http://www.perl.org/ ✦http://www.cpan.org/ Essendo un interpreted language, un programma scritto in Perl non necessita di una compilazione prima della sua esecuzione. Per questo motivo questi programmi sono definiti "Perl scripts".

Le limitazioni passate. I virtual packages presentavano in passato diverse limitazioni, tra le quali la più importante era l'assenza del numero identificativo della versione.
Riprendendo l'esempio precedente, il mancato riconoscimento del numero identificativo della versione comportava ad esempio che la dichiarazione <<Depends:libdigest-md5-perl (>=1.6)>> non fosse sufficiente per far considerare al packaging system la dipendenza soddisfatta, nonostante la presenza ad esempio di Perl 5.10 – e di conseguenza nonostante fosse di fatto soddisfatta.
Reso inconsapevole dal suddetto limite, il packaging system, che adotta ancora oggi un metodo decisionale con una preferenza di scelte a rischio minimo, ipotizzava che le versioni non fossero compatibili.
Questa limitazione è stata risolta con dpkg 1.17.11. e la sua presenza è stata del tutto rimossa a partire da Jessie. I pacchetti finalmente possono assegnare ai virtual packages, indicati nel loro Provides Field, un numero identificativo della versione, ad esempio <<Provides:libdigest-md5-perl (=1.8)>>.

5.2.1.5 Sostituzione dei Files: il Replaces Field (il campo "Sostituisce...")

Il Replaces field dichiara i files contenuti dal pacchetto stesso che, pur essendo presenti in altri pacchetti, è comunque legittimamente autorizzato a sostituire. Senza un'espressa dichiarazione nel suddetto field dpkg interrompe la sua esecuzione, annunciando a sua volta di non potere sovrascrivere i files di un altro pacchetto (anche se teoricamente è possibile comunque farlo, nonostante non sia considerata una procedura da prendere come modello, aggiungendo al comando dpkg --force-overwrite). Il flop di dpkg consente l'identificazione di potenziali anomalie ed impone al maintainer di studiare la questione prima di aggiungere qualcosa nei fields.
L'uso di questo field è giustificato quando il nome di un pacchetto è cambiato o quando un pacchetto è incluso in un altro. In particolare quest'ultime evenienze possono palesarsi anche quando il manutentore decide di distribuire i files in modo diverso tra diversi pacchetti binari prodotti dallo stesso pacchetto sorgente: si precisa che un file sostituito non appartiene più al vecchio pacchetto, ma solo a quello nuovo.
Se tutti i files in un pacchetto installato sono stati sostituiti, il pacchetto è considerato eliminato.
Infine, questo campo richiede anche a dpkg di eliminare il pacchetto sostituito in caso di conflitto.

ANDANDO OLTRE
Il field Tag

Anche nell'esempio soprastante possiamo scorgere la presenza del field Tag, che non è stato ancora descritto. Questo field non è impiegato per evidenziare un mero confronto/rapporto da i pacchetti, bensì è un metodo pratico per catalogare i pacchetti in base ad una tassonomia tematica. Tale classificazione, messa in atto secondo diversi criteri (tipo di interfaccia, linguaggio di programmazione, l'area di competenza, ecc.) è disponibile da diverso tempo. Ma, nonostante ciò, non tutti i pacchetti hanno delle tags [metadati - parole chiave] accurate e/o non tutti i tools di Debian sono stati integrati con tale sistema; aptitude ne consente sia la visione, sia di disporne come metodo di ricerca. Per coloro che sono allergici ai metodi di ricerca di aptitude possono usufruire del seguente database tags on line:
✦http://debtags.alioth.debian.org/

5.2.2. Scripts di Configurazione

Oltre al file control, all'archivio control.tar.gz i pacchetti Debian possono contenere una quantità di scripts, richiamati da dpkg durante le differenti fasi dell'elaborazione di un pacchetto. La Debian Policy descrive dettagliatamente tutti i casi che possono potenzialmente verificarsi, indicando specificamente gli scripts richiamati e gli argomenti da questi accettati. Questi scripts potrebbero essere talmente complessi da costringere dpkg ad interrompere l'installazione o la rimozione in corso ed a tornare ad uno status [del pacchetto] soddisfacente (per quanto possa essere ancora possibile).
[Negli scripts un **ciclo** (o **iterazione**) è una ripetizione di una determinata azione per un dato numero di volte (**ciclo "for"**). Diversamente **un ciclo o iterazione "while"**, è una ripetizione di una determinata azione sino a quando una data condizione non sarà soddisfatta.]

ANDANDO OLTRE
Perl, un linguaggio di programmazione

Tutti gli scripts di configurazione dei pacchetti installati sono immagazzinati nella directory /var/lib/dpkg/info sotto forma di file che ha come prefisso il nome del pacchetto associato. La suddetta directory include anche il file con estensione .list di ogni pacchetto installato, contenente la lista dei files che appartengono al pacchetto associato.
Il file /var/lib/dpkg/status contiene una serie di blocco dati (nello stesso formato degli headers mail, RFC 2822) che descrivono lo status di ogni pacchetto. Le informazioni contenute nel file control dei pacchetti installati sono riprese anche nel suddetto file.

Solitamente, lo script preinst è eseguito prima di installare il pacchetto, mentre lo script postinst viene eseguito dopo l'installazione. Allo stesso modo, lo script prerm è invocato prima della rimozione di un pacchetto, mentre lo script postrm è invocato dopo una rimozione. L'effetto dell'aggiornamento di un pacchetto determina la rimozione della versione non più aggiornata e l'installazione della più recente compatibile. Per ovvie ragioni non è possibile enunciare tutti i possibili scenari in questa sede, ma possiamo limitarci a trattare i due più comuni: l'installazione/aggiornamento e la rimozione.

ATTENZIONE I nomi "simbolici" [sotto forma di nomenclatura rappresentativa] degli scripts	Le sequenze descritte in questo paragrafo richiamano gli scripts di configurazione attraverso degli specifici nomi, come ad esempio old-prerm o new-postinst. Questi due nomi in particolare si riferiscono rispettivamente il primo al prerm script, contenuto nella vecchia versione del pacchetto (installata di conseguenza prima del suo aggiornamento) ed il secondo al postinst script, contenuto nella nuova versione del pacchetto (installata attraverso l'aggiornamento).

SUGGERIMENTO I Diagrammi di Stato	Manoj Srivastava ha realizzato i diagrammi di stato che descrivono come gli scripts sono richiamati da dpkg. Dei diagrammi di stato inerenti alla summenzionata tematica sono stati realizzati anche dal Debian Women Project; sono un po' più semplici da comprendere, ma meno completi. ✦https://people.debian.org/~srivasta/MaintainerScripts.html ✦https://wiki.debian.org/MaintainerScripts

[Un **diagramma di stato** (anche detto **pallogramma**) è un tipo di diagramma usato in informatica per descrivere il comportamento dei sistemi, attraverso l'analisi e la rappresentazione di una serie di eventi associati a ciascuno stato. Il sistema deve essere però composto da un numero finito di stati e qualora così non fosse lo si può descrivere tramite astrazione].

5.2.2.1 Installazione e Aggiornamento

Quanto esposto in seguito descrive cosa avviene durante un installazione (o aggiornamento):
1. Per effettuare l'aggiornamento, dpkg chiama lo script `old-prerm upgrade new-version`.
2. Durante l'aggiornamento dpkg esegue anche `new-preinst upgrade old-version`; diversamente durante una prima installazione esegue `new-preinst install`; qualora il pacchetto fosse stato installato e rimosso da tempo dpkg aggiungerà l'ultimo parametro "old-version" (ciò significa che i files di configurazione, nonostante la rimozione del pacchetto, sono stati mantenuti ovvero che il pacchetto non è stato "purgato") [in gergo linux "purge" viene utilizzato per eliminare del tutto ed in modo definitivo un pacchetto].
3. I files del nuovo pacchetto vengono quindi "spacchettati". Se uno di questi files già esiste sulla macchina, viene sostituito, ma una copia dello stesso file sostituito viene conservata temporaneamente.
4. Per continuare l'aggiornamento dpkg esegue `old-postrm upgrade new-version`.
5. Dpkg aggiorna tutti i dati (l'elenco dei files, gli scripts di configurazione, ecc.) e rimuove le copie dei file sostituiti. Raggiunta questa fase dpkg non può ritornare più sui suoi passi in quanto non ha più accesso ai files necessari per ripristinare lo status precedente.
6. Dpkg aggiornerà anche i files di configurazione, chiedendo il permesso all'utente qualora non fosse autorizzato ad eseguire questa attività automaticamente. I dettagli di questa procedura sono trattati nel paragrafo 5.2.3, "Checksum, Elenco dei Files di Configurazione" a pag. 83.
7. Infine, dpkg configura il pacchetto eseguendo `new-postinst configure last-version-configured`.

5.2.2.2 Rimozione di un Pacchetto

Quanto esposto in seguito descrive cosa avviene durante una rimozione:

1. Dpkg chiama `prerm remove`.
2. Dpkg rimuove tutti i files del pacchetto, tranne i files di configurazione e gli scripts di configurazione.
3. Dpkg esegue `postrm remove`. Tutti gli script di configurazione, eccetto postrm, sono rimossi. Se l'utente durante la rimozione non ha usato l'opzione "purge", il processo finisce qui.
4. Quando viene "purgato" completamente il pacchetto (attraverso il comando `dpkg --purge` oppure attraverso `dpkg -P`), anche i files di configurazione vengono rimossi, così come alcune specifiche copie (`*.dpkg-top`,`*.dpkg-old`, `*.dpkg-new`) ed i files temporanei; dpkg poi esegue `postrm purge`.

DIZIONARIO
Purge, una rimozione completa

Quando un pacchetto Debian è rimosso, i files di configurazione sono conservati allo scopo di rendere possibile una sua re-installazione. Anche i dati generati da un demone (ad esempio il contenuto di una directory di un LPDA server o di un database SQL) sono comunemente conservati.
Per rimuovere tutti i dati relativi ad un pacchetto, è indispensabile "purgare" lo stesso pacchetto con i comandi: `dpkg -P nomedelpacchetto`; `apt-get remove --purge nomedelpacchetto` oppure `aptitude purge nomedelpacchetto`.
Data la sua efficacia permanente l'opzione "purge" non dovrebbe essere presa alla leggera.

I quattro scripts sopra descritti sono integrati da un config script, di cui sono muniti gli stessi pacchetti che si intendono rimuovere ed ottenuto sempre da questi pacchetti grazie all'utilizzo di debconf per acquisire informazioni per la configurazione dall'utente. Difatti durante la loro installazione, questo script raccoglie dettagliatamente le risposte alle domande poste attraverso debconf. Le risposte sono registrate nel database debconf per agevolare la loro consultazione. Lo script è generalmente eseguito da apt prima di installare i pacchetti uno ad uno, allo scopo di raccogliere tutte le risposte e conoscere tutto sulle preferenze dell'utente prima dell'inizio del processo di installazione. Gli scripts pre e post installazione possono quindi utilizzare poi queste informazioni per svolgere i loro processi in linea con le suddette preferenze utente.

STRUMENTI
TOOLS
debconf

In passato debconf fu creato per risolvere una ricorrente anomalia di Debian. In pratica tutti i pacchetti Debian, inabili di svolgere le loro funzioni senza le informazioni essenziali per la loro configurazione, erano soliti porre delle domande negli scripts shell postinst (ed in altri scripts simili), attraverso chiamate ai comandi `echo` e `read`. Ciò comportava che l'utente fosse obbligato a stare "incollato" al proprio computer durante le installazioni più complesse o gli aggiornamenti per poter rispondere alle domande che potevano presentarsi senza preavviso. Oggi invece si è quasi del tutto dispensati da questa interazione manuale grazie allo strumento debconf.
Inoltre debconf offre diverse funzionalità degne di nota: impone allo sviluppatore di definire dettagliatamente l'interazione con l'utente; consente di effettuare la localizzazione di tutte le stringhe mostrate all'utente (tutte le traduzioni sono contenute nel file template che definisce le summenzionate interazioni); ha differenti frontends per le diverse modalità con cui possono essere esposte le domande all'utente (testuale, grafica o non interattiva); consente la creazione di un archivio risposte principiale attraverso cui condividere la stessa configurazione fra diverse computers... ma la cosa più meritevole è che debconf pone in successione queste domande all'utente prima che inizi un lungo processo d'installazione o di aggiornamento. L'utente può quindi occuparsi della propria attività lavorativa, mentre il sistema gestisce l'installazione dei pacchetti in base alle sue preferenze, senza essere costretto a stare davanti lo schermo in attesa di ricevere delle domande.

[In informatica un **LDAP (Lightweight Directory Access Protocol)** è un protocollo standard per l'interrogazione e la modifica dei servizi di directory, ossia in generale per l'interrogazione o la

modifica di qualsiasi raggruppamento di informazioni che può essere espresso come record di dati ed organizzato in modo gerarchico.
Mentre il **debconf**, in altre parole, è un “assistente all'installazione” che, durante l'installazione o aggiornamento di ogni singolo pacchetto o di un gruppo di pacchetti, pone all'utente delle domande di configurazione, una ad una e tutte in successione, memorizzando le preferenze dell'utente in un database. Durante l'installazione o l'aggiornamento dei pacchetti, gli scripts, degli stessi pacchetti in fase di installazione o aggiornamento, utilizzano le preferenze di configurazione, raccolte nel database, per generare i files di configurazione e compiere attività amministrative, senza la necessità di configurazioni manuali ex post o il presentarsi di domande poste durante l'installazione o l'aggiornamento che ne prolungherebbero i tempi o peggio, in caso di temporanea assenza dell'utente, ne impedirebbero la stessa installazione o aggiornamento.]

5.2.3 Checksum, Elenco dei Files di Configurazione

Oltre ai control data ed agli scripts di configurazione già menzionati nei paragrafi precedenti, l'archivio control.tar.gz dei pacchetti Debian contiene altri files meritevoli di essere citati.
Il primo fra questi è l'md5sums che contiene il checksum in MD5 di tutti i files del pacchetto. Il suo principale scopo è di consentire la validazione dei suddetti files tramite il comando `dpkg --verify` (trattato a pagina 386 dal paragrafo 14.3.3.1 "La validazione dei pacchetti tramite il comando `dpkg --verify`") ed accertarsi che non siano stati modificati dopo l'installazione. Si fa presente che quando questo file non esiste, viene generato automaticamente durante l'installazione (e conservato nel database dpkg insieme agli altri control files).
`conffiles` elenca i files del pacchetto che devono essere gestiti come files di configurazione. I files di configurazione possono essere modificati dall'amministratore, e dpkg tenterà di preservare tali modifiche durante gli aggiornamenti del pacchetto.
Di conseguenza dpkg nel suddetto contesto cerca di comportarsi nel modo più efficiente possibile: se il modello a cui il file di configurazione fa riferimento non è cambiato sia nella versione non più recente e precedentemente installata, sia nella nuova versione aggiornata che sta per essere installata, dpkg non compie alcuna azione. Diversamente se il file è stato modificato tenterà di aggiornarlo. Sono quindi possibili due scenari: il primo in cui dpkg, non avendo l'amministratore modificato il file, esegue l'aggiornamento; il secondo in cui dpkg, avendo l'amministratore modificato il file, chiede allo stesso amministratore se desidera mantenere la versione del file modificato oppure sostituirla con il file incluso nel pacchetto aggiornato. Per facilitare la decisione dell'amministratore dpkg gli propone anche di consultare un "diff" che mostra le differenze fra le due versioni. Se l'utente sceglie di mantenere la vecchia versione, la nuova versione sarà comunque conservata nella stessa collocazione come file con suffisso `.dpkg-dist` Se l'utente sceglie la nuova versione, la vecchia versione è conservata come file con suffisso `.dpkg-old` Un'ulteriore azione disponibile all'utente consiste nell'interrompere temporaneamente dpkg nella scrittura in corso del file e tentare di ripristinare le modifiche più importanti (precedentemente identificate tramite diff).

ANDANDO OLTRE
Come forzare dpkg a porre le domande per il file di configurazione

L’opzione `--force-confask` impone a dpkg di porre le domande inerenti i files di configurazione, anche qualora non fosse necessario. Così facendo, durante la re-installazione del pacchetto dpkg porrà nuovamente le domande sui files di configurazione modificati dall’amministratore. Ciò fa comodo quando sono stati rimossi permanentemente i files di configurazione originali e non si hanno delle copie: difatti una normale reinstallazione non sarà sufficiente a ripristinarli in quanto dpkg considererà la rimozione come una modifica legittima, non installando in sostanza la configurazione automatica.

ANDANDO OLTRE
Come evitare le domande per il file di configurazione

dpkg gestisce gli aggiornamenti del file di configurazione ed a tale scopo interrompe il processo per ricevere gli input dall'amministratore. Questo potrebbe determinare degli inconvenienti a chi sceglie la modalità non interattiva. Per questo motivo il programma offre anche la possibilità di far rispondere automaticamente alle domande direttamente il sistema: `--force-confnew` utilizza la nuova versione del file (queste scelte sono rispettate anche se l'amministratore di fatto non ha effettuato modifiche, pur essendo tale evenienza rara). Aggiungendo invece `--force-confdef`, consentirete a dpkg di scegliere da solo le risposte quando è possibile (ovvero quando il file originale non è stato modificato), pertanto utilizzate `--force-confold` e `--force-confnew` solo per gli altri casi.
Queste opzioni appartengono a dpkg, ma il più delle volte un amministratore utilizza altri programmi come aptitude o apt-get. Di conseguenza, qualora si intendesse utilizzare uno di questi programmi e trasmettere queste opzioni a dpkg, è di primaria importanza conoscere la sintassi corretta (anche se i comandi sono simili):

```
# apt -o DPkg::options::="--force-confdef" -o DPkg::options
-> ::="--force-confold" full-upgrade
```

Queste opzioni possono essere impartite modificando direttamente il file configurazione di apt, che si trova in /etc/apt/apt.conf.d/local :

```
DPkg::options { "--force-confdef"; "--force-confold"; }
```

In questo modo non avrete la necessità di digitarle di volta in volta, saranno valide anche per la modalità grafica e di conseguenza pure per aptitude.

5.3 Struttura di un Pacchetto Sorgente

5.3.1 Struttura

Un pacchetto sorgente comprende generalmente tre files: un file .dsc, un file .orig.tar.gz e un file .debian.tar.gz (o .diff.gz). Questi tre files consentono la creazione dei pacchetti in formato binario (i files .deb precedentemente descritti) a partire dai files del codice sorgente di un programma, scritti in linguaggio di programmazione.
Il file .dsc (Debian Source Control) è un breve file di testo che contiene un header RFC 2822 (proprio come il control file analizzato nel paragrafo 5.2.1 "Descrizione: il control File" a pag. 74) che a sua volta descrive il pacchetto sorgente e specifica gli altri files che ne fanno parte. È firmato dal maintainer che ne certifica l'autenticità. Per maggiori informazioni sull'argomento recatevi al paragrafo 6.5 "Verifica dell'autenticità del pacchetto" a pag. 121.

Esempio 5.1 Un file .dsc

```
----- BEGIN PGP SIGNED MESSAGE -----
Hash: SHA256

Format: 3.0 (quilt)
Source: zim
```

```
Binary: zim
Architecture: all
Version: 0.62-3
Maintainer: Emfox Zhou <emfox@debian.org>
Uploaders: Raphaël Hertzog <hertzog@debian.org>
Homepage: http://zim-wiki.org
Standards-Version: 3.9.6
Vcs-Browser: http://anonscm.debian.org/gitweb/?p=collab-maint/zim.git
Vcs-Git: git://anonscm.debian.org/collab-maint/zim.git
Build-Depends: debhelper (>= 9), xdg-utils, python (>= 2.6.6-3~), libgtk2.0-0
(>=
->2.6), python-gtk2, python-xdg
Package-List:
zim deb x11 optional arch=all
Checksums-Sha1:
ad8de170826682323c10195b65b9f1243fd75637 1772246 zim_0.62.orig.tar.gz
a4f70d6f7fb404022c9cc4870a4e62ea3ca08388 14768 zim_0.62-3.debian.tar.xz
Checksums-Sha256:
19d62aebd2c1a92d84d80720c6c1dcdb779c39a2120468fed01b7f252511bdc2 1772246
zim_0.62.orig.tar.gz
fc2e827e83897d5e33f152f124802c46c3c01c5158b75a8275a27833f1f6f1de 14768 zim
_0.62-3.debian.tar.xz
Files:
43419efba07f7086168442e3d698287a 1772246 zim_0.62.orig.tar.gz
725a69663a6c2961f07673ae541298e4 14768 zim_0.62-3.debian.tar.xz

-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2
Comment: Signed by Raphael Hertzog

iQEcBAEBCAAGBQJUR2jqAAoJEAOIHavrwpq5WFcH/RsdzCHc1oXXxHitU23hEqMj
T6ok29M1UFDJDowMXW75jQ1nT4WPUtvEGygkCHeoO/PvjEvB0sjU8GQlX+N9ddSB
aHfqfAYmVhADNGxrXQT5inZXUa8qGeeq2Sqf6YcWtsnuD56lDbvxkyf/XYopoIEl
oltfl05z/AI+vYsW482YrCz0fxNAKAvkyuPhDebYI8jnKWeAANoqmKpsNc/HYyvT
+ZiA5o57OiGdOKT6XGy3/FiF3dkHiRY8lXW7xdr1BbIgulwl9UmiUNwuxwOYbQO7
edtjiTJqOaFUA0x1zB/XGv5tHr1MjP8naT+kfVoVHTOox51CDbeu5D3DZY4imCY=
=Wtoa

-----END PGP SIGNATURE-----
```

Si precisa che anche il pacchetto sorgente ha delle dipendenze (Build-Depends) completamente distinte da quelle dei pacchetti binari e che tali dipendenze specificano gli strumenti necessari per compilare il software in questione e costruire il relativo pacchetto binario.

ATTENZIONE Distinti namespaces	È importante farvi notare che non c'è una corrispondenza fra il nome del pacchetto binario e quello del pacchetto sorgente da cui il primo è stato generato. Di conseguenza è facilmente intuibile che da un pacchetto sorgente potrebbero essere creati diversi pacchetti binari. Difatti il file .dsc possiede dei Field distinti, Source e Binary, per puntualizzare il nome del pacchetto sorgente (Source Field) ed allo stesso tempo elencare i pacchetti binari (Binary Field) che il pacchetto sorgente, precedentemente individuato, è in grado di generare.

[Un namespace è un insieme di simboli che vengono utilizzati per organizzare oggetti di vario tipo, in modo che questi possano essere specificati per nome. Un namespace garantisce che tutti i suoi identificatori abbiano nomi univoci in modo che questi possano essere facilmente distinti.]

CULTURA

Le ragioni per cui il contenuto di un pacchetto sorgente viene diviso in diversi pacchetti

Molto frequentemente un pacchetto sorgente (di un determinato software) può generare diversi pacchetti binari. La suddivisione è giustificata dalla possibilità di usare il software (o delle sue parti) in contesti differenti.
Prendiamo in considerazione una libreria condivisa, potrebbe essere installata per far funzionare un'applicazione (ad esempio, libc6) oppure per sviluppare un nuovo programma (libc6-deb di conseguenza sarà il pacchetto "corretto" ovvero modificato e/o migliorato). Troviamo la stessa logica per i servizi client/server in cui vogliamo installare la parte server su una macchina e la parte client su altre (questo è il caso, ad esempio, di **`openssh-server`** e **`openssh-client`**).
[La logica in informatica è in generale l'insieme dei principi scelti per la progettazione di un sistema di eleborazione automatica di dati.]
Altrettanto frequentemente, la documentazione viene fornita in un pacchetto dedicato: l'utente può installarlo indipendentemente dal software e può in qualsiasi momento scegliere di rimuoverlo per risparmiare spazio sul disco. Inoltre, ciò consente di risparmiare spazio sul disco dei mirrors Debian, poiché il pacchetto di documentazione sarà condiviso tra tutte le architetture (invece di avere la documentazione duplicata nei pacchetti per ciascuna architettura).

IN PROSPETTIVA

I formati dei pacchetti sorgente sono diversi

Inizialmente esisteva un solo formato di pacchetto sorgente. Era il formato 1.0, che associava un archivio .orig.tar.gz ad una patch.diff.gz per "debianize" ["debianizzare" un pacchetto non nato per Debian ossia eseguire una "debianizaion"] (di tale formato esiste anche una variante, costituita da un singolo archivio .tar.gz, che viene automaticamente utilizzata se non è disponibile un archivio orig.tar.gz).
A partire da Debian Squeeze, gli sviluppatori Debian hanno avuto l'opportunità di utilizzare dei nuovi formati che hanno corretto molti problemi del formato storico. Ad esempio il formato 3.0 (quilt) può combinare più archivi a monte nello stesso pacchetto sorgente: inoltre può includere oltre ai soliti archivi .orig.tar.gz anche degli archivi supplementari .orig-component.tar.gz. Tale sistema è utile in particolare con quel software che è distribuito in più componenti a monte, ma che si preferisce redistribuirlo in un singolo pacchetto sorgente. Questi archivi possono anche essere compressi con bzip2 o xz anziché con gzip, consentendo di risparmiare spazio sul disco e risorse di rete. Infine, la monolitic patch [in un unico blocco], .diff.gz è stata sostituita da un archivio .debian.tar.gz contenente le istruzioni di compilazione e la serie di patches up-stream [a monte] fornite dal manutentore del pacchetto. E queste ultime vengono salvate in un formato compatibile con **quilt**, un tool, che non a caso, facilita la gestione di una serie di patch.

Il file .orig.tar.gz è un archivio contenente il codice sorgente fornito dallo sviluppatore originale. Ai manutentori dei pacchetti Debian viene chiesto di non modificare questo archivio per poter verificare facilmente l'origine e l'integrità del file (mediante un semplice confronto con un checksum) e rispettare le preferenze di alcuni autori.
Il file .debian.tar.gz contiene tutte le modifiche apportate dal manutentore Debian e per di più una directory "debian" contenente le istruzioni da eseguire per mettere insieme il pacchetto Debian.

STRUMENTI TOOL
Come decomprimere un pacchetto sorgente

Se avete un pacchetto sorgente, potete usare il comando dpkg-source (dal pacchetto dpkg-dev) per decomprimerlo:

```
$ dpkg-source -x package_0.7-1.dsc
```

Potete anche usare apt-get per scaricare un pacchetto sorgente e spacchettarlo istantaneamente. Tale metodo tuttavia richiede la presenza delle righe appropriate deb-src nel file elenco /etc/apt/source.list (per ulteriori dettagli, consultate il paragrafo 6.1., “Come compilare il file sources.list” a pag. 100). Questi files .list sono usati per elencare le “fonti” dei pacchetti sorgente (ovvero i servers sui quali viene ospitato un gruppo di pacchetti sorgente).

```
$ apt-get source package
```

5.3.2. Utilizzo in Debian

Ogni cosa su Debian si basa sui pacchetti sorgente. Tutti i pacchetti Debian provengono da un pacchetto sorgente e ogni modifica in un pacchetto Debian è la conseguenza di una modifica apportata al pacchetto sorgente. I manutentori Debian operano direttamente sul pacchetto sorgente, essendo consapevoli, tuttavia, degli effetti delle loro azioni sui pacchetti binari. Di conseguenza i frutti delle loro fatiche sono riconoscibili nei pacchetti sorgente disponibili su Debian: quindi qualora lo desideriate potrete sempre recuperare i pacchetti sorgente, in quanto, si reitera, tutto deriva da loro.
Quando una nuova versione di un pacchetto (pacchetto sorgente ed uno o più pacchetti binari) arriva sul server Debian, il pacchetto sorgente è il componente più importante. Infatti, questi sarà utilizzato da diverse macchine di architettura diversa connesse in una rete allo scopo di compilarlo in base alle varie architetture supportate da Debian. Il fatto che lo sviluppatore invii anche uno o più pacchetti binari per una data architettura (di solito i386 o amd64) è piuttosto irrilevante, dal momento che tali pacchetti potrebbero anche essere stati generati automaticamente.

5.4 Manipolazione dei Pacchetti con dpkg

Dpkg è il comando su cui si fonda la gestione dei pacchetti Debian sul sistema. Quando avrete a che fare con i pacchetti .deb, sarà dpkg che ve ne consentirà l'installazione o l'analisi dei loro contenuti. Ma questo programma ha solo una visione parziale dell'universo Debian: è in grado di riconoscere cosa è installato sul sistema e di rispondere alle istruzioni sulla riga di comando, ma non è in grado di recepire informazioni in merito alla disponibilità di altri pacchetti. Pertanto fallirà se una dipendenza non è soddisfatta. Diversamente, uno strumento come apt stabilirà l'elenco delle dipendenze per poter installare tutto ciò che è disponibile nel modo più automatico possibile.

NOTA
Dpkg o APT? Quale scegliere?

Dpkg altri non è che uno strumento di sistema (backend), mentre APT è uno strumento più vicino alle esigenze dell’utente, essendo in grado di colmare le lacune del primo. Questi strumenti di fatto lavorano insieme, ciascuno con le proprie competenze, specializzate in determinate attività.

[In informatica i termini front end e back end indicano rispettivamente: il primo la parte visibile all'utente di un programma con cui può interagire (ossia un'interfaccia utente); il secondo la parte non visibile all'utente di un programma, che però permette l'effettivo funzionamento delle interazioni con lo stesso utente.]

5.4.1 Installazione dei Pacchetti

dpkg è innanzitutto lo strumento per installare un pacchetto Debian già disponibile (in quanto al bisogno non è in grado di scaricare nulla). Per effettuare un'installazione dovrete utilizzare l'opzione `-i` o `--install`.

Esempio 5.2 Installazione di un pacchetto tramite dpkg

```
# dpkg -i man-db_2.7.0.2-5_amd64.deb
(Reading database ... 86425 files and directories currently installed.)
Preparing to unpack man-db_2.7.0.2-5_amd64.deb ...
Unpacking man-db (2.7.0.2-5) over (2.7.0.2-4) ...
Setting up man-db (2.7.0.2-5) ...
Updating database of manual pages ...
Processing triggers for mime-support (3.58) ...
```

Dall'esempio soprastante pare evidente che potrete visualizzare i diversi passaggi effettuati da dpkg; in questo modo potrete verificare in quale fase potrebbe essersi verificato un errore. L'installazione può essere anche effettuata in due fasi: nella prima lo spacchettamento, mentre nella seconda la configurazione. Ciò avvantaggia apt-get che può limitare il numero di chiamate a dpkg (dal momento che ciascuna chiamata determina un costo in termini di caricamento del database in memoria, in particolare dell'elenco dei files già installati).

Esempio 5.3 Spacchettamento e configurazioni separatamente

 # dpkg --unpack man-db_2.7.0.2-5_amd64.deb
(Reading database ... 86425 files and directories currently installed.)
Preparing to unpack man-db_2.7.0.2-5_amd64.deb ...
Unpacking man-db (2.7.0.2-5) over (2.7.0.2-5) ...
Processing triggers for mime-support (3.58) ...
 # dpkg --configure man-db
Setting up man-db (2.7.0.2-5) ...
Updating database of manual pages ...

A volte dpkg non è in grado di installare un pacchetto, comunicando l'evento all'utente sotto forma di errore; se l'utente gli ordina di ignorarlo, emetterà semplicemente un avvertimento: ecco a cosa servono le varie opzioni `--force-*`. Il comando `dpkg --force-help` o la documentazione di questo comando vi forniranno l'elenco completo di queste opzioni. L'errore più comune, che prima o poi incontrerete, è il "file collision". Si verifica quando un pacchetto contiene un file già installato da un altro pacchetto e dpkg rifiuta di installarlo. Vengono quindi visualizzati i seguenti messaggi:

Unpacking libgdm (from .../libgdm_3.8.3-2_amd64.deb) ...
dpkg: error processing /var/cache/apt/archives/libgdm_3.8.3-2_amd64.deb (--unpack):
trying to overwrite '/usr/bin/gdmflexiserver', which is also in package gdm3 3.4.1-9

In questo caso, qualora riteniate che la sovrascrittura di questo file non rappresenti un rischio significativo per la stabilità del vostro sistema (il che capita spesso), potrete utilizzare l'opzione `--force-overwrite`, che imporrà a dpkg di ignorare questo errore e di sovrascrivere il file. Sebbene esistano molte opzioni `--force-*`, è probabile che utilizzerete solo `--force-overwrite` regolarmente. Queste opzioni esistono solo per situazioni eccezionali ed è consigliabile farne a meno il più possibile in modo da rispettare le regole imposte dal complesso delle parti che costituiscono il pacchetto. Difatti non dovete dimenticare, che tali regole garantiscono la coerenza e la stabilità del sistema.

ATTENZIONE
L'uso efficace di `--force-*`

L'opzione `--force-*`, se non usata con parsimonia, può compromettere un sistema al punto tale che i comandi della famiglia APT si rifiuteranno di funzionare. Difatti, alcune di queste opzioni consentono di installare un pacchetto anche quando una dipendenza non è soddisfatta o nonostante la presenza di un conflitto. Le conseguenze di ciò saranno un sistema non compatto in termini di dipendenze ed il rifiuto dei comandi APT di eseguire qualsiasi azione, eccetto quelle che ne consentano il ripristino del sistema ad uno status coerente (il che spesso consiste nell'installazione della dipendenza mancante o nella rimozione del pacchetto problematico). Ciò si traduce spesso ad esempio in un messaggio come il sottostante, in questo caso dovuto a seguito dell'installazione di una nuova versione di rdesktop priva della sua dipendenza libc6 in una versione più recente:

```
# apt full-upgrade
[...]
You might want to run 'apt-get -f install' to correct these.
The following packages have unmet dependencies:
rdesktop: Depends: libc6 (>= 2.5) but 2.3.6.ds1-13etch7 is
installed
E: Unmet dependencies. Try using -f.
```

Gli amministratori avventati che sono certi della correttezza della loro analisi possono scegliere di ignorare una dipendenza od un conflitto e quindi utilizzare l'opzione `--force-*` relativa. In questo caso, per poter continuare ad usare apt o aptitude, dovranno però modificare `/var/lib/dpkg/status` per rimuovere o modificare la dipendenza od il conflitto di cui hanno scelto di eseguirne l'override. Questa manipolazione è un hack squallido, che non dovrebbe essere mai usato, se non in casi di estrema necessità. Spesso, una soluzione più adeguata è ricompilare il pacchetto che crea problematiche (consultate il paragrafo 15.1, "Ricompilazione di un Pacchetto dalle sue Sorgenti" a pagina 420) o persino recuperare una versione più recente (potenzialmente corretta) da un repository come ad esempio stable-backports (andate a vedere il paragrafo 6.1.2.4, "Stable Backport" a pagina 103).

5.4.2 Rimozione di un Pacchetto

Invocando dpkg con l'opzione `-r` o `--remove` seguita da un nome di un pacchetto, quest'ultimo sarà rimosso. Questa rimozione non è tuttavia completa in quanto rimarranno conservati tutti i files che il pacchetto rimosso gestiva ovvero: i files di configurazione, gli scripts di configurazione, i files log (i registri di sistema) e tutti i dati utente. Questo metodo consente di disinstallare un programma e di reinstallarlo rapidamente con la stessa configurazione. Per rimuovere il programma ed i suoi contenuti permanentemente dovrete usare l'opzione `-P` o `--purge` seguita dal nome del pacchetto.

Esempio 5.4 Le opzioni Remove e Purge sul pacchetto debian-cd.

```
# dpkg -r debian-cd
(Reading database ... 97747 files and directories currently installed.)
Removing debian-cd (3.1.17) ...
# dpkg -P debian-cd
(Reading database ... 97401 files and directories currently installed.)
Removing debian-cd (3.1.17) ...
Purging configuration files for debian-cd (3.1.17) ...
```

5.4.3 Querying (Richieste) per il Database di dpkg ed ispezione dei Files .deb

BASILARE La sintassi delle options	La maggior parte delle opzioni sono disponibili nella versione “lunga” (una o più parole significative, precedute da un doppio trattino) o “breve” (una singola lettera, spesso l’iniziale di una parola della versione lunga, preceduta da un singolo trattino). Questa convenzione è così comune che è diventata uno standard POSIX.

Prima di concludere questo paragrafo, abbiamo descritto, attraverso il sottostante esempio, una serie di opzioni di dpkg che "interrogano" il suo database interno per ottenere informazioni. Prima ivi citeremo le opzioni lunghe e poi le corrispondenti opzioni brevi (che ovviamente se utilizzate accettano gli stessi argomenti) ovvero: `--listfiles` (o `-L`) `nomedelpacchetto`, che mostra l'elenco dei files installati da questo pacchetto; `--search` (o `-S`) `nomedelfile`, che trova il pacchetto da cui proviene il file; `--status` (o `-s`) `nomedelpacchetto`, che mostra l'header di un pacchetto installato; `--list` (o `-l`), che visualizza l'elenco dei pacchetti noti al sistema e il loro status di installazione; `--contents` (o `-c`) `nomedelfile.deb`, che mostra l'elenco dei files contenuti nel pacchetto Debian specificato; `--info` (o `-I`) `nomedelfile.deb`, che mostra le intestazioni del pacchetto Debian indicato.

Esempio 5.5 Le diverse queries via dpkg

```
$ dpkg -L base-passwd
/.
/usr
/usr/sbin
/usr/sbin/update-passwd
/usr/share
/usr/share/lintian
/usr/share/lintian/overrides
/usr/share/lintian/overrides/base-passwd
/usr/share/doc-base
/usr/share/doc-base/users-and-groups
/usr/share/base-passwd
/usr/share/base-passwd/group.master
/usr/share/base-passwd/passwd.master
```

```
/usr/share/man
/usr/share/man/pl/
/usr/share/man/pl/man8
/usr/share/man/pl/man8/update-passwd.8.gz
/usr/share/man/ru
/usr/share/man/ru/man8
/usr/share/man/ru/man8/update-passwd.8.gz
/usr/share/man/ja
/usr/share/man/ja/man8
/usr/share/man/ja/man8/update-passwd.8.gz
/usr/share/man/fr
/usr/share/man/fr/man8
/usr/share/man/fr/man8/update-passwd.8.gz
/usr/share/man/es
/usr/share/man/es/man8
/usr/share/man/es/man8/update-passwd.8.gz
/usr/share/man/de
/usr/share/man/de/man8
/usr/share/man/de/man8/update-passwd.8.gz
/usr/share/man/man8
/usr/share/man/man8/update-passwd.8.gz
/usr/share/doc
/usr/share/doc/base-passwd
/usr/share/doc/base-passwd/users-and-groups.txt.gz
/usr/share/doc/base-passwd/changelog.gz
/usr/share/doc/base-passwd/copyright
/usr/share/doc/base-passwd/README
/usr/share/doc/base-passwd/users-and-groups.html
```

```
$ dpkg -S /bin/date
coreutils: /bin/date
$ dpkg -s coreutils
Package: coreutils
Essential: yes
Status: install ok installed
Priority: required
Section: utils
Installed-Size: 13855
Maintainer: Michael Stone <mstone@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 8.23-3
Replaces: mktemp, realpath, timeout
Pre-Depends: libacl1 (>=2.2.51-8), libattr1 (>=1:2.4.46-8), libc6 (>=2.17), libselinux1
(>=2.1.13)
Conflicts: timeout
Description: GNU core utilities
This package contains the basic file, shell and text manipulation utilities which are
expected to exist on every operating system.
```

```
Specifically, this package includes:
arch base64 basename cat chcon chgrp chmod chown chroot cksum comm cp
csplit cut date dd df dir dircolors dirname du echo env expand expr
factor false flock fmt fold groups head hostid id install join link ln
logname ls md5sum mkdir mkfifo mknod mktemp mv nice nl nohup nproc numfmt
od paste pathchk pinky pr printenv printf ptx pwd readlink realpath rm
rmdir runcon sha*sum seq shred sleep sort split stat stty sum sync tac
tail tee test timeout touch tr true truncate tsort tty uname unexpand
uniq unlink users vdir wc who whoami yes
Homepage: http://gnu.org/software/coreutils
$ dpkg -l 'b*'
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
[...]
||/ Name              Version              Architecture         Description
+++-===============  ===================  ===================  ==================
un backupninja        <none>               <none>               (no description
                                                                available)

ii backuppc           3.3.0-2              amd64                high-performance,
                                                                enterprise-grade
                                                                system for backin


un base available     <none>               <none>               (no description
                                                                available

un base-config        <none>               <none>               (no description
                                                                available

ii base-files         8                    amd64                Debian base system
                                                                miscellaneous
                                                                files

ii base-passwd        3.5.37               amd64                Debian base system
                                                                master password
                                                                and group files
```

```
$ dpkg -c /var/cache/apt/archives/gnupg_1.4.18-6_amd64.deb
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./lib/
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./lib/udev/
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./lib/udev/rules.d/
-rw-r--r-- root/root         2711 2014-12-04 23:03   ./lib/udev/rules.d/60-gnupg.rules
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./usr/
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./usr/lib/
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./usr/lib/gnupg/
-rwxr-xr-x root/root         39328 2014-12-04 23:03   ./usr/lib/gnupg/gpgkeys_ldap
-rwxr-xr-x root/root         92872 2014-12-04 23:03   ./usr/lib/gnupg/gpgkeys_hkp
-rwxr-xr-x root/root         47576 2014-12-04 23:03   ./usr/lib/gnupg/gpgkeys_finger
-rwxr-xr-x root/root         84648 2014-12-04 23:03   ./usr/lib/gnupg/gpgkeys_curl
-rwxr-xr-x root/root          3499 2014-12-04 23:03   ./usr/lib/gnupg/gpgkeys_mailto
drwxr-xr-x  root/root            0 2014-12-04 23:03   ./usr/bin/
-rwxr-xr-x root/root         60128 2014-12-04 23:03   ./usr/bin/gpgsplit
-rwxr-xr-x root/root       1012688 2014-12-04 23:03   ./usr/bin/gpg
[...]
```

```
$ dpkg -I /var/cache/apt/archives/gnupg_1.4.18-6_amd64.deb
```

```
new debian package, version 2.0.
size 1148362 bytes: control archive=3422 bytes.
   1264 bytes,  26 lines   control
   4521 bytes,  65 lines   md5sum
    479 bytes,  13 lines * postinst  #!/bin/sh
    473 bytes,  13 lines * preinst   #!/bin/sh
Package: gnupg
Version: 1.4.18-6
Architecture: amd64
Maintainer: Debian GnuPG-Maintainers <pkg-gnupg-maint@lists.alioth.debian.org>
Installed-Size: 4888
Depends: gpgv, libbz2-1.0, libc6 (>= 2.15), libreadline6 (>= 6.0), libusb-0.1-4
(>= 2:0.1.12), zliblg (>= 1:1.1.4)
Recommends: gnupg-curl, libldap-2.4-2 (>= 2.4.7)
Suggests: gnupg-doc, libpcsclitel, parcimonie, xloadimage | imagemagick | eog
Section: utils
Priority: important
Multi-Arch: foreign
Homepage: http://www.gnupg.org
Description: GNU privacy guard - a free PGP replacement
 GnuPG is GNU's tool for secure communication and data storage.
 It can be used to encrypt data and to create digital signatures.
 It includes an advanced key management facility and is compliant
 with the proposed OpenPGP Internet standard as described in RFC 4880.
[...]
```

ANDANDO OLTRE
La comparazione delle versioni

Dal momento che dpkg è il programma di riferimento per la gestione dei pacchetti Debian, sempre a questi è affidato anche il compito di fornire l'implementazione [ossia la procedura a partire dagli studi preliminari fino alla sua messa in opera definitiva] del metodo logico comparativo dei numeri di versione. Questo è il motivo per cui dpkg ha un'opzione `--compare-version` che può essere utilizzata dai programmi esterni (ed in particolare dagli scripts di configurazione eseguiti dallo stesso dpkg). Questa opzione richiede tre parametri: un numero di versione, un operatore di confronto e un secondo numero di versione. I diversi operatori possibili sono `lt` (minore a), `le` (minore o uguale a), `eq` (uguale a), `ne` (diverso da), `ge` (maggiore o uguale a) e `gt` (maggiore di). Se l'operatore di confronto impostato è corretto, dpkg restituisce come valore di ritorno 0 (esito positivo); in caso contrario restituisce un valore diverso da zero (che indica un errore).

```
$ dpkg --compare-versions 1.2-3 gt 1.1-4
$ echo $?
0
$ dpkg --compare-versions 1.2-3 lt 1.1-4
$ echo $?
1
$ dpkg --compare-versions 2.6.0pre3-1 lt 2.6.0-1
$ echo $?
1
```

Notate il fallimento imprevisto dell'ultimo confronto: per dpkg l'espressione `pre`, che indica in genere una pre-rilascio, non ha alcun significato particolare; questo programma confronta i caratteri alfabetici allo stesso modo dei numeri `(a<b<c ...)`, in ordine lessicografico [è un criterio di ordinamento di stringhe costituite da una sequenza di simboli per cui è già presente un ordine interno; il suddetto criterio logico di ordinamento corrisponde a quello utilizzato nei dizionari, da cui deriva il nome, anche se è esteso ad un qualunque insieme di simboli]. Per questo motivo dpkg

considera “Opre3” maggiore di “0”. Pertanto quando si vuole specificare nel numero di versione di un pacchetto che si tratta di una pre-rilascio, si usa il carattere tilde “~”:

```
$ dpkg --compare-versions 2.6.0-pre3-1 lt 2.6.0-1
$ echo $?
0
```

5.4.4 Il Log File di dpkg

dpkg tiene un registro [log, in inglese, significa registro] di tutte le sue azioni, in /var/log/dpkg.log. Questo registro è estremamente "verboso", perché descrive in dettaglio ciascuna delle fasi attraverso cui dpkg gestisce i pacchetti. Oltre ad offrire un metodo per verbalizzare le funzioni di dpkg, tale registro facilità soprattutto il mantenimento della cronologia sul development del sistema: difatti attraverso questi è possibile conoscere l'esatto momento in cui ciascun pacchetto è stato installato o aggiornato e queste informazioni possono essere estremamente utili per comprendere le ragioni di un recente cambiamento nel suo funzionamento. Inoltre, considerato che tutte le versioni sono documentate, è semplice eseguire un confronto incrociato delle informazioni con il changelog.Debian.gz dei pacchetti in questione, anche con le segnalazioni di bugs disponibili online.

5.4.5 Supporto Multi-Arch

Tutti i pacchetti Debian hanno un campo Architecture nel loro control information. Questo campo può dichiarare sia il valore "all" (per i pacchetti che non dipendono da una particolare architettura) o altrimenti il nome dell'architettura a cui il pacchetto è destinato (ad esempio amd64, armhf, ecc.). In quest'ultimo caso, per impostazione predefinita, dpkg accetterà di installare il pacchetto solo se l'architettura dichiarata soddisfa l'architettura dell'host ovvero quella restituita dal comando `dpkg --print-architecture`.
Questa restrizione garantisce che gli utenti non finiscano con pacchetti binari compilati per un'architettura errata. Tale metodo potrebbe essere certamente quello definitivo se non fosse che esistono alcuni computer in grado di eseguire i pacchetti binari compilati anche per altre architetture: sia nativamente (un sistema amd64 può eseguire programmi i386); sia tramite emulatori.

5.4.5.1 Abilitare Multi-Arch

Il supporto multi-architettura di dpkg consente all'amministratore di definire le "foreign architectures" [architetture non native che possono essere aggiunte] i cui pacchetti possono essere installati sul sistema. Tale configurazione può essere impartita all'host tramite il comando `dpkg --add-architecture` come mostrato nel seguente esempio. Esiste anche il comando per disabilitare il supporto per l'architettura aggiunta ovvero `dpkg --remove-architecture`, ma può essere utilizzato solo se non è rimasto installato alcun pacchetto di quell'architettura.

```
# dpkg --print-architecture
amd64
# dpkg --print-foreign-architectures
# dpkg -i gcc-4.9-base.4.9.1-19_armhf.deb
dpkg: error processing archive gcc-49-base_4.9.1-19_armhf.deb (--install):
 package architecture (armhf) does not match system (amd64)
Errors were encountered while processing:
dpkg -I /var/cache/apt/archives/gnupg_1.418-6_amd64.deb
```

```
gcc-4.9-base_4.9.1-19_armhf.deb
# dpkg --add-architecture armhf
# dpkg --add-architecture armel
# dpkg --print-foreign-architectures
armhf
armel
# dpkg -i gcc-4.9-base_4.9.1-19_armhf.deb
Selecting previously unselected package gcc-4.9-base:armhf.
(Reading database ...86425 files and directories currently installed.)
Preparing to unpack gcc-4.9-base_4.9.1-19_armhf.deb
Unpacking gcc-4.9-base:armhf (4.9.1-19) ...
Setting up gcc-4.9-base:armhf (4.9.1-19) ...
# dpkg --remove-architecture armhf
dpkg: error: cannot remove architecture 'armhf' currently in use by the database
# dpkg --remove-architecture armel
# dpkg --print-foreign-architectures
armhf
```

NOTA Il supporto multi arch di APT	APT, se il supporto per le "foreign architectures" tramite dpkg è stato configurato, sarà in grado: di rilevare automaticamente le "foreign architectures" e di scaricare automaticamente i files dei pacchetti corrispondenti durante il processo di aggiornamento. Quindi i pacchetti di una specifica foreign architectures, potranno essere installati con il semplice comando : `apt install nomedelpacchetto:architettura`

IN PRATICA Come usare i pacchetti proprietario con binario i386 su architetture amd64	Esistono diverse ragioni per usufruire del supporto multi-architettura fra le quali la più comune è la possibilità di eseguire i pacchetti binari a 32 bit (i386) su sistemi a 64 bit (amd64), in particolare quando applicazioni popolari (come Skype) sono disponibili in versioni a 32 bit. [Il testo originale in inglese fa riferimento agli use cases, ovvero una serie di azioni o di eventi che definiscono tipicamente le interazioni tra un ruolo (denominato attore nell'Unified Modeling Language - UML) e un sistema indirizzato al raggiungimento di un obiettivo]

[*Per il supporto Multi-Arch vedete anche pagina 97]

5.4.5.2 I cambiamenti dovuti a Multi-Arch

Affinché il supporto multi-architettura sia davvero utile e utilizzabile, le librerie devono essere rimpacchettate e spostate in una directory destinata all'architettura (ovvero in base ad una strategia mirata alle differenti architetture), in modo da poter installare più copie della stessa libreria, ma compilate per architetture diverse, allo stesso tempo. In questo modo i pacchetti aggiornati saranno etichettati nell'header con il campo "Multi-Arch:same" che avvisa il sistema di gestione dei pacchetti che più architetture dello stesso pacchetto possono essere co-installate in modo sicuro (e che questi pacchetti a loro volta possono soddisfare solo le dipendenze dei pacchetti con la stessa architettura). Dal momento che il supporto Multi-Arch è attivo solo a partire da Debian Wheezy, non tutte le librerie sono state ancora convertite.

```
$ dpkg -s gcc-4.9-base
dpkg-query: error: --status needs a valid package name but 'gcc-4.9-base' is not:
ambiguous package name 'gcc-4.9-base' with more than one installed instance

Use --help for help about querying packages.
$ dpkg -s gcc-4.9-baseamd64 gcc-4.9-basearmhf | grep ^Multi
Multi-Arch: same
Multi-Arch: same
```

```
$ dpkg -L libgcc1:amd64 |grep .so
/lib/x86_64-linux-gnu/libgcc_s.so.1
dpkg -S /usr/share/doc/gcc-4.9-base/copyright
gcc-4.9-base:amd64, gcc-4.9-base:armhf: /usr/share/doc/gcc-4.9-base/copyright
```

Occorre precisare che i pacchetti "Multi-Arch: same" sono identificabili in modo inequivocabile solo se il loro nome è qualificato con la loro architettura. Questi pacchetti hanno anche la possibilità di condividere i files con le altre istanze dello stesso pacchetto; dpkg si assicura che questi files condivisi fra i pacchetti siano identici bit per bit. Infine, ricordiamo che tutte le istanze dello stesso pacchetto devono avere la stessa versione e quindi devono essere aggiornate contemporaneamente.

Il supporto multi-architettura comporta anche alcune complicazioni nella gestione delle dipendenze. Una dipendenza per essere soddisfatta richiede una delle seguenti due alternative: che un pacchetto sia contrassegnato come "`Multi-Arch:foreign`" oppure un pacchetto con architettura identica a quella del pacchetto che ne dichiara la dipendenza (durante questo processo di risoluzione delle dipendenze, i pacchetti con architettura indipendente si presume siano della stessa architettura dell'host). Una dipendenza può anche essere resa più flessibile in modo da poter essere soddisfatta da un pacchetto di diversa architettura, attraverso la sintassi "`package:any`", ma i pacchetti di architetture aggiunte possono soddisfare tale dipendenza solo se sono contrassegnati come "`Multi-Arch:allowed`".

5.5 Coesistenza con altri Packaging Systems

I pacchetti Debian non sono gli unici pacchetti software utilizzati nel mondo del software libero. Il principale concorrente è il formato RPM della distribuzione Red Hat Linux ed i suoi numerosi derivati. Red Hat Linux è una distribuzione commerciale a cui spesso si fa riferimento. È sovente che il software di terze parti sia distribuito come pacchetto RPM, piuttosto che come pacchetto Debian.

In questo caso, dovreste sapere che il programma `rpm`, che vi permette di gestire i pacchetti RPM, esiste anche come pacchetto Debian; è quindi possibile usare i pacchetti di questo formato su una macchina Debian. Ma allo stesso tempo, dovreste prestare attenzione a limitarvi a questo tipo di pratiche, dall'estrazione di informazioni dal pacchetto o alla verifica della sua integrità. Difatti non è conveniente usare rpm per installare un pacchetto RPM su un sistema Debian in quanto i pacchetti RPM usano i propri database, separati da quelli del software nativo Debian (tra cui dpkg). Questo è il motivo per cui non è possibile garantire una stabile coesistenza tra i due sistemi di gestione pacchetti.
Inoltre l'utility **`alien`** può convertire i pacchetti RPM in pacchetti Debian e viceversa.

COMUNITÀ Come incoraggiare l'adozione di .deb	Se usate regolarmente il comando **alien** per installare i pacchetti RPM provenienti da terze parti a cui fate riferimento, non esitate a comunicare loro gentilmente la vostra preferenza per il formato .deb. Si noti, tuttavia, che il formato del pacchetto non è tutto: un pacchetto .deb creato con alien, o preparato per un'altra versione di Debian rispetto a quella in uso, o anche per una distribuzione derivata come Ubuntu, probabilmente non offrirà lo stesso livello di qualità ed integrazione rispetto ad un pacchetto sviluppato appositamente per Debian Jessie

```
$ fakeroot alien --to-deb phpMyAdmin-2.0.5-2.noarch.rpm
phpmyadmin_2.0.5-2_all.deb generated
$ ls -s phpmyadmim_2.0.5-2_all.deb
  64 phpmyadmim_2.0.5-2_all.deb
```

Scoprirete che questo processo è estremamente semplice. Tuttavia, dovete sapere, che il pacchetto generato non ha informazioni sulle dipendenze, poiché le dipendenze dei due formati del pacchetto non hanno corrispondenze sistematiche fra loro. Spetta quindi all'amministratore assicurarsi personalmente che il pacchetto convertito funzioni correttamente ed è per questo che si dovrebbero evitare, per quanto possibile, i pacchetti Debian generati in questo modo. Fortunatamente, Debian ha la più grande raccolta di pacchetti software di tutte le distribuzioni ed è probabile che ciò che state cercando esista già nel formato nativo.
Se leggerete la pagina man del comando alien, potrete anche notare che questo programma supporta altri formati (di pacchetto), in particolare quello della distribuzione Slackware (realizzati semplicemente con un archivio tar.gz).
È la stabilità del software configurato attraverso il tool dpkg a contribuire alla fama di Debian. La suite di strumenti APT, descritta nel prossimo capitolo, preserva il summenzionato merito dispensando l'amministratore dalla gestione dello status dei pacchetti, un'attività necessaria ma difficile.

[*
Per il supporto multi-arch, specialmente con distro derivate come Ubuntu, anche se con Debian non ce n'è bisogno, è possibile modificare manualmente anche il file `apt.conf` in `/etc/apt/` o in `/etc/apt/apt.conf.d` ed il file sources.list in `/etc/apt/` o in `/etc/apt/sources.list.d/`.]

Sul file apt.conf basta aggiungere le architetture nel seguente modo:
```
APT::Architecture="<arch> <arch>"
```

Sul file sources.list basta aggiungere le architetture nel seguente modo:
```
[arch=amd64,i386]  deb http://deb.debian.org/debian buster main contrib non-free
[arch=amd64,i386]  deb-src http://deb.debian.org/debian buster main contrib non-free

[arch=amd64,i386]  deb http://deb.debian.org/debian-security buster/updates main contrib
non-free
[arch=amd64,i386]  deb-src http://deb.debian.org/debian-security buster/updates main
contrib non-free

[arch=amd64,i386]  deb http://deb.debian.org/debian buster-updates main contrib non-free
[arch=amd64,i386]  deb-src http://deb.debian.org/debian buster-updates main contrib non-
free
```

Dopodiché se intendete utilizzare le soprastanti sorgenti su distro derivate da Debian dovrete scaricare ed installare il keyring di Debian per convalidare i pacchetti scaricati dai repositories Debian e viceversa. Ad esempio se intendete installare i pacchetti da Kali Linux su Debian dovrete aggiungere sia i repositories ufficiali di Kali come sopra mostrato, nonché il keyring di Kali per convalidarne i pacchetti su Debian:

```
wget https://http.kali.org/pool/main/k/kali-archive-keyring/kali-archive-
keyring_***_all.deb
sudo dpkg -i kali-archive-keyring_***_all.deb
```

Infine vi basterà aggiornare il sistema e la macchina con
```
sudo apt-get update && sudo apt-get upgrade && sudo update-initramfs -u && sudo update-
grub && sudo reboot
```

Occorre tenere in considerazione però che il sistema installerà i pacchetti della “release” predefinita. Dovrete pertanto specificare per i pacchetti non nativi la distro di origine tramite il comando apt oppure gestire il tutto attraverso pinning.]

Parole chiave

apt
apt-get
apt-cache
aptitude
synaptic
sources.list
apt-cdrom