L'installazione delle componenti di un sistema GNU/Linux avviene attraverso dei «pacchetti», rappresentati esteriormente da file singoli. L'installazione di questi pacchetti può essere semplice o complessa, a seconda della circostanza.

7.1 Applicativi originali distribuiti in forma sorgente

La maggior parte dei programmi per sistemi Unix, il cui utilizzo viene concesso gratuitamente, viene distribuita in forma sorgente. Ciò significa che per poterli utilizzare, questi programmi devono essere compilati. Fortunatamente, di norma è disponibile il compilatore GNU C (conforme allo standard) che permette di uniformare questo procedimento di compilazione.

Un programma originale distribuito in forma sorgente si trova di solito confezionato in un archivio il cui nome ha un'estensione '`.tar.gz`' o '`.tgz`' (ottenuto attraverso '**`tar`**' e '**`gzip`**'), oppure ancora con un'estensione '`.tar.bz2`' ('**`tar`**' e '**`bzip2`**'). Prima di poter procedere con la sua compilazione deve essere estratto il suo contenuto. Solitamente si fa questo in una directory di lavoro. Nell'esempio che segue, si fa riferimento a un pacchetto ipotetico archiviato nel file '`pacch.tar.gz`' che si trova nella directory '`~/tmp/`':

```
$ cd ~/tmp [Invio]
$ tar xzvf pacch.tar.gz [Invio]
```

Oppure:

```
$ cat pacch.tar.gz | gunzip | tar xvf - [Invio]
```

Se invece si trattasse dell'archivio '`pacch.tar.bz2`', sarebbe necessario decomprimerlo attraverso un comando leggermente diverso:

```
$ cd ~/tmp [Invio]
$ tar xjvf pacch.tar.bz2 [Invio]
```

Oppure:

```
$ cat pacch.tar.bz2 | bunzip2 | tar xvf - [Invio]
```

Di solito si ottiene una struttura ad albero più o meno articolata in sottodirectory, dove nella directory principale di questa struttura si trovano:

- uno o più file di documentazione ('`README`', '`INSTALL`', ecc.) che servono per ricordare il procedimento corretto per ottenere la compilazione;
- uno o più script preparati per facilitare questo procedimento;
- il file-make (o *makefile*).

Seguendo l'esempio visto poco prima, dovrebbe essere stata creata la directory '`pacch/`'.

```
$ cd pacch [Invio]
```



```
$ ls [Invio]
```

I file di testo che si trovano nella directory principale del pacchetto contenente il programma sorgente, servono per presentare brevemente il programma e per riassumere le istruzioni necessarie alla sua compilazione. Di solito, queste ultime sono contenute nel file '`INSTALL`'. In ogni caso, tutti questi file vanno letti, in particolare quello che spiega il procedimento per la compilazione e l'installazione.

Il modo più semplice per leggere un file è l'utilizzo del programma '**`less`**':

```
$ less INSTALL [Invio]
```

In sua mancanza si può usare '**`more`**':

```
$ more INSTALL [Invio]
```

La composizione classica di un pacchetto distribuito in forma sorgente, prevede la presenza di uno script il cui scopo è quello di costruire un file-make adatto all'ambiente in cui si vuole compilare il programma. Ciò è necessario perché i vari sistemi Unix sono diversi tra loro per tanti piccoli dettagli.

Spesso questo script è in grado di accettare argomenti. Ciò può permettere, per esempio, di definire una directory di destinazione del programma, diversa da quella predefinita.

Quando questo script di preparazione manca, occorre modificare manualmente il file-make in modo che sia predisposto correttamente per la compilazione nel proprio sistema.

Per evitare ambiguità, questo script viene sempre avviato indicando un percorso preciso: '`./configure`'.

Il file-make, o *makefile*, è quel file che viene letto da Make, precisamente dall'eseguibile '**`make`**', allo scopo di coordinare le varie fasi della compilazione ed eventualmente anche per l'installazione del programma compilato. Il nome di questo file può essere diverso, generalmente si tratta di '`Makefile`', oppure di '`makefile`'.

Se manca lo script '`./configure`', o altro equivalente per la creazione automatica del file-make, significa che il file-make esistente deve essere controllato e, se necessario, modificato prima della compilazione.

Spesso è bene controllare il contenuto del file-make anche quando questo è stato generato automaticamente.

7.2 Fasi tipiche di una compilazione e installazione

I pacchetti più comuni si compilano e si installano con tre semplici operazioni; tuttavia, è sempre indispensabile leggere le istruzioni che si trovano nei file di testo distribuiti insieme ai sorgenti dei programmi.

1. `$ ./configure [Invio]`

   Genera automaticamente il file-make.

2. `$ make [Invio]`

   Esegue la compilazione generando i file eseguibili.

3. `# make install [Invio]`

   Installa gli eseguibili e gli altri file necessari nella loro destinazione prevista per il funzionamento: l'ultima fase deve essere eseguita con i privilegi dell'utente '**`root`**'.

I problemi maggiori si possono incontrare derivano dalla mancanza di uno script '`./configure`' o simile, per cui si è costretti a modificare direttamente il file-make. In altri casi, il file-make potrebbe non prevedere la fase di installazione ('**`make install`**'), costringendo

a installare il programma copiando pezzo per pezzo nella destinazione corretta. Inoltre, spesso l'installazione non rispetta la struttura standard del proprio file system. Ciò nel senso che magari vengono piazzati file che devono poter essere modificati, all'interno di una zona che si voleva riservare per l'accesso in sola lettura.

Quando si utilizza una distribuzione GNU ben organizzata, si trova una gestione dei pacchetti installati che permette l'eliminazione e l'aggiornamento di questi senza rischiare di lasciare file inutilizzati in giro. Se si installa un programma distribuito in forma originale, viene a mancare il sostegno della gestione dei pacchetti. In questi casi si cerca di installare tali pacchetti al di sotto della directory '`/opt/`', o almeno al di sotto di '`/usr/local/`'.

7.3 File di differenze, o «patch»

Quando si ha a che fare con programmi il cui aggiornamento è frequente, come avviene nel caso del kernel, si possono anche trovare aggiornamenti in forma di file di differenze (*patch* o «pezze»), cioè di file che contengono solo le variazioni da una versione all'altra. Queste variazioni si applicano ai file di una versione per ottenerne un'altra.

Se la versione da aggiornare è stata espansa a partire dall'ipotetica directory '`~/tmp/`', per applicarvi una modifica è necessario posizionarsi sulla stessa directory e poi eseguire il comando seguente:

```
patch < file_di_differenze
```

Può darsi che la posizione in cui ci si deve trovare sia diversa o che i sorgenti da aggiornare debbano trovarsi in una posizione precisa. Per capirlo, dovrebbe bastare l'osservazione diretta del contenuto del file di differenze.

Spesso, il file di differenze è realizzato considerando che i file da aggiornare si trovino all'interno di una certa directory. Di solito, si raggira il problema usando l'opzione '**`-p`**'. Per esempio, si può immaginare di avere la directory '`/tmp/prova/esempio/`' che contiene qualcosa da aggiornare attraverso il file di differenze '`/tmp/prova/esempio-007.diff`', che fa riferimento a file contenuti nella directory '`prova-007/`' (come percorso relativo). Per fare in modo che l'aggiornamento venga eseguito correttamente, è necessario spostarsi nella directory '`/tmp/prova/esempio/`':

```
$ cd /tmp/prova/esempio [Invio]
```

Quindi si esegue l'aggiornamento ignorando la prima directory del percorso:

```
$ patch -p1 < /tmp/prova/esempio-007.diff [Invio]
```

L'applicazione delle variazioni, può fallire. Se non si vuole perdere il rapporto degli errori, questi possono essere ridiretti in un file specifico.

```
patch < file_di_differenze 2> file_degli_errori
```

Se gli aggiornamenti sono più di uno, occorre applicare le modifiche in sequenza. La sezione 22.12 tratta meglio questo problema.

7.4 Installazione di programmi già compilati

Alcuni programmi vengono distribuiti in forma già compilata (e senza sorgenti) soprattutto quando si tratta di prodotti proprietari, ma anche in questi casi si possono incontrare problemi nell'installazione.[1]

7.4.1 Scelta della piattaforma

Quando si cerca del software per il proprio sistema che può essere ottenuto solo in forma già compilata, occorre fare attenzione alla piattaforma. Infatti, non basta che si tratti di programmi compilati per il proprio sistema operativo, ma occorre che gli eseguibili siano adatti al tipo di elaboratore su cui il sistema operativo è in funzione.

Normalmente, per identificare l'architettura dei «PC» (x86), si utilizza la sigla i386 nel nome dei file degli archivi. Sigle come i486, i586, i686,... rappresentano la stessa architettura basata però su un livello particolare del microprocessore.

7.4.2 Eseguibili e variabili di ambiente

Un programma distribuito in forma binaria, deve essere estratto normalmente dall'archivio compresso che lo contiene. A volte è disponibile uno script o un programma di installazione, altre volte è necessario copiare manualmente i file nelle varie destinazioni finali. Quando si può scegliere, è preferibile collocare tutto quanto a partire da un'unica directory discendente da '`/opt/`'.

A volte, perché il programma possa funzionare, è necessario predisporre o modificare il contenuto di alcune variabili di ambiente. Il caso più comune è costituito da ***PATH*** che deve (o dovrebbe) contenere anche il percorso necessario ad avviare il nuovo programma. Spesso, i file di documentazione che accompagnano il software indicano chiaramente tutte le variabili che devono essere presenti durante il loro funzionamento.

La dichiarazione di queste variabili può essere collocata direttamente in uno dei file di configurazione della shell utilizzata (per esempio '`/etc/profile`', oppure '`~/.bash_profile`' o altri ancora a seconda di come è organizzato il proprio sistema).

7.4.3 Librerie dinamiche

Alcuni programmi utilizzano delle librerie non standard che spesso vengono collocate al di fuori delle directory predisposte per contenerle. Per fare in modo che queste librerie risultino disponibili, ci sono due modi possibili:

1. modificare la configurazione di '`/etc/ld.so.cache`';
2. utilizzare la variabile di ambiente ***LD_LIBRARY_PATH***.

Per agire secondo la prima possibilità, occorre comprendere come è organizzato questo meccanismo. Il file '`/etc/ld.so.cache`' viene creato a partire da '`/etc/ld.so.conf`' che contiene semplicemente un elenco di directory destinate a contenere librerie. Il programma '**`ldconfig`**' serve proprio a ricreare il file '`/etc/ld.so.cache`' leggendo '`/etc/ld.so.conf`', pertanto viene avviato solitamente dalla stessa procedura di inizializzazione del sistema, allo scopo di garantire che questo file sia sempre aggiornato.

Dovrebbe essere chiaro, ormai, il modo giusto di includere nuovi percorsi di librerie nel file '`/etc/ld.so.cache`': occorre indicare questa o queste directory nel file '`/etc/ld.so.conf`' e quindi basta avviare il programma '**`ldconfig`**'.

L'utilizzo della variabile di ambiente ***LD_LIBRARY_PATH*** è meno impegnativo, ma soprattutto, in questo modo si può intervenire facilmente attraverso dei semplici script. Ciò permette, per esempio, di fare in modo che solo un certo programma «veda» certe librerie. In ogni caso, quando si intende usare questa variabile di ambiente, è importante ricordare di includere tra i vari percorsi anche quelli standard: '`/lib/`', '`/usr/lib/`' e '`/usr/local/lib/`'. L'esempio seguente rappresenta un pezzo di uno script (potrebbe trattarsi di '`/etc/profile`') in cui viene assegnata la variabile di ambiente in questione.

```
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/opt/mio_prog/lib
export LD_LIBRARY_PATH
```

Se un certo programma richiede determinate librerie che potrebbero entrare in conflitto con altri programmi, è indispensabile l'utilizzo della variabile di ambiente ***LD_LIBRARY_PATH***, configurandola esclusivamente nell'ambito del processo di quel programma. In pratica, si tratta di avviare il programma attraverso uno script che genera l'ambiente adatto, in modo che non si rifletta negli altri processi, come mostrato nell'esempio seguente:

```
#! /bin/sh
#
# Modifica il percorso di ricerca delle librerie.
#
LD_LIBRARY_PATH="/opt/mio_programma/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH
#
# Avvia il programma.
#
mio_programma
#
# Al termine dello script non restano tracce dei
# cambiamenti.
#
```

Avviando lo script, viene modificata la variabile di ambiente ***LD_LIBRARY_PATH*** per quel processo e per i suoi discendenti (viene esportata), quindi, al termine del programma termina lo script e con lui anche gli effetti di queste modifiche.

Si osservi in particolare il fatto che nella nuova definizione del percorso delle librerie, viene posto all'inizio quello per le librerie specifiche del programma, in modo che venga utilizzato per primo; subito dopo, viene inserito l'elenco dei percorsi eventualmente già esistente.

Teoricamente, è necessario soltanto che i file delle librerie siano accessibili in lettura, perché possano essere utilizzate dai programmi; tuttavia, per motivi di sicurezza, potrebbe essere richiesto anche il permesso di esecuzione attivo. In particolare, con i sistemi GNU/Linux è indispensabile che le librerie utilizzate da Init (‘`/sbin/init`’) dispongano dei permessi di esecuzione.

7.4.4 Verifica delle librerie utilizzate dagli eseguibili

Il programma ‘**`ldd`**’ emette l'elenco delle librerie condivise richieste dai programmi indicati come argomenti della riga di comando:

```
ldd [opzioni] programma...
```

Si utilizza ‘**`ldd`**’ per determinare le dipendenze di uno o più programmi dalle librerie. Naturalmente, si può verificare anche la dipendenza di un file di libreria da altre librerie.

L'esempio seguente mostra le dipendenze dalle librerie di ‘`/bin/bash`’. Il risultato ottenuto indica il nome delle librerie e la collocazione effettiva nel sistema, risolvendo anche eventuali collegamenti simbolici.

```
$ ldd /bin/bash [Invio]

        libncurses.so.5 => /lib/libncurses.so.5 (0x40019000)
        libdl.so.2 => /lib/libdl.so.2 (0x40059000)
        libc.so.6 => /lib/libc.so.6 (0x4005c000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

7.5 Aggiornamento delle librerie standard

Oltre al problema delle librerie specifiche di un programma particolare, ci può essere la necessità di aggiornare le librerie standard del sistema operativo, a seguito di qualche aggiornamento di altro software.

Normalmente, la propria distribuzione GNU dovrebbe offrire questi aggiornamenti in forma di archivi già pronti, installabili attraverso il proprio sistema di gestione dei pacchetti. Ciò garantendo la sistemazione di dettagli importanti.

Quando si è costretti a fare da soli è importante essere attenti. In particolare, dovendo intervenire in ciò che risiede nella directory ‘`/lib/`’, se si commette un errore, si rischia di bloccare il sistema senza possibilità di rimedio.

I file delle librerie sono organizzati normalmente con un numero di versione piuttosto articolato. Quando ciò accade, è normale che

questi file siano affiancati da dei collegamenti simbolici strutturati in modo che si possa accedere a quelle librerie anche attraverso l'indicazione di versioni meno dettagliate, oppure semplicemente attraverso nomi differenti. I collegamenti in questione sono molto importanti perché ci sono dei programmi che dipendono da questi; quando si aggiorna una libreria occorre affiancare la nuova versione a quella vecchia, quindi si devono modificare i collegamenti relativi. Solo alla fine, sperando che tutto sia andato bene, si può eliminare eventualmente il vecchio file di libreria.

Per fare un esempio pratico, si suppone di disporre della libreria ‘`libc.so.9.8.7`’, articolata nel modo seguente:

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7
```

Volendo sostituire questa libreria con la versione 9.8.10, il cui file ha il nome ‘`libc.so.9.8.10`’, occorre procedere come segue:

```
# ln -s -f libc.so.9.8.10 libc.so.9 [Invio]
```

Come si vede, per generare il collegamento, è stato necessario utilizzare l'opzione ‘**`-f`**’ che permette di sovrascrivere il collegamento preesistente. Infatti, non sarebbe stato possibile eliminare prima il collegamento vecchio, perché così si sarebbe rischiato il blocco del sistema.

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.10
libc.so.9.8.7
libc.so.9.8.10
```

In generale, per sicurezza, è meglio lasciare le librerie vecchie, perché ci potrebbero essere ugualmente dei programmi che ne hanno ancora bisogno.

Quando la libreria da aggiornare ha subito un aggiornamento molto importante, per cui i numeri delle versioni sono molto distanti rispetto a quanto si utilizzava prima, conviene evitare di sostituirle, mentre è solo il caso di affiancarle. Volendo ritornate all'esempio precedente, si può supporre che la libreria da aggiornare sia arrivata alla versione 10.1.1, con il file ‘`libc.so.10.1.1`’. Intuitivamente si comprende che il collegamento simbolico ‘`libc.so.9`’ non può puntare a questa nuova libreria, mentre resta il dubbio per ‘`libc.so`’.

In generale è meglio lasciare stare le cose come sono, a meno di scoprire che qualche programma cerca proprio la libreria ‘`libc.so`’ e si lamenta perché non si tratta della versione adatta a lui. Comunque, sempre seguendo l'esempio, sarebbe il caso di riprodurre un collegamento equivalente a ‘`libc.so.9`’, denominato ovviamente ‘`libc.so.10`’.

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7
libc.so.10 -> libc.so.10.1.1
libc.so.10.1.1
```

7.6 Pacchetti applicativi confezionati appositamente per le distribuzioni GNU

Ogni distribuzione GNU utilizza un metodo per il confezionamento dei pacchetti (blocchi) che compongono l'intero sistema. Il problema principale è quello di tenere traccia della collocazione dei file di ogni applicazione, delle sue dipendenze da altri pacchetti e di permetterne l'aggiornamento o l'eliminazione senza danneggiare il sistema e senza lasciare file ignoti inutilizzati.

Ogni distribuzione GNU particolare può avere un'attenzione differente rispetto alla preparazione e alla gestione del sistema che si occupa di installare e disinstallare questi pacchetti. È il caso di citare la distribuzione Debian, a cui si rifanno altre distribuzioni derivate, in cui tale sistema è abbastanza complesso. Naturalmente, una gestione troppo semplificata dei pacchetti di applicativi è un incentivo all'utilizzo della distribuzione per un principiante, ma poi tutto questo

si traduce in gravi difficoltà nel momento in cui si vuole aggiornare la distribuzione, o semplicemente si desidera fare qualcosa di più rispetto al solito.

7.6.1 Distinguere tra «pacchetti» e «archivi»

Per evitare di fare confusione, sarebbe bene distinguere tra il «pacchetto», che rappresenta un componente installato, da installare, o da eliminare dal sistema, rispetto al suo contenitore, noto come «archivio». Per esempio, si può dire che l'archivio '`make_3.77-4.deb`' contenga il pacchetto '**`make`**' nella versione 3.77-4.

Purtroppo, questa distinzione non viene utilizzata da tutti; ci sono distribuzioni in cui si parla indifferentemente di «pacchetto» per fare riferimento all'archivio che lo contiene e a ciò che si ottiene installandolo. Questa anomalia, poi, la si riscontra anche nelle sigle usate nelle opzioni della riga di comando, dove potrebbe capitare che si utilizzi la lettera «p» (*package*) per fare riferimento ai file degli archivi.

7.6.2 Binari e sorgenti

Gran parte del software distribuito con i sistemi GNU è sottoposto alla licenza GNU-GPL (*GNU general public license*, appendice A), che impone la disponibilità dei sorgenti. Per questo motivo, una distribuzione GNU, oltre a organizzare i pacchetti compilati e archiviati opportunamente, quando richiesto dalla licenza deve mettere a disposizione i sorgenti, assieme alle modifiche eventuali, generalmente in forma di file di differenze. Si distingue così tra pacchetti binari (archiviati in qualche modo) e pacchetti sorgenti.

Il pacchetto binario si compone dei file già compilati e pronti per essere collocati dove previsto. Il pacchetto sorgente è qualcosa di diverso: contiene l'archivio originale dell'applicativo (quello dei sorgenti), assieme a tutte le informazioni necessarie per modificarlo e per compilarlo nel modo più appropriato per la distribuzione GNU in cui deve essere installato. Inoltre, dovrebbe contenere le informazioni necessarie a generare il pacchetto binario relativo.

In generale, quando si parla di «pacchetti», si fa riferimento implicitamente a quelli contenenti i binari, o comunque i file finali da installare.

7.6.3 Interdipendenza tra i pacchetti

I pacchetti, ovvero i vari blocchi in cui è suddiviso il software, devono convivere in modo armonico nel sistema. Questo fatto sembra ovvio, ma la cosa più difficile da definire è proprio la relazione corretta tra questi.

Con il termine «dipendenza», si fa riferimento al fatto che un pacchetto può dipendere da altri per il suo funzionamento. In pratica, se il pacchetto «A» richiede che sia presente anche il pacchetto «B», si dice che «A» dipende da «B». Con il termine «incompatibilità», si fa riferimento al fatto che un pacchetto non può coesistere con un altro per qualche ragione. Per esempio, se il pacchetto «A» non può stare assieme a «C» si dice che «A» è incompatibile con «C».

I due concetti sono abbastanza semplici, ma a questi se ne aggiunge un altro: la dipendenza prima dell'installazione. Infatti, un pacchetto potrebbe dipendere da un altro che deve essere già presente prima che questo venga installato. A questo proposito, si parla a volte di «pre-dipendenza». Questo tipo di dipendenza impone quindi un ordine nell'installazione dei pacchetti.

In certi casi, un pacchetto può dipendere da una funzionalità che può essere offerta da diversi altri pacchetti. Per esempio, un programma può richiedere la presenza del comando '**`mail`**' per inviare dei messaggi; più in generale questo dipenderebbe dalla funzionalità di invio della posta elettronica. Nel caso della distribuzione Debian, si parla di «pacchetti virtuali», per fare riferimento a queste funzionalità generiche da cui possono dipendere altri pacchetti reali.

7.6.4 Fasi dell'installazione e della disinstallazione di un pacchetto

Da quanto esposto, si possono intuire alcune delle fasi riferite all'installazione e alla disinstallazione di un pacchetto:

- prima dell'installazione occorre verificare che siano rispettate le dipendenze e che non ci siano incompatibilità;
- prima della disinstallazione occorre verificare che non ci siano altri pacchetti che rimangono installati e dipendono da quello che si vuole eliminare.

Ma i problemi non si limitano a questi. Infatti, un pacchetto che si installa può richiedere la predisposizione di qualcosa, come dei collegamenti simbolici, dei file di dispositivo nella directory '`/dev/`' e dei file di configurazione. In generale, gli archivi dei pacchetti utilizzati dalle distribuzioni GNU contengono degli script realizzati specificatamente per questo, cioè per sistemare le cose in fase di installazione e anche quando si disinstalla un pacchetto. Volendo si può arrivare a distinguere tra quattro script corrispondenti ad altrettante fasi:

1. uno script da eseguire prima dell'estrazione dell'archivio contenente il pacchetto da installare;
2. uno script da eseguire dopo l'estrazione dell'archivio contenente il pacchetto da installare;
3. uno script da eseguire prima della cancellazione dei file che compongono un pacchetto da disinstallare;
4. uno script da eseguire dopo la cancellazione dei file che compongono un pacchetto da disinstallare.

Naturalmente, dipende dalle caratteristiche di un pacchetto il fatto che siano necessari o meno questi script. In generale, la configurazione rappresenta un problema che viene affrontato in maniera differente dalle varie distribuzioni GNU.

La configurazione può richiedere di fatto la creazione o la modifica di un file di testo, secondo una sintassi determinata, oppure l'interazione con un programma apposito (che si occupa di fare le domande necessarie e di memorizzare le risposte nel modo più appropriato). I file che contengono le informazioni sulla configurazione di un pacchetto, fanno parte del pacchetto stesso e sono candidati per la cancellazione nel momento in cui si decide di disinstallarlo. Tuttavia, il sistema di gestione dei pacchetti potrebbe distinguere opportunamente il caso in cui si vuole disinstallare un pacchetto conservando però i file di configurazione, rispetto al caso in cui si vuole eliminare tutto senza porsi problemi di alcun tipo.

A parte il dettaglio importante relativo al fatto di trattare in modo distinto i file di configurazione nel momento della disinstallazione, le distribuzioni GNU possono differenziarsi in modo notevole in base alla gestione della configurazione stessa. In pratica si potrebbero avere due estremi:

- definire una configurazione minima e indispensabile **prima** di iniziare una nuova installazione della distribuzione GNU, lasciando che il resto venga fatto dall'utilizzatore quando vuole, dopo che l'installazione è terminata;
- definire la configurazione mano a mano che i pacchetti vengono installati.

Nel primo caso, la procedura di installazione si limiterebbe a chiedere le informazioni indispensabili per il completamento della stessa (i dischi, le partizioni, la tastiera, eventualmente la rete, ecc.); successivamente verrebbero installati i pacchetti senza disturbare più l'utilizzatore, che alla fine deve configurare per conto proprio i servizi che gli interessano.

Nel secondo caso, ogni volta che si installa un pacchetto che richiede una configurazione (indipendentemente dal fatto che si tratti della

prima installazione della distribuzione o che si tratti di un lavoro fatto in seguito), gli script che lo corredano interrogano l'utilizzatore su come configurare, almeno in modo grossolano, ciò che serve.

Tra i due estremi ci sono delle situazioni intermedie, nelle quali si possono fissare alcune informazioni che tornano utili ai pacchetti più importanti, già in fase di prima installazione, in modo da alleggerire il carico di notizie da fornire nel momento della configurazione finale legata all'installazione del singolo pacchetto.

L'esempio tipico di una distribuzione GNU in cui la configurazione avviene mano a mano che i pacchetti vengono installati è quello della Debian. Quando si installa un pacchetto nuovo in un sistema GNU già funzionante, il fatto che durante l'installazione vengano richieste (eventualmente) le informazioni necessarie a dargli una configurazione minima, è sicuramente un fatto positivo. Tuttavia, quando l'utente inesperto tenta di installare per la prima volta questa distribuzione dopo avere selezionato una grande quantità di pacchetti, questo si trova disorientato di fronte alla quantità di cose che devono essere configurate e che non sono state previste, oltre all'eccessiva quantità di tempo necessaria per completare l'installazione.

Da quanto scritto si intuisce che: di fronte a una distribuzione GNU organizzata in modo da gestire la configurazione dei pacchetti mano a mano che questi vengono installati, è indispensabile, in fase di prima installazione del sistema, iniziare con la selezione del minimo possibile, riservandosi di aggiungere ciò che manca in un momento successivo.

7.6.5 Caratteristiche di un pacchetto nei confronti di un sistema funzionante

Un sistema sofisticato di gestione dei pacchetti di una distribuzione GNU, potrebbe non limitarsi a riportare il fatto che un pacchetto sia installato o meno, dando qualche informazione in più. Un pacchetto potrebbe essere:

- non installato;
- installato (correttamente);
- non installato, ma con i file di configurazione ancora presenti (in pratica, è stato installato e successivamente disinstallato senza eliminare i file di configurazione);
- installato in parte (l'archivio è stato estratto, ma gli script necessari al completamento della procedura hanno rilevato un qualche tipo di errore, per cui il pacchetto potrebbe non essere utilizzabile).

7.6.6 Aggiornamento

L'aggiornamento di un pacchetto implica la sostituzione di quello installato con uno di una versione più aggiornata. Si tratta di un problema comune, tuttavia pone dei problemi importanti. Un aggiornamento, perché non vada a danno di chi lo fa, dovrebbe preservare la sua configurazione precedente. In pratica, se il pacchetto «A» utilizza il file di configurazione '`/etc/A.conf`', è bene che questo file non venga sovrascritto, o almeno venga conservato in qualche modo.

La politica delle distribuzioni GNU può essere varia:

- i file di configurazione potrebbero essere sostituiti senza salvare quelli precedenti;
- i file di configurazione potrebbero essere sostituiti salvandone una copia a cui viene data un'estensione particolare;
- i file di configurazione potrebbero non essere sostituiti, affiancando eventualmente la nuova versione standard di questi file con un'estensione particolare.

Tanto per fare un esempio pratico, le distribuzioni basate su archivi RPM (*Red Hat package manager*) salvano i file di configurazione precedenti utilizzando l'estensione '`.rpmorig`', mentre le distribuzioni basate su pacchetti Debian affiancano eventualmente una copia della configurazione nuova, distinguendola con l'aggiunta dell'estensione '`.dpkg-dist`'.[2]

7.6.7 File di configurazione comuni

Alcuni pacchetti potrebbero condividere uno stesso file di configurazione, oppure potrebbero dipenderne in qualche modo. Ciò comporta dei problemi che non sono facili da risolvere in generale, tanto che si cerca di evitare il più possibile tale sovrapposizione. Il caso più evidente di una dipendenza del genere è quello dei file '`/etc/passwd`' e '`/etc/group`', per i quali potrebbe essere richiesta una modifica ogni volta che si installa un servizio particolare a cui si deve associare un utente fittizio specifico, oppure un gruppo.

7.7 Pacchetti Debian

I pacchetti della distribuzione GNU/Linux Debian sono archiviati in modo differente, a seconda che si tratti di pacchetti binari o di pacchetti sorgenti. I pacchetti binari, cioè tutto quello che può essere usato così come si trova (compresa la documentazione), è archiviato nel formato Debian ('`.deb`'), mentre i sorgenti sono composti da terne di file: una descrizione contenuta in un file con estensione '`.dsc`', l'archivio dei sorgenti originali in un file con estensione '`.orig.tar.gz`' e un file di differenze da applicare ai sorgenti originali, in questo caso con l'estensione '`.diff.gz`'.

Il nome di un archivio contenente un pacchetto binario Debian ha una struttura riassumibile nello schema seguente:

***nome_versione*-*revisione*.deb**

Un pacchetto Debian binario, oltre ai file che compongono il pacchetto e che vengono installati, contiene:

- un file di «controllo» ('`control`'), con quasi tutte le informazioni relative al pacchetto, in particolare le sue dipendenze;
- un file contenente l'elenco dei file di configurazione, per i quali occorre avere un occhio di riguardo ('`conffiles`');
- due coppie di script che controllano la fase di installazione e quella di disinstallazione del pacchetto ('`preinst`', '`postinst`', '`prerm`', '`postrm`').

7.7.1 Priorità di un pacchetto

A ogni pacchetto Debian viene attribuita una priorità, rappresentata da una definizione standard. Questa permette di facilitare la scelta dei pacchetti da installare. Si usano le parole chiave seguenti:

Parola chiave	Significato	Descrizione
`required`	richiesto	indica un pacchetto necessario per il funzionamento elementare del sistema;
`important`	importante	indica un pacchetto importante per il buon funzionamento del sistema;
`standard`	standard	indica un pacchetto che fa parte di software comune in un sistema GNU, senza richiedere la presenza del sistema grafico X;
`optional`	opzionale	indica un pacchetto opzionale;
`extra`	extra	indica un pacchetto destinato a un uso specializzato, o che va in conflitto con altri pacchetti di livello precedente.

È interessante osservare che il livello di priorità è un'informazione che normalmente non è contenuta nei pacchetti, ma viene attribuita all'esterno di questi. La si ritrova nei file '`Packages`' e '`Packages.cd`', la cui descrizione viene fatta in seguito.

7.7.2 Dipendenze secondo i pacchetti Debian

«

Come accennato, il file di controllo contiene in particolare le informazioni sulle dipendenze del pacchetto. Secondo la logica di Debian, le dipendenze avvengono sempre in relazione a «pacchetti»: quando si tratta di una dipendenza da una funzionalità, questa viene identificata attraverso un «pacchetto virtuale». L'esempio seguente è un estratto delle informazioni relative al pacchetto '**apache-ssl**', dove si vede l'uso delle definizioni di quasi tutti i tipi di dipendenza e di incompatibilità:

```
Depends: libc6 (>= 2.0.7u-6), libssl09, mime-support, …
Suggests: apache-doc, lynx
Conflicts: apache-modules, php3 (<= 3.0.3-1), ↩
↪libapache-mod-perl (<= 1.15-2.1)
Replaces: apache-modules
Provides: httpd
```

Le varie parole chiave hanno il significato seguente:

Parola chiave	Significato	Descrizione
`depends`	dipende	indica un elenco di pacchetti indispensabili, eventualmente con il livello di versione richiesto, che devono essere presenti perché questo possa funzionare;
`recommends`	raccomanda	indica un elenco di pacchetti raccomandati, anche se non indispensabili, perché il pacchetto che si installa possa essere utilizzato opportunamente;
`suggests`	suggerisce	indica un elenco di pacchetti suggeriti, che starebbero bene assieme a quello che si installa;
`conflicts`	contrasta	indica un elenco di pacchetti che non possono convivere assieme a questo;
`replaces`	sostituisce	indica un elenco di pacchetti che vengono rimpiazzati da questo;
`provides`	fornisce	indica un elenco di pacchetti che rappresentano le funzionalità offerte da questo (si tratta in pratica di pacchetti virtuali).

Le parole chiave utilizzate sono verbi posti alla terza persona singolare, come dire che «il pacchetto A: dipende da... raccomanda... suggerisce... va in conflitto con... sostituisce... fornisce le funzionalità...». Osservando l'esempio, il pacchetto in questione fornisce le funzionalità '**httpd**' (in questo caso un pacchetto virtuale), ovvero quelle di un servente HTTP, è incompatibile con il pacchetto '**apache-modules**' e con altri se hanno una versione troppo vecchia, inoltre va a sostituire lo stesso pacchetto '**apache-modules**'.

7.7.3 Stato di un pacchetto

«

Secondo la logica del sistema di gestione dei pacchetti Debian, lo stato di un pacchetto può avere tre gruppi di caratteristiche: lo stato in relazione a ciò che è installato nel sistema, lo stato di selezione e alcune caratteristiche speciali. Rispetto al sistema, un pacchetto può essere:

Parola chiave	Significato	Descrizione
`installed`	installato	il pacchetto risulta installato correttamente nel sistema e anche la configurazione è stata completata;
`half-installed`	semi-installato	l'installazione del pacchetto non è stata completata per qualche ragione;
`not-installed`	non installato	il pacchetto non risulta installato;
`unpacked`	estratto	il pacchetto risulta estratto dall'archivio, ma non è stato configurato;
`half-configured`	semi-configurato	il pacchetto risulta estratto dall'archivio e la configurazione non è stata completata per qualche ragione;
`config-files`	file di configurazione	del pacchetto sono presenti solo i file di configurazione.

Il sistema di installazione e disinstallazione dei pacchetti Debian è in realtà una procedura, con la quale si «prenotano» delle operazioni che poi vengono eseguite in sequenza. Sotto questo aspetto, un pacchetto che in qualche modo sia «conosciuto» da questa procedura ha anche uno stato di selezione (come già accennato) che può essere:

Parola chiave	Significato	Descrizione
`unknown`	sconosciuto	quando non è mai stata richiesta la sua installazione (e di conseguenza non è nemmeno installato);
`install`	da installare	quando è stata richiesta la sua installazione, o il suo aggiornamento;
`remove` `deinstall`	da togliere	quando è stata richiesta la sua disinstallazione normale, cioè senza cancellare i file di configurazione;
`purge`	da eliminare completamente	quando è stata richiesta la sua eliminazione totale, comprendendo anche i file di configurazione.

Infine, un pacchetto può essere stato marcato in modo che non venga aggiornato o sostituito con un'altra versione, '**hold**', oppure può essere stato marcato dal sistema di gestione dei pacchetti perché risulta danneggiato in qualche modo e in tal senso viene indicato come candidato alla reinstallazione, '**reinst-required**'.

Per conoscere lo stato di un pacchetto si può usare '**dpkg**' richiedendo l'azione '**-l**'. L'esempio seguente mostra l'elenco di alcuni pacchetti con l'indicazione del loro stato:

```
$ dpkg -l[Invio]

Desired=Unknown/Install/Remove/Purge
| Status=Not/Installed/Config-files/Unpacked/Failed-config/
|/       Half-installed
|| Err?=(none)/Hold/Reinst-required/X=both-problems
||/     (Status,Err: uppercase=bad)
||| Name            Version   Description
+++-===============-=========-==============================
ii  gnome-admin     1.0.1-1   Gnome Admin Utilities (gulp an
ii  gnome-bin       1.0.3-1   Miscellaneous binaries used by
rc  gnome-card-game 1.0.1-4   Gnome card games - Solitaire g
rc  gnome-control-c 0.30-2    The Gnome Control Center
ii  gnome-core      1.0.1-0.3 Common files for Gnome core ap
ii  gnome-dev-doc   1.0.3-1   Gnome developers documentation
pn  gnome-games     <none>    (no description available)
ii  gnome-games-loc 1.0.1-4   The locale databases for the g
rc  gnome-gnibbles  1.0.1-4   A cute little game that has no
rc  gnome-gnobots   1.0.1-4   Gnome version of text based ro
pn  gnome-guile     <none>    (no description available)
un  gnome-gxnsnmp   <none>    (no description available)
pn  gnome-gxsnmp    <none>    (no description available)
ii  gnome-terminal  1.0.1-0.3 The Gnome terminal emulator ap
```

Tabella 7.13. Significato delle lettere utilizzate nelle prime tre colonne del rapporto generato con '**dpkg -l**'.

Colonna	Sigla	Sintesi	Significato
1	u	*unknown*	Pacchetto sconosciuto.
1	i	*install*	Pacchetto da installare.
1	r	*remove*	Pacchetto da rimuovere (lasciando la configurazione).
1	p	*purge*	Pacchetto da eliminare completamente.
2	n	*not installed*	Pacchetto non installato.

Colonna	Sigla	Sintesi	Significato
2	i	*installed*	Pacchetto installato.
2	c	*configuration*	Sono presenti solo i file di configurazione.
2	u	*unpacked*	Pacchetto estratto dall'archivio, ma non configurato.
2	f	*failed configuration*	Configurazione interrotta.
2	h	*hold*	Installazione interrotta.

Colonna	Sigla	Sintesi	Significato
3			Nessuno stato particolare.
3	h	*hold*	Segnato per la conservazione alla versione attuale.
3	r	*reinstall*	Si richiede la reinstallazione.
3	x	*hold and reinstall*	Equivalente a «h» e a «r» messi assieme.

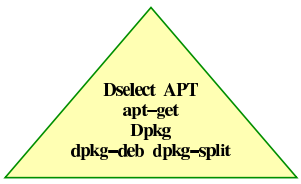
7.7.4 Disponibilità di un pacchetto

La gestione dei pacchetti Debian, richiede che i programmi che ne consentono l'installazione, siano in grado di sapere quali sono i pacchetti effettivamente disponibili. I pacchetti disponibili sono quelli che si trovano in una distribuzione su CD-ROM, DVD-ROM, in una copia locale nel file system (eventualmente condiviso in rete), in un sito Internet,... In ogni caso, tali informazioni sono contenute in un file che accompagna i pacchetti (uno per ogni raggruppamento principale) e si tratta di '`Packages`', o '`Packages.cd`' nel caso di distribuzioni suddivise su più dischi (CD-ROM, DVD-ROM o unità di memorizzazione di altro tipo che possono essere sostituite durante l'installazione).

7.7.5 Stratificazione degli strumenti di gestione dei pacchetti Debian

Gli strumenti per la gestione dei pacchetti Debian sono molti e sono fatti per intervenire a livelli diversi. La figura 7.14 mostra la piramide, alla base della quale si trovano gli strumenti fondamentali.

Figura 7.14. Gerarchia semplificata tra gli strumenti di gestione dei pacchetti Debian.



In breve:

- '**`dpkg-deb`**' interviene solo al livello di archivi Debian, consentendo l'estrazione e l'archiviazione in questo formato;
- '**`dpkg-split`**' è uno strumento aggiuntivo in grado di suddividere e riassemblare assieme gli archivi Debian, in modo da poterli gestire in file più piccoli, soprattutto quando questi devono essere trasportati su unità di memorizzazione a capacità limitata;
- Dpkg (l'eseguibile '**`dpkg`**') interviene nei pacchetti Debian, a livello elementare, consentendone l'installazione e la loro disinstallazione, avvalendosi eventualmente di '**`dpkg-deb`**' quando necessario;
- '**`apt-get`**' interviene nei pacchetti Debian, a livello più evoluto di Dpkg, essendo in grado di risolvere da solo molti problemi di dipendenze;
- Dselect si trova al livello più alto per la gestione dei pacchetti (assieme a APT) e si avvale di tutti gli strumenti inferiori;
- APT è un sistema di strumenti paralleli a Dselect, composto da diversi programmi frontali alternativi che poi si avvalgono di '**`apt-get`**' per lo svolgimento dei loro compiti.

7.7.6 Gestione elementare attraverso gli strumenti fondamentali

Per l'installazione e la disinstallazione dei pacchetti Debian, lo strumento più banale è dato da Dpkg,[3] composto dall'eseguibile '**`dpkg`**'. Questo è in grado di utilizzare a sua volta anche '**`dpkg-deb`**', quando si vuole intervenire a livello di archivio Debian.

`dpkg` [*opzioni*] *azione*

Alla fine della riga di comando dell'eseguibile '**`dpkg`**' deve apparire l'indicazione di un'azione, che può essere preceduta da alcune opzioni. Come si intuisce, l'azione stabilisce cosa debba essere fatto, mentre le opzioni ne alterano l'ambito. Qui viene mostrato solo l'uso di alcune azioni, perché per le altre conviene agire attraverso strumenti di livello superiore. Non viene mostrato l'uso delle opzioni, comunque si può approfondire l'uso di Dpkg consultando la pagina di manuale *dpkg(8)*.

Tabella 7.15. Alcune opzioni.

Opzione	Descrizione
`-c` *archivio_debian*… `--contents` *archivio_debian*…	Questa azione richiama l'utilizzo di '**`dpkg-deb`**' allo scopo di mostrare l'elenco dei file contenuti nell'archivio Debian indicato come argomento.
`-f` *archivio_debian*… `--field` *archivio_debian*…	Questa azione richiama l'utilizzo di '**`dpkg-deb`**' allo scopo di mostrare le informazioni di controllo sui pacchetti contenuti negli archivi elencati.
`-I` *archivio_debian*… `--info` *archivio_debian*…	Questa azione richiama l'utilizzo di '**`dpkg-deb`**' allo scopo di mostrare tutte le informazioni disponibili sui pacchetti contenuti negli archivi elencati.
`-i` *archivio_debian*… `--install` *archivio_debian*…	Installa o aggiorna i pacchetti contenuti negli archivi indicati come argomento. Questa azione può essere abbinata all'opzione '**`-R`**', o '**`--recursive`**', allo scopo di installare tutti i pacchetti i cui archivi si trovano in una directory, che diventerebbe quindi l'argomento di questa azione. L'installazione del pacchetto include anche la configurazione dello stesso.
`--configure` *pacchetto*…	Richiede espressamente di eseguire la configurazione dei pacchetti indicati (ammesso che questi non risultino già installati e configurati correttamente).
`-r` *pacchetto*… `--remove` *pacchetto*…	Rimuove i pacchetti indicati, senza eliminare i file di configurazione.
`-P` *pacchetto*… `--purge` *pacchetto*…	Elimina completamente i pacchetti indicati, compresi i file di configurazione.
`-l` [*modello_pacchetti*…] `--list` [*modello_pacchetti*…]	Mostra l'elenco dei pacchetti corrispondenti ai modelli indicati (si usano i metacaratteri comuni, proteggendoli in modo che la shell non li interpreti direttamente). Se vengono indicati dei modelli, vengono fornite informazioni su tutti i pacchetti conosciuti, non solo quelli installati, mentre utilizzando l'azione senza argomenti, si ottengono informazioni solo sui pacchetti installati, o che comunque hanno mantenuto i file di configurazione.

Opzione	Descrizione
`-s` ***pacchetto***… `--status` ***pacchetto***…	Mostra le informazioni sullo stato dei pacchetti indicati, aggiungendo anche la descrizione se il pacchetto risulta installato. In questo caso non si possono più utilizzare i metacaratteri.
`-L` ***pacchetto***… `--listfiles` ***pacchetto***…	Elenca i file che appartengono a un pacchetto, in base a quanto contenuto nell'archivio originale. Tuttavia, non è possibile conoscere in questo modo quali file sono stati creati dagli script di installazione del pacchetto stesso.
`-S` ***modello_file***… `--search` ***modello_file***…	Permette di fare una ricerca per trovare a quali pacchetti appartengono i file indicati come argomento, con o senza l'ausilio di metacaratteri. In particolare, se si indica un nome o un modello senza l'indicazione di un percorso, si ottengono tutti i pacchetti che hanno file o directory con quel nome.
`-C` `--audit`	Controlla i pacchetti installati per determinare quali sono stati installati solo parzialmente.
`--compare-versions` ***versione_1*** ↩ ↪***operatore versione_2***	Si tratta di una richiesta particolare con la quale si ottiene il confronto tra due stringhe che rappresentano una versione (completa di revisione). Ciò può essere molto utile nella realizzazione di script. Se il confronto dà un risultato *Vero*, l'eseguibile '**`dpkg`**' restituisce zero, mentre negli altri casi restituisce un valore maggiore. Gli operatori che possono essere utilizzati sono elencati nella tabella 7.16.

Tabella 7.16. Utilizzo dell'opzione '**`--compare-versions`**' con '**`dpkg`**'.

Espressione	Descrizione
`--compare-versions` ***ver_1*** `eq` ***ver_2***	*Vero* se le versioni sono uguali.
`--compare-versions` ***ver_1*** `ne` ***ver_2***	*Vero* se le versioni sono differenti.
`--compare-versions` ***ver_1*** `lt` ***ver_2***	*Vero* se la prima versione è minore della seconda.
`--compare-versions` ***ver_1*** `le` ***ver_2***	*Vero* se la prima versione è minore o uguale alla seconda.
`--compare-versions` ***ver_1*** `gt` ***ver_2***	*Vero* se la prima versione è maggiore della seconda.
`--compare-versions` ***ver_1*** `ge` ***ver_2***	*Vero* se la prima versione è maggiore o uguale alla seconda.

Segue la descrizione di alcuni esempi.

- `$` **`dpkg -c zsh_3.1.2-10.deb`** [*Invio*]

  Mostra l'elenco dei file che compongono il pacchetto '**`zsh`**', contenuti nell'archivio indicato, esclusi i file che vengono creati dagli script del pacchetto stesso.

- `$` **`dpkg -I zsh_3.1.2-10.deb`** [*Invio*]

  Mostra tutte le informazioni disponibili sull'archivio indicato.

- `#` **`dpkg -i zsh_3.1.2-10.deb`** [*Invio*]

  Installa, o aggiorna, il pacchetto contenuto nell'archivio indicato, ammesso che ciò sia possibile in relazione alle dipendenze di questo.

- `#` **`dpkg -r zsh`** [*Invio*]

  Rimuove il pacchetto indicato, senza eliminare i file di configurazione.

- `#` **`dpkg -P zsh`** [*Invio*]

  Elimina completamente il pacchetto indicato, compresi i file di configurazione.

- `$` **`dpkg -l`** [*Invio*]

  Elenca lo stato di tutti i pacchetti installati, o dei quali rimangono i file di configurazione.

- `$` **`dpkg -l z\*`** [*Invio*]

  Elenca lo stato di tutti i pacchetti conosciuti che iniziano con la lettera «z». Si osservi l'uso della barra obliqua inversa per proteggere l'asterisco contro l'interpretazione da parte della shell.

- `$` **`dpkg -s zsh`** [*Invio*]

  Mostra le informazioni sullo stato del pacchetto indicato, in modo più dettagliato.

- `$` **`dpkg -L zsh`** [*Invio*]

  Elenca i file che appartengono al pacchetto '**`zsh`**'.

- `$` **`dpkg -S /bin/cat`** [*Invio*]

  Cerca di scoprire a chi appartiene il file '`/bin/cat`' (scoprendo che appartiene al pacchetto '**`textutils`**').

7.7.7 Gestione più evoluta dei pacchetti: organizzazione di una copia della distribuzione

Per una gestione più evoluta dei pacchetti, occorre definire la fonte di questi, dalla quale deve essere ottenibile il file '`Packages`'. Per comprendere la cosa, è necessario prima di tutto conoscere in che modo dovrebbe essere organizzato un CD-ROM o una copia nel file system della distribuzione. Lo schema seguente rappresenta l'essenziale (la metavariabile ***arch*** viene sostituita dalla sigla dell'architettura per cui sono fatti i pacchetti):

```
debian/
 |-- .disk/
 |    `-- info
 `-- dists/
      |-- stable/
      |    |-- main/
      |    |    `-- binary-arch/
      |    |         |-- Packages
      |    |         |-- Packages.gz
      |    |         |-- Packages.cd
      |    |         |-- Packages.cd.gz
      |    |         `-- *.deb
      |    |-- contrib/
      |    |    `-- binary-arch/
      |    |         |-- Packages*
      |    |         `-- *.deb
      |    |-- non-free/
      |    |    `-- binary-arch/
      |    |         |-- Packages*
      |    |         `-- *.deb
      |    |-- non-US/
      |    |    `-- binary-arch/
      |    |         |-- Packages*
      |    |         `-- *.deb
      |    `-- local/
      |         `-- binary-arch/
      |              |-- Packages*
      |              `-- *.deb
      `--local/
          `-- local --> ../stable/local
```

Il file '`debian/.disk/info`' è indispensabile quando la distribuzione è suddivisa su più dischi. Questo file contiene una riga che serve a identificare l'unità di memorizzazione. I file '`debian/dists/stable/*/binary-`***arch***`/Packages`', assieme alle loro versioni compresse con '**`gzip`**', contengono le informazioni sui pacchet-

ti contenuti negli archivi che si trovano nella stessa directory, o in quelle successive, mentre i file '`debian/dists/stable/*/binary-`***arch***`/Packages.cd`' contengono le informazioni di tutti i file '`Packages`' delle stesse directory per tutti i dischi in cui si articola la distribuzione.

Di tutta questa struttura, la directory '`debian/`' è la radice, o l'inizio, e questa è la posizione che viene richiesta dai programmi per la gestione dei pacchetti quando devono attingere le informazioni sulla disponibilità dei pacchetti.

Come si vede dalla struttura mostrata, una distribuzione Debian si articola anche in componenti, a causa di possibili limitazioni nell'uso e nella distribuzione del software relativo. In generale, gli archivi che si trovano a partire da '`debian/dists/stable/main/`' sono quelli principali che non contengono limitazioni particolari, mentre per gli altri valgono considerazioni differenti. Le varie componenti in cui si articola una distribuzione sono identificate dai nomi delle directory che si diramano da '`debian/dists/stable/`'.

7.7.8 APT a livello essenziale

Il sistema APT si basa sul programma di servizio '**`apt-get`**', che a sua volta si avvale di '**`dpkg`**'.

Per funzionare, '**`apt-get`**' richiede la presenza di un file di configurazione, '`/etc/apt/sources.list`', all'interno del quale vanno elencate le fonti da cui si possono ottenere delle distribuzioni Debian. Questo file può contenere commenti, preceduti dal simbolo '`#`', righe vuote che vengono ignorate come i commenti e righe contenenti ognuna l'indicazione di un'origine, espressa secondo la sintassi seguente:

```
deb uri_inizio_distribuzione distribuzione componente...
```

Per esempio, per indicare l'utilizzo della distribuzione '**`stable`**' (come mostrato fino a questo punto negli esempi) contenuta a partire da '`/home/pippo/debian/`', della quale si vogliono tutte le componenti normali, si può utilizzare la direttiva seguente:

```
deb file:/home/pippo/debian/ stable main contrib non-free non-US local
```

Nello stesso modo si potrebbero indicare degli URI riferiti a siti FTP o HTTP. Inoltre, è importante tenere presente che si possono indicare molte fonti differenti, come si vede dall'esempio seguente:

```
deb file:/home/pippo/1/debian/ stable main contrib non-free non-US local
deb file:/home/pippo/2/debian/ stable main contrib non-free non-US local
deb http://ftp.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable non-US
```

Dopo la configurazione, '**`apt-get`**' richiede espressamente che gli sia ordinato di leggere le sorgenti per aggiornare l'elenco locale dei pacchetti disponibili, in modo da disporre di una visione di ciò che esiste e delle dipendenze relative. Si osservi lo schema sintattico per l'utilizzo di '**`apt-get`**':

```
apt-get [opzioni] [comando] [pacchetto...]
```

Da quello che si vede, nella riga di comando di '**`apt-get`**' non si fa mai riferimento direttamente ad archivi Debian, ma sempre solo a pacchetti.

Per comprendere il funzionamento di '**`apt-get`**' è bene cominciare da alcuni esempi. Normalmente si inizia aggiornando l'elenco locale dei pacchetti disponibili:

```
# apt-get update [Invio]
```

Quindi potrebbe essere conveniente chiedere l'aggiornamento dei pacchetti, riferiti alla stessa versione della distribuzione che si sta utilizzando:

```
# apt-get upgrade [Invio]
```

Per installare o aggiornare un pacchetto specifico, soddisfacendo le dipendenze necessarie, si può intervenire come nell'esempio seguente, dove si mostra in che modo installare o aggiornare il pacchetto '**`zsh`**', rispettando le dipendenze:

```
# apt-get install zsh [Invio]
```

Infine, per aggiornare il proprio sistema a una nuova versione della distribuzione, si utilizza il comando seguente:

```
# apt-get -f dist-upgrade [Invio]
```

Questo è seguito generalmente da una richiesta esplicita di configurazione dei pacchetti che ne avessero bisogno, per mezzo di '**`dpkg`**':

```
# dpkg --configure --pending [Invio]
```

Tabella 7.20. Alcuni comandi.

Opzione	Descrizione
`update`	Risincronizza l'elenco locale dei pacchetti rispetto alle origini dichiarate nel file di configurazione. In generale, prima di un comando '**`upgrade`**' o '**`dist-upgrade`**', occorrerebbe eseguire un comando '**`update`**'.
`upgrade`	Aggiorna tutti i pacchetti che risultano già installati, per i quali è disponibile un aggiornamento. L'aggiornamento non viene fatto se questo può provocare degli inconvenienti.
`dist-upgrade`	Aggiorna tutti i pacchetti che risultano già installati, per i quali è disponibile un aggiornamento. L'aggiornamento viene fatto tenendo conto delle dipendenze, che nel frattempo potrebbero essere cambiate (di solito quando si tratta di un aggiornamento che coinvolge l'intera distribuzione). Di solito, dopo questo tipo di operazione, si avvia il comando '**`dpkg --configure --pending`**' allo scopo di procedere con la configurazione di ciò che richiede tale passaggio.
`install` ***pacchetto...***	Installa i pacchetti indicati come argomento, provvedendo a sistemare anche le dipendenze. È bene sottolineare che vanno indicati i nomi dei pacchetti e non i nomi degli archivi che li contengono.
`check`	Esegue un controllo, anche sui pacchetti che sono stati installati in modo errato.
`clean`	APT utilizza un deposito transitorio per gli archivi utilizzati per l'installazione o l'aggiornamento. Si tratta della directory '`/var/cache/apt/archives/`', che va ripulita periodicamente attraverso il comando '**`clean`**'.

Tabella 7.21. Alcune opzioni.

Opzione	Descrizione
`-f` `--fix-broken`	Con l'uso di questa opzione si fa in modo che '**`apt-get`**' cerchi di sistemare i problemi legati alle dipendenze. Questa opzione può essere usata da sola, senza l'indicazione di un comando, oppure con un comando, di solito '**`dist-upgrade`**', per richiedere espressamente la sistemazione dei problemi che si possono generare con lo stesso.
`-s` `--simulate`	Simula l'esecuzione del comando, in modo da mostrare cosa si otterrebbe.

7.7.9 Ricerca dei file che apparentemente non appartengono ad alcun pacchetto

Con l'aiuto di Dpkg è possibile cercare di individuare a quale pacchetto appartiene questo o quel file. Precisamente si usa l'opzione '**`-S`**', come nell'esempio seguente:

```
$ dpkg -S /etc/timezone [Invio]
```

In questo caso si fa riferimento al file '`/etc/timezone`' e si dovrebbe ottenere una segnalazione simile a quella seguente, da cui si comprende che il file è abbinato al pacchetto '**`timezones`**':

```
timezones: /etc/timezone
```

Per avere una visione di insieme dei file che potrebbero essere stati abbandonati inavvertitamente, si può usare Cruft,[4] che scandisce il file system e genera un rapporto. Naturalmente, non tutto ciò che viene indicato è necessariamente un file superfluo, ma è comunque il punto di partenza per la propria ricerca.

```
cruft [opzioni]
```

L'eseguibile '**`cruft`**' può essere usato solo dall'utente '**`root`**' e l'elenco che genera, di file sospetti, viene emesso attraverso lo standard output, a meno che siano usate delle opzioni particolari.

7.7.10 Ricerca dei pacchetti che possono essere disinstallati senza problemi di dipendenze

L'individuazione dei pacchetti che possono essere eliminati senza problemi di dipendenze, è un compito piuttosto complesso senza l'utilizzo di uno strumento apposito. Per questo si può usare Deborphan[5] che, se usato senza argomenti, si limita a elencare i pacchetti delle librerie che, apparentemente, non servono:

```
deborphan [opzioni] [pacchetto]...
```

Tabella 7.23. Alcune opzioni significative per l'utilizzo di Deborphan.

Opzione	Descrizione
`-d` `--show-deps`	Mostra le dipendenze dei pacchetti indicati alla fine della riga di comando.
`-n` `--nice-mode`	Elenca anche i pacchetti che non sono indispensabili, pur essendo suggeriti o raccomandati da altri.
`-a` `--all-packages`	Elenca tutti i pacchetti che risultano dall'interrogazione richiesta, senza limitare il risultato alle sole librerie.
`-P` `--show-priority`	Mostra la priorità dei pacchetti che risultano dall'interrogazione.
`-s` `--show-section`	Mostra la sezione a cui appartengono i pacchetti ottenuti dall'interrogazione.

Eventualmente, si può controllare il funzionamento di Deborphan tramite Orphaner,[6] che viene distribuito assieme a Deborphan stesso, il quale si comporta da programma frontale interattivo:

```
orphaner [--purge] [opzioni_deborphan]
```

Con l'opzione '**`--purge`**' si specifica di voler eliminare completamente i pacchetti che devono essere selezionati successivamente (eliminando anche i file di configurazione), mentre ogni opzione successiva viene passata tale e quale al programma '**`deborphan`**', allo scopo di cambiare l'elenco dei pacchetti selezionabili.

Figura 7.24. L'aspetto di '**`orphaner`**' durante il suo funzionamento. In questo caso non sono state indicate opzioni di alcun tipo, pertanto vengono elencati soltanto i nomi delle librerie inutili.

```
.--------------------Orphaner V1.1-----------------------.
| Select Packages for removal or cancel to quit:          |
| .-------------------------------------------------------. |
| |             [ ] fftw2     main/libs                   | |
| |             [ ] ldso      main/oldlibs                | |
| |             [ ] libdb4.1  main/libs                   | |
| |             [ ] libpng3   main/libs                   | |
| |                                                       | |
| |                                                       | |
| |                                                       | |
| |                                                       | |
| |                                                       | |
| |                                                       | |
| |                                                       | |
| '-------------------------------------------------------' |
|-----------------------------------------------------------|
|  <   OK   >    <Simulate>    < Cancel >    <  Help  >    |
'-----------------------------------------------------------'
```

7.7.11 Riconfigurazione

L'installazione di un pacchetto prevede anche una fase di configurazione, che in molti casi è determinata in modo automatico, mentre il resto delle volte richiede un'interazione con un programma. Questa comunicazione tra utente e programma di installazione serve a creare al volo i file di configurazione necessari al funzionamento dei programmi che compongono il pacchetto; di solito, dopo tale operazione è possibile intervenire a mano in questi file, ma spesso ciò non è conveniente. Per ripetere la configurazione di un pacchetto già installato, si può usare '**`dpkg-reconfigure`**', che fa parte del pacchetto Debconf:[7]

```
dpkg-reconfigure [opzioni] pacchetto
```

Per esempio, si può usare il comando seguente per riconfigurare il pacchetto '**`debconf`**':

```
# dpkg-reconfigure debconf [Invio]
```

L'esempio mostra una situazione abbastanza comune, dal momento che il pacchetto '**`debconf`**' serve proprio a definire il comportamento della distribuzione in fase di configurazione dei pacchetti:

```
.----------------[?] Configuring Debconf-------------------.
| Packages that use debconf for configuration share a      |
| common look and feel. You can select the type of user    |
| interface they use.                                      |
|                                                          |
| The dialog frontend is a full-screen, character based    |
| interface, while the readline frontend uses a more       |
| traditional plain text interface, and the gnome frontend |
| is a modern X interface. The editor frontend lets you    |
| configure things using your favorite text editor. The    |
| noninteractive frontend never asks you any questions.    |
|                                                          |
| What interface should be used for configuring packages?  |
| .-------------------------------------------------------. |
| |                    Dialog                             | |
| |                    Readline                           | |
| |                    Gnome                              | |
| |                    Editor                             | |
| |                    Noninteractive                     | |
| '-------------------------------------------------------' |
|----------------------------------------------------------|
|               <  OK  >            <Cancel>               |
'----------------------------------------------------------'
```

Selezionando la voce *Dialog* si passa a un'altra richiesta, in cui si specifica il livello di attenzione che si vuole dare alla configurazione:

```
.------------------[?] Configuring Debconf-----------------.
| Debconf prioritizes the questions it asks you. Pick the  |
| lowest priority of question you want to see:             |
|   - 'critical' only prompts you if the system might      |
|     break. Pick it if you are a newbie, or in a hurry.   |
|   - 'high' is for rather important questions             |
|   - 'medium' is for normal questions                     |
|   - 'low' is for control freaks who want to see          |
|     everything                                           |
|                                                          |
| Note that no matter what level you pick here, you will   |
| be able to see every question if you reconfigure a       |
| package with dpkg-reconfigure.                           |
|                                                          |
| See only questions that are of what priority and higher? |
| .--------------------------------------------------. |
| |                      medium                      | |
| |                      critical                    | |
| |                      high                        | |
| |                      low                         | |
| '--------------------------------------------------' |
|----------------------------------------------------------|
|              <  OK  >           <Cancel>                |
'----------------------------------------------------------'
```

È disponibile anche un programma frontale per accedere alla configurazione dei pacchetti, partendo dalla classificazione a cui questi appartengono, con Configure-debian:[8]

```
configure-debian [opzioni]
```

Le opzioni che si possono indicare nella riga di comando vengono passate tali e quali al programma '**dpkg-reconfigure**', nel momento in cui questo viene avviato. Segue un esempio di utilizzo del programma:

```
# configure-debian [Invio]
```

```
.-------Configure Packages---------.
|                                   |
|                                   |
| Which subsection do you want?     |
| .-------------------------------. |
| |         admin                 | |
| |         base                  | |
| |         comm                  | |
| |         devel                 | |
| |         doc                   | |
| |         editors               | |
| |         libs                  | |
| |         misc                  | |
| |         net                   | |
| |         otherosfs             | |
| |         shells                | |
| |         sound                 | |
| '---------.(+)-----------------' |
|-----------------------------------|
|      <  OK  >    <Cancel>         |
'-----------------------------------'
```

Scorrendo l'elenco viene scelta la voce '**tex**':

```
.------------Configure Packages--------------.
|                                             |
|                                             |
| Which program do you want to configure?     |
| .-----------------------------------------. |
| |              tetex-base                 | |
| |              tetex-bin                  | |
| '-----------------------------------------' |
|---------------------------------------------|
|          <  OK  >      <Cancel>             |
'---------------------------------------------'
```

Viene selezionata la voce '**tetex-base**', che così viene configurato. Al termine viene proposto di configurare un altro pacchetto:


```
.--------------Configure Packages-----------------.
|                                                 |
|                                                 |
| Would you like to configure another program?    |
|                                                 |
|                                                 |
|-------------------------------------------------|
|            <  Yes  >       <  No  >            |
'-------------------------------------------------'
```

7.7.12 Pacchetti Debian sorgenti

I pacchetti sorgenti messi a disposizione dalla distribuzione GNU/Linux Debian sono composti generalmente da tre file:

***nome_versione*-*revisione*.dsc**

***nome_versione*-*revisione*.orig.tar.gz**

***nome_versione*-*revisione*.diff.gz**

Il file con estensione '`.dsc`' contiene informazioni essenziali sul pacchetto, con le firme eventuali, per garantire la sua integrità. Il file con estensione '`.orig.tar.gz`' contiene i sorgenti originali, archiviati attraverso tar+gz. Il file con estensione '`.diff.gz`' è un file di differenze da applicare per l'adattamento alla distribuzione; eventualmente questo file potrebbe mancare se il pacchetto nasce espressamente per la distribuzione Debian.

Per compilare un pacchetto occorre prima estrarlo, applicandogli le modifiche previste. Per farlo nel modo corretto, si usa il comando seguente, dove si suppone che nella directory corrente siano disponibili i tre file del pacchetto che interessa:

```
$ dpkg-source -x nome_versione-revisione.dsc [Invio]
```

In questo modo si ottiene la directory '***nome*-*versione*/**', all'interno della quale bisogna entrare per avviare uno script che viene creato proprio con l'applicazione delle modifiche:

```
$ cd nome-versione [Invio]

$ su [Invio]

# debian/rules binary [Invio]
```

Come si vede, è stato necessario acquisire i privilegi dell'utente '**root**' per procedere alla compilazione.

In alternativa, se manca il file con estensione '`.dsc`', si può rimediare nel modo seguente:

```
$ tar xzvf nome_versione-revisione.orig.tar.gz [Invio]

$ cd nome-versione [Invio]

$ cat ../nome_versione-revisione.diff.gz | gunzip | patch -p1 [Invio]

$ chmod a+x debian/rules [Invio]

$ su [Invio]

# debian/rules binary [Invio]
```

Come si vede, si devono applicare le modifiche manualmente, quindi occorre attribuire al file '`debian/rules`' i permessi di esecuzione. Il resto funziona regolarmente.

Al termine si ottiene il file '***nome_versione*-*revisione_arch*.deb**', contenente il pacchetto binario pronto per l'installazione.

È evidente che, se si vogliono apportare delle modifiche ulteriori al sorgente, queste vanno fatte dopo l'estrazione e dopo l'applicazione delle modifiche già previste per la distribuzione Debian. Volendo ricostruire un pacchetto sorgente corretto, si interviene secondo la sequenza seguente.

1. Si estrae l'archivio originale e si applicano le modifiche già previste:

   $ **tar xzvf** ***nome_versione-revisione*****.orig.tar.gz** [*Invio*]

   $ **cd** ***nome-versione*** [*Invio*]

   $ **cat** ***../nome_versione-revisione*****.diff.gz** ↩
   ↪ **| gunzip | patch -p1** [*Invio*]

   $ **cd ..** [*Invio*]

2. Si fanno le modifiche aggiuntive che si ritengono necessarie.
3. Si cancella il file con estensione '`.diff.gz`' e '`.dsc`':

   $ **rm** ***nome_versione-revisione*****.diff.gz** [*Invio*]

   $ **rm** ***nome_versione-revisione*****.dsc** [*Invio*]

4. Si ricostruisce tutto con '**`dpkg-source`**':

   $ **dpkg-source -b** ***nome-versione*** [*Invio*]

   In pratica, l'argomento dell'opzione '**`-b`**' è il nome della directory contenente i sorgenti modificati.

Al termine si ottiene un file '***nome_versione*****-*****revisione*****`.diff.gz`**' nuovo, assieme al file '***nome_versione*****-*****revisione*****`.dsc`**' appropriato.

7.8 Ricompilazione dei pacchetti Debian sorgenti

Può capitare di avere la necessità di ricompilare un pacchetto con opzioni differenti. Attraverso il sistema APT è possibile gestire in modo semplice la ricompilazione dei pacchetti sorgenti della distribuzione GNU/Linux Debian.

Come per i pacchetti compilati, anche per accedere ai pacchetti sorgenti attraverso il sistema APT, è necessario predisporre il file '`/etc/apt/sources.list`'. In questo caso, servono direttive che iniziano con la parola chiave '**`deb-src`**':

```
deb-src uri_inizio_distribuzione distribuzione componente...
```

Ecco alcuni esempi:

```
deb-src http://mi.mirror.garr.it/mirrors/debian stable    main contrib
deb-src http://mi.mirror.garr.it/mirrors/debian testing   main contrib
deb-src http://mi.mirror.garr.it/mirrors/debian unstable  main contrib
```

Naturalmente, nello stesso file '`/etc/apt/sources.list`' possono convivere anche le direttive relative all'origine di pacchetti già compilati e pronti per l'uso.

Ogni pacchetto sorgente ha bisogno dei propri strumenti di sviluppo per poter procedere alla sua ricompilazione. Di certo sono necessari due pacchetti: '**`fakeroot`**' e '**`build-essential`**':

apt-get install fakeroot build-essential [*Invio*]

Per il resto, fortunatamente, c'è un modo semplice per installare esattamente ciò che serve:

apt-get build-dep ***nome*** [*Invio*]

Per esempio, si suppone di voler installare ciò che serve per ricompilare il pacchetto '**`mc`**' (Midnight Commander):

apt-get build-dep mc [*Invio*]

Il pacchetto dei sorgenti può essere acquisito senza privilegi particolari, pertanto si può operare in qualità di un utente comune. Ci si posiziona all'interno di una directory che si considera appropriata, quindi si procede:

$ **apt-get source** ***nome*** [*Invio*]

Continuando con l'ipotesi di voler ricompilare il pacchetto '**`mc`**' (Midnight Commander), il comando è il seguente:

$ **apt-get source mc** [*Invio*]

Supponendo di accedere alla versione '**`4.6.1-2`**', si ottengono i file '`mc_4.6.1.orig.tar.gz`', '`mc_4.6.1-2.dsc`' e '`mc_4.6.1-2.diff.gz`'; inoltre si ottiene la sottodirectory '`mc-4.6.1/`', contenente i sorgenti estratti; il tutto a partire dalla directory corrente.

Per modificare i sorgenti si entra nella directory che li contiene e si procede con le modifiche che si ritengono necessarie. Nella maggior parte dei casi si tratta solo di modificare il modo in cui viene avviato lo script '**`configure`**', ma per questo occorre intervenire nel file '`debian/rules`', oppure, come nel caso particolare di '**`mc`**', nel file '`debian/rocks`'.

$ **cd mc-4.6.1** [*Invio*]

...

Dopo le modifiche ci si posiziona all'inizio della directory dei file sorgenti, quindi si procede con il comando seguente:

$ **dpkg-buildpackage -rfakeroot -uc -us** [*Invio*]

Si ottiene la ricompilazione e la creazione di un pacchetto Debian pronto per l'installazione, ma nella directory che precede la posizione corrente (la directory genitrice).

Supponendo di avere ottenuto il file '`../mc_4.6.1-2_i386.deb`', lo si può installare così:

dpkg -i ../mc_4.6.1-2_i386.deb [*Invio*]

Come si vede dall'esempio, ovviamente, per installare il pacchetto servono i privilegi dell'amministratore del sistema.

7.9 Dselect

Dselect[9] è uno strumento molto importante per la gestione dei pacchetti Debian, nel momento in cui si vuole tenere sotto controllo il quadro completo della situazione. Si tratta di un programma frontale che gestisce direttamente Dpkg e può avvalersi della configurazione di APT per quanto riguarda l'origine da cui prelevare i pacchetti da installare.

È fondamentale la comprensione di molti dettagli di funzionamento di questo programma, per poterlo sfruttare al meglio, considerato che il suo utilizzo non è molto intuitivo.

7.9.1 Menù iniziale

Dselect è un programma interattivo, rappresentato dall'eseguibile '**`dselect`**', che di solito si avvia senza argomenti:

dselect [*Invio*]

Il suo funzionamento è suddiviso in sei passaggi, rappresentati dal suo menù iniziale, che in generale dovrebbero essere eseguiti secondo la sequenza prevista:

1. scelta del metodo di accesso agli archivi della distribuzione;
2. aggiornamento dell'elenco dei pacchetti disponibili;
3. selezione dei pacchetti da installare, da rimuovere o da eliminare dal sistema;
4. installazione o aggiornamento dei pacchetti, in base alla selezione fatta;
5. richiesta esplicita di eseguire la configurazione dei pacchetti per i quali questa operazione non fosse stata eseguita, o che non fosse stata completata normalmente (di solito, l'installazione provvede anche alla loro configurazione);
6. rimozione o eliminazione dei pacchetti in base alle richieste fatte in fase di selezione.

In generale, al completamento di uno di questi passaggi, Dselect suggerisce automaticamente di passare al successivo. La figura 7.31 mostra il menù introduttivo di Dselect.

Figura 7.31. Il menù iniziale di Dselect.

```
Debian Linux 'dselect' package handling frontend.

 * 0. [A]ccess  Choose the access method to use.
   1. [U]pdate  Update list of available packages, if possible.
   2. [S]elect  Request which packages you want on your system.
   3. [I]nstall Install and upgrade wanted packages.
   4. [C]onfig  Configure any packages that are unconfigured.
   5. [R]emove  Remove unwanted software.
   6. [Q]uit    Quit dselect.

Use ^P and ^N, cursor keys, initial letters, or digits to
select. Press ENTER to confirm selection. ^L to redraw screen.
```

Se si utilizza Dselect con un monitor monocromatico, è molto importante ricordarsi di configurare correttamente la variabile di ambiente ***TERM***, in modo che questa contenga un nome di un terminale monocromatico (potrebbe trattarsi del nome '**vt100**' o '**vt220**', basta verificare nella directory '`/usr/share/terminfo/v/`'), altrimenti si fa molta fatica a decifrare il significato delle varie informazioni che appaiono sullo schermo, soprattutto si fa fatica a individuare il cursore di selezione. Per esempio, si potrebbe usare il comando seguente:

```
# TERM=vt100 ; dselect [Invio]
```

La selezione della voce desiderata avviene in modo molto semplice: per spostare il cursore di selezione si possono usare i tasti freccia, le cifre numeriche corrispondenti, o le iniziali; per selezionare la voce evidenziata basta premere [*Invio*].

7.9.2 Metodo di accesso ai pacchetti della distribuzione

Come accennato, è bene seguire l'ordine prestabilito, per cui si comincia dalla selezione del metodo di accesso agli archivi della distribuzione. La figura 7.32 mostra il menù che si potrebbe presentare dopo tale selezione.

Figura 7.32. Il menù di selezione del metodo di accesso alla distribuzione.

```
#### dselect - list of access methods ################################
  Abbrev.     Description
  cdrom       Install from a CD-ROM.
* multi_cd    Install from a CD-ROM set.
  nfs         Install from an NFS server (not yet mounted).
  multi_nfs   Install from an NFS server (using the CD-ROM set)
  harddisk    Install from a hard disk partition (not yet mounted).
  mounted     Install from a filesystem which is already mounted.
  multi_mount Install from a mounted partition with changing contents.
  floppy      Install from a pile of floppy disks.
  apt         APT Acquisition [file,http,ftp]

#### Access method 'multi_cd'. ######################################
multi_cd - Install from a CD-ROM set.

Installation from a CD-ROM set containing a Debian distribution.  The
CD-ROMs may be or not be mounted already and should contain a standard
ISO9660 CD-ROM filesystem.

#### explanation of multi_cd ########################################
```

Come si può osservare, l'immagine sullo schermo è suddivisa in due parti: quella superiore che contiene l'elenco dei metodi di accesso, dove si esegue la selezione, e quella inferiore che contiene la descrizione della voce su cui si trova il cursore di selezione in quel dato momento (in questo caso si tratta della voce '**multi_cd**'). Come nel menù generale, il cursore si può spostare utilizzando i tasti freccia e anche altre combinazioni (è disponibile una guida interna accessibile attraverso la pressione di [*F1*] o di [*?*], dalla quale si esce premendo la [*barra-spaziatrice*]); infine, premendo [*Invio*] si seleziona la voce in cui si trova il cursore.

È bene sottolineare che l'elenco dei metodi di accesso potrebbe essere composto da un numero maggiore o minore di possibilità, in base alla disponibilità effettiva. Qui vengono descritti solo alcuni metodi di accesso, dal momento che poi il meccanismo di selezione si ripete con una logica simile anche negli altri.

7.9.3 Accesso attraverso un disco ottico unico

Il metodo di accesso corrispondente alla sigla '**cdrom**' è ormai un po' antiquato, utilizzabile solo con una distribuzione Debian ridotta a un solo disco ottico. Una volta fatta la selezione, viene richiesto quale file di dispositivo deve essere utilizzato per l'innesto del disco e di solito viene proposto già '`/dev/cdrom`':

```
  I see that /dev/cdrom exists and is a block device.

Insert the CD-ROM and enter the block device name
[/dev/cdrom]:
```

Se la realtà corrisponde a quanto proposto (lo si vede tra parentesi quadre), basta confermare premendo [*Invio*], diversamente si scrive il percorso del file di dispositivo adatto e quindi si conferma sempre con [*Invio*]. Se tutto procede regolarmente, Dselect tenta di accedere al disco e dovrebbe riuscirci anche se questo è già stato innestato in precedenza.

```
Insert the CD-ROM and enter the block device name
[/dev/cdrom]: [Invio]

  All directory names should be entered relative to the root
  of the CD-ROM.

  I would like to know where on the CD-ROM the top level of
  the Debian distribution is (eg. 'dists/stable') - this
  directory usually contains the Packages-Master file.

  If the CD-ROM is badly organised and doesn't have a
  straightforward copy of the distribution you may answer
  'none' and we'll go through the parts I need individually.

  Last time you said '/debian/dists/stable', and that looks
  plausible.

Distribution top level ? [/debian/dists/stable]:
```

A questo punto, viene richiesta l'indicazione della directory di inizio della distribuzione, tenendo conto che il percorso deve essere indicato partendo dalla directory radice del disco stesso. Di solito, un disco di una distribuzione Debian dovrebbe essere organizzato esattamente come propone Dselect, per cui dovrebbe bastare premere [*Invio*].

```
Distribution top level ? [/debian/dists/stable]: [Invio]
```

Se il disco è organizzato esattamente come si aspetta Dselect, allora tutto va bene, altrimenti si è costretti a inserire ogni singola directory e ogni singolo percorso dei vari blocchi in cui può essere divisa la distribuzione.

```
  Using '/debian/dists/stable/main/binary-i386' as main binary dir.
  Using '/debian/dists/stable/main/binary-i386/Packages.gz' for main.
  Using '/debian/dists/stable/contrib/binary-i386' as contrib binary dir.
  Using '/debian/dists/stable/contrib/binary-i386/Packages.gz' for contrib.
  Using '/debian/dists/stable/non-free/binary-i386' as non-free binary dir.
  Using '/debian/dists/stable/non-free/binary-i386/Packages.gz' for non-free.
  Using '/debian/dists/stable/non-US/binary-i386' as non-US binary dir.
  Using '/debian/dists/stable/non-US/binary-i386/Packages.gz' for non-US.
  Using '/debian/dists/stable/local/binary-i386' as local binary dir.
  Using '/debian/dists/stable/local/binary-i386/Packages.gz' for local.

Hit RETURN to continue.
```

Quelle che si vedono sono le informazioni che Dselect richiede; se queste directory non sono esattamente lì dove Dselect si aspetta che siano, occorre indicarle tutte una per una, inoltre per ognuna occorre specificare dove si trova il file '`Packages.gz`' relativo. Comunque, alla fine la cosa si conclude qui e si torna al menù iniziale.

```
Hit RETURN to continue. [Invio]
```

7.9.4 Accesso attraverso un insieme di dischi

Il metodo di accesso corrispondente alla sigla '**multi_cd**' è quello più comune, data l'attuale dimensione di una distribuzione Debian completa, che impone la riproduzione su diversi dischi. Scegliendo

tale modalità di installazione, in questa fase, si deve inserire l'**ultimo** dei dischi che compongono la distribuzione. Dopo aver selezionato il dispositivo (esattamente come nel caso del metodo '**cdrom**'), occorre indicare dove iniziano le distribuzioni. Quindi, non è più come prima: occorre indicare la directory all'interno della quale poi si può trovare la gerarchia '`dists/`'.

```
All directory names should be entered relative to the root of
the CD-ROM.

I would like to know where on the CD-ROM the top level of
the Debian distribution is - this will usually contain the
'dists' directory.

If the CD-ROM is badly organised and doesn't have a
straightforward copy of the distribution you may answer
'none' and we'll go through the parts I need individually.
Last time you said '/debian', and that looks plausible.
Distribution top level ? [/debian]
```

Così come propone Dselect, si tratta della directory '`/debian/`', sempre intesa come riferita all'inizio del disco stesso. In questo caso basta confermare premendo [*Invio*].

```
Distribution top level ? [/debian] [Invio]

Ok, this is Debian GNU/Linux 2.1 "slink" - disco 2 di 2.

Using '/debian/dists/stable/main/binary-i386' as main binary
      dir from disk 'Debian GNU/Linux 2.1 "slink" -
      disco 2 di 2'
Using '/debian/dists/stable/main/binary-i386/Packages.cd.gz'
      for main.
Using '/debian/dists/stable/contrib/binary-i386' as contrib
      binary dir from disk 'Debian GNU/Linux 2.1 "slink" -
      disco 2 di 2'
Using '/debian/dists/stable/contrib/binary-i386/Packages.cd.gz'
      for contrib.
Using '/debian/dists/stable/non-free/binary-i386' as
      non-free binary dir from disk 'Debian GNU/Linux 2.1
      "slink" - disco 2 di 2'
Using '/debian/dists/stable/non-free/binary-i386/Packages.cd.gz'
      for non-free.
Using '/debian/dists/stable/non-US/binary-i386' as non-US
      binary dir from disk 'Debian GNU/Linux 2.1 "slink" -
      disco 2 di 2'
Using '/debian/dists/stable/non-US/binary-i386/Packages.cd.gz'
      for non-US.
Using '/debian/dists/local/local/binary-i386' as local
      binary dir from disk 'Debian GNU/Linux 2.1 "slink" -
      disco 2 di 2'
Using '/debian/dists/local/local/binary-i386/Packages.cd.gz'
      for local.

Hit RETURN to continue.
```

Anche in questo caso sarebbe bene che il disco fosse organizzato esattamente come si aspetta Dselect, altrimenti si è costretti a inserire i percorsi delle directory e dei file '`Packages.cd.gz`'. Si osservi in particolare, che rispetto al metodo '**cdrom**', in questo caso si utilizzano i file '`Packages.cd.gz`' e non più i file '`Packages.gz`'. Al termine, si torna al menù principale.

```
Hit RETURN to continue. [Invio]
```

7.9.5 Accesso attraverso un file system di rete

I metodi '**nfs**' e '**multi_nfs**' permettono di accedere a un file system NFS che non è stato ancora innestato. Potrebbe trattarsi di un elaboratore che offre in condivisione l'accesso al suo lettore di dischi ottici, che comunque, lì, deve essere stato innestato. A differenza dei metodi che coinvolgono il lettore di dischi ottici locale, si deve aggiungere l'indicazione del nome o dell'indirizzo dell'elaboratore che offre l'accesso attraverso il protocollo NFS.

Nel primo caso Dselect si aspetta di trovare una distribuzione unica e completa, per cui servono i file '`Packages.gz`', mentre nel secondo si tratta di dati che possono cambiare (per esempio perché viene sostituito il disco nell'elaboratore remoto), per cui contano i file '`Packages.cd.gz`'.

7.9.6 Accesso attraverso un file system già innestato localmente

I metodi '**mounted**' e '**multi_mount**' sono riferiti a una distribuzione accessibile all'interno del file system, dove Dselect non deve occuparsi di innestare dei dischi.

Nel primo caso si deve trattare di una distribuzione unica e completa, per cui Dselect va a cercare i file '`Packages.gz`', mentre nel secondo si tratta di dati che possono cambiare (per esempio perché si interviene per mezzo di collegamenti simbolici), per cui contano i file '`Packages.cd.gz`'.

7.9.7 Accesso attraverso APT

Il metodo '**apt**' corrisponde all'utilizzo degli strumenti offerti dal sistema APT (in questo caso si usa precisamente '**apt-get**'). In effetti, quando si dispone di una distribuzione accessibile attraverso APT, questa è la scelta migliore, dal momento che basta configurare il file '`/etc/apt/sources.list`' per risolvere il problema con Dselect.

Dselect propone di modificare il file, ma sarebbe bene che questo fosse già predisposto correttamente prima di iniziare, in modo da poter confermare semplicemente la configurazione attuale:

```
 see you already have a source list.
 ----------------------------------------------------------------------
deb file:/home/tizio/DEBIAN/1/debian stable main contrib non-free non-US local
deb file:/home/tizio/DEBIAN/2/debian stable main contrib non-free non-US local
 ----------------------------------------------------------------------
Do you wish to change it?[y/N]
```

In questo caso, la distribuzione si trova suddivisa in due parti, a partire dalle directory '`/home/tizio/DEBIAN/1/debian/`' e '`/home/tizio/DEBIAN/2/debian/`'. Per confermare, basta premere [*Invio*], dal momento che '**n**' (no) è la scelta predefinita.

```
Do you wish to change it?[y/N] [Invio]
```

7.9.8 Aggiornamento dell'elenco locale dei pacchetti

La fase successiva richiede di aggiornare l'elenco dei pacchetti disponibili selezionando la seconda voce del menù iniziale (quella con il numero uno).

```
  1. [U]pdate   Update list of available packages, if possible.
```

Questa operazione non richiede un'interazione con l'utilizzatore. Al massimo si può verificare che sia terminata o meno con successo, in base ai messaggi che si possono leggere.

7.9.9 Selezione dei pacchetti

Si passa quindi alla fase più complessa: quella della selezione dei pacchetti da installare o da disinstallare.

```
  2. [S]elect   Request which packages you want on your system.
```

Prima di mostrare la schermata di selezione dei pacchetti, data la complessità della cosa, Dselect presenta una guida, di cui si vede la schermata iniziale nella figura 7.41.

Figura 7.41. Introduzione alla guida interna di Dselect per la selezione dei pacchetti da installare.

```
Help: Introduction to package selections####################
Welcome to dselect's main package listing.

You will be presented with a list of packages which are
installed or available for installation.  You can navigate
around the list using the cursor keys, mark packages for
installation (using '+') or deinstallation (using '-').
Packages can be marked either singly or in groups; initially
you will see that the line 'All packages' is selected.  '+',
'-' and so on will affect all the packages described by the
highlighted line.

Press ? for help menu, <space> for next topic, <enter> to
exit help.
```

In pratica, si esce da questa guida premendo [*Invio*], mentre si possono scorrere le varie pagine di informazioni premendo la [*barra-spaziatrice*]. Per richiamare questa guida durante la selezione dei pacchetti, basta premere il punto interrogativo, [*?*], o il tasto [*F1*].

Dopo avere superato la schermata di presentazione della guida si ottiene finalmente il pannello di selezione dei pacchetti (figura 7.42).

Figura 7.42. Pannello di selezione dei pacchetti.

```
dselect - main package listing (status, priority)
mark:+/=/- verbose:v help:?
EIOM Pri Section Package  Inst.ver   Avail.ver  Description
    - All packages -
    --- Installed packages ---
    ----- Installed Required packages -----
    ------- Installed Required packages in section base ----
 *** Req base base-files  2.1.0      2.1.0      Debian Base
 *** Req base base-passwd 2.0.3.3    2.0.3.3    Debian Base
 *** Req base bash        2.01.1-4.1 2.01.1-4.1 The GNU Bour
 *** Req base bsdutils    4.4.1.1    4.4.1.1    Basic utilit
bash         installed;  install (was: install).  Required
bash - The GNU Bourne Again SHell

Bash is an sh-compatible command language interpreter that
executes commands read from the standard input or from a
file.  Bash also incorporates useful features from the Korn
and C shells (ksh and csh).
#### description of bash ###################################
```

Prima di mettersi a selezionare i pacchetti, occorre comprendere come «muoversi» in questa fase. Per cominciare, si deve osservare che la schermata dovrebbe apparire divisa in due parti, come si vede nella figura, dove nella parte superiore appare un cursore (che in questa figura non si vede, ma è posizionato sul pacchetto '**bash**') e nella parte inferiore appare la descrizione di ciò che si trova in corrispondenza del cursore.

Il cursore si sposta, «intuitivamente», utilizzando i tasti: [*freccia-su*], [*freccia-giù*], [*pagina-su*], [*pagina-giù*], [*Inizio*] e [*Fine*]. Alle volte, non riuscendo a leggere tutte le informazioni (come nel caso dell'esempio), è possibile spostare la visualizzazione in orizzontale, utilizzando i tasti [*freccia-sinistra*] e [*freccia-destra*].

```
EIOM Pri Section  Package    Inst.ver  Avail.ver Description
```

In condizioni normali, l'intestazione dell'elenco è piuttosto oscura. I primi quattro caratteri rappresentano rispettivamente: uno stato di errore, lo stato del pacchetto installato, la selezione precedente e la selezione attuale. La tabella 7.44 raccoglie queste sigle e mostra i simboli che possono essere abbinati nelle colonne rispettive; la tabella 7.45 mostra invece il significato dei simboli che possono essere attribuiti a questi indicatori.

Tabella 7.44. Indicatori di un pacchetto.

Indicatore	Significato	Valori possibili
E	Stato di errore	spazio, R
I	Stato di installazione	spazio, *, −, U, C, I
O	Selezione precedente	*, −, =, _, n
M	Selezione attuale	*, −, =, _, n

Tabella 7.45. Simboli riferiti agli indicatori e loro significato.

Indicatore	Simbolo	Significato
E	spazio	Nessun errore grave.
E	R	Errore grave: reinstallare.
I	spazio	Non installato.
I	*	Installato regolarmente.
I	− (meno)	Ci sono solo i file di configurazione.
I	U	Estratto, ma non configurato.
I	C	Semi-configurato: si è verificato un errore.
I	I	Semi-installato: si è verificato un errore.
O, M	*	Segnato per l'installazione o l'aggiornamento.
O, M	− (meno)	Segnato per la rimozione, lasciando la configurazione.
O, M	=	Segnato per non essere toccato (*hold*).

Indicatore	Simbolo	Significato
O, M	– (trattino basso)	Segnato per l'eliminazione totale.
O, M	n	Il pacchetto è nuovo e non è stato stabilito nulla.

La selezione dei pacchetti avviene intervenendo sull'ultimo di questi indicatori, «M», dove però si utilizzano tasti differenti rispetto ai simboli che si usano per rappresentarne la selezione. La tabella 7.46 mostra questi tasti e il risultato che se ne ottiene.

Tabella 7.46. Tasti di selezione dei pacchetti.

Tasto di selezione	Effetto
[+], [*Ins*]	Richiede l'installazione o l'aggiornamento del pacchetto.
[-], [*Canc*]	Richiede la disinstallazione del pacchetto lasciando la configurazione.
[_]	Richiede l'eliminazione totale del pacchetto.
[=], [*H*]	Richiede che il pacchetto venga congelato (*hold*).
[:], [*G*]	Toglie lo stato di pacchetto congelato.

Le colonne successive indicano: la priorità del pacchetto, generalmente in forma abbreviata, la sezione, ovvero il raggruppamento a cui questo risulta abbinato, il nome del pacchetto, la versione installata, la versione disponibile e infine la descrizione.

L'aspetto di questo pannello di selezione può essere cambiato; in particolare si possono ottenere le descrizioni estese dei primi quattro indicatori, così come della colonna della priorità, utilizzando il tasto [*v*]; in modo simile, il tasto [*V*] serve a mostrare più o meno colonne di informazioni. Oltre a questo è possibile cambiare il genere di informazioni che appaiono nella parte inferiore dello schermo, che si riferiscono al pacchetto evidenziato dal cursore, oppure è possibile dedicare tutto lo schermo all'elenco o solo alla descrizione del pacchetto. Questi tasti sono elencati nella tabella 7.47.

Tabella 7.47. Tasti per il controllo della visualizzazione del pannello di selezione.

Tasto	Effetto
[*i*]	Cambia il tipo di informazione che appare nella parte inferiore dello schermo.
[*I*]	Dedica lo schermo solo all'elenco o solo alla descrizione.
[*o*], [*O*]	Cambiano l'ordine dell'elenco dei pacchetti.
[*v*]	Espande o contrae le prime colonne.
[*V*]	Seleziona la visualizzazione di alcune colonne finali.

Vale la pena di osservare in che modo è possibile cambiare ordine all'elenco dei pacchetti visualizzati. Con il tasto [*O*] si individuano alcune forme di raggruppamento globale:

- Installato o meno ---> obsoleto, aggiornato;
- Installato, rimosso, eliminato o mai installato;
- nessuna classificazione.

Con il tasto [*o*] si definiscono delle sotto-classificazioni ulteriori:

- priorità ---> sezione;
- sezione ---> priorità;
- nessuna classificazione.

È molto importante selezionare la combinazione giusta dell'ordine in cui vengono classificati i pacchetti, altrimenti ci si perde alla loro ricerca. Tuttavia, fortunatamente, è possibile eseguire una ricerca rapida di un pacchetto il cui nome contiene una stringa data. Si ottiene questo con il tasto [/], che permette di inserire la stringa e di premere [*Invio*] per avviare la ricerca; mentre con il tasto [\] si ripete la ricerca con l'ultimo modello inserito.

Tabella 7.48. Funzionalità varie.

Tasto	Effetto
[/]	Cerca in base a una stringa.
[\ ]	Ripete la ricerca.
[*Ctrl l*]	Ripulisce l'immagine sullo schermo.
[*?*], [*F1*]	Richiama la guida interna.

Quando si seleziona o si deseleziona un pacchetto e questo fatto comporta delle ripercussioni sugli altri, perché occorre soddisfare in qualche modo delle dipendenze, o delle incompatibilità, viene presentato un sotto-pannello riepilogativo dei pacchetti coinvolti in queste dipendenze, in cui si offre una soluzione del problema. Il funzionamento di questi sotto-pannelli è identico a quello principale; ciò che conta è comprendere che ogni pannello ha un suo contesto e solo quando si giunge alla conferma delle selezioni fatte su quello generale, si esce da questa fase di utilizzo di Dselect.

Figura 7.49. Nel momento in cui viene rivelato un problema di dipendenze, si presenta questa guida, dalla quale si esce premendo [*Invio*].

```
Help: Introduction to conflict/dependency resolution
sub-list Dependency/conflict resolution - introduction.

One or more of your choices have raised a conflict or
dependency problem - some packages should only be installed
in conjunction with certain others, and some combinations of
packages may not be installed together.

You will see a sub-list containing the packages involved.
The bottom half of the display shows relevant conflicts and
dependencies; use 'i' to cycle between that, the package
descriptions and the internal control information.
```

Figura 7.50. Esempio di un sotto-pannello di selezione per risolvere un problema di dipendenze o di incompatibilità.

```
dselect - recursive package listing
mark:+/=/- verbose:v help:?
EIOM Pri Section  Package      Description
  ** Opt non-free unzip        De-archiver for .zip files
  *_ Opt non-us/u unzip-crypt  De-archiver for .zip files
  *_ Opt non-us/u zip-crypt    Archiver for .zip files


unzip      not installed;  install (was: install).  Optional
unzip conflicts with unzip-crypt


interrelationships affecting unzip
```

In generale, quando si sta operando con un pannello, o un sotto-pannello di selezione, con il tasto [*Invio*] si confermano le scelte fatte, a patto che siano state rispettate le dipendenze, mentre per annullare le selezioni si usa il tasto [*R*] (si veda la tabella 7.51). Al termine si torna al menù iniziale.

Tabella 7.51. Conferma o ripristino delle selezioni in un pannello.

Tasto	Effetto
[*Invio*]	Conferma le scelte fatte e chiude la selezione nel pannello corrente.
[*Q*]	Impone le scelte fatte indipendentemente dalle dipendenze che vengono violate.
[*X*], [*Esc*]	Abbandona le modifiche ed esce.
[*R*]	Ripristina allo stato precedente del pannello attuale.
[*U*]	Pone tutto allo stato suggerito.
[*D*]	Pone tutto allo stato richiesto espressamente.

7.9.10 Installazione e aggiornamento dei pacchetti selezionati

Il passo successivo è l'installazione dei pacchetti che sono stati indicati per questo nella fase precedente.

```
3. [I]nstall      Install and upgrade wanted packages.
```

L'installazione prevede anche la configurazione, per cui inizia qui una procedura che può risultare anche abbastanza lunga, durante la quale bisogna essere sempre pronti a rispondere alle domande che vengono fatte. In generale, quando si aggiorna un pacchetto o lo si installa, mentre sono presenti ancora i file di configurazione vecchi, questi vengono mantenuti ed eventualmente gli vengono affiancati i file nuovi con un'estensione differente ('`.dpkg-dist`').[10]

Dal momento che un pacchetto può avere anche delle «predipendenze» che possono impedirgli l'installazione se prima non è già stato installato qualcos'altro, può darsi che l'installazione debba essere ripetuta più volte per riuscire a installare tutto ciò che è stato richiesto.

7.9.11 Configurazione

Dopo l'installazione si può richiedere espressamente la configurazione dei pacchetti che per qualche motivo non sono stati configurati nel passaggio precedente.

```
4. [C]onfig   Configure any packages that are unconfigured.
```

7.9.12 Cancellazione dei pacchetti

Infine, si può richiedere la rimozione o l'eliminazione dei pacchetti segnati per questo nella fase di selezione.

```
5. [R]emove   Remove unwanted software.
```

7.10 APT

APT[11] è un sistema di gestione dei pacchetti Debian più evoluto di Dselect, per il quale sono disponibili diversi programmi frontali.

Tutto il sistema si avvale di due file di configurazione: '`/etc/apt/apt.conf`' e '`/etc/apt/sources.list`'. Il secondo di questi due file serve a stabilire la fonte da cui attingere per installare i pacchetti, mentre il primo è un file di configurazione generale.

Quando si utilizzano programmi interattivi che sfruttano la console, se si dispone soltanto di uno schermo monocromatico occorre ricordarsi di intervenire sulla variabile di ambiente ***TERM***, in modo che questa contenga un nome di un terminale monocromatico (potrebbe trattarsi del nome '**`vt100`**' o '**`vt220`**', basta verificare nella directory '`/usr/share/terminfo/v/`'), altrimenti può essere faticoso interpretare o leggere le informazioni.

7.10.1 Configurazione generale

Come già accennato, i file di configurazione più importanti del sistema APT sono '`/etc/apt/apt.conf`' e '`/etc/apt/sources.list`'. Per entrambi questi file vengono ignorate le righe bianche e quelle vuote, mentre cambia il modo di rappresentare i commenti: per '`/etc/apt/sources.list`' si rappresentano precedendoli con il simbolo '`#`', mentre '`/etc/apt/apt.conf`' ignora quanto contenuto tra '`/*`' e '`*/`', oppure quanto preceduto da '`//`'.

Le direttive del file '`/etc/apt/apt.conf`' sono organizzate a gruppi e sottogruppi, secondo una forma simile a quella seguente:

gruppo **::** ***sottogruppo*** **::** ***opzione valore_assegnato*** **;**

In pratica, il valore che si assegna alla direttiva complessiva può essere una stringa delimitata da apici doppi, oppure senza delimitazione se questo non è necessario.

Per evitare di dover riscrivere ogni volta il gruppo e il sottogruppo di un insieme di direttive, si può usare una notazione alternativa:

```
gruppo {
    sottogruppo {
        opzione valore_assegnato ;
        opzione valore_assegnato ;
        ...
    }
    sottogruppo {
        opzione valore_assegnato ;
        opzione valore_assegnato ;
        ...
    }
    ...
}
```

La classificazione in gruppi e sottogruppi serve a definire il contesto a cui riferire le opzioni, permettendo uno sviluppo indipendente della configurazione da parte dei programmi che compongono il sistema APT.

In generale non dovrebbe essere necessario modificare il file '`/etc/apt/apt.conf`'; a ogni modo, la pagina di manuale *apt.conf(5)* descrive bene i vari gruppi e sottogruppi, mentre sono disponibili degli esempi nella directory '`/usr/share/doc/apt/`'.

Il file '`/etc/apt/sources.list`' serve a definire come raggiungere i pacchetti della distribuzione. Si distinguono due situazioni: pacchetti Debian binari e pacchetti Debian sorgenti.

```
deb uri percorso_distribuzione [componente]...
```

```
deb-src uri percorso_distribuzione [componente]...
```

A seconda che si tratti di pacchetti binari o sorgenti, la direttiva deve iniziare con la parola chiave '**`deb`**' oppure '**`deb-src`**'.

Queste direttive servono a definire il percorso necessario a raggiungere i pacchetti, come se fossero scritte nel modo seguente:

```
uri/percorso_distribuzione/componente_1
uri/percorso_distribuzione/componente_2
uri/percorso_distribuzione/componente_3
...
```

Per esempio, se la distribuzione è raggiungibile presso *http://ftp.it.debian.org/debian/stable/* che poi si articola nelle directory '`main/`', '`contrib/`' e '`non-free/`', la direttiva potrebbe essere espressa nel modo seguente:

```
deb http://ftp.it.debian.org/debian stable main contrib non-free
```

Nello stesso modo, se ciò può risultare più chiaro, si potrebbero utilizzare tre direttive come nell'esempio seguente, avendo cura di indicare il punto finale che rappresenta la directory corrente:

```
deb http://ftp.it.debian.org/debian stable/main        .
deb http://ftp.it.debian.org/debian stable/contrib     .
deb http://ftp.it.debian.org/debian stable/non-free    .
```

Nell'ambito dei percorsi che si indicano in queste direttive, si può usare la stringa '**`$(ARCH)`**', che viene rimpiazzata con la sigla della propria architettura.

Gli URI possono essere indicati in diversi modi, che vengono elencati nel seguito.

- `file:`*percorso_assoluto* | `file://`*percorso_assoluto*

  Permette di fare riferimento a una distribuzione accessibile nell'ambito del file system. Il percorso assoluto inizia con la barra obliqua, per cui si possono avere URI del tipo '**`file:/uno/due`**', oppure '**`file:///uno/due`**' indifferentemente.

- `copy:`*percorso_assoluto* | `copy://`*percorso_assoluto*

  Si tratta di una variante del tipo '**`file`**', in cui si prevede di copiare i file nella memoria di transito, costituita normalmente dalla directory '`/var/cache/apt/archives/`'. Alcuni programmi come Aptitude potrebbero avere difficoltà ad accedere a un URI del tipo '**`file`**', preferendo invece questa seconda alternativa.

- `http://`*nodo*`[:`*porta*`][/`*percorso*`]`

  `ftp://`*nodo*`[:`*porta*`][/`*percorso*`]`

  Si tratta di un accesso a un servizio HTTP o FTP.

Esiste anche un altro tipo di URI, che inizia con la parola chiave '**`cdrom`**', allo scopo di fare riferimento a distribuzioni su CD-ROM o DVD-ROM. Tuttavia, le direttive di questo tipo sono troppo complesse e vanno realizzate con l'aiuto del programma '**`apt-cdrom`**'.

7.11 Approfondimento: raccolta di pacchetti Debian

La distribuzione Debian, per la sua natura collaborativa, è molto dinamica e nel susseguirsi delle sue edizioni viene prodotta una grande quantità di pacchetti aggiornati. Data la dimensione che ha la distribuzione, è improbabile che si riesca ad averne sempre una copia completa; piuttosto è facile trovare nelle riviste dei CD-ROM o dei DVD-ROM con questo o quel gruppo di applicativi, più o meno aggiornati. Volendo realizzare una propria copia locale della distribuzione (su disco fisso, o su dischi rimovibili), occorre realizzare qualche script per gestire un po' meglio la cosa.

7.11.1 Composizione del nome degli archivi Debian

Il nome degli archivi Debian è organizzato secondo la struttura seguente:

```
{nome_del_pacchetto}_{versione_e_revisione}[_architettura].deb
```

Come si vede, la parte finale, che specifica l'architettura per la quale è predisposto un archivio, potrebbe anche essere omessa, se il contesto rende implicita l'informazione.

Alle volte, per qualche ragione, il nome degli archivi che si trovano in circolazione non è conforme a questo modello; tuttavia, con l'aiuto delle informazioni contenute negli archivi stessi, è possibile riprodurre il nome standard. I comandi seguenti, che utilizzano '**`dpkg`**', permettono di ottenere le informazioni necessarie a ricostruire il nome di un archivio Debian:

- `dpkg --field` ***archivio*** `package`

  restituisce il nome del pacchetto;

- `dpkg --field` ***archivio*** `version`

  restituisce la stringa che rappresenta la versione e la revisione del pacchetto.

- `dpkg --field` ***archivio*** `architecture`

  restituisce la stringa che rappresenta l'architettura per cui è fatto il pacchetto.

Quello che segue è l'esempio di uno script in grado di scandire gli archivi contenuti nella directory corrente, allo scopo di modificarne il nome se questo non corrisponde al modello standard. La scansione viene fatta in due fasi, per verificare alla fine quali archivi non sono stati corretti (una copia di questo file, dovrebbe essere disponibile presso *allegati/debian-nomi*).

```
#!/bin/sh
## Interviene nella directory *corrente* correggendo i nomi
## degli archivi che sembrano non essere coerenti. Si
## utilizza in particolare:
##       dpkg --field <archivio> package
##       dpkg --field <archivio> version
##       dpkg --field <archivio> architecture

# Inizializza le variabili di ambiente che servono per
# accumulare i valori per il confronto.
ARCHIVIO=""
PACCHETTO=""
VERSIONE=""
ARCHITETTURA=""
NOME_CORRETTO=""
# Inizia il ciclo di scansione degli archivi Debian che si
# trovano nella directory corrente. Prima fase silenziosa.
for ARCHIVIO in *.deb
do
    # Se il nome è «*.deb», non ci sono file del genere.
    if [ "$ARCHIVIO" = "*.deb" ]
    then
        # Non si fa nulla.
        exit
    fi
    # Estrae il nome del pacchetto.
    PACCHETTO=‘dpkg --field $ARCHIVIO package‘
    # Estrae la versione del pacchetto.
    VERSIONE=‘dpkg --field $ARCHIVIO version‘
    # Estrae la sigla dell’architettura.
    ARCHITETTURA=‘dpkg --field $ARCHIVIO architecture‘
    # Compone il nome teorico.
    NOME_CORRETTO=\
"${PACCHETTO}_${VERSIONE}_${ARCHITETTURA}.deb"
    # Confronta con il nome dell’archivio.
    if [ "$NOME_CORRETTO" != "$ARCHIVIO" ]
    then
        # I nomi sono differenti.
        # Modifica il nome solo se è possibile.
        echo "n" | mv -i "$ARCHIVIO" "$NOME_CORRETTO" \
            2> /dev/null
    fi
done
# Inizia il ciclo di scansione degli archivi Debian che si
# trovano nella directory corrente. Seconda fase.
for ARCHIVIO in *.deb
do
    # Estrae il nome del pacchetto.
    PACCHETTO=‘dpkg --field $ARCHIVIO package‘
    # Estrae la versione del pacchetto.
    VERSIONE=‘dpkg --field $ARCHIVIO version‘
    # Estrae la sigla dell’architettura.
    ARCHITETTURA=‘dpkg --field $ARCHIVIO architecture‘
    # Compone il nome teorico.
    NOME_CORRETTO=\
"${PACCHETTO}_${VERSIONE}_${ARCHITETTURA}.deb"
    # Confronta con il nome dell’archivio.
    if [ "$NOME_CORRETTO" != "$ARCHIVIO" ]
    then
        # A quanto pare, il nome di questo archivio non è
        # stato corretto.
        echo "Non è stato possibile correggere il nome"
        echo "dell’archivio $ARCHIVIO, che dovrebbe"
        echo "chiamarsi $NOME_CORRETTO."
    fi
done
```

7.11.2 Eliminazione delle versioni precedenti

La versione di un pacchetto è composta da due parti: la versione (vera e propria) e la revisione. Si tratta di due stringhe unite da un trattino:

```
versione - revisione
```

In generale, non è facile confrontare questi valori e per fortuna viene in aiuto ‘**dpkg**’ che è in grado di affermare quale sia più recente. In pratica, si utilizza uno dei comandi seguenti, per determinare se una versione è maggiore, minore o uguale all’altra:


```
dpkg --compare-versions versione1 gt versione2
```

```
dpkg --compare-versions versione1 lt versione2
```

```
dpkg --compare-versions versione1 eq versione2
```

Lo script seguente serve a scandire gli archivi Debian contenuti nella directory corrente, allo scopo di eliminare quelli che contengono uno stesso pacchetto ma di una versione precedente a quanto già disponibile. Durante la scansione, si presume che i nomi degli archivi siano composti correttamente, in modo tale che gli archivi di uno stesso pacchetto si trovino di seguito, nella sequenza alfabetica (una copia di questo file, dovrebbe essere disponibile presso *allegati/debian-doppi*).

```
#!/bin/sh
## Interviene nella directory *corrente* eliminando i file
## doppi, più vecchi. Il confronto viene fatto utilizzando:
##       dpkg --field <archivio> package
##       dpkg --field <archivio> version
##       dpkg --field <archivio> architecture
##       dpkg --compare-versions <versione1> eq <versione0>
##       dpkg --compare-versions <versione1> gt <versione0>
##       dpkg --compare-versions <versione1> lt <versione0>

# Inizializza le variabili di ambiente che servono per
# accumulare i valori per il confronto.
ARCHIVIO0=""
ARCHIVIO1=""
PACCHETTO0=""
PACCHETTO1=""
VERSIONE0=""
VERSIONE1=""
ARCHITETTURA0=""
ARCHITETTURA1=""
# Inizia il ciclo di scansione degli archivi Debian che si
# trovano nella directory corrente.
for ARCHIVIO1 in *.deb
do
    # Se il nome è «*.deb», non ci sono file del genere.
    if [ "$ARCHIVIO1" = "*.deb" ]
    then
        # Non si fa nulla.
        exit
    fi
    # Estrae il nome del pacchetto.
    PACCHETTO1=‘dpkg --field $ARCHIVIO1 package‘
    # Estrae la versione del pacchetto.
    VERSIONE1=‘dpkg --field $ARCHIVIO1 version‘
    # Estrae l’architettura.
    ARCHITETTURA1=‘dpkg --field $ARCHIVIO1 architecture‘
    # Confronta con il pacchetto precedente.
    if [ "$PACCHETTO1" == "$PACCHETTO0" ]
    then
        if dpkg --compare-versions "$VERSIONE1" \
                                   eq "$VERSIONE0"
        then
            # Si tratta di un’anomalia in cui si deve
            # intervenire a mano.
            echo "Archivi con la stessa versione:"
            echo "    $ARCHIVIO0"
            echo "    $ARCHIVIO1"
            # In questo caso, non occorre spostare i valori
            # nelle variabili.
        elif dpkg --compare-versions "$VERSIONE1" \
                                   gt "$VERSIONE0"
        then
            # Si elimina l’archivio del pacchetto più
            # vecchio.
            rm -f "$ARCHIVIO0"
            echo "Eliminato $ARCHIVIO0"
            # Sposta i valori nelle variabili.
            ARCHIVIO0="$ARCHIVIO1"
            PACCHETTO0="$PACCHETTO1"
            VERSIONE0="$VERSIONE1"
            ARCHITETTURA0="$ARCHITETTURA1"
        elif dpkg --compare-versions "$VERSIONE1" \
```

```
                                    lt "$VERSIONE0"
        then
            # Si elimina l'archivio del pacchetto più
            # vecchio.
            rm -f "$ARCHIVIO1"
            echo "Eliminato $ARCHIVIO1"
            # In questo caso, non occorre spostare i valori
            # nelle variabili.
        else
            # Questo caso non dovrebbe verificarsi.
            echo "C'è un errore nel confronto degli"
            echo "archivi seguenti:"
            echo "    $ARCHIVIO0"
            echo "    $ARCHIVIO1"
            # In questo caso, non occorre spostare i valori
            # nelle variabili.
        fi
    else
        # Dal momento che i pacchetti sono differenti, si
        # devono salvare le variabili prima di procedere.
        ARCHIVIO0="$ARCHIVIO1"
        PACCHETTO0="$PACCHETTO1"
        VERSIONE0="$VERSIONE1"
        ARCHITETTURA0="$ARCHITETTURA1"
    fi
done
```

7.11.3 Distribuzione completa su un supporto unico

Per comprendere come realizzare una copia locale della distribuzione GNU/Linux Debian, occorre andare per gradi, partendo dal caso in cui questa è disponibile completamente in un supporto unico, per poi arrivare a comprendere le differenze che si devono introdurre per ottenere una versione distribuita su più unità di memorizzazione.

Si suppone che l'utente '**tizio**' voglia predisporre una copia locale della distribuzione Debian, a partire dalla directory '`/home/tizio/DEBIAN/`'. La struttura minima che dovrebbe articolarsi a partire da questa directory è quella che si vede nella figura 7.59.

Figura 7.59. Struttura minima di una distribuzione GNU/Linux Debian organizzata in modo da utilizzare un supporto unico.

```
debian/
  |-- dists/
  |     |-- stable --> ../codename
  |     '-- codename/
  |           |-- main/
  |           |     |-- disks-arch/
  |           |     |     '-- current/
  |           |     '-- binary-arch/
  |           |           |-- Packages
  |           |           |-- Packages.gz
  |           |           |-- admin/
  |           |           |-- base/
  |           |           |-- comm/
  |           |           :
  |           |-- contrib/
  |           |     '-- binary-arch/
  |           |           |-- Packages
  |           |           |-- Packages.gz
  |           |           :
  |           |-- non-free/
  |           |     :
  |           |-- non-US/
  |           |     :
  |           '-- local/
  |                 :
  '-- indices/
        '-- override.gz
```

In breve, viene descritto il senso di questa struttura.

- La directory '`debian/dists/stable/main/disks-i386/current/`' contiene i file delle immagini dei dischetti (per l'architettura x86) da utilizzare per avviare l'installazione e per riprodurre il sistema minimo precedente alla selezione dei pacchetti.
- Le directory '`debian/dists/stable/*/binary-i386/`' contengono la gerarchia finale dei pacchetti di ogni gruppo

('`main/`', '`contrib/`', '`non-free/`', '`non-US/`' e '`local/`'), che potrebbe essere suddivisa o meno in sezioni ('`admin/`', '`base/`', ecc.). Da quel punto, poi, si inseriscono gli archivi Debian.

- I file '`Packages`' e '`Packages.gz`', contenuti nelle directory '`debian/dists/stable/*/binary-i386/`', sono indispensabili per fornire ai programmi come Dselect e APT le informazioni importanti sui pacchetti.
- Il file, o i file '`debian/indices/override*.gz`' servono per ricostruire correttamente i file '`Packages`'.

Il problema nella riproduzione di una distribuzione Debian sta nella creazione dei file '`Packages`' (e di conseguenza anche '`Packages.gz`'). Per arrivare a questo risultato, occorre definire una stringa che serva a individuare la distribuzione, per esempio:

```
Debian GNU/Linux 2.1 slink personalizzata
```

Inoltre occorre un file *override* ('`debian/indices/override.gz`' o un altro nome simile), contenente l'abbinamento tra i pacchetti, la priorità e la classificazione in sezioni, come nell'estratto seguente:

```
at              important       admin
cron            important       admin
locales         standard        admin
ncurses-term    standard        admin
acct            optional        admin
adjtimex        optional        admin
    …
adduser         required        base
ae              required        base
amiga-fdisk     required        base
    …
lilo            important       base
silo            important       base
    …
newt0.25        optional        base
ppp             optional        base
pppconfig       optional        base
syslinux        optional        base
    …
```

Infine occorrono i pacchetti, che si trovano lì dove sono. I file *override* vanno prelevati da una delle varie riproduzioni speculari, tenendo presente che possono essere aggiornati frequentemente. Di solito, questi file sono suddivisi in base ai raggruppamenti principali in cui si articola una versione: '`main/`', '`contrib/`',... Per semplificare le operazioni, può convenire la realizzazione di un file unico, come è stato mostrato nella struttura di esempio. A questo file si fa riferimento come '`override.gz`'.

I file *override* originali della distribuzione '**slink**' hanno i nomi: '`override.slink.gz`', '`override.slink.contrib.gz`', '`override.slink.non-free.gz`' e '`override.slink.non-US.gz`'. Per realizzare la propria copia della distribuzione, nulla vieta di fonderli tutti assieme in un file unico, come descritto in questi esempi, dove si fa riferimento a un solo file '`override.gz`'.

Disponendo di questo materiale, si può utilizzare '**dpkg-scanpackages**' per rigenerare i file '`Packages`'. Eventualmente si può vedere la pagina di manuale *dpkg−scanpackages(8)*.

```
tizio$ cd /home/tizio/DEBIAN/debian [Invio]
```

Ci si posiziona nella directory principale della distribuzione.

```
tizio$ dpkg-scanpackages ↩
↪      -m "Debian GNU/Linux 2.1 slink personalizzata" ↩
↪      dists/stable/main/binary-i386 indices/override.gz ↩
↪      > dists/stable/main/binary-i386/Packages [Invio]
```

Si genera il file ‘Packages’ per il gruppo di pacchetti della classificazione ‘main/’ (il comando è stato mostrato suddiviso su più righe per motivi tipografici).

```
tizio$ cat dists/stable/main/binary-i386/Packages ↩
↪      | gzip ↩
↪      | dists/stable/main/binary-i386/Packages.gz [ Invio ]
```

Si genera il file ‘Packages.gz’, comprimendo ‘Packages’ creato precedentemente.

In seguito, si fa la stessa cosa per i raggruppamenti ‘contrib/’, ‘non-free/’, ‘non-US/’ e ‘local/’.

Anche se alcuni di questi raggruppamenti non vengono utilizzati, nel senso che non si vogliono tenere pacchetti che vi appartengano, è molto importante che siano predisposte le directory vuote e anche i file ‘Packages*’, per facilitare le operazioni con Dselect e APT.

Il programma ‘**dpkg-scanpackages**’ può generare delle segnalazioni di errore, in particolare quando trova un pacchetto che non è indicato nel file *override*. In generale questo non provoca conseguenze gravi, tranne la mancanza di qualche informazione per quel pacchetto.

L’esempio seguente è un estratto di uno dei file ‘Packages’, dove si vede la descrizione del pacchetto ‘**wget**’. Si deve osservare in particolare che le informazioni dei campi ‘**Priority**’ e ‘**Section**’ sono state determinate in base al file *override*, mentre la descrizione del campo ‘**X-Medium**’ è stata ottenuta dall’opzione ‘**-m**’ di ‘**dpkg-scanpackages**’.

```
Package: wget
Version: 1.5.3-1.1
Priority: optional
Section: web
Maintainer: Nicolás Lichtmaier <nick@feedback.net.ar>
Depends: libc6
Architecture: i386
Filename: dists/stable/main/binary-i386/wget_1.5.3-1.1.deb
Size: 221932
MD5sum: 323962a35dabbf88edfe665ad70eb382
Description: utility to retrieve files from the WWW via HTTP and FTP
 Wget [formerly known as Geturl] is a freely available network utility
 to retrieve files from the World Wide Web using HTTP and FTP, the two
 most widely used Internet protocols.  It works non-interactively, thus
 enabling work in the background, after having logged off.
 .
 The recursive retrieval of HTML pages, as well as FTP sites is
 supported -- you can use Wget to make mirrors of archives and home
 pages, or traverse the web like a WWW robot (Wget understands
 /robots.txt).
installed-size: 535
X-Medium: Debian GNU/Linux 2.1 slink personalizzata
```

Per accedere facilmente a questa distribuzione locale, basta configurare APT attraverso il file ‘/etc/apt/sources.list’:

```
deb file:/home/tizio/DEBIAN/debian stable main contrib non-free non-US local
```

Se le dimensioni lo consentono, si può trasferire una copia della gerarchia ‘/home/tizio/DEBIAN/’ in un disco rimovibile, come un CD-ROM o un DVD-ROM.

7.11.4 Distribuzione suddivisa su diversi supporti

Se si vuole produrre un CD/DVD, o comunque si vuole fare una copia della distribuzione suddividendola in più supporti, le cose si complicano. Per prima cosa si deve iniziare da una copia locale organizzata già nelle suddivisioni che si vogliono ottenere. Supponendo di partire dalla directory ‘/home/tizio/DEBIAN/’, conviene aggiungere altre sottodirectory ulteriori, una per ogni suddivisione che si vuole ottenere: ‘1/’, ‘2/’,...

La struttura della gerarchia che si articola a partire da queste sottodirectory deve essere la stessa, anche quando alcuni gruppi di pacchetti (‘main/’, ‘contrib/’, ecc.) risultano senza archivi. La figura 7.64 mostra evidenziate le varianti rispetto al modello già mostrato.

Figura 7.64. Struttura minima di una distribuzione GNU/Linux Debian organizzata in modo da utilizzare più supporti.

```
debian/
  |-- .disk/
  |     ‘-- info
  |-- local/
  |     ‘-- local --> ../stable/local
  |-- dists/
  |     |-- stable --> ../codename
  |     ‘-- codename/
  |           |-- main/
  |           |     |-- disks-arch/
  |           |     |     ‘-- current/
  |           |     ‘-- binary-arch/
  |           |           |-- Packages
  |           |           |-- Packages.gz
  |           |           |-- Packages.cd
  |           |           |-- Packages.cd.gz
  |           |           :
  |           |-- contrib/
  |           |     :
  |           |-- non-free/
  |           |     :
  |           |-- non-US/
  |           |     :
  |           ‘-- local/
  |                 :
  ‘-- indices/
        ‘-- override.gz
```

Rispetto alla situazione precedente, si aggiunge il file ‘debian/.disk/info’, che deve contenere la stringa di descrizione del supporto, una cosa simile ai due esempi seguenti:

```
Debian GNU/Linux 2.1 slink personalizzata disco 1
```

```
Debian GNU/Linux 2.1 slink personalizzata disco 2
```

Nelle directory ‘debian/dists/stable/*/binary-i386/’ appaiono dei file nuovi: ‘Packages.cd’ e ‘Packages.cd.gz’. Infine, il raggruppamento di pacchetti ‘**local**’, dovrebbe trovarsi nella directory ‘debian/dists/local/local/’. Probabilmente, conviene realizzare un collegamento simbolico per portarlo nella collocazione normale.

I supporti distinti, vengono riconosciuti in base alla stringa contenuta nel file ‘debian/.disk/info’, che va scelta opportunamente e va utilizzata anche per la definizione del campo ‘**X-Medium**’.

Si comincia dalla preparazione dei file ‘Packages’ e ‘Packages.gz’, più o meno come è stato fatto nella situazione precedente:

```
tizio$ cd /home/tizio/DEBIAN/1/debian [ Invio ]

tizio$ dpkg-scanpackages -m ‘cat .disk/info‘ ↩
↪      dists/stable/main/binary-i386 indices/override.gz ↩
↪      > dists/stable/main/binary-i386/Packages [ Invio ]

tizio$ cat dists/stable/main/binary-i386/Packages ↩
↪      | gzip ↩
↪      | dists/stable/main/binary-i386/Packages.gz [ Invio ]
```

Come prima, si fa la stessa cosa per gli altri gruppi di pacchetti e poi si ripete il procedimento per la copia contenuta nella directory ‘/home/tizio/DEBIAN/2/’ (si suppone che si tratti di una suddivisione in due soli supporti):

```
tizio$ cd /home/tizio/DEBIAN/2/debian [ Invio ]

tizio$ dpkg-scanpackages -m ‘cat .disk/info‘ ↩
↪      dists/stable/main/binary-i386 indices/override.gz ↩
↪      > dists/stable/main/binary-i386/Packages [ Invio ]

tizio$ cat dists/stable/main/binary-i386/Packages ↩
↪      | gzip ↩
↪      | dists/stable/main/binary-i386/Packages.gz [ Invio ]
```

Alla fine, si devono realizzare i file ‘Packages.cd’, che si compongono della somma dei file ‘Packages’ di ogni gruppo:

```
tizio$ cd /home/tizio/DEBIAN/ [ Invio ]

tizio$ cat 1/debian/dists/stable/main/binary-i386/Packages ↩
↪      2/debian/dists/stable/main/binary-i386/Packages ↩
```

```
↪     > 1/debian/dists/stable/main/binary-i386/Packages.cd [Invio]

tizio$ cat 1/debian/dists/stable/main/binary-i386/Packages ↩
↪     2/debian/dists/stable/main/binary-i386/Packages ↩
↪     > 2/debian/dists/stable/main/binary-i386/Packages.cd [Invio]

tizio$ cat 1/debian/dists/stable/main/binary-i386/Packages.cd ↩
↪     | gzip ↩
↪     | 1/debian/dists/stable/main/binary-i386/Packages.cd.gz [Invio]

tizio$ cat 2/debian/dists/stable/main/binary-i386/Packages.cd ↩
↪     | gzip ↩
↪     | 2/debian/dists/stable/main/binary-i386/Packages.cd.gz [Invio]
```

In pratica, i file ‘`Packages.cd`’ contengono le informazioni su tutti i pacchetti del proprio gruppo; sia quelli presenti effettivamente nel supporto, sia quelli che si trovano negli altri. I programmi come Dselect distinguono il supporto in base al nome che gli è stato attribuito, indicato nel file ‘`debian/.disk/info`’ e riportato nel campo ‘**`X-Medium`**’ dei file ‘`Packages.cd*`’.

Per accedere facilmente a questa distribuzione locale, spezzata in due o più parti, basta configurare APT attraverso il file ‘`/etc/apt/sources.list`’:

```
deb file:/home/tizio/DEBIAN/1/debian stable main contrib non-free non-US local
deb file:/home/tizio/DEBIAN/2/debian stable main contrib non-free non-US local
…
```

Per copiare le due strutture in dischi separati, basta trasferire una copia delle gerarchie ‘`/home/tizio/DEBIAN/*/`’.

7.11.5 Un esempio banalizzato, ma funzionante

Viene proposta in questa sezione un’organizzazione semplificata per ospitare i pacchetti Debian che si raccolgono qua e là. Si parte da una struttura, di alcuni file e directory, che per il momento viene semplificata:

```
.
|-- DIST
|-- Packages.cd.gz
|-- dists/
|   |-- 00 -> /tmp
|   |-- 01/
|   |-- 02/
|   `-- 03/
|-- indices/
|   |-- Maintainers.gz
|   |-- override.gz
|   `-- pubring.pgp
`-- make-packages
```

Si può osservare che la directory ‘`dists/00/`’ è in realtà un collegamento simbolico alla directory temporanea (‘`/tmp/`’).

Il file ‘`DIST`’ deve contenere una riga, con la denominazione della distribuzione, che eventualmente può limitarsi a essere:

```
Debian GNU/Linux
```

I file ‘`Packages.cd.gz`’, ‘`indices/Maintainers.gz`’ e ‘`indices/override.gz`’ sono file compressi con ‘**`gzip`**’, a partire da file completamente vuoti; il file ‘`indices/pubring.pgp`’ è vuoto.

All’interno delle directory ‘`dists/01/`’, ‘`dists/02/`’ e ‘`dists/03/`’ ci sono dei collegamenti simbolici identici nei tre casi:

```
.
|-- dists/
|   |-- 01/
|   |   |-- binary-all -> .
|   |   |-- binary-i386 -> .
|   |   |-- contrib -> .
|   |   |-- local -> .
|   |   |-- main -> .
|   |   |-- non-US -> .
|   |   `-- non-free -> .
:   :
```

Con questa struttura, si possono mettere i file degli archivi Debian nelle directory ‘`dists/01/`’, ‘`dists/02/`’ e ‘`dists/03/`’, tenendo conto che in presenza di pacchetti doppi, ma con versione differente, quelli riferiti alla versione più vecchia vengono spostati a un livello precedente tramite uno script. In pratica, secondo questa sistemazione, i pacchetti che si raccolgono vanno inseriti soltanto nella directory ‘`dists/03/`’, dalla quale i più vecchi vengono spostati automaticamente all’interno di ‘`dists/02/`’, di ‘`dists/01/`’ e di ‘`dists/00/`’, ma dato che ‘`dists/00`’ è in realtà un collegamento simbolico a ‘`/tmp/`’, l’ultimo passaggio è semplicemente un modo per eliminarli. Così facendo, anche se si aggiornano i pacchetti, rimane una copia di scorta delle versioni precedenti.

Lo script che fa il lavoro è ‘**`make-packages`**’, che va avviato mentre la directory corrente corrisponde all’inizio della struttura; pertanto va avviato così:

```
$ ./make-packages [Invio]
```

Lo script crea anche il file ‘`.disk/info`’ e il file ‘`error-messages`’ (una copia di questo file dovrebbe essere disponibile presso *allegati/make-packages*).

Listato 7.71. Lo script ‘**`make-packages`**’.

```
#!/bin/sh
# La directory di partenza deve essere quella da cui si
# dirama la directory "dists/"
START_DIR=‘pwd‘
# Definisce il file da usare per i messaggi di errore e lo
# svuota.
ERROR_MESSAGES="$START_DIR/error-messages"
echo > $ERROR_MESSAGES
# Definisce il contenuto dei file ".disk/info"
# ATTENZIONE! il file ".disk/info" è obbligatorio per
# "dselect"!
DISTRIBUTION=‘cat $START_DIR/DIST‘
DATE=‘date "+%Y.%m.%d %H:%M"‘
rm -rf $START_DIR/.disk
mkdir $START_DIR/.disk
echo "$DISTRIBUTION $DATE" > $START_DIR/.disk/info
# Carica il nome che identifica i supporti.
# ATTENZIONE! il file ".disk/info" è obbligatorio per
# "dselect"!
MEDIA=‘cat .disk/info‘
# Dichiara la funzione che sistema i file.
function debian_arrange_files ()
{
    DISTRIBUTION="$1"
    PREVIOUS_LEVEL="$2"
    # Inizializza le variabili di ambiente che servono per
    # accumulare i valori per il confronto.
    ARCHIVE_0=""
    ARCHIVE_1=""
    PACKAGE_0=""
    PACKAGE_1=""
    VERSION_0=""
    VERSION_1=""
    ARCHITECTURE_0=""
    ARCHITECTURE_1=""
    CORRECT_NAME_0=""
    CORRECT_NAME_1=""
    # Inizia il ciclo di scansione degli archivi Debian che
    # si trovano nella directory corrente.
    for ARCHIVE_1 in *.deb
    do
        # Se il nome è «*.deb», non ci sono file del genere.
        if [ "$ARCHIVE_1" = "*.deb" ]
        then
            # Non si fa nulla.
            return
        fi
        # Estrae il nome del pacchetto.
        PACKAGE_1=‘dpkg --field $ARCHIVE_1 package‘
        # Estrae la versione del pacchetto.
        VERSION_1=‘dpkg --field $ARCHIVE_1 version‘
        # Estrae il nome dell’architettura.
        ARCHITECTURE_1=\
‘dpkg --field $ARCHIVE_1 architecture‘
        # Compone il nome teorico.
        CORRECT_NAME_1=\
"${PACKAGE_1}_${VERSION_1}_${ARCHITECTURE_1}.deb"
        # Confronta con il nome dell’archivio.
        if [ "$CORRECT_NAME_1" != "$ARCHIVE_1" ]
        then
            # I nomi sono differenti.
```

```
            # Modifica il nome forzando la situazione (prima
            # si usava l'opzione -i).
            echo "n" | mv -f "$ARCHIVE_1" \
                            "$CORRECT_NAME_1" 2> /dev/null
            echo "$ARCHIVE_1 ---> $CORRECT_NAME_1"
            echo "$ARCHIVE_1 ---> $CORRECT_NAME_1" \
                 >> $ERROR_MESSAGES
            # Meglio annotare il nome nuovo nella variabile.
            ARCHIVE_1="$CORRECT_NAME_1"
        fi
        # Confronta con il pacchetto precedente.
        if [ "$PACKAGE_1" == "$PACKAGE_0" ]
        then
            if dpkg --compare-versions "$VERSION_1" \
                                       eq "$VERSION_0"
            then
                # Si tratta di un'anomalia in cui si deve
                # intervenire a mano.
                echo "Same version: $ARCHIVE_0 $ARCHIVE_1"
                echo "Same version: $ARCHIVE_0 $ARCHIVE_1"\
                     >> $ERROR_MESSAGES
                # In questo caso, non occorre spostare i
                # valori nelle variabili.
            elif dpkg --compare-versions "$VERSION_1" \
                                         gt "$VERSION_0"
            then
                # Si sposta l'archivio doppio nel livello
                # precedente.
                echo \
"$DISTRIBUTION/$ARCHIVE_0 --> $PREVIOUS_LEVEL"
                mv $ARCHIVE_0 ../$PREVIOUS_LEVEL/$ARCHIVE_0
                # Sposta i valori nelle variabili.
                ARCHIVE_0="$ARCHIVE_1"
                PACKAGE_0="$PACKAGE_1"
                ARCHITECTURE_0="$ARCHITECTURE_1"
                VERSION_0="$VERSION_1"
                CORRECT_NAME_0="$CORRECT_NAME_1"
            elif dpkg --compare-versions "$VERSION_1" \
                                         lt "$VERSION_0"
            then
                # Si sposta l'archivio doppio nel livello
                # precedente.
                echo \
"$DISTRIBUTION/$ARCHIVE_1 --> $PREVIOUS_LEVEL"
                mv $ARCHIVE_1 ../$PREVIOUS_LEVEL/$ARCHIVE_1
                # In questo caso, non occorre spostare i
                # valori nelle variabili.
            else
                # Questo caso non dovrebbe verificarsi.
                echo "There is an error matching the"
                echo "following archives:"
                echo "    $ARCHIVE_0"
                echo "    $ARCHIVE_1"
                echo "There is an error matching the"\
                echo "following archives:"
                     >> $ERROR_MESSAGES
                echo "    $ARCHIVE_0"\
                     >> $ERROR_MESSAGES
                echo "    $ARCHIVE_1"\
                     >> $ERROR_MESSAGES
                # In questo caso, non occorre spostare i
                # valori nelle variabili.
            fi
        else
            # Dal momento che i pacchetti sono differenti,
            # si devono salvare le variabili prima di
            # procedere.
            ARCHIVE_0="$ARCHIVE_1"
            PACKAGE_0="$PACKAGE_1"
            ARCHITECTURE_0="$ARCHITECTURE_1"
            VERSION_0="$VERSION_1"
            CORRECT_NAME_0="$CORRECT_NAME_1"
            # mostra cosa è stato fatto
            echo "$PACKAGE_1"
        fi
    done
}
# Dichiara la funzione che esegue le operazioni ripetitive
# su ogni sezione di ogni distribuzione, per ogni disco.
function process_disk_distribution_section ()
{
    MEDIA="$1"
    DISTRIBUTION="$2"
    PREVIOUS_LEVEL="$3"
    SECTION="$4"
    echo "$MEDIA, $DISTRIBUTION, $SECTION"
    cd $START_DIR/dists/$DISTRIBUTION/$SECTION/binary-i386
    debian_arrange_files $DISTRIBUTION $PREVIOUS_LEVEL
    cd $START_DIR
    dpkg-scanpackages                                    \
       dists/$DISTRIBUTION/$SECTION/binary-i386          \
       $START_DIR/indices/override.gz                    \
       > dists/$DISTRIBUTION/$SECTION/binary-i386/Packages \
       2>> $ERROR_MESSAGES
    cat dists/$DISTRIBUTION/$SECTION/binary-i386/Packages  \
        | gzip -9                                        \
    > dists/${DISTRIBUTION}/$SECTION/binary-i386/Packages.gz
}
# Funzione frontale.
function process_section () {
    DISTRIBUTION="$1"
    PREVIOUS_LEVEL="$2"
    SECTION="$3"
    process_disk_distribution_section \
        "$MEDIA" $DISTRIBUTION $PREVIOUS_LEVEL $SECTION
    # Prepara i file "Packages.cd" e "Packages.cd.gz"
    cd $START_DIR
    cat dists/$DISTRIBUTION/$SECTION/binary-i386/Packages  \
      > dists/$DISTRIBUTION/$SECTION/binary-i386/Packages.cd
    cd $START_DIR/dists/$DISTRIBUTION/$SECTION/binary-i386
    cat Packages.cd | gzip -9 > Packages.cd.gz
    cd $START_DIR
}
# Inizia il programma.
cd $START_DIR
process_section 03 02 main
process_section 02 01 main
process_section 01 00 main
```


7.12 Riferimenti

- *The Debian GNU/Linux FAQ*, http://ftp.it.debian.org/debian/doc/FAQ/
- Ian Jackson, Christian Schwarz, *Debian Po, al* http://www.debian.org/doc/debian-policy/

[1] Quando si installa un programma fornito in forma già compilata per la propria piattaforma, si possono incontrare dei problemi se il pacchetto di distribuzione del programma non è stato predisposto specificatamente per l'organizzazione della propria distribuzione GNU, perché diversamente è prassi normale che il pacchetto in questione contenga tutte le informazioni sulle dipendenze e le eventuali incompatibilità.

[2] In fase di aggiornamento dei pacchetti di una distribuzione Debian, di norma si viene interrogati sul da farsi, a proposito della configurazione preesistente.

[3] **Dpkg** GNU GPL

[4] **Cruft** GNU GPL

[5] **Deborphan** GNU GPL o Artistic

[6] **Orphaner** GNU GPL o Artistic

[7] **Debconf** software libero con licenza speciale

[8] **Configure-debian** GNU GPL

[9] **Dselect** GNU GPL

[10] Per la precisione, si usa l'estensione '`.dpkg-dist`' quando il file in questione rappresenta la configurazione proposta dalla distribuzione, mentre si usa '`.dpkg-old`', quando il file rappresenta la configurazione precedente che è stata sostituita a seguito di una risposta affermativa da parte di colui che esegue l'aggiornamento.

[11] **APT** GNU GPL