APT HOWTO

Gustavo Noronha Silva <kov@debian.org>

Traduzione di Luca Monducci

1.8.10.4 - Marzo 2005

Estratto

Questo documento intende fornire una buona base sul funzionamento del sistema per la gestione dei pacchetti fornito da Debian, APT. Il suo scopo è facilitare la vita ai nuovi utenti Debian e aiutarli ad approfondire le loro conoscenze su come si amministrano i pacchetti. Questo HOWTO fa parte del progetto Debian e ha lo scopo di migliorare il supporto disponibile agli utenti di questa distribuzione.

Avviso di Copyright

Indice

Capitolo 1

Introduzione

In principio esistevano i .tar.gz. Gli utenti dovevano compilare ogni programma che volevano usare sui loro sistemi GNU/Linux. Quando fu creata Debian, fu ritenuto necessario che il sistema incorporasse un metodo di gestione dei pacchetti installati sulla macchina. A questo sistema fu dato il nome `dpkg`. Fu così che nacque il famoso "pacchetto" nel mondo GNU/Linux, poco prima che Red Hat decidesse di creare il proprio "rpm".

Rapidamente un nuovo dilemma si fece strada nelle menti degli sviluppatori di GNU/Linux. A loro serviva un modo rapido, pratico ed efficiente per installare i programmi, che gestisse automaticamente le dipendenze e che avesse cura di mantenere i file di configurazione esistenti mentre si effettuavano i vari aggiornamenti. Ancora una volta Debian ha aperto la strada dando vita a APT (Advanced Packaging Tool), che poi è stato adattato da Conectiva per usarlo insieme a rpm e in seguito è stato adottato anche da altre distribuzioni.

Questo HOWTO non tratta di apt-rpm, così è chiamato l'adattamento fatto da Conectiva di APT, ma eventuali patch a questo documento saranno ben accolte.

Questo manuale è basato sulla prossima release Debian, `Sarge`.

Capitolo 2

Configurazione di base

2.1 Il file /etc/apt/sources.list

Nello svolgere le sue operazioni, APT usa un file che contiene la lista delle "sorgenti" dalle quali può attingere i pacchetti. Questo file è `/etc/apt/sources.list`.

Il contenuto di questo file ha normalmente il seguente formato:

```
deb http://host/debian distribuzione sezione1 sezione2 sezione3
deb-src http://host/debian distribuzione sezione1 sezione2 sezione3
```

È da osservare che l'esempio sopra è fittizio e non dovrebbe essere usato. La prima parola di ogni riga, `deb` o `deb-src`, indica il tipo di archivio: se contiene pacchetti binari (`deb`), che sono i pacchetti già compilati che normalmente usiamo, o se l'archivio contiene i pacchetti sorgente (`deb-src`), che sono il codice originale del programma con l'aggiunta del file di controllo (`.dsc`) e del file `diff.gz` che contiene i cambiamenti necessari per "debianizzare" il programma.

Generalmente questo è il contenuto predefinito di sources.list:

```
# See sources.list(5) for more information, especially
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
deb http://http.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-fre
deb http://security.debian.org stable/updates main contrib non-free

# Uncomment if you want the apt-get source function to work
#deb-src http://http.us.debian.org/debian stable main contrib non-free
#deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib no
```

Queste righe sono quelle necessarie per un'installazione base. La prima delle righe che iniziano con `deb` punta all'archivio ufficiale, la seconda all'archivio non-US e la terza all'archivio degli aggiornamenti di sicurezza Debian.

Le ultime due righe sono commentate (con un "#" all'inizio), così verranno ignorate da apt-get. Queste iniziano con `deb-src` e puntano agli archivi che contengono i pacchetti sorgente. Se si scaricano spesso i sorgenti di un programma per fare dei test o per ricompilarlo, è necessario togliere il "#" all'inizio delle righe.

In `/etc/apt/sources.list` si possono specificare diversi tipi di righe. APT sa come trattare i seguenti tipi di archivio `http`, `ftp`, `file` (file locali, per esempio una directory su cui è montato un CD-ROM) e `ssh`.

Non scordarsi di lanciare `apt-get update` ogni volta che si modifica il file `/etc/apt/source.list`. Questa operazione forza APT ad aggiornare la lista dei pacchetti che può reperire dalle sorgenti specificate nel file stesso.

2.2 Come usare APT localmente

Può capitare di avere dei pacchetti .deb da installare e lo si vuole fare usando APT così che le dipendenze siano risolte automaticamente.

Per farlo bisogna creare una directory e metterci dentro i .deb da trattare. Per esempio:

```
# mkdir /root/debs
```

È possibile modificare le opzioni nel file di controllo dei pacchetti contenuti nell'archivio locale usando il file di `override`. Dentro questo file si possono definire alcune opzioni che hanno precedenza su quelle predefinite nei pacchetti. Il suo formato dovrebbe assomigliare a quello che segue:

```
pacchetto priorità sezione
```

dove "pacchetto" è il nome del pacchetto, "priorità" può essere low, medium o high e "sezione" è la sezione alla quale appartiene. Il nome di questo file non è importante, successivamente dovrà essere passato come argomento a `dpkg-scanpackages`. Se non si vuole scrivere o usare un file di `override`, è sufficiente usare `/dev/null` quando si richiama `dpkg-scanpackages`.

Rimanendo nella directory `/root` eseguire:

```
# dpkg-scanpackages debs file | gzip > debs/Packages.gz
```

Nella riga sopra, dove *file* è il file di `override`, il comando genera il file `Packages.gz`, questo contiene delle informazioni sui pacchetti che saranno usate da APT in seguito. Infine, per usare i nostri pacchetti, aggiungere questa riga a `/etc/apt/source.list`:

```
deb file:/root debs/
```

Adesso si può usare APT normalmente. Anche per i pacchetti sorgente si può generare un archivio locale. Per farlo si segue la stessa procedura, ma bisogna ricordarsi di avere i file `.orig.tar.gz`, `.dsc` e `.diff.gz` nella directory di lavoro e bisogna usare `Sources.gz` al posto di `Packages.gz`. Anche il programma da usare è diverso: per questa operazione va utilizzato `dpkg-scansources`. La riga di comando sarà simile a questa:

```
# dpkg-scansources debs | gzip > debs/Sources.gz
```

Notare che `dpkg-scansources` non necessita del file di `override`. Infine aggiungere a `/etc/apt/source.list`:

```
deb-src file:/root debs/
```

2.3 Scegliere il miglior mirror da inserire in sources.list: netselect, netselect-apt

Un dubbio molto frequente, in particolare per nuovi utenti, è: "quale mirror inserire in `sources.list`?". Ci sono molti modi per deciderlo. Gli utenti esperti probabilmente hanno uno script che misura il tempo di ping verso svariati mirror. Ma c'è un programma che fa questo per noi: `netselect`.

Per installare netselect, generalmente si esegue:

```
# apt-get install netselect
```

Eseguendo netselect senza alcun argomento viene mostrato un breve elenco delle opzioni. Eseguendolo con una lista di host separati da uno spazio come argomento visualizzerà un punteggio e il nome di uno degli host. Il punteggio prende in considerazione il tempo di ping e gli hop (numero di nodi della rete che si deve attraversare per raggiungere la destinazione) ed è inversamente proporzionale alla velocità di download stimata (il più basso è il migliore). Il nome mostrato è quello dell'host che ha il punteggio più basso (la lista completa dei punteggi si ottiene usando l'opzione -vv). Osservare questo esempio:

```
# netselect ftp.debian.org http.us.debian.org ftp.at.debian.org download.unes
  365 ftp.debian.org.br
#
```

Significa che, fra i mirror indicati nella riga di comando, `ftp.debian.org.br` era il migliore, con un punteggio di 365. (Attenzione!! Questo risultato è stato ottenuto su un certo computer

e per un certo tipo d'accesso alla rete, quindi questo valore non è assolutamente di validità generale).

Ora è necessario aggiungere il mirror più veloce nel file `/etc/apt/sources.list` (vedere 'Il file /etc/apt/sources.list' nella pagina 3) e seguire i suggerimenti in 'Gestione dei pacchetti' nella pagina 9.

Nota: la lista completa dei mirror Debian si può trovare a questo indirizzo `http://www.debian.org/mirror/mirrors_full`.

Dalla versione 0.3.ds1 il pacchetto sorgente netselect include il pacchetto binario **netselect-apt** che esegue automaticamente il processo descritto sopra. È sufficiente passare come parametro il ramo della distribuzione (stable è il valore predefinito) e verrà generato un nuovo `sources.list` con i mirror migliori per main e per non-US nella directory attuale. Il prossimo esempio genera un `sources.list` per la distribuzione stable:

```
# ls sources.list
ls: sources.list: File or directory not found
# netselect-apt stable
(...)
# ls -l sources.list
sources.list
#
```

Da ricordare: il file `sources.list` è creato nella directory attuale e deve essere spostato nella directory `/etc/apt`.

Infine, seguite i suggerimenti in 'Gestione dei pacchetti' nella pagina 9.

2.4 Aggiungere un CD-ROM al file sources.list

Se si vuole usare un CD-ROM per installare i pacchetti o per aggiornare il sistema automaticamente con APT, va inserito nel file `sources.list`. Per farlo si può usare il programma `apt-cdrom` in questo modo:

```
# apt-cdrom add
```

con un CD-ROM di Debian nel lettore. Questa istruzione eseguirà il mount del CD-ROM e, se il CD è valido, verranno cercate le informazioni sui pacchetti che sono sul disco. Se la configurazione del CD-ROM è un po' inusuale, si possono usare le seguenti opzioni:

```
-h           - Questo aiuto
-d directory - Mount point del CDROM
-r           - Rinomina un CDROM già riconosciuto
-m           - Nessun montaggio
-f           - Modalità veloce, non controlla i file dei pacchetti
-a           - Scansione in modalità accurata
```

Per esempio:

```
# apt-cdrom -d /home/kov/mycdrom add
```

Si può controllare l'identità di un CD-ROM, senza inserirlo nella lista:

```
# apt-cdrom ident
```

Notare che questo programma funziona solo se il CD-ROM è correttamente configurato in `/etc/fstab`.

Capitolo 3

Gestione dei pacchetti

3.1 Aggiornare la lista dei pacchetti disponibili

Il sistema di gestione dei pacchetti usa un suo database per tenere traccia di quali pacchetti sono installati, quali non lo sono e quali sono disponibili per l'installazione. Il programma `apt-get` usa questo database per capire come installare i pacchetti richiesti dall'utente e per scoprire quali ulteriori pacchetti sono necessari per farli funzionare correttamente.

Per aggiornare questo elenco, va usato il comando `apt-get update`. Questo comando controlla le liste dei pacchetti presenti negli archivi elencati in `/etc/apt/sources.list`; vedere 'Il file /etc/apt/sources.list' nella pagina 3 ulteriori informazioni su questo file.

È una buona idea eseguire questo comando regolarmente per avere un sistema informato su possibili aggiornamenti dei pacchetti, in particolare sugli aggiornamenti di sicurezza.

3.2 Installazione dei pacchetti

Con il file sources.list pronto e l'elenco dei pacchetti disponibili aggiornato, tutto quello che è necessario fare per installare un pacchetto è eseguire `apt-get`. Per esempio, si può lanciare:

```
# apt-get install xchat
```

APT cercherà nel suo database la versione più recente del pacchetto e lo recupererà, come specificato in `sources.list`, dal corrispondente archivio. Nel caso che il pacchetto dipenda da altri, come nel caso dell'esempio, APT controllerà le dipendenze e installerà i pacchetti necessari, come nell'esempio successivo:

```
# apt-get install nautilus
Reading Package Lists... Done
Building Dependency Tree... Done
```

```
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 0 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 17.2MB will be used.
Do you want to continue? [Y/n]
```

Il pacchetto `nautilus` dipende dalle librerie citate sopra, quindi per prima cosa APT preleverà queste dall'archivio. Se si era specificato sulla riga di comando di `apt-get` il nome di queste librerie, APT non avrebbe chiesto conferma per continuare, infatti avrebbe installato esattamente ciò che viene chiesto dall'utente.

Questo significa che APT chiede la conferma solo quando è necessario installare pacchetti che non sono specificati sulla riga di comando.

Le seguenti opzioni di apt-get possono essere utili:

```
-h  Questo aiuto
-d  Solamente download - NON installa o decomprime gli archivi
-f  Tenta di continuare se il controllo di integrità fallisce
-s  Nessuna azione. Simula i passi
-y  Assume sì a tutte le domande e non chiede conferma
-u  Mostra anche una lista dei pacchetti da aggiornare
```

Sulla riga di comando si possono specificare più pacchetti. I file scaricati dalla rete sono riposti nella directory `/var/cache/apt/archives` per la successiva installazione.

Sulla medesima riga di comando si possono specificare anche i pacchetti da rimuovere, è sufficiente mettere "-" immediatamente dopo il nome del pacchetto che dev'essere rimosso, come in quest'esempio:

```
# apt-get install nautilus gnome-panel-
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Fare riferimento a 'Rimozione dei pacchetti' a fronte per altri dettagli sulla rimozione dei pacchetti.

Se in qualche modo si danneggia un pacchetto già installato, o semplicemente si vuole che i file di un pacchetto siano rimpiazzati da quelli della versione più recente che è disponibile, si può usare l'opzione `--reinstall`, in questo modo:

```
# apt-get --reinstall install gdm
Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 1 reinstalled, 0 to remove and 1  not
Need to get 0B/182kB of archives. After unpacking 0B will be used.
Do you want to continue? [Y/n]
```

3.3 Rimozione dei pacchetti

Se non si vuole più usare un pacchetto, lo si può rimuovere dal sistema usando APT. Per farlo eseguire: `apt-get remove package`. Per esempio:

```
# apt-get remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets gnome-panel gnome-panel-data gnome-session
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Come si può vedere nell'esempio sopra, APT si occupa di rimuovere anche i pacchetti che dipendono da quello che si è chiesto di rimuovere. Non c'è modo, usando APT, di rimuovere un pacchetto senza che siano rimossi anche i pacchetti che dipendono da questo.

Eseguendo `apt-get` come sopra, saranno rimossi i pacchetti ma i loro file di configurazione, se ci sono, rimarranno intatti sul sistema. Per una rimozione completa del pacchetto, eseguire:

```
# apt-get --purge remove gnome-panel
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
0 packages upgraded, 0 newly installed, 4 to remove and 1  not upgraded.
Need to get 0B of archives. After unpacking 14.6MB will be freed.
Do you want to continue? [Y/n]
```

Notare gli "*" dopo i nomi. Indicano che anche i file di configurazione di questi pacchetti saranno rimossi.

Come nel caso del metodo `install`, si può usare un simbolo per invertire il significato del metodo per un particolare pacchetto. Nel caso di `remove` se si aggiunge "+" alla fine del nome del pacchetto, il pacchetto sarà installato anziché essere rimosso.

```
# apt-get --purge remove gnome-panel nautilus+
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
The following packages will be REMOVED:
  gnome-applets* gnome-panel* gnome-panel-data* gnome-session*
The following NEW packages will be installed:
  bonobo libmedusa0 libnautilus0 nautilus
0 packages upgraded, 4 newly installed, 4 to remove and 1  not upgraded.
Need to get 8329kB of archives. After unpacking 2594kB will be used.
Do you want to continue? [Y/n]
```

Notare che `apt-get` elenca i pacchetti extra che saranno installati (cioè i pacchetti di cui è necessaria l'installazione per il corretto funzionamento dei pacchetti richiesti), i pacchetti che saranno rimossi e i nuovi pacchetti che saranno installati (compresi, ancora una volta, i pacchetti extra).

3.4 Aggiornamento dei pacchetti

Gli aggiornamenti dei pacchetti sono un grande successo di APT. Possono essere realizzati con un singolo comando: `apt-get upgrade`. Si può usare questo comando sia per aggiornare i pacchetti della stessa distribuzione, sia per aggiornare a una nuova distribuzione, benché per questo scopo si consiglia: `apt-get dist-upgrade`; vedere 'Aggiornare a una nuova release' nella pagina successiva per ulteriori dettagli.

È utile eseguire questo comando con l'opzione `-u`. Questa opzione forza APT a mostrare la lista completa dei pacchetti che saranno aggiornati, senza l'aggiornamento avverrà senza mostrare nulla. APT scaricherà la versione più recente di ogni pacchetto e li installerà nel giusto ordine. È importante eseguire sempre `apt-get update` prima di provare l'aggiornamento. A questo proposito consultare 'Aggiornare la lista dei pacchetti disponibili' nella pagina 9. Osservare questo esempio:

```
# apt-get -u upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages have been kept back
  cpp gcc lilo
The following packages will be upgraded
  adduser ae apt autoconf debhelper dpkg-dev esound esound-common ftp indent
```

```
  ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0 libesd0-dev
  libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev liborbit0
  libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit procps psmisc
29 packages upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 5055B/5055kB of archives. After unpacking 1161kB will be used.
Do you want to continue? [Y/n]
```

Il procedimento è molto semplice. Notare che nelle prime righe, `apt-get` informa che alcuni pacchetti saranno trattenuti (`kept back`). Questo significa che esistono nuove versioni di questi pacchetti ma che non saranno installate per qualche ragione. Motivi possibili sono la presenza di dipendenze che non possono essere soddisfatte (uno dei pacchetti da cui dipende non è disponibile per il download) o nuove dipendenze (il pacchetto ha aggiunto una nuova dipendenza nell'ultima versione).

Non c'è una soluzione semplice nel primo caso. Nel secondo caso è sufficiente lanciare `apt-get install` per il pacchetto in questione e le dipendenze saranno risolte. Una soluzione altrettanto valida è usare `dist-upgrade`. Controllare in 'Aggiornare a una nuova release' in questa pagina.

3.5 Aggiornare a una nuova release

Questa caratteristica di APT permette di aggiornare l'intero sistema in una volta sola, sia attraverso internet che attraverso un CD (acquistato o scaricato come immagine ISO).

È usata anche quando ci sono delle modifiche alle relazioni fra i pacchetti installati. Con `apt-get upgrade`, questi pacchetti saranno lasciati intoccati (`kept back`).

Per esempio, supponiamo che si stia usando la release 0 della versione stable di Debian e che abbiamo comprato un CD con la release 3. Si può usare APT per aggiornare il sistema da questo nuovo CD. Si usa `apt-cdrom` (guardare 'Aggiungere un CD-ROM al file sources.list' nella pagina 6) per aggiungere il CD al file `/etc/apt/sources.list` e poi si esegue `apt-get dist-upgrade`.

È importante notare che APT cerca sempre la versione più recente dei pacchetti. Quindi, se nel file `/etc/apt/sources.list` è elencato un archivio che contiene una versione di un pacchetto più recente di quella sul CD, APT vorrà scaricare il pacchetto da quell'archivio.

Nell'esempio mostrato in 'Aggiornamento dei pacchetti' nella pagina precedente, abbiamo visto che alcuni pacchetti possono essere trattenuti (`kept back`). Adesso questo problema verrà risolto con il metodo `dist-upgrade`:

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following NEW packages will be installed:
```

```
  cpp-2.95 cron exim gcc-2.95 libident libopenldap-runtime libopenldap1
  libpcre2 logrotate mailx
The following packages have been kept back
  lilo
The following packages will be upgraded
  adduser ae apt autoconf cpp debhelper dpkg-dev esound esound-common ftp gcc
  indent ipchains isapnptools libaudiofile-dev libaudiofile0 libesd0
  libesd0-dev libgtk1.2 libgtk1.2-dev liblockfile1 libnewt0 liborbit-dev
  liborbit0 libstdc++2.10-glibc2.2 libtiff3g libtiff3g-dev modconf orbit
  procps psmisc
31 packages upgraded, 10 newly installed, 0 to remove and 1 not upgraded.
Need to get 0B/7098kB of archives. After unpacking 3118kB will be used.
Do you want to continue? [Y/n]
```

Adesso va osservato che alcuni pacchetti saranno aggiornati e altri nuovi saranno installati (le nuove dipendenze dei pacchetti già installati). Notare che `lilo` continua a essere trattenuto, probabilmente ha qualche altro problema che è più serio di una nuova dipendenza. Lo si può scoprire eseguendo:

```
# apt-get -u install lilo
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be REMOVED:
  debconf-tiny
The following NEW packages will be installed:
  cron debconf exim libident libopenldap-runtime libopenldap1 libpcre2
  logrotate mailx
The following packages will be upgraded
  lilo
1 packages upgraded, 9 newly installed, 1 to remove and 31 not upgraded.
Need to get 225kB/1179kB of archives. After unpacking 2659kB will be used.
Do you want to continue? [Y/n]
```

Come si può notare sopra, la nuova versione di `lilo` è in conflitto con `debconf-tiny`, questo significa che non può essere installato (o aggiornato) senza rimuovere `debconf-tiny`.

Per sapere quali pacchetti saranno rimossi o non-aggiornati si può usare:

```
# apt-get -o Debug::pkgProblemResolver=yes dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Starting
```

```
Starting 2
Investigating python1.5
Package python1.5 has broken dep on python1.5-base
  Considering python1.5-base 0 as a solution to python1.5 0
  Holding Back python1.5 rather than change python1.5-base
Investigating python1.5-dev
Package python1.5-dev has broken dep on python1.5
  Considering python1.5 0 as a solution to python1.5-dev 0
  Holding Back python1.5-dev rather than change python1.5
 Try to Re-Instate python1.5-dev
Done
Done
The following packages have been kept back
  gs python1.5-dev
0 packages upgraded, 0 newly installed, 0 to remove and 2  not upgraded.
```

In questo modo è facile vedere che il pacchetto python1.5-dev non può essere installato perché c'è una dipendenza non soddisfatta: python1.5.

3.6 Rimozione dei file dei pacchetti: apt-get clean e autoclean

Quando si installa un pacchetto APT recupera tutti i file necessari delle sorgenti specificate in /etc/apt/sources.list, gli immagazzina in un archivio locale (`/var/cache/apt/archives/`) e poi procede con l'installazione. Vedere 'Installazione dei pacchetti' nella pagina 9.

Con il tempo l'archivio locale cresce e può occupare molto spazio sul disco. Fortunatamente APT fornisce gli strumenti per la gestione di questo archivio: i metodi `clean` e `autoclean` di `apt-get`.

`apt-get clean` elimina qualunque cosa, tranne i file lock, dalle directory `/var/cache/apt/archives/` e `/var/cache/apt/archives/partial/`. Di conseguenza, se si volesse reinstallare un pacchetto APT dovrà nuovamente scaricarlo.

`apt-get autoclean` cancella solo i file dei pacchetti che non possono più essere scaricati.

Il prossimo esempio mostra come lavora apt-get autoclean:

```
# ls /var/cache/apt/archives/logrotate* /var/cache/apt/archives/gpm*
logrotate_3.5.9-7_i386.deb
logrotate_3.5.9-8_i386.deb
gpm_1.19.6-11_i386.deb
```

Nella directory /var/cache/apt/archives ci sono due file che forniscono il pacchetto `logrotate` e uno che fornisce il pacchetto `gpm`.

```
# apt-show-versions -p logrotate
logrotate/stable uptodate 3.5.9-8
# apt-show-versions -p gpm
gpm/stable upgradeable from 1.19.6-11 to 1.19.6-12
```

`apt-show-versions` mostra che `logrotate_3.5.9-8_i386.deb` fornisce la versione aggiornata di `logrotate`, quindi il file `logrotate_3.5.9-7_i386.deb` è inutile. Anche il file `gpm_1.19.6-11_i386.deb` è inutile perché è disponibile una versione più recente.

```
# apt-get autoclean
Reading Package Lists... Done
Building Dependency Tree... Done
Del gpm 1.19.6-11 [145kB]
Del logrotate 3.5.9-7 [26.5kB]
```

Infine, `apt-get autoclean`, rimuove solo i file inutili. Vedere 'Come aggiornare i pacchetti di una specifica distribuzione Debian' nella pagina 18 per ulteriori informazioni su apt-show-versions.

3.7 Usare APT insieme a dselect

`dselect` è un programma che aiuta gli utenti a scegliere quali pacchetti installare. È ritenuto piuttosto complicato e abbastanza noioso, ma con la pratica si può acquisire familiarità con la sua interfaccia basata su ncurses.

Una caratteristica di dselect è che sa usare la capacità che hanno i pacchetti Debian di "raccomandare" e "consigliare" l'installazione di altri pacchetti. Per avviare il programma, usare `dselect` da root, scegliere "apt" come metodo di accesso (non è obbligatorio, ma se non si sta usando un CD-ROM e si vuole scaricare i pacchetti da internet, è il miglior modo d'usare dselect).

Per approfondire l'uso di dselect, leggere la documentazione sul sito Debian `http://www.debian.org/doc/ddp`.

Dopo aver fatto le scelte con dselect, usare:

```
# apt-get -u dselect-upgrade
```

come nell'esempio sotto:

```
# apt-get -u dselect-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
The following packages will be REMOVED:
```

```
  lbxproxy
The following NEW packages will be installed:
  bonobo console-tools-libs cpp-3.0 enscript expat fingerd gcc-3.0
  gcc-3.0-base icepref klogd libdigest-md5-perl libfnlib0 libft-perl
  libgc5-dev libgcc300 libhtml-clean-perl libltdl0-dev libsasl-modules
  libstdc++3.0 metamail nethack proftpd-doc psfontmgr python-newt talk tidy
  util-linux-locales vacation xbill xplanet-images
The following packages will be upgraded
  debian-policy
1 packages upgraded, 30 newly installed, 1 to remove and 0  not upgraded.
Need to get 7140kB of archives. After unpacking 16.3MB will be used.
Do you want to continue? [Y/n]
```

Questo è quello che si ottiene esguendo `apt-get dist-upgrade` sullo stesso sistema:

```
# apt-get -u dist-upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
Calculating Upgrade... Done
The following packages will be upgraded
  debian-policy
1 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
Need to get 421kB of archives. After unpacking 25.6kB will be freed.
Do you want to continue? [Y/n]
```

Notare che molti dei pacchetti elencati sopra sono installati perché sono "consigliati" o "raccomandati" da altri pacchetti. Altri sono installati o rimossi (nell'esempio lbxproxy) per le scelte che sono state fatte nella lista dei pacchetti con dselect. Dselect può essere uno strumento davvero potente se usato insieme ad APT.

3.8 Come mantenere un sistema misto

Alcune volte le persone vogliono usare una delle versioni Debian come distribuzione per il sistema principale e uno o più pacchetti di un altro ramo.

Per impostare qual è la versione di Debian principale è necessario modificare la seguente riga del file `/etc/apt/apt.conf`:

```
APT::Default-Release "versione";
```

Dove *versione* è la versione di Debian che si desidera usare come distribuzione principale. Le versioni che si possono usare sono `stable`, `testing` e `unstable`. Poi, per installare pacchetti di un'altra versione, si dovrà usare APT nel seguente modo:

```
# apt-get -t distribuzione install pacchetto
```

Per funzionare è necessario specificare nel file `/etc/apt/sources.list` fra le sorgenti per APT almeno una riga per la distribuzione da cui si vuole prelevare il pacchetto e che il pacchetto sia disponibile tramite quella fonte.

È anche possibile richiedere l'installazione di una specifica versione di un pacchetto usando la seguente sintassi:

```
# apt-get install pacchetto=versione
```

Per esempio, con la prossima riga è possibile installare la versione `2.2.4-1` del pacchetto `nautilus`:

```
# apt-get install nautilus=2.2.4-1
```

IMPORTANTE: la versione "unstable" è la versione di Debian che contiene i nuovi pacchetti e i più recenti aggiornamenti. Questa distribuzione segue tutti i cambiamenti dei pacchetti, dai più piccoli ai più drastici che possono influenzare molti altri pacchetti o l'intero sistema. Per questo motivo, questa versione della distribuzione *non* dovrebbe essere usata da utenti inesperti o da chi necessita di una collaudata stabilità.

La distribuzione "testing" non è necessariamente migliore di "unstable", dato che non riceve rapidamente gli aggiornamenti di sicurezza. Per i server e per i sistemi di produzione si dovrebbe usare sempre la distribuzione "stable".

3.9 Come aggiornare i pacchetti di una specifica distribuzione Debian

`apt-show-versions` fornisce agli utenti delle distribuzioni miste un modo sicuro per aggiornare i loro sistemi senza ottenere poi una distribuzione meno stabile di quella che hanno in mente. Per esempio, è possibile aggiornare solo i pacchetti di unstable eseguendo dopo aver installato il pacchetto `apt-show-versions`:

```
# apt-get install `apt-show-versions -u -b | grep unstable | cut -d ' ' -f 1`
```

3.10 Come bloccare una specifica versione di un pacchetto installata (i pin)

Può capitare di modificare qualcosa in un pacchetto e che non si abbia voglia o tempo, di apportare questi cambiamenti a una nuova versione del programma. Oppure, per esempio,

si vuole aggiornare la distribuzione alla 3.0, ma si vuole continuare a usare alcuni pacchetti nella versione presente nella Debian 2.2. Per far questo si può "fissare" questi pacchetti nella versione installata in modo che non saranno aggiornati.

Usare questo mezzo è semplice. Basta solo modificare il file `/etc/apt/preferences`.

Il formato è semplice:

```
Package: <pacchetto>
Pin: <definizione del pin>
Pin-Priority: <priorità del pin>
```

Ogni elemento deve essere separato dagli altri con una riga vuota. Per esempio, per mantenere il pacchetto `sylpheed` che si è modificato per usare "reply-to-list" nella versione 0.4.99 si aggiunge:

```
Package: sylpheed
Pin: version 0.4.99*
```

Notare che si è usato un `*` (asterisco). Questo è un "carattere jolly"; indica che questo "pin" è valido per tutte le versioni che iniziano con 0.4.99. Questo perché Debian numera i suoi pacchetti usando un "numero di revisione" e non si vuole impedire l'installazione delle revisioni successive. Così, per esempio, le revisioni 0.4.99-1 e 0.4.99-10 saranno installate non appena sono disponibili. Notare che se si è modificato il pacchetto non si vuole che questo accada.

La priorità del pin determina se un pacchetto che verifica le righe "Packages:" e "Pin:" verrà installato; più è alta la priorità e maggiori sono le possibilità che il pacchetto sia installato. Si può leggere `apt_preferences(7)` per una profonda discussione sulle priorità, ma alcuni esempi possono rendere l'idea. Di seguito sono descritti gli effetti di diversi valori del campo priorità per l'esempio precedente.

1001 La versione 0.4.99 di sylpheed non sarà mai sostituita da apt. Se disponibile apt installerà la versione 0.4.99 anche sostituendo una versione successiva del pacchetto già installata. Solo i pacchetti con priorità superiore a 1000 possono far retrocedere un pacchetto già installato.

1000 L'effetto è lo stesso della priorità con l'eccezione che apt si rifiuta di retrocedere la versione installata alla versione 0.4.99.

990 La versione 0.4.99 verrà sostituita solo da una versione successiva disponibile per la release impostata come preferita tramite la variabile "APT::Default-Release" (vedere 'Come mantenere un sistema misto' nella pagina 17).

500 Qualsiasi versione successiva alla 0.4.99 di sylpheed disponibile da una qualunque release avrà precedenza sulla versione 0.4.99, ma la 0.4.99 ha comunque la precedenza su tutte le versioni precedenti.

100 Le versioni successive di sylpheed disponibili da qualsiasi release hanno precedenza sulla versione 0.4.99 di conseguenza verrà installata qualsiasi versione successiva di sylpheed, la versione 0.4.99 verrà installata solo se non c'è una versine già installata. Questa è la priorità dei pacchetti una volta che sono installati.

-1 Sono consentiti anche valori negativi per la priorità, impediscono che 0.4.99 sia installato.

Un pin può specificare la versione (`version`), la release (`release`) o l'origine (`origin`) di un pacchetto.

Il pin su `version`, come abbiamo visto, ammette sia cifre che caratteri jolly per specificare più versioni contemporaneamente.

L'opzione `release` dipende dal file Release nell'archivio usato da APT o nel CD. Questa opzione non può essere usata se l'archivio che si usa non fornisce questo file. Si può controllare il contenuto dei file Release in `/var/lib/apt/lists/`. I parametri per release sono: `a` (archive), `c` (components), `v` (version), `o` (origin) e `l` (label).

Un esempio:

```
Package: *
Pin: release v=2.2*,a=stable,c=main,o=Debian,l=Debian
Pin-Priority: 1001
```

In questo esempio abbiamo scelto la versione 2.2* (che può essere 2.2r2, 2.2r3; questi sono i "point release" che tipicamente includono le fix di sicurezza e altri importanti aggiornamenti), la versione `stable`, la sezione `main` (al posto di `contrib` o `non-free`) e come origine e etichetta Debian. Origine (o) definisce il nome della distribuzione: Debian per la Debian e Progeny per Progeny, per esempio. Un esempio del file Release:

```
$ cat /var/lib/apt/lists/ftp.debian.org.br_debian_dists_potato_main_binary-i3
Archive: stable
Version: 2.2r3
Component: main
Origin: Debian
Label: Debian
Architecture: i386
```

Capitolo 4

Aiuti molto utili

4.1 Come installare i pacchetti compilati in proprio: equivs

Qualche volta si vuole usare una specifica versione di un programma disponibile solo tramite i sorgenti, senza alcun pacchetto Debian. Ma per il sistema di gestione dei pacchetti questo può essere un problema. Supponiamo che si voglia compilare una nuova versione del server e-mail. Non ci sarebbero problemi, ma molti pacchetti di Debian dipendono da un MTA (Mail Transport Agent) e, quando si installa qualcosa che abbiamo compilato in proprio, il sistema di gestione dei pacchetti non può sapere nulla del nuovo software.

Qui entra in scena `equivs`. Per usarlo installare il pacchetto omonimo. Equivs crea un pacchetto vuoto che è capace di soddisfare qualunque dipendenza, facendo credere al sistema di gestione dei pacchetti che le dipendenze sono soddisfatte.

Prima di iniziare è bene ricordare che esistono modi più sicuri per compilare un programma già impacchettato per Debian con opzioni differenti e che non si dovrebbe usare equivs per rimpiazzare le dipendenze se non sa quali possono essere le conseguenze sul sistema. Vedere 'Lavorare con i pacchetti sorgente' nella pagina 31 per ulteriori informazioni.

Continuando con l'esempio del MTA: si è appena installato il neo-compilato `postfix` e ora si vuole installare `mutt`, però `mutt` richiede di installare un altro MTA. Ma ce ne già uno installato.

Cambiare directory (`/tmp`, per esempio) ed eseguire:

```
# equivs-control nome
```

Mettere al posto di *name* il nome del file di controllo da creare. Il file che sarà creato è come quello che segue:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1
```

```
Package: <enter package name; defaults to equivs-dummy>
Version: <enter version here; defaults to 1.0>
Maintainer: <your name and email address; defaults to username>
Pre-Depends: <packages>
Depends: <packages>
Recommends: <packages>
Suggests: <package>
Provides: <(virtual)package>
Architecture: all
Copyright: <copyright file; defaults to GPL2>
Changelog: <changelog file; defaults to a generic changelog>
Readme: <README.Debian file; defaults to a generic one>
Extra-Files: <additional files for the doc directory, comma-seperated>
Description: <short description; defaults to some wise words>
 long description and info
 .
 second paragraph
```

Ora è necessario modificarlo in modo che faccia quello che si vuole. Qui verrà mostrato il formato dei soli campi necessari:

```
Section: misc
Priority: optional
Standards-Version: 3.0.1

Package: mta-local
Provides: mail-transport-agent
```

`mutt` dipende da `mail-transport-agent`, che è un pacchetto virtuale fornito da tutti i MTA, si potrebbe semplicemente nominare il pacchetto `mail-transport-agent`, ma è preferibile usare lo schema dei pacchetti virtuali tramite "Provides".

Ora serve solo creare il pacchetto:

```
# equivs-build name
dh_tstdir
touch build-stamp
dh_testdir
dh_testroot
dh_clean -k
# Add here commands to install the package into debian/tmp.
touch install-stamp
dh_testdir
dh_testroot
```

```
dh_installdocs
dh_installchangelogs
dh_compress
dh_fixperms
dh_installdeb
dh_gencontrol
dh_md5sums
dh_builddeb
dpkg-deb: building package 'name' in '../name_1.0_all.deb'.

The package has been created.
Attention, the package has been created in the current directory,
```

e installare il `.deb` finale.

Come si può vedere ci sono molti usi di `equivs`. Per esempio, si può creare un pacchetto `my-fovorites`, che dipende dai programmi che abitualmente si installano, per esempio. Si ha la più completa libertà di scelta, ma si deve valutare attentamente quello che si ha intenzione di fare.

Infine notare che ci sono vari esempi di file di controllo in `/usr/share/doc/equivs/example`.

4.2 Rimuovere i file di localizzazione inutilizzati: localepurge

Molti utenti Debian usano una sola impostazione internazionale. Un utente brasiliano, per esempio, generalmente usa *locale* settato come `pt_BR` tutto il tempo e non gli interessa avere a disposizione anche `es`.

`localepurge` è uno strumento molto utile per questi utenti, infatti possono liberare molto spazio mantenendo solo i file di localizzazione che realmente usano. Per installare localepurge: `apt-get install localepurge`.

Questo pacchetto è veramente semplice da configurare dato che debconf guida l'utente passo dopo passo nella sua configurazione. Tuttavia si deve porre attenzione alle risposte delle prime domande, che, se non corrette, potrebbero causare la rimozione di tutti i file di localizzazione, compresi quelli che si indendeva mantenere. In questo caso, l'unico modo per ripristinare questi file è reinstallare i pacchetti che li forniscono.

4.3 Come sapere quali pacchetti potrebbero essere aggiornati

`apt-show-versions` è un programma che mostra quali pacchetti del sistema potrebbero essere aggiornati e altre utili informazioni. L'opzione `-u` visualizza un elenco dei pacchetti aggiornabili:

```
$ apt-show-versions -u
libeel0/unstable upgradeable from 1.0.2-5 to 1.0.2-7
libeel-data/unstable upgradeable from 1.0.2-5 to 1.0.2-7
```

Capitolo 5

Ottenere informazioni sui pacchetti

Ci sono vari front-end per APT che rendono più semplice avere la lista dei pacchetti disponibili per l'installazione o di quelli già installati, così come scoprire a quale sezione appartiene un pacchetto, qual è la sua priorità, qual è la sua descrizione, ecc.

Ma lo scopo è imparare ad usare APT. Come si può conoscere il nome di un pacchetto che si desidera installare?

Sono disponibili alcuni programmi ognuno con il suo specifico compito. Inizieremo con `apt-cache`. Questo programma è usato da APT per gestire il suo database. Faremo un breve riassunto sulle sue applicazioni più comuni.

5.1 Avere informazioni sui nomi dei pacchetti

Per esempio, supponiamo che si voglia ricordare i bei vecchi tempi dell'Atari 2600. Si vuole usare APT per installare un emulatore dell'Atari e per scaricare qualche gioco. Si può fare così:

```
# apt-cache search atari
atari-fdisk-cross - Partition editor for Atari (running on non-Atari)
circuslinux - The clowns are trying to pop balloons to score points!
madbomber - A Kaboom! clone
tcs - Character set translator.
atari800 - Atari emulator for svgalib/X/curses
stella - Atari 2600 Emulator for X windows
xmess-x - X binaries for Multi-Emulator Super System
```

Sono presenti parecchi pacchetti che riguardano quello che si stava cercando, e tutti hanno una breve descrizione. Per avere più informazioni su un pacchetto, si può usare:

```
# apt-cache show stella
Package: stella
```

```
Priority: extra
Section: non-free/otherosfs
Installed-Size: 830
Maintainer: Tom Lear <tom@trap.mtview.ca.us>
Architecture: i386
Version: 1.1-2
Depends: libc6 (>= 2.1), libstdc++2.10, xlib6g (>= 3.3.5-1)
Filename: dists/potato/non-free/binary-i386/otherosfs/stella_1.1-2.deb
Size: 483430
MD5sum: 11b3e86a41a60fa1c4b334dd96c1d4b5
Description: Atari 2600 Emulator for X windows
 Stella is a portable emulator of the old Atari 2600 video-game console
 written in C++.  You can play most Atari 2600 games with it.  The latest
 news, code and binaries for Stella can be found at:
 http://www4.ncsu.edu/~bwmott/2600
```

Questo è l'output con molti più dettagli sul pacchetto che si desidera (o non si desidera) installare. Se il pacchetto è già installato sul sistema e ne esiste una nuova versione saranno mostrate le informazioni su entrambe le versioni. Per esempio:

```
# apt-cache show lilo
Package: lilo
Priority: important
Section: base
Installed-Size: 271
Maintainer: Russell Coker <russell@coker.com.au>
Architecture: i386
Version: 1:21.7-3
Depends: libc6 (>= 2.2.1-2), debconf (>=0.2.26), logrotate
Suggests: lilo-doc
Conflicts: manpages (<<1.29-3)
Filename: pool/main/l/lilo/lilo_21.7-3_i386.deb
Size: 143052
MD5sum: 63fe29b5317fe34ed8ec3ae955f8270e
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.

Package: lilo
Status: install ok installed
Priority: important
Section: base
```

```
Installed-Size: 190
Maintainer: Vincent Renardias <vincent@debian.org>
Version: 1:21.4.3-2
Depends: libc6 (>= 2.1.2)
Recommends: mbr
Suggests: lilo-doc
Description: LInux LOader - The Classic OS loader can load Linux and others
 This Package contains lilo (the installer) and boot-record-images to
 install Linux, OS/2, DOS and generic Boot Sectors of other OSes.
 .
 You can use Lilo to manage your Master Boot Record (with a simple text scree
 or call Lilo from other Boot-Loaders to jump-start the Linux kernel.
```

Notare che il primo pacchetto nella lista dell'esempio precedente è la versione disponibile per l'installazione e il secondo è quello già installato. Per avere delle informazioni generali su un pacchetto si può usare:

```
# apt-cache showpkg penguin-command
Package: penguin-command
Versions:
1.4.5-1(/var/lib/apt/lists/download.sourceforge.net_debian_dists_unstable_mai

Reverse Depends:
Dependencies:
1.4.5-1 - libc6 (2 2.2.1-2) libpng2 (0 (null)) libsdl-mixer1.1 (2 1.1.0) libs
Provides:
1.4.5-1 -
Reverse Provides:
```

E per sapere solo quali sono le dipendenze:

```
# apt-cache depends penguin-command
penguin-command
  Depends: libc6
  Depends: libpng2
  Depends: libsdl-mixer1.1
  Depends: libsdl1.1
  Depends: zlib1g
```

Ricapitolando, abbiamo un'ampia gamma di strumenti che possiamo usare per trovare il nome dei pacchetti che vogliamo.

5.2 Usare dpkg per avere informazioni sui pacchetti

Un modo per individuare un pacchetto è conoscere il nome di un file che può essere trovato al suo interno. Per esempio, per trovare il pacchetto che fornisce un particolare ".h" necessario per compilare un programma si può eseguire:

```
# dpkg -S stdio.h
libc6-dev: /usr/include/stdio.h
libc6-dev: /usr/include/bits/stdio.h
perl: /usr/lib/perl/5.6.0/CORE/nostdio.h
```

oppure:

```
# dpkg -S /usr/include/stdio.h
libc6-dev: /usr/include/stdio.h
```

Per avere i nomi dei pacchetti installati nel sistema, utile in previsione di "pulizia" dei dischi, si può lanciare:

```
# dpkg -l | grep mozilla
ii  mozilla-browse 0.9.6-7        Mozilla Web Browser
```

Il problema di questo commando è che può "spezzare" il nome del pacchetto. Nell'esempio sopra il nome intero del pacchetto è `mozilla-browser`, per sistemare questo inconveniente si può usare la variabile d'ambiente `COLUMNS` in questo modo:

```
$ COLUMNS=132 dpkg -l | grep mozilla
ii  mozilla-browser             0.9.6-7                      Mozilla Web Brows
```

o fare una ricerca sulla descrizione o su parte di essa:

```
# apt-cache search "Mozilla Web Browser"
mozilla-browser - Mozilla Web Browser
```

5.3 Come installare pacchetti "on demand"

Si sta compilando un programma e, a un certo punto, boom! C'è un errore perché è necessario un file `.h` che non è installato. Il programma `auto-apt` può essere utile in questa situazione. Questo programma ferma l'esecuzione del processo principale, chiede di installare i pacchetti necessari e, una volta installati, prosegue nell'esecuzione del processo principale.

Quello che si deve fare è lanciare:

```
# auto-apt run command
```

Dove “command” è il comando che deve essere eseguito a cui potrebbero mancare alcuni file. Per esempio:

```
# auto-apt run ./configure
```

Questo chiederà di installare i pacchetti necessari e, se li vogliamo installare, chiama automaticamente apt-get. Gli utenti che usano X possono usare anche un’interfaccia grafica anziché la normale interfaccia testuale.

Per funzionare correttamente auto-apt usa un database che deve essere tenuto aggiornato. Questo può essere fatto usando i commandi `auto-apt update`, `auto-apt updatedb` e `auto-apt update-local`.

5.4 Come scoprire a quale pacchetto appartiene un file

Se si vuole installare un pacchetto, di cui non si riesce a scoprire il nome usando `apt-cache`, ma di cui si conosce il nome del programma o di un file che è contenuto nel pacchetto, si può usare `apt-file` per trovare il nome del pacchetto in questo modo:

```
$ apt-file search file
```

Funziona come `dpkg -S`, con la differenza che la ricerca è effettuata anche nei pacchetti non installati. Può anche essere usato per scoprire quali pacchetti contengono i file mancanti quando si compilano dei programmi, anche se `auto-apt` sia un metodo molto migliore per risolvere questo genere di problemi, vedere ‘Come installare pacchetti “on demand”’ a fronte.

È anche possibile vedere contenuto di un pacchetto eseguendo:

```
$ apt-file list pacchetto
```

`apt-file` tiene un database del contenuto di tutti i pacchetti, come fa auto-apt, e questo dev’essere tenuto aggiornato. Lo si fa eseguendo:

```
# apt-file update
```

`apt-file` usa lo stesso database di `auto-apt`, controllare ‘Come installare pacchetti “on demand”’ nella pagina precedente.

5.5 Come essere informato sui cambiamenti nei pacchetti

Ogni pacchetto installa, nella sua directory dedicata alla documentazione (`/usr/share/doc/pacchetto`), un file chiamato `changelog.Debian.gz` che contiene l'elenco dei cambiamenti fatti al pacchetto sino all'ultima versione. È possibile leggere questo file usando, per esempio, `zless` però non è semplice, dopo un aggiornamento completo del sistema, cercare i changelog di ogni pacchetto che è stato aggiornato.

C'è un modo per automatizzare questo lavoro tramite l'uso di `apt-listchanges`. Per prima cosa si deve installare il pacchetto `apt-listchanges`. Durante l'installazione, debconf chiederà come lo si vuole configurare, anche se alcune domande, in base al livello di priorità impostato in Debconf, potrebbero non essere visualizzate, si consiglia di ricordare i seguenti suggerimenti mentre si risponde.

la prima domanda chiede come apt-listchanges deve essere mostrare le modifiche, è possibile che siano inviate per posta (ideale per gli aggiornamenti automatici) o che siano mostrate in un impaginatore come `less` (in modo da consultarli prima di completare l'aggiornamento). Se non si vuole che `apt-listchanges` sia eseguito automaticamente durante gli aggiornamenti si può rispondere `nessuno`.

Dopo aver installato apt-listchanges, appena i pacchetti sono disponibili per l'installazione (scaricati, presi da un CD, ecc.) apt mostrerà i changelog di questi pacchetti prima di installarli.

Capitolo 6

Lavorare con i pacchetti sorgente

6.1 Scaricare i pacchetti sorgente

Spesso nel mondo del software libero si fanno correzioni ai programmi che contengono dei bug. Per fare questo è necessario scaricare il sorgente del programma: APT fornisce un modo semplice per scaricare i sorgenti di un programma, compresi i file necessari per creare il pacchetto .deb.

Un altro uso comune dei sorgenti è compilare su una distribuzione stable i pacchetti di testing o di unstable dato che, quelli presenti in quest'ultime, sono più recenti. Infatti, compilando un pacchetto su stable, sarà creato un .deb le cui dipendenze sono automaticamente aggiustate per operare in questa distribuzione.

Per poter fare questo dev'essere specificata almeno una riga `deb-src` in `/etc/apt/sources.list` che punti ai pacchetti di testing o di unstable. Vedere 'Il file /etc/apt/sources.list' nella pagina 3.

Per scaricare un pacchetto sorgente, si può usare il seguente comando:

```
$ apt-get source nomepacchetto
```

Questo scaricherà tre file: un `.orig.tar.gz`, un `.dsc` e un `.diff.gz`. Se un pacchetto è fatto specificamente per Debian, l'ultimo fra quelli elencati non è scaricato e al primo della lista mancherà `orig` nel nome.

Il file `.dsc` è usato da dpkg-source per decomprimere il pacchetto sorgente nella directory *nomepacchetto-versione*. Dentro tutti i pacchetti sorgente, una volta scompattati, c'è dentro una directory `debian/` che contiene i file necessari per la costruzione del pacchetto .deb.

Per automatizzare la costruzione del pacchetto, dopo aver scaricato i sorgenti, è sufficiente aggiungere l'opzione `-b` nella riga di comando:

```
$ apt-get -b source nomepacchetto
```

Se invece non si vuole creare il .deb al termine del download, lo si può creare successivamente lanciando:

```
$ dpkg-buildpackage -rfakeroot -uc -b
```

dopo essere entrati nella directory creata al termine del download del pacchetto. Per installare il pacchetto creato con i precedenti comandi si deve usare direttamente il gestore di pacchetti in questo modo:

```
# dpkg -i file.deb
```

C'è una differenza fra il metodo `source` e gli altri messi a disposizione da `apt-get`: tale metodo può essere usato da tutti gli utenti senza che questi abbiano i privilegi di root. I file sono scaricati nella directory dalla quale è stato lanciato il comando `apt-get source pacchetto`.

6.2 Soddisfare le dipendenze per compilare un pacchetto sorgente

Normalmente per compilare un pacchetto sorgente è necessario aver installato header e librerie appropriate. Tutti i pacchetti sorgente hanno nei loro file di controllo un campo chiamato "Build-Depends:" che indica quali pacchetti devono essere installati per poter costruire il pacchetto dai sorgenti.

APT possiede un metodo veramente semplice per scaricare questi pacchetti. È sufficiente lanciare `apt-get build-dep pacchetto`, dove "pacchetto" è il nome del pacchetto che si vuole compilare. Per esempio:

```
# apt-get build-dep gmc
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  comerr-dev e2fslibs-dev gdk-imlib-dev imlib-progs libgnome-dev libgnorba-de
  libgpmg1-dev
0 packages upgraded, 7 newly installed, 0 to remove and 1  not upgraded.
Need to get 1069kB of archives. After unpacking 3514kB will be used.
Do you want to continue? [Y/n]
```

Saranno installati i pacchetti necessari per la corretta compilazione del pacchetto `gmc`. È importante notare che questo comando non recupera i file sorgente del pacchetto che si vuole compilare, per scaricarlo va lanciato prima `apt-get source`.

Se invece si è interessati solo a controllare quali pacchetti sono necessari per compilare un certo pacchetto c'è una variante del comando `apt-cache show` (vedere 'Ottenere informazioni sui pacchetti' nella pagina 25), che vi mostrerà fra le varie informazioni la riga `Build-Depends` in cui sono elencati questi pacchetti.

```
# apt-cache showsrc pacchetto
```

Capitolo 7

Come trattare gli errori

7.1 Errori banali

Gli errori sono sempre accaduti, molti di questi sono causati da utenti che non prestano attenzione. Quella che segue è una breve lista degli errori più frequenti e di come si possono risolvere.

Se viene visualizzato un messaggio d'errore che assomiglia a questo quando si sta provando a installare un pacchetto con `apt-get install package`...

```
Reading Package Lists... Done
Building Dependency Tree... Done
W: Couldn't stat source package list 'http://people.debian.org unstable/ Pack
W: You may want to run apt-get update to correct these missing files
E: Couldn't find package penguineyes
```

vuol dire che non si è lanciato `apt-get update` dopo l'ultima modifica al file `/etc/apt/sources.list`.

Se l'errore assomiglia a questo:

```
E: Could not open lock file /var/lib/dpkg/lock - open (13 Permission denied)
E: Unable to lock the administration directory (/var/lib/dpkg/), are you root
```

quando si tenta di usare `apt-get` con qualunque metodo (tranne `source`), significa che non si hanno i permessi dell'utente root, cioè si sta eseguendo `apt-get` come un normale utente.

C'è anche un errore simile al precedente che avviene quando si tenta di avviare più di un processo `apt-get` o quando `apt-get` è lanciato mentre è già attivo un processo `dpkg`. L'unico modo possibile per avere più processi `apt-get` attivi è usare il metodo `source`.

Se un'installazione si blocca e non è più possibile né installare né rimuovere il pacchetto, provare con questi due comandi:

```
# apt-get -f install
# dpkg --configure -a
```

E poi tentare di nuovo. Potrebbe essere necessario ripetere più volte i comandi precedenti. Questa è una lezione importante per gli avventurieri che usano "unstable".

Se si riceve l'errore "E: Dynamic MMap ran out of room" durante l'esecuzione di `apt-get update`, aggiungete la seguente riga a `/etc/apt/apt.conf`:

```
APT::Cache-Limit 10000000;
```

7.2 Dove posso trovare aiuto?

In caso di dubbi, consultare la vasta documentazione disponibile per il sistema di gestione dei pacchetti Debian. `--help` e le pagine di manuale possono essere d'aiuto, inoltre si può leggere la documentazione contenuta nelle directory `/usr/share/doc` e in particolare `/usr/share/doc/apt`.

Se questa documentazione non è sufficiente, si può provare a cercare la risposta nell'archivio delle mailing list. Altre informazioni sulle mailing list si possono trovare sul sito web di Debian: http://www.debian.org.

Tuttavia, queste mailing list dovrebbero essere usate solo dagli utenti Debian; gli utenti di altri sistemi troveranno un miglior supporto dalle communità che si occupano della propria distribuzione.

Capitolo 8

Quali distribuzioni supportano APT?

Di seguito sono elencati i nomi di alcune distribuzioni che usano APT:

Debian GNU/Linux (http://www.debian.org) - la distribuzione per la quale è stato sviluppato APT

Conectiva (http://www.conectiva.com.br) - è stata la prima distribuzione a fondere insieme APT con rpm

Libranet (http://www.libranet.com)

Mandrake (http://www.mandrake.com)

PLD (http://www.pld.org.pl)

Vine (http://www.vinelinux.org)

APT4RPM (http://apt4rpm.sf.net)

Alt Linux (http://www.altlinux.ru/)

Red Hat (http://www.redhat.com/)

Sun Solaris (http://www.sun.com/)

SuSE (http://www.suse.de/)

Yellow Dog Linux (http://www.yellowdoglinux.com/)

Capitolo 9

Ringraziamenti

Un grande grazie va ai miei amici del progetto Debian-BR e a Debian in sé, che sono un aiuto costante per me e che sempre mi danno la forza per continuare a lavorare per il miglioramento dell'umanità, così come mi aiutano nel mio scopo di salvare il mondo. :)

Voglio ringraziare il CIPSGA per l'enorme aiuto che ha dato al nostro progetto e a tutti i progetti liberi che nascono da grandi idee.

E speciali ringraziamenti vanno a:

Yooseong Yang <yooseong@debian.org>

Michael Bramer <grisu@debian.org>

Bryan Stillwell <bryan@bokeoa.com>

Pawel Tecza <pawel.tecza@poczta.fm>

Hugo Mora <h.mora@melix.com.mx>

Luca Monducci <luca.mo@tiscali.it>

Tomohiro KUBOTA <kubota@debian.org>

Pablo Lorenzzoni <spectra@debian.org>

Steve Langasek <vorlon@netexpress.net>

Arnaldo Carvalho de Melo <acme@conectiva.com.br>

Erik Rossen <rossen@freesurf.ch>

Ross Boylan <RossBoylan@stanfordalumni.org>

Matt Kraai <kraai@debian.org>

Aaron M. Ucko <ucko@debian.org>

Jon Åslund <d98-jas@nada.kth.se>

Capitolo 10

Nuove versioni di questo tutorial

Questo manuale è stato creato dal progetto Debian-BR (http://www.debian-br.org), con lo scopo di agevolare l'uso quotidiano di Debian.

Nuove versioni di questo documento saranno disponibili alla pagina del Progetto di Documentazione Debian http://www.debian.org/doc/ddp.

Commenti e critiche possono essere inviati per email all'autore <kov@debian.org>.