Terza Edizione Ottobre 2007

Se i tempi non chiedono la tua parte migliore
inventa altri tempi

Baolian, libro II, vv. 16-17

Note e Convenzioni

#	questo simbolo indica un comando eseguito da un utente con privilegi di amministratore di sistema (*root*)
$	questo simbolo indica un comando eseguito da un utente normale

Dopo la scrittura di ogni comando è sottintesa la pressione del tasto invio (*enter*).
L'eventuale equivalente inglese dei termini è indicato in corsivo, così come nelle righe precedenti.
I caratteri inseriti da tastiera e tutto quanto mostrato a video dal computer sono indicati nello stile `macchina da scrivere`.
Si assume che il lettore disponga (fisicamente su CD/DVD oppure via rete) del corredo completo dei pacchetti software componenti l'attuale distribuzione Debian stabile (Debian 4.0 Etch) nella variante che include GNU/Linux; nel documento si fa riferimento alla versione in lingua inglese per piattaforma x86.
Il documento è stato scritto con LyX e LaTeX 2_ε, mentre la figura della copertina è stata elaborata con GIMP (affettuosamente noto come *The GIMP*).

Martin F. Krafft è l'autore della figura sugli archivi Debian, basata a sua volta su quella di Kevin Mark.
Un grazie a Paolo Didonè e Pierpaolo Toniolo, per l'aiuto prestato quando ho cominciato ad usare Debian e Linux. La sezione sulla compilazione del kernel deve molto all'articolo di Jesse Goerz del progetto "Newbiedoc".
Hanno collaborato: Roberto Rossetto, Darren Salt, Riccardo Brigo, Sandro Tosi.
Il presente manuale esiste in forma di pacchetto non ufficiale debian binario e sorgente a cura dell'autore; il pacchetto sorgente **debianadv** è reperibile presso:

`http://mentors.debian.net`

mentre quello binario dovrebbe essere scaricabile da:

`http://www.itisnatta.eu/~rob66`

e vi installerà automaticamente questo manuale nei formati testo e pdf, che poi potrete consultare in **dwww** (vedasi paragrafo 5.1 a pagina 49) dal menù Documentazione Debian/Debian. Per installare il pacchetto binario **debianadv** si può fare riferimento al paragrafo 3.4 a pagina 39.

Prefazione

Quest'opera nasce dallo studio appassionato, dal contenuto dei classici tre grossi blocchi di appunti (compilati con quanto si andava leggendo dalle liste di posta, dalle riviste, dalla varia documentazione disponibile sul sistema e in rete), e naturalmente dall'esperienza d'uso accumulata dall'autore.

L'opera è diretta principalmente all'utente domestico "semplice", che usa il computer e Linux per scrivere, giocare, navigare, ma che vuole approfondire le tematiche di un sistema Debian; è tuttavia probabile che essa possa anche essere utile all'utente esperto proveniente da altra distribuzione Linux, che vuole rapidamente orientarsi in Debian. In effetti, la dispersione dell'informazione e la scarsità di documentazione unificatrice sono stati i motivi ispiratori dell'opera.

Da questa impostazione deriva la scelta di non trattare né argomenti inerenti i servizi di rete lato server, che interessano appunto le macchine server, né l'installazione del sistema operativo nel computer, perché si ritiene che il lettore abbia già superato questa fase, autonomamente o mediante l'aiuto di persone più esperte, né, a meno di rare eccezioni, la configurazione personalizzata dei programmi, per la quale di solito esiste ampia documentazione, appannaggio peraltro di utenti molto esperti. Come dice il titolo, viene trattato solo l'uso "normale" dei programmi e dei vari comandi, facendo riferimento al loro funzionamento predefinito, spesso direttamente tramite concisi esempi d'uso.

Un utilizzo avanzato delle caratteristiche di Debian deve necessariamente passare per l'uso, ancorché moderato, della shell, che viene pertanto introdotta fin dalle primissime pagine: ciò giustifica la presenza di alcuni argomenti che sono parte del sistema operativo Linux più che specificità di Debian.

Si è cercato di realizzare una facile esposizione, facendo però l'importante assunzione che il lettore sia a conoscenza dei concetti base dell'informatica (come ad es. quelli di file, periferica, software di sistema, software applicativo).

Molti argomenti complessi, come la compilazione dei programmi, sono accennati a livello introduttivo, in modo che il lettore li possa comunque sperimentare, apprezzare, per poi eventualmente approfondirli per proprio conto, studiando i testi consigliati.

Lo scopo dell'opera è mostrare sinteticamente ed in modo discorsivo ed accessibile le principali notevoli caratteristiche avanzate di Linux e Debian, in modo da permettere al lettore di progredire rapidamente nelle sue conoscenze informatiche meno elementari, acquisendo nel contempo quella visione d'insieme necessaria per muoversi nei confini dell'universo Debian.

MAURO DARIDA <mauro.darida (ad) istruzione.it>

Indice

1 In principio fu Debian

1.1 DFSG: Linee Guida di Debian per il Software Libero

Debian è un significativo distributore di Linux, con uno statuto, un contratto sociale[1] (`http://www.debian.org/social_contract`) e documenti di politica di organizzazione del progetto stesso. Il progetto Debian si è sempre posto obiettivi elevati, per produrre una distribuzione che sia veramente all'altezza del nome e dello spirito di Linux: ogni versione stabile[2] di Debian viene rilasciata esclusivamente sulla base della qualità del sistema e dell'assenza di significativi problemi.

Un utente è opportuno che sia a conoscenza della filosofia e ideologia che sta dietro il flessibile e potente sistema Debian, il quale viene (apparentemente) regalato senza contropartita alcuna. Ogni software contenuto in un pacchetto Debian deve essere accompagnato da una licenza che lo possa far definire software libero[3]. Debian definisce libero un software se la sua licenza soddisfa tutti i punti delle DFSG. Le DFSG sono state la base, con lievi modifiche, di quella che è oggi nota come *Open Source Definition* ad opera di Bruce Perens (Fig.1.1), uno dei primi "direttori" di Debian, dall'aprile 1996 al dicembre 1997[4].
Per comodità del lettore vengono qui riportate le DFSG (*Debian Free Software Guidelines).*

1. Libera distribuzione

La licenza di un componente Debian non può porre restrizioni sulla vendita o cessione del software, come componente di una distribuzione di software aggregato, contenente pro-

[1] Anche digitando: `$ less /usr/share/doc/debian/social-contract.1.1.txt`.
[2] Si veda il paragrafo 1.2 seguente.
[3] In realtà ci sono delle deroghe a questo, ma il software che deroga (entro certi limiti) non è parte integrante della distribuzione Debian. Esso viene inserito nelle sezioni denominate *non-free* e *contrib* e fruisce delle risorse di Debian, ma non lo troverete sui cd/dvd.
[4] Per approfondire il passato di Debian si veda il pacchetto debian-history.

Figura 1.1: B. Perens

grammi provenienti da fonti diverse. La licenza non può richiedere nessun pagamento aggiuntivo o altra tassa su tale vendita.

2. Codice sorgente
Il programma deve includere il codice sorgente e permetterne la distribuzione sia in forma di codice binario che di sorgente.

3. Opere derivate
La licenza deve permettere modifiche e opere derivate, e deve permettere che esse siano distribuite sotto i medesimi termini della licenza del software originale.

4. Integrità del codice sorgente
La licenza può proibire che il codice sorgente venga distribuito in forma modificata solo se la licenza permette la distribuzione di patch file con il codice sorgente allo scopo di modificare il programma al momento della costruzione. La licenza può richiedere che opere derivate siano designate con un nome o numero di serie diverso da quello del software originale (purtroppo questa clausola è molto tecnica e oscura ma comunque rappresenta un compromesso poco importante: da notare che la GPL non ne ha bisogno per qualificarsi).

5. Nessuna discriminazione contro persone o gruppi
La licenza non può porre discriminazioni verso persone o gruppi.

6. Nessuna discriminazione contro campi di attività
La licenza non può porre restrizioni sull'uso del programma in uno specifico campo di attività. Per esempio, non può proibire l'uso del programma nella ricerca genetica.

7. Distribuzione della licenza
I diritti connessi al programma si devono trasferire a tutti coloro ai quali il programma è distribuito, senza bisogno di istituire licenze addizionali verso terzi.

8. La licenza non deve essere specifica per Debian
I diritti connessi al programma non devono essere subordinati al fatto che il programma sia parte di un sistema Debian. Se il programma è estratto da Debian e usato e distribuito al di fuori di Debian, ma secondo i termini della licenza del programma, a tutti coloro ai quali è distribuito il programma vanno riconosciuti gli stessi diritti di coloro che lo usano in un sistema Debian.

9. La licenza non deve contaminare altro software
La licenza di un software non deve porre restrizioni sulla distribuzione con altro software. Per esempio, la licenza non deve pretendere che altri programmi distribuiti sul medesimo supporto siano software libero.

10. Esempio di (valide) licenze
GPL, BSD, Artistica, sono esempi di licenze che qualificano il software come libero.

Come si vede, Debian favorisce uno scenario digitale aperto e moderno, mentre potenti forze spingono su una tecnologia informatica in possesso di poche multinazionali, capaci di condizionare pesantemente le cosiddette riviste del settore e gli organi istituzionali, negando la condivisione decentrata del sapere alla base della scienza moderna, in nome del massimo profitto.

In Italia esistono norme legislative, anche di recente introduzione, che ignorano i concetti del software libero, di fatto penalizzando questa categoria di software. Non è detto che sia moralmente accettabile, come si sente spesso dire, che come cittadini siamo obbligati a rispettare delle leggi manifestamente ingiuste, anche se come cittadini siamo naturalmente obbligati a sopportare le conseguenze della nostra insubordinazione[5]. Vorrei qui sottolineare che quando si afferma che Richard Stallman ed altri hanno ***legalmente*** sviluppato concretamente l'idea del software libero, ci si dimentica di notare che ciò è accaduto quando Linux non aveva alcuna importanza economica ed era usato da una ristrettissima minoranza; la comparsa di rilevanti interessi economici impedisce tale sviluppo innanzitutto rendendolo **illegale** mediante l'approvazione di leggi ad hoc .

Su piani diversi da quello legale invece si impedisce tale sviluppo con azioni tendenti a confondere e stemperare l'idea iniziale (**tecniche FUD**: *Fear Uncertainty and Doubt*): a tale proposito cito le numerose riviste e i libri con "**tutti i diritti riservati**" che si occupano di Linux (e di Debian) e di informatica; del resto, il concetto di software libero (*open source*, o comunque si voglia chiamarlo) è in netto contrasto con alcune pratiche di marketing consolidate come l'esclusiva (non a caso chiamata **privativa** dagli addetti ai lavori) sui diritti di sfruttamento commerciale, sui diritti di riproduzione, sui diritti di distribuzione, che minacciano non solo di vanificare le enormi possibilità della nuova economica riproduzione digitale per la diffusione dei prodotti dell'ingegno umano, ma anche di cambiare cose più antiche come il modo di condurre la ricerca scientifica e la libertà di pubblicazione dei ricercatori sui risultati delle loro ricerche scientifiche[6]. Se non vi sono circostanze drammatiche legate alla sopravvivenza di una società "gli scienziati di tutti i paesi dovrebbero opporsi a qualsiasi limitazione della libertà di comunicazione scientifica e di diffusione dell'innovazione tecnologica. Essi si devono impegnare a rispettare le regole di un'etica che sostanzialmente concepisca il lavoro dello scienziato come un'espressione dell'esigenza dell'homo sapiens di **comprendere** e quella del tecnologo come un'espressione della tendenza dell'homo sapiens ad **applicare** le conoscenze scientifiche per migliorare le condizioni di vita sue e della sua progenie".[7]

1.2 Le tre facce di Debian: stable, testing, unstable

Attualmente gli sviluppatori hanno organizzato Debian in tre distribuzioni: stabile (*stable*), di prova (*testing*), instabile (*unstable*). Lo scopo è quello di facilitare il collaudo dei vari pezzi della futura versione della distribuzione stabile, che mediamente è composta da migliaia di pacchetti software.

In questo schema il nuovo software (per esempio, una nuova versione di OpenOffice) viene introdotto nella distribuzione instabile; dopo qualche tempo (circa dieci giorni), se il software funziona bene, viene introdotto anche nella distribuzione di prova. La distribuzione stabile rimane invece sempre uguale a se stessa, subendo solo minimi aggiornamenti per migliorarne la sicurezza, almeno finché non si decide che la distribuzione di prova è abbastanza matura da divenire la nuova versione della distribuzione stabile. In questo caso la distribuzione di prova viene congelata (*frozen testing*) e non riceve più nuovo software dalla

[5] Il concetto non è nuovo; si veda H.D.Thoreau, Disobbedienza Civile.
[6] Si veda anche D. Nelkin, Science as intellectual property, Macmillan, 1984
[7] Alfonso Maria Liquori, Etica ed estetica della scienza, Di Renzo editore, Roma, 2003

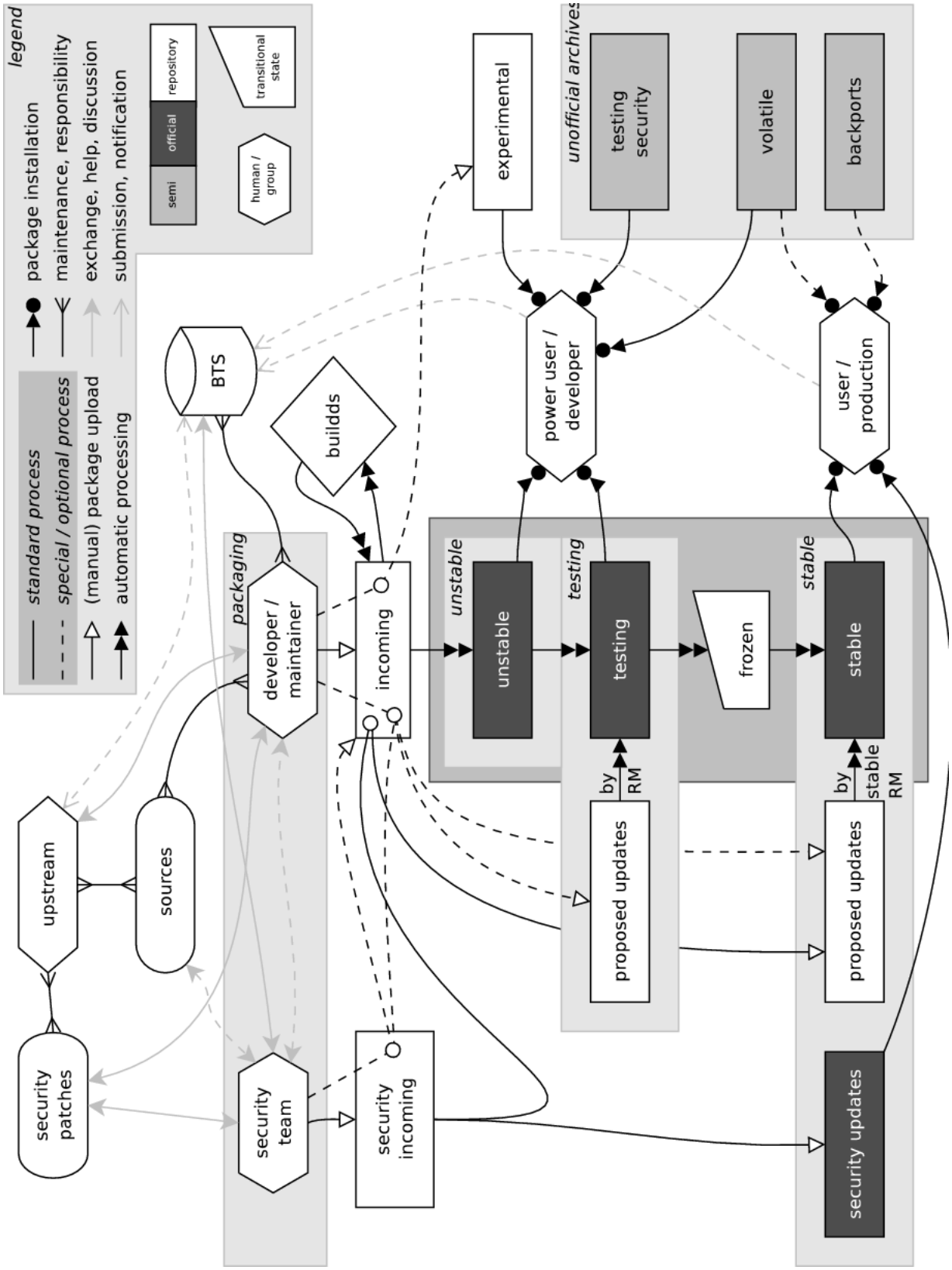

Figura 1.2: Gli archivi Debian (©2005 Martin F. Krafft)

Tabella 1.1: Le piattaforme di Debian

Piattaforma	Note
ia64	Computer con processore Intel Itanium a 64 bit
alpha	Workstation DEC Alpha e compatibili
arm	Sharp Zaurus, Acorn Risc PC, RiscStation
hppa	Vecchie workstation di HP con processore PA-RISC
s390	Computer mainframe di IBM mod. S/390
x86	AMD/Intel sui comuni computer e portatili
amd64	Computer con processore AMD64 o Intel EM64T
mips/mipsel	Workstation di Silicon Graphics
powerpc	Moderni Apple (Power Macintosh)
sparc	Computer con i processsori di Sun Microsystems

instabile (Fig.1.2) mentre nel contempo si effettuano degli ultimi cicli di collaudo (*test cycles*) nell'imminenza del rilascio.

Tutte e tre le distribuzioni sono pubblicamente disponibili in rete. L'intuizione che stabilità e sicurezza aumentino andando dalla distribuzione instabile a quella stabile è errata. Infatti, se si verificano delle falle di sicurezza in qualche software, esse sono coperte quasi immediatamente in *stable*, mentre *testing* deve aspettare. Inoltre se qualche programma che è filtrato in *testing* viene a manifestare dei bachi[8], si determina spesso una situazione, a causa delle intricate dipendenze, tale che i bachi in *testing* resteranno finchè non verrà introdotta da *unstable* la nuova versione corretta. Questo stato di cose rende la distribuzione di prova la meno usabile e sicura delle tre, tanto che qualcuno ha proposto di renderla non pubblica, per sfruttarla solo per gestire la transizione della distribuzione di prova alla nuova versione della stabile.

Qualunque sia la decisione che prenderete in merito, considerate che la distribuzione stabile è caratterizzata da elevata stabilità e sicurezza, ma tende però ad essere costituita da software piuttosto datato, visti i tempi di rilascio delle versioni stabili (superiori ad un anno, finora). Viene consigliata per le macchine server, ma nulla vieta di usarla su una macchina da scrivania di uso quotidiano; lo svantaggio della vecchiezza del software si può superare ricorrendo a qualche pacchetto non ufficiale compilato per la distribuzione stabile (*backport for stable*).

Per contro la distribuzione instabile, continuamente aggiornata, è caratterizzata dall'avere sempre le ultimissime versioni del software in circolazione. Viene consigliata per le macchine di uso quotidiano, perché ritenuta più "divertente", ammesso che vi diverta stare a smanettare sul computer perchè qualcosa non funziona.

Ogni distribuzione Debian è anche caratterizzata da un nome in codice: quello della distribuzione instabile rimane sempre Sid (Sid è il nome del ragazzo della porta accanto che rompe i giocattoli nel film animazione "Toy Story") mentre gli altri nomi cambiano con la successione nel tempo delle distribuzioni stabili e di prova.

[8] Si veda il paragrafo 8.3 a pagina 69.

E' molto probabile, anzi desiderabile, che abbiate installato la distribuzione stabile[9]. Se avete dei dubbi, basta andare a guardare cosa c'è nel file /etc/apt/sources.list; dovreste vedere delle righe che contengono la parola "stable", simili alla seguente:

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

Se invece, per intenderci, state facendo girare la distribuzione instabile, vedreste qualcosa del tipo[10]:

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

Fate attenzione al fatto che aggiornare tutto il sistema alla distribuzione instabile costituisce un passo irreversibile: il gestore dei pacchetti aggiornerà automaticamente tutto il sistema alla instabile, ma non sarà possibile degradare il sistema automaticamente seguendo il percorso inverso. Pertanto, se avete aggiornato tutto il sistema alla instabile e vi siete pentiti, l'unico modo per tornare alla distribuzione stabile è di installare da capo l'intero sistema.

Ogni distribuzione stabile è disponibile per molte piattaforme, a ciscuna delle quali corrisponde un corredo di pacchetti software in formato binario e sorgente; attualmente sono trattate quelle descritte in tabella 1.1.

Talvolta potreste infine imbattervi nella "distribuzione" *experimental*: non si tratta di una vera e propria distribuzione, ma di un ambiente in cui gli sviluppatori certe volte pongono dei componenti software che considerano appunto sperimentali; di norma un componente software entra negli archivi Debian direttamente da *unstable* (Fig.1.2).

Dalla versione 4.0 Debian mette a disposizione il servizio di debian-volatile, che come suggerisce il nome, risolve il problema di aggiornare basi di dati in continuo mutamento, come ad esempio la base di dati di virus di pacchetti come l'antivirus clamav, pur nell'ambito statico di una distribuzione stabile, ove non ci sono mai nuove versioni di software; viene pure garantito l'aggiornamento senza problemi alle future versioni stabili di Debian. Per avvalersi del servizio bisogna modificare il file /etc/apt/sources.list aggiungendo la seguente riga:

deb http://volatile.debian.org/debian-volatile etch/volatile main contrib non-free

[9] La distribuzione instabile non è adatta ai principianti.

[10] Purtroppo questo metodo potrebbe non funzionare su fonti di tipo "cdrom" che fanno riferimento ad *unstable*, pur contenendo di fatto un archivio *stable*. In questo caso fa fede ciò che dice il fornitore dei cdrom.

2 Linux precipitevolissimevolmente

2.1 Anatomia di un file

Nei sistemi Linux l'utente ha accesso ad un'unica collezione di oggetti organizzata ad albero, la cui radice è la cosiddetta "cartella radice" (*root directory*), identificata dal carattere "/", all'interno della quale risiedono tutti i file e le cartelle del nostro sistema. Se facessimo elencare gli attributi di un file al sistema vedremmo qualcosa del tipo:

```
-rw-r--r-- 1 mario staff 0 Aug 23 12:56 prova
```

Possiamo osservare che gli attributi del file sono raggruppati in sette colonne a partire da sinistra verso destra. La prima colonna rappresenta il tipo di file e i permessi ad esso associati: il primo carattere individua il tipo di file, che in questo caso è un file normale, essendoci il trattino.

I rimanenti nove caratteri costituiscono appunto i permessi associati al file. I permessi sono, nell'ordine, di tre tipi:

r (*read*) sta per lettura

w (*write*) sta per scrittura

x (*execute*) sta per esecuzione

I primi tre caratteri definiscono i permessi accordati all'utente proprietario del file: nell'esempio sono i permessi di lettura e scrittura, mentre il trattino successivo sta a significare che è negato il permesso di esecuzione[1]. I successivi caratteri definiscono i permessi accordati al gruppo del file: nell'esempio solo la lettura, perchè scrittura ed esecuzione sono sostituiti dal solito trattino. Gli ultimi tre caratteri della prima colonna definiscono i permessi accordati a tutti gli altri utenti (**escluso l'utente root, che ha sempre pieno accesso a qualunque risorsa**): nell'esempio solo la lettura. Ciò significa che il file "prova" può essere modificato, cioè "scritto", solo dall'utente mario proprietario del file, mentre tutti gli altri utenti potranno solo limitarsi a leggerne il contenuto. Bisogna dire che il concetto di esecuzione varia a seconda che sia applicato ad un file oppure ad una cartella; nel primo caso indica che si tratta di un codice eseguibile (un programma), nel secondo invece x assume il significato di permesso di accesso alla cartella. Ciò non è da confondere col permesso di lettura, che continua ad essere gestito da r.
La seconda colonna indica il numero di collegamenti effettivi (*hard links*) al file.

[1] Un errore spesso commesso dai principianti è quello di dimenticare di attivare il permesso di esecuzione di uno script dopo averlo creato, con conseguente rifiuto del sistema di eseguirlo.

La terza e quarta colonna indicano rispettivamente l'utente e il gruppo cui il file appartiene: nell'esempio sono l'utente mario e il gruppo staff.
La quinta colonna indica le dimensioni in byte del file: zero nel nostro esempio.
La sesta colonna fornisce la data e l'ora in cui il file ha subito l'ultima modifica; infine la settima colonna, come avrete capito, indica il nome attribuito al file ("prova" nel caso dell'esempio).

In un sistema Debian ci sono numerosi gruppi predefiniti, in aggiunta a quelli che eventualmente possono essere creati dall'amministratore di sistema (*root*). Nell'elenco seguente trovate molti gruppi predefiniti di Debian, assieme alla loro funzione:

root associato all'amministratore di sistema

daemon associato ad alcuni demoni di sistema che scrivono file su disco

bin mantenuto per ragioni storiche

sys mantenuto per ragioni storiche

adm membri di questo gruppo possono leggere numerosi file in /var/log/

tty associato a tutti i terminali reali e virtuali

disk accesso a basso livello al disco fisso; equivale praticamente al gruppo root

lp associato ai demoni di stampa

mail associato alle caselle di posta in /var/mail/

news associato ai programmi per gestire i gruppi Usenet

uucp associato al sottosistema UUCP, ormai in disuso

proxy associato ad alcuni demoni del servizio di proxy

kmem vestigia BSD associabile a programmi che leggono la memoria di sistema

dialout accesso diretto alle porte seriali

fax membri di questo gruppo possono usare il fax

voice associato alle segreterie telefoniche via e-mail

floppy membri di questo gruppo possono usare il lettore di dischetti

tape membri di questo gruppo possono usare unità a nastro

sudo associato al programma sudo
(vedere /usr/share/doc/sudo/OPTIONS)

audio membri di questo gruppo possono usare dispositivi audio del sistema

cdrom membri di questo gruppo possono usare il masterizzatore/lettore di CD-ROM

dip membri di questo gruppo possono accedere a programmi come ppp, wvdial e ad una connessione internet

postgres associato al programma Postgresql

www-data associato ad alcuni web-server, come ad es. Apache

backup associato ai programmi di archiviazione dati

operator associato all'utente operator, l'unico che storicamente poteva fare il login da remoto

list associato ad alcuni programmi di posta elettronica

irc associato ad alcuni demoni IRC

src associato ai file di codice sorgente in /usr/src

gnats associato all'omonimo programma GNU

shadow associato al file /etc/shadow, contenente le password cifrate di sistema

utmp associato al file /var/run/utmp

video membri di questo gruppo possono usare dispositivi video

camera membri di questo gruppo hanno accesso alla memoria di una fotocamera collegata

scanner membri di questo gruppo hanno accesso allo scanner

staff membri di questo gruppo hanno accesso in lettura e scrittura alle cartelle /usr/local/ e /home/

games associato a programmi di giochi

users mantenuto per ragioni storiche

nogroup associato a demoni che non possiedono alcun file

postfix associato al server di posta Postfix (se lo avete installato al posto di Exim)

postdrop associato al server di posta Postfix (se lo avete installato al posto di Exim)

Alcuni gruppi sono associati a determinati programmi detti demoni (*daemons*); in gergo informatico un demone è un programma che offre servizi agli utenti funzionando senza la connessione ad un terminale. I demoni sono programmi molto comuni nei sistemi tipo Unix e derivati, perciò non c'è da stupirsi se li trovate anche in Debian. Un esempio di demone è il programma cupsd, in perenne attesa di un documento da mandare alla coda di stampa.

I gruppi più importanti per il lettore sono **floppy**, **cdrom**, **audio**, **dip**, **video**, dei quali un utente deve far parte per poter usare le tipiche periferiche di un moderno calcolatore: lettore di dischetti, masterizzatore, altoparlanti, modem... Per aggiungere per esempio l'utente mario a questi gruppi inserire[2]:

[2]Sono comandi shell, come spiegato nel paragrafo 2.2 successivo.

```
# adduser mario floppy
# adduser mario cdrom
# adduser mario audio
# adduser mario dip
# adduser mario video
```

Da notare che nel comando si può specificare un solo gruppo alla volta. Se siete già membri di questi gruppi il sistema non farà altro che confermarne la vostra appartenenza, mentre in caso contrario eseguirà quanto richiesto.

2.2 La shell dei sistemi Unix

Un utente avanzato che voglia definirsi tale deve necessariamente interagire con la riga di comando: la shell dei sistemi Unix. Tutti i comandi dei paragrafi seguenti sono digitati dalla console della shell. Per accedere ad una console potete usare due modi: cliccare sul simbolo (tipicamente una conchiglia[3]) disponibile nel vostro ambiente grafico preferito KDE/GNOME per accedere ad una console grafica oppure premere contemporaneamente Ctrl-Alt-F1 (per tornare alla grafica premere Ctrl-Alt-F7) per accedere ad una console di testo. Se usate il secondo metodo, dovrete rifare l'ingresso (*login*) nel sistema per ottenere l'invito (*prompt*) di una console di testo, reinserendo nome[4] e parola d'ordine (*password*)[5]:

```
login: mario
Password: ******
```

La shell offre un'interfaccia a caratteri, rappresentata dalla riga di comando, che permette all'utente di interagire con certe caratteristiche di Linux in modo efficiente. La shell BASH (*Bourne Again SHell*) è la più usata in ambiente Linux, pur essendo una delle tante disponibili. Per sapere che shell state usando digitate:

```
$ echo $SHELL
```

BASH è un progetto GNU[6] che venne sviluppato negli anni ottanta conservando la compatibilità con la shell Bourne (donde il nome), molto popolare all'epoca in ambiente Unix, includendo numerose caratteristiche presenti in altre shell. La shell può essere usata in tre modalità di funzionamento diverse:

shell di login

Questa è la modalità usata se si fa il login nel sistema da una console di testo. Il sistema leggerà le impostazioni nel file /etc/profile e leggerà poi anche /home/<utente>/.bash_profile se esiste. Il primo file contiene impostazioni valide per tutti gli utenti del sistema, il secondo contiene impostazioni valide solo per un determinato utente.

[3] *Shell* significa, appunto, conchiglia.
[4] Ovviamente dovete sostituire mario con il vostro nome utente e con la corrispondente parola d'ordine.
[5] In realtà, nelle funzioni interne del kernel, l'utente è definito mediante un valore numerico univoco, lo UID (*User Id*); all'utente root è associato lo UID=0.
[6] http://www.gnu.org

shell interattiva

Questa è la modalità usata quando si fa partire un emulatore di terminale (la console); vengono lette le impostazioni nel file ~/.bashrc se esiste[7]. In pratica, dopo aver fatto il login ed ottenuto l'invito dal sistema voi usate la shell in modalità interattiva, se digitate dei comandi.

shell non interattiva

Questa è la modalità usata quando si usa la shell come un interprete[8], per l'esecuzione di script. In gergo Unix uno script è un file di testo, contenente istruzioni che l'interprete shell può capire, che l'utente ha il permesso di mandare in esecuzione. Se si invia in esecuzione un file di questo tipo, il sistema cerca una prima linea nella forma:

```
#!/bin/sh
```

e se esiste esegue automaticamente le istruzioni successive, utilizzando come standard input il file stesso, escludendo qualsiasi possibilità di interazione da parte dell'utente.

Segue un semplice esempio per chiarire il concetto di script. Creiamo **da root** il file misc_custom col nostro editor preferito scrivendoci dentro:

```
#!/bin/sh
hdparm -d1 -k1 /dev/hda
```

e salviamo il file col nome indicato. Il comando che abbiamo inserito in seconda riga attiva il DMA sul disco fisso identificato come "hda". Aggiungiamo al file appena creato il permesso di esecuzione:

```
# chmod u+x misc_custom
```

Se ora digitiamo il nome del file misc_custom e premiamo il tasto invio:

```
# misc_custom
```

ciò equivale a digitare[9]:

```
# hdparm -d1 -k1 /dev/hda
```

ottenendo in risposta dal sistema:

```
/dev/hda:
 setting using_dma to 1 (on)
 setting keep_settings to 1 (on)
 using_dma    =  1 (on)
 keepsettings =  1 (on)
```

[7] Il simbolo ~ denota il percorso /home/<utente>/, vedere paragrafo 2.4 a pagina 21.

[8] Un linguaggio di programmazione è tradotto dal codice sorgente in codice binario, comprensibile al calcolatore, da un programma che può essere un interprete oppure un compilatore. La differenza non è importante in questo contesto.

[9] Se il sistema non riconosce il comando vuol dire che non avete installato il pacchetto **hdparm**.

che è proprio l'effetto del comando contenuto nello script da noi creato. Qualsiasi programma (non solo uno script) che viene lanciato da una console si può interrompere premendo Ctrl-C. Se volete maggiori informazioni su questo o altri comandi shell sono sempre disponibili le pagine Unix man accessibili in qualunque momento inserendo nel nostro caso:

```
$ man hdparm
```

Per saperne di più sul comando "man" inserire:

```
$ man man
```

Può essere utile sapere che se viene aggiunta una "&" alla fine di un comando, la shell è in grado di fornire informazioni su eventuali problemi di malfunzionamenti del programma avviato appunto in secondo piano (*background mode*). Esempio:

kppp è un programma a corredo del KDE per la connessione internet via modem avviato cliccando col mouse sull'apposita icona. Supponiamo che cliccando sull'icona il programma non si avvii. Ebbene, per investigare sul perché avviamolo allora dalla console:

```
$ kppp &
bash: /usr/bin/kppp : permission denied
```

e veniamo così a sapere che il programma non parte perchè non abbiamo configurato bene i permessi[10] dell'utente che ha avviato il programma.

2.3 Il filesystem, codesto sconosciuto

Il filesystem, letteralmente "sistema di archivio", definisce i metodi e le strutture dati che un sistema operativo usa per tenere traccia dei file su disco: un po' come il tipo di armadio e di appendiabiti da usare per i nostri vestiti; esempi di filesystem (armadi) sono l'usatissimo ext3 di Linux, l'ISO9660 dei CD-ROM. Linux riconosce un numero elevato di filesystem[11], i quali differiscono tra loro per il modo in cui i dati sono organizzati fisicamente sul disco; a complicare (o semplificare?) le cose c'è da dire che queste differenze sono mascherate all'utente dal filesystem "virtuale"; è come avere davanti una maschera di carnevale: la maschera appare sempre la stessa anche se viene indossata da persone diverse.

Filesystem è anche il termine applicato all'insieme di dati gestito da un sistema software di gestione dei file che implementa una struttura gerarchica ad albero rovesciato, dove i file sono le foglie dell'albero e il punto d'inizio è, guarda caso, *root*, che significa radice. Quest'unica struttura gerarchica per l'organizzazione dei file e delle cartelle è stata oggetto di un notevole sforzo di standardizzazione che ha prodotto lo FHS (*Filesystem Hierarchy Standard*). Esso rappresenta il punto di arrivo di un processo di standardizzazione del filesystem di Linux cominciato nel 1993 e giunto poi a comprendere tutti i sistemi tipo Unix per l'interessamento della comunità BSD[12]. La piena conformità allo FHS rientra tra gli obiettivi di Debian, che organizza il proprio filesystem secondo le cartelle principali seguenti:

"/" cartella radice

[10] Molto probabilmente l'utente che ha lanciato il programma non fa parte del gruppo **dip**.
[11] Se, come probabile, non siete degli esperti, vi conviene adottare l'ext3.
[12] BSD (nelle sue varie incarnazioni) è un altro sistema operativo libero.

bin contenente i comandi essenziali

boot contenente i file statici del *bootloader*

dev contenente i dispositivi per usare le periferiche

etc contenente i file di configurazione del sistema

home contenente le cartelle degli utenti

cdrom punto di montaggio del lettore CD-ROM

floppy punto di montaggio del lettore di dischetti

lib contenente le librerie condivise essenziali e i moduli del kernel

mnt punto di montaggio per esigenze locali

opt inizialmente vuota, destinata a contenere software non presente nella distribuzione Debian

root cartella dell'ammnistratore di sistema

sbin contenente comandi essenziali per l'amministratore di sistema

tmp contenente file temporanei

usr cartella contenente numerosissime sotto-cartelle con gran parte dei file binari del sistema; evidenziamo in particolare:

/usr/share/doc/ contenente la documentazione dei programmi installati

/usr/local/ cartella inizialmente vuota ma destinata a software installato dagli utenti

/usr/src/ contenente codice sorgente (tipicamente il sorgente del kernel)

var contenente dati variabili

Ricordate sempre che pur avendo la possibilità, in qualità di amministratore, di creare qualunque sotto-cartella o modificare qualunque file in qualunque luogo del filesystem, tutte le cartelle sopra elencate, ad eccezione di /opt, /usr/local, /home, /mnt, sono riservate al sistema e **conviene non toccarle a meno che non si sappia bene cosa si sta facendo**. Si sottolinea infine che il carattere "/" indica sia la cartella radice che l'elemento separatore delle sotto-cartelle: ciò che potrebbe essere motivo di confusione.

2.4 Indirizzare un file sul filesystem

Quando volete andare a trovare un amico dovete sapere in che via e a quale numero civico si trova la sua casa; così anche un sistema Linux vuole sapere le coordinate precise di un file che gli chiedete di individuare. L'indirizzamento di un file sul filesystem si può fare in tre differenti modalità:

1. assoluta

2. relativa
3. attraverso il “tilde”, ossia il carattere ~

La prima specifica l’indirizzo del file a partire dalla cartella radice. Esempio:

```
$ ls -l /home/mario/doc/morena.txt
```

indirizza il file morena.txt sotto la cartella “doc”, che a sua volta è contenuta nella cartella “mario”, che a sua volta è contenuta nella cartella “home”; il sistema può così individuare il file ed elencarlo, seguendo quello che si chiama il percorso (*path*) del file. Per esempio per sapere il percorso del comando reboot inserite:

```
# type reboot
reboot is /sbin/reboot
```

La seconda modalità di indirizzamento parte dalla cartella in cui siamo nel momento in cui impartiamo il comando, identificata da un punto. Esempio:

```
$ ls -l ./doc/morena.txt
```

indirizza il file morena.txt sotto la cartella “doc” a partire dalla cartella corrente, supposta essere /home/mario/. Alternativo al precedente e col medesimo effetto è il comando:

```
$ ls -l doc/morena.txt
```

Le due sintassi non sono però sempre equivalenti, poichè i comandi shell vengono eseguiti solo se si è in certe cartelle; pertanto in caso di dubbio preferire sempre la prima.

La terza modalità di indirizzamento consente di riferirsi ad un file presente nella cartella “home” dell’utente, in qualunque posizione del filesystem l’utente si trovi. Per esempio, spostiamoci nella cartella “opt”:

```
$ cd /opt
```

adesso dalla cartella “opt” l’utente mario può indirizzare il file morena.txt con:

```
$ ls -l ~/doc/morena.txt
```

in quanto del tutto equivalente a:

```
$ ls -l /home/mario/doc/morena.txt
```

Provate, per esercizio, la successione dei comandi di cui sopra, creando nella vostra cartella “home” il file morena.txt inserendo:

```
$ touch morena.txt
```

che produrrà un file vuoto chiamato, appunto, morena.txt. Ripetiamo che nei sistemi Linux il simbolo “/” denota sia la cartella radice che la separazione tra gli elementi costituenti un percorso.

2.5 Creare un collegamento simbolico

Un collegamento simbolico (*soft link*) è un file speciale che punta ad un altro file; se si legge il contenuto del collegamento, esso fornisce quello del file a cui punta. Si può tranquillamente cancellare un collegamento: l'operazione di cancellazione non riguarda il file a cui punta, ma solo il suo collegamento. Provate il seguente esempio:

Portiamoci nella cartella /tmp

```
$ cd /tmp
```

e creiamo un file vuoto di prova chiamandolo "prova"

```
$ touch prova
```

torniamo poi alla nostra "home"

```
$ cd ~
```

e creiamo il collegamento simbolico "mia" al file "prova"

```
$ ln -s /tmp/prova mia
```

ora visualizziamo l'avvenuta creazione del collegamento con

```
$ ls -l mia
lrwxrwxrwx 1 mario staff 10 Aug 23 12:58 mia -> /tmp/prova
```

Se successivamente cancelliamo il collegamento

```
$ rm mia
```

possiamo accertarci che il file "prova" a cui puntava è rimasto tal quale

```
$ cd /tmp
$ ls -l prova
-rw-r--r-- 1 mario staff 0 Aug 23 12:56 prova
```

2.6 Montaggio di dispositivi

Ad ogni dispositivo hardware (*device*) del sistema è associato un file nella cartella /dev. Il disco fisso è inquadrato da Linux come un qualunque dispositivo, in aggiunta ad altri dispositivi come il lettore di dischetti o il lettore di CD-ROM, elencati in tabella 2.1 nella pagina seguente. Occorre inoltre informare il sistema perchè inserisca un dispositivo nel punto giusto (punto di montaggio) dell'albero dei file. Queste informazioni il sistema le va a cercare nel file /etc/fstab, che bisogna modificare se si vogliono aggiungere ulteriori punti di montaggio rispetto a quelli predefiniti.

L'operazione di montaggio in Debian si effettua manualmente, per motivi di sicurezza. Per esempio per agganciare il CD-ROM e poterne leggere il contenuto digitare:

Tabella 2.1: Designazione di alcuni dispositivi

Dispositivo	**Designazione**
Primo dischetto	/dev/fd0
Primo disco SCSI/SATA	/dev/sda
Primo CD-ROM SCSI	/dev/scd0
Primo disco IDE	/dev/hda
Secondo disco IDE	/dev/hdc
Primo disco IDE (*slave*)	/dev/hdb
Prima porta seriale	/dev/ttyS0
Mouse	/dev/input/mice

```
# mount /dev/cdrom
```

mentre per smontare il dispositivo scrivere:

```
# umount /dev/cdrom
```

Ricordatevi di smontare un dispositivo che avete agganciato, prima di spegnere il computer, a pena di possibili malfunzionamenti del sistema. **Nel caso del lettore di CD-ROM il computer vi impedirà di estrarre il CD se prima non smontate il dispositivo**. Le operazioni di montaggio/smontaggio si possono fare anche mediante il mouse; la procedura da seguire dipende dall'ambiente grafico in cui si opera.

Una menzione a parte merita l'aggancio di partizioni del disco. Una partizione è una sorta di disco fisso indipendente pur essendo fisicamente una parte in cui è stato diviso un disco fisso. Le informazioni relative alle partizioni di un disco fisso sono contenute nel disco stesso, all'interno della tavola delle partizioni (*partition table*).

Non è possibile per un dato disco avere più di quattro partizioni, dette primarie. Per incrementare il numero di partizioni di un disco si può ricorrere ad altri tipi di partizione, dette estese e logiche. In questo caso per ogni dato disco agganciato al sistema si possono avere fino ad un **massimo** di 15 partizioni in grado di contenere un filesystem: tre partizioni primarie, una estesa (che prende il posto dell'eventuale ultima primaria e non può essere montata) contenente un massimo di dodici logiche. La partizione estesa non può essere formattata (è solo un contenitore per le partizioni logiche) donde il numero massimo di quindici partizioni usabili.

Ad ogni partizione del disco Linux associa un identificativo costituito da "hd" (che sta per *hard disk*, disco fisso) più un'altra lettera che identifica il disco seguita da un numero (da uno a quattro per le primarie e da cinque in poi per le logiche) che identifica la partizione. Ad esempio con "hda1" il sistema rappresenterà la prima partizione primaria nel primo disco agganciato. Nel caso particolare di dischi SCSI/SATA Linux associerà un identificativo con le prime lettere costituite da "sd".

2.7 Gestione delle librerie di sistema

Il comune di una città allo scopo di evitare di far acquistare ai cittadini un numero enorme di libri può mettere a loro disposizione un certo numero di testi in una biblioteca pubblica e concederli in prestito a chi ne faccia richiesta. Allo stesso modo Linux, per evitare che numerosi programmi con le stesse funzioni abbiano le stesse sezioni di codice replicate in memoria centrale, utilizza delle librerie[13] (*libraries*) di funzioni caricate una volta per tutte dal kernel; tutti i programmi che vogliono usarle ne richiedono semplicemente l'uso.

In effetti la traduzione di *library* è biblioteca, ma qualche ignorante l'ha tradotto libreria ed ormai così è entrato nell'uso comune.

Bisogna però che il meccanismo di caricamento delle librerie condivise del sistema operativo le localizzi. La gestione delle librerie condivise è comandata dal file /etc/ld.so.conf che è un semplice file di testo contenente l'indirizzamento completo delle librerie.

Se si rileva che Linux non riesce a trovare una libreria, provare a cercarla con il comando:

```
$ cat /etc/ld.so.conf|xargs -i find {} -name <nomelibreria>\*
```

ricordandosi eventualmente di cercarla anche in /usr/lib:

```
$ find /usr/lib -name <nomelibreria>\*
```

Se questi comandi vi sembrano troppo astrusi e vi mettono a disagio è possibile fare la ricerca in ambiente grafico cliccando col mouse sull'icona di "Trova" e simili. A questo punto ci sono tre possibilità:

a. la libreria non c'è e allora bisogna installarla

b. la libreria esiste e allora si dovrà aggiungere una riga in /etc/ld.so.conf con l'indirizzamento completo della libreria e poi far girare il programma **ldconfig**[14]

c. la libreria esiste ma Linux non riesce a caricarla; in questo caso (improbabile) la libreria è danneggiata e va sostituita

In realtà ci sarebbe anche la possibilità che la libreria esiste ma non ha esattamente il nome richiesto[15]; è sconsigliabile creare un collegamento simbolico col nome richiesto perché si rischia di provocare malfunzionamenti dovuti a differenze nelle versioni delle librerie.

Può essere utile anche conoscere il comando per elencare le librerie da cui dipende un determinato programma:

```
# ldd <nomeprogramma>
```

Esempio:

```
# ldd /usr/sbin/sshd
```

[13] Più precisamente vengono dette librerie dinamiche, per distinguerle dalle librerie cosiddette statiche.
[14] Inserire semplicemente: `# ldconfig`
[15] Il nome con cui vengono indirizzate le librerie dinamiche è detto *soname*.

che visualizza:

```
linux-gate.so.1 =>  (0xffffe000)
libwrap.so.0 => /lib/libwrap.so.0 (0xb7fb0000)
libpam.so.0 => /lib/libpam.so.0 (0xb7fa8000)
libdl.so.2 => /lib/tls/libdl.so.2 (0xb7fa4000)
libselinux.so.1 => /lib/libselinux.so.1 (0xb7f8f000)
libresolv.so.2 => /lib/tls/libresolv.so.2 (0xb7f7c000)
libcrypto.so.0.9.8 => /usr/lib/i686/cmov/libcrypto.so.0.9.8
libutil.so.1 => /lib/tls/libutil.so.1 (0xb7e3d000)
libz.so.1 => /usr/lib/libz.so.1 (0xb7e29000)
libnsl.so.1 => /lib/tls/libnsl.so.1 (0xb7e13000)
libcrypt.so.1 => /lib/tls/libcrypt.so.1 (0xb7de5000)
libgssapi_krb5.so.2 => /usr/lib/libgssapi_krb5.so.2 (0xb7dc9000)
libkrb5.so.3 => /usr/lib/libkrb5.so.3 (0xb7d4c000)
libk5crypto.so.3 => /usr/lib/libk5crypto.so.3 (0xb7d27000)
libcom_err.so.2 => /lib/libcom_err.so.2 (0xb7d24000)
libkrb5support.so.0 => /usr/lib/libkrb5support.so.0 (0xb7d1f000)
libc.so.6 => /lib/tls/libc.so.6 (0xb7bed000)
/lib/ld-linux.so.2 (0xb7fd2000)
libsepol.so.1 => /lib/libsepol.so.1 (0xb7bac000)
```

elencando le librerie usate dal demone del protocollo SSH.

2.8 Gestione dei processi

Si definisce processo un qualsiasi programma in esecuzione. Ad ogni processo il sistema associa un numero univoco, chiamato pid (*process identification*). Per visualizzare l'elenco dei processi attivi nel sistema digitare:

```
$ ps aux
```

che farà apparire qualcosa di simile:

```
USER    PID %CPU %MEM   VSZ  RSS TTY  STAT START   TIME COMMAND
root      1  0.1  0.0  1272  484 ?     S    15:18   0:05 init [2]
root      2  0.0  0.0     0    0 ?     SW   15:18   0:00 [keventd]
root      4  0.0  0.0     0    0 ?     SW   15:18   0:00 [kswapd]
root      5  0.0  0.0     0    0 ?     SW   15:18   0:00 [bdflush]
root      6  0.0  0.0     0    0 ?     SW   15:18   0:00 [kupdated]
root     10  0.0  0.0     0    0 ?     SW   15:18   0:00 [knodemgrd_0]
root     11  0.0  0.0     0    0 ?     SW   15:18   0:00 [kjournald]
root     72  0.0  0.0     0    0 ?     SW   15:19   0:00 [kjournald]
root     83  0.0  0.0     0    0 ?     SW   15:19   0:00 [khubd]
root    256  0.0  0.1  2036  780 ?     S    15:19   0:00 /sbin/syslogd
root    319  0.0  0.2  2244 1360 ?     S    15:19   0:00 /sbin/klogd
root    328  0.0  0.0  1248  428 ?     S    15:19   0:00 /usr/sbin/inetd
root    343  0.0  0.3  4664 1672 ?     S    15:19   0:00 /usr/sbin/cupsd
root    594  0.0  0.2  2788 1212 ?     S    15:19   0:00 /usr/sbin/sshd
daemon 601  0.0  0.1  1384  580 ?     S    15:19   0:00 /usr/sbin/atd
```

```
root   604  0.0  0.1  1652  680 ?     S    15:19   0:00 /usr/sbin/cron
root   614  0.0  0.1  2504  724 ?     S    15:19   0:00 /usr/bin/kdm
mario  735  0.0  0.3  2752 1652 pts/1 S    15:20   0:00 /bin/bash
mario 1972  0.0  0.3  3740 1756 pts/1 R    16:29   0:00 ps aux
```

L'elenco evidenzia nell'ordine l'utente proprietario del processo, il numero di identificazione del processo (il suo pid), la percentuale di potenza del processore utilizzata, la percentuale di memoria usata, altre informazioni meno importanti fino all'ultima colonna che elenca il comando che ha avviato il processo.

Tutto ciò è molto utile se un programma si blocca e non si riesce a farlo sparire con i soliti metodi. In questa eventualità il comando precedente consente di conoscere il pid ad esso associato per poi inserire:

```
# kill -9 <pid>
```

equivalente a:

```
# kill -kill <pid>
```

La sintassi generale del comando[16] è del tipo:

```
# kill <segnale> <pid>
```

L'equivalenza tra segnale e numero intero (come sopra illustrato) è mostrata digitando:

```
$ kill -l
```

che visualizza l'elenco dei segnali che possiamo inviare:

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR      31) SIGSYS      32) SIGRTMIN    33) SIGRTMIN+1
34) SIGRTMIN+2  35) SIGRTMIN+3  36) SIGRTMIN+4  37) SIGRTMIN+5
38) SIGRTMIN+6  39) SIGRTMIN+7  40) SIGRTMIN+8  41) SIGRTMIN+9
42) SIGRTMIN+10 43) SIGRTMIN+11 44) SIGRTMIN+12 45) SIGRTMIN+13
46) SIGRTMIN+14 47) SIGRTMIN+15 48) SIGRTMAX-15 49) SIGRTMAX-14
50) SIGRTMAX-13 51) SIGRTMAX-12 52) SIGRTMAX-11 53) SIGRTMAX-10
54) SIGRTMAX-9  55) SIGRTMAX-8  56) SIGRTMAX-7  57) SIGRTMAX-6
58) SIGRTMAX-5  59) SIGRTMAX-4  60) SIGRTMAX-3  61) SIGRTMAX-2
62) SIGRTMAX-1  63) SIGRTMAX
```

ove si vede che il segnale kill (SIGKILL) corrisponde al numero nove. Se non riuscite neanche così ad arrestare il programma impazzito allora riavviate la macchina con:

```
# reboot
```

[16] Per ulteriori informazioni occorre il solito `man kill`.

Potreste infine avere bisogno di sospendere temporaneamente un processo per poi riavviarlo in seguito. Debian fornisce una serie di script nella cartella /etc/init.d (vedere il paragrafo 2.8.2 nella pagina successiva) che permettono di avviare e fermare molti processi al vostro comando, **senza bisogno di riavviare la macchina**. Esempio:

Per fermare il servizio di web server di Apache[17]:

```
# /etc/init.d/apache2 stop
```

Allo stesso modo per riavviare il servizio[18]:

```
# /etc/init.d/apache2 start
```

2.8.1 Cron: il dio del tempo al vostro servizio

I sistemi Debian sono equipaggiati con Vixie Cron, un programma capace di eseguire una qualsiasi istruzione ad un giorno ed orario prefissato. Ovviamente nei sistemi che non rimangono sempre accesi (in pratica tutti i computer domestici) potrebbe capitare che all'orario stabilito di esecuzione il computer sia spento e quindi l'operazione non possa essere eseguita. In particolare, Debian installa cron predisponendolo ad eseguire tutte le operazioni a circa le sei del mattino, per non interferire con le normali attività del sistema; ciò va benissimo per un server acceso 24 ore su 24, ma non va affatto bene per il nostro mostro da scrivania, che non le eseguirebbe in pratica mai!

Studiamo allora un poco la sintassi di controllo di cron, per fare qualche piccolo aggiustamento. Il file di controllo di cron si chiama /etc/crontab; qui sono elencate le operazioni da eseguire e a quali giorni e orari:

```
25 6 * * * root run-parts --report /etc/cron.daily
47 6 * * 7 root run-parts --report /etc/cron.weekly
52 6 1 * * root run-parts --report /etc/cron.monthly
```

Pertanto, come si vede, le operazioni da eseguire sono degli script concentrati nelle cartelle /etc/cron.daily, /etc/cron.weekly, /etc/cron.monthly. Quello che interessa sono le prime cinque colonne che individuano il giorno e l'ora di esecuzione; da sinistra a destra abbiamo rispettivamente:

minuti da 0 a 59

ore da 0 a 23

giorno (del mese) da 1 a 31

mese da 1 a 12

giorno (della sett.) da 0 a 7 ove il sette corrisponde a domenica

***** tutti i valori validi per quel campo (valore jolly)

[17] Naturalmente dovete avere installato Apache sul vostro sistema.
[18] Di solito è disponibile anche `restart` che esegue uno stop-start in sequenza.

Adesso pertanto risulta chiaro che cron è impostato ad eseguire le operazioni giornaliere alle 6:25 del mattino, quelle settimanali alle 6:47 del mattino della domenica, quelle mensili alle 6:52 del primo giorno di ogni mese. Siccome noi non siamo così mattinieri, impostiamo cron ad eseguire tutte le operazioni all'incirca alle quattro del pomeriggio modificando **da root** il file /etc/crontab come segue:

```
25 16 * * * root run-parts --report /etc/cron.daily
47 16 * * 7 root run-parts --report /etc/cron.weekly
00 18 1 * * root run-parts --report /etc/cron.monthly
```

salvando il file così modificato con il vostro editor preferito. Con questa impostazione cron eseguirà le operazioni giornaliere alle 4:25 del pomeriggio, quelle settimanali alle 4:47 del pomeriggio del venerdì, quelle mensili alle 6:00 del pomeriggio del primo giorno di ogni mese.

Ora che cron vi è così familiare sapete a chi rivolgervi quando dovete automatizzare qualche operazione che non sia da eseguire all'avvio del sistema.

2.8.2 Debian System V Init

Il programma init, avviato dal kernel, si occupa di gestire le fasi di avvio del sistema operativo dopo l'accensione del computer. Se non avete installato un'immagine da mostrare a schermo all'avvio (*splash screen*) Linux vi informa di quello che sta accadendo facendo scorrere velocemente del testo sullo schermo; lo scorrimento è veloce, tanto veloce che può essere difficile riuscire a leggere tutto quello che scrive. Niente paura! Dopo che avete fatto il *login*, se all'invito della shell inserite[19]:

```
$ dmesg | less
```

potrete, mediante i tasti cursore, comodamente scorrere la lista dei messaggi stampati. In questo modo Linux consente all'utente di avere una maggiore padronanza del sistema, poichè egli può controllare istante per istante cosa è accaduto all'avvio della macchina. Ciò è molto comodo qualora si verificassero dei problemi, dei quali si potrà facilmente identificare la causa (per esempio un demone che non è partito).

Un tempo c'erano due modalità d'avvio tra loro incompatibili: il modello System V Init e il modello BSD Init. Bisognava scegliere e il mondo Unix scelse System V che è poi divenuto lo standard anche di Linux[20].

In Debian, che segue il modello System V, tutti gli script che avviano programmi all'accensione sono ubicati nella cartella /etc/init.d mentre le cartelle:

/etc/rcS.d

/etc/rc0.d

/etc/rc1.d

/etc/rc2.d

[19] Se il sistema non riconosce il comando dovete installare il pacchetto **less**.
[20] I sistemi BSD e altri su di esso basato usano ancora il modello BSD Init.

/etc/rc3.d

/etc/rc4.d

/etc/rc5.d

/etc/rc6.d

contengono ciascuna dei collegamenti simbolici ai file in /etc/init.d/. Il nome del collegamento comincia per "S" (*Start*) seguito da un numero di due cifre seguito dal nome del file presente in /etc/init.d se il programma è da avviare; oppure per "K" (*Kill*) seguito da un numero di due cifre seguito dal nome del file presente in /etc/init.d se il programma è da arrestare (evidentemente ci si riferisce alla fase si spegnimento o di riavvio della macchina). Esempio:

Se abbiamo installato il web server Apache, in /etc/rc2.d troveremo un collegamento del tipo **S91apache** che punta allo script chiamato **apache** in /etc/init.d che avvierà il programma Apache; in /etc/rc2.d troveremo allora anche un collegamento del tipo **K20apache** che punta al medesimo script **apache** in /etc/init.d che però arresterà il programma Apache. Gli script vengono invocati passando il parametro "start" se il nome comincia per "S" o passando il parametro "stop" se il nome comincia per "K".

La cartella del tipo /etc/rc?.d viene selezionata in base al valore di una variabile detta *runlevel*, che prende il nome di *initdefault* in /etc/inittab, il file di configurazione del programma init. I possibili valori di *runlevel* sono[21]:

N	post-accensione
S	modalità singolo utente (da non attivare direttamente)
0	arresto del sistema (*halt*)
1	modalità singolo utente
2-5	modalità multi-utente
6	riavvio del sistema (*reboot*)

Nella fase di avvio la sequenza di esecuzione di init è la seguente:

a. esegue quanto presente in /etc/rcS.d in ordine alfabetico ed in subordine numerico. Il valore di *runlevel* è posto pari a "N".

b. aggiorna il *runlevel* al valore specificato in /etc/inittab, di solito pari a due; pertanto eseguirà quanto presente nella cartella /etc/rc2.d/.

Come si vede in Debian il valore predefinito di runlevel è pari a due; gli altri valori fino a cinque sono identici ed è lasciato all'amministratore il compito di personalizzarli, se così desidera. La conoscenza della sequenza di avvio può servire per automatizzare l'avvio di processi che debbano essere eseguiti all'accensione della macchina. E' superfluo ricordare

[21] Valori superiori sono possibili ma non utilizzati.

che intervenire su questi file richiede i privilegi di amministratore e cura particolare deve essere posta per evitare di inficiare seriamente la procedura di avvio del sistema. Esempio:

Supponiamo di aver preparato lo script misc_custom come spiegato nel paragrafo 2.2 a pagina 18. E' evidente che sarebbe molto scomodo dover digitare a mano il comando per attivare il DMA del disco fisso ogni volta che accendiamo il computer; sarebbe molto più comodo che lo facesse il computer all'accensione al posto nostro! Piazziamo allora lo script nella cartella /etc/init.d; per fare eseguire lo script all'accensione della macchina dovremmo come minimo creare un collegamento in /etc/rc2.d. Per essere sicuri, però, di eseguire lo script all'avvio qualunque sia il *runlevel* dovremmo piazzare il medesimo collegamento simbolico in ciascuna delle cartelle /etc/rc?.d sopra elencate. Esiste in Debian un comodo script; basterà infatti inserire:

```
# update-rc.d misc_custom defaults
```

per avere il medesimo risultato, anzi meglio: ciò lancerà lo script /etc/init.d/misc_custom eseguendo **hdparm** nel runlevel 2,3,4 o 5 ed "arrestandolo" nel runlevel 0,1,6 con una priorità di sequenza predefinita pari a venti.

Per cancellare i collegamenti testé creati, qualora non ci servissero più, digitare invece:

```
# update-rc.d -f misc_custom remove
```

Bisogna sottolineare che sono solo i collegamenti a venire cancellati; lo script `/etc/init.d/misc_custom` rimane intatto. In verità lo script in questione ha solo valore didattico, in quanto si può attivare il DMA del disco fisso direttamente nel menu di configurazione del kernel (vedasi paragrafo 7.1 a pagina 59).

3 Gestione dei pacchetti Debian

3.1 Comandi base e avanzati

Un pacchetto di un componente software binario, riconoscibile per il suffisso .**deb**, contiene tutto ciò che serve a installare, ed eventualmente rimuovere, il software in questione in modo efficiente, cioè senza danneggiare il sistema, senza lasciare file inutili ad occupare spazio su disco, senza entrare in conflitto con altri componenti software, le cosiddette dipendenze (pararagrafo 3.2 seguente). Un interessante effetto secondario è che è molto semplice avere una chiara visione di tutto il software installato sul sistema, in maniera sintetica e pulita. Rispetto alla tradizione storica in ambito Unix di fornire il codice sorgente del software, che poi doveva essere compilato sulle singole macchine, il pacchetto costituisce un notevole salto tecnologico verso la fruizione del software da parte di un'utenza inesperta, in quanto esso permette, con un semplice comando, di rendere il programma immediatamente usabile dall'utente. Per fare un paragone, è come se si fosse passati dall'orticello di casa al supermercato.

Il sistema che in Debian gestisce l'installazione e disinstallazione dei pacchetti è relativamente complesso ma estremamente potente e avanzato, consentendo:

- ➢ verifica in ogni momento di nuove versioni
- ➢ installazione automatica delle dipendenze
- ➢ installazione automatica degli aggiornamenti
- ➢ mantenimento delle configurazioni dell'utente
- ➢ aggiornamento dei programmi anche mentre sono in uso[1]

Lo strumento principe di gestione è APT (*Advanced Package Tool*) consistente in vari programmi CLI[2] apt- e il programma a menu di testo **aptitude**[3], che può usarsi anche da riga di comando. Nella tabella 3.1 sono elencati alcuni degli strumenti disponibili più adeguati a giudizio dell'autore. APT si appoggia comunque a **dpkg** (che funge da strumento di basso livello) per eseguire i suoi compiti.

Da notare che per gestire i pacchetti non è necessario conoscere i comandi shell che verranno illustrati per le interfacce a riga di comando, perché esistono anche ottime e comode interfacce grafiche specifiche, come ad esempio **kpackage** del KDE, **synaptic** di GNOME, e l'ottimo **dpkg-www** di Massimo Dal Zotto; tuttavia alcune delle funzionalità più avanzate sono accessibili solo dalla riga di comando. Il comando in assoluto più usato è quello, semplice, per installare un pacchetto, che prende la seguente sintassi:

[1] Non si ha bisogno della modalità singolo utente per aggiornare un sistema Debian.
[2] *Command Line Interface*, Interfaccia a riga di comando.
[3] Tecnicamente, APT non è altro che una libreria, usata da molti programmi.

Tabella 3.1: Strumenti di gestione dei pacchetti

Nome	Note
dpkg	interfaccia a basso livello
tasksel	installa un gruppo di pacchetti
aptitude	CLI e interfaccia a menu per APT

```
# aptitude install <nomepacchetto>
```

Per esempio, la documentazione relativa ad APT è disponibile in italiano installando il pacchetto apt-howto-it:

```
# aptitude install apt-howto-it
```

mentre la variante:

```
# aptitude -s install apt-howto-it
```

mostrerà **cosa** si andrà ad installare (il componente software richiesto più le eventuali dipendenze) senza installare alcunché.

Il programma **dpkg**, invece, è quello che è apparso sulla scena per primo e per questo, per certi versi, è limitato: a differenza di APT, interrompe l'installazione di un pacchetto se le sue dipendenze non sono soddisfatte. D'altra parte esso offre delle funzionalità uniche di accesso a basso livello al contenuto dei pacchetti. Per conoscere per esempio i file che compongono un determinato pacchetto inserire:

```
# dpkg -L <nomepacchetto>
```

Esempio:

```
# dpkg -L apt-howto-it
```

Un'altra eventualità che si potrebbe presentare è quella di voler conoscere a quale pacchetto installato corrisponde un determinato file; in questo caso serve il comando:

```
# dpkg -S <nomefile-anche-approssimato>
```

Provate:

```
# dpkg -S crontab
```

Notevole è poi il fatto che in ogni momento è possibile riconfigurare un pacchetto già installato[4] inserendo:

```
# dpkg-reconfigure --p=medium <nomepacchetto>
```

[4]Solo se usa **debconf**, il programma di servizio che presiede alla configurazione dei pacchetti.

Esempio:

```
# dpkg-reconfigure man-db
```

che riconfigura l'archivio delle pagine Unix man. Vale la pena osservare che purtroppo non esiste un sistema centralizzato per la configurazione; per tutti quei pacchetti che fanno uso di debconf, si può installare il seguente programma:

```
# aptitude install configure-debian
# configure-debian
```

che esegue automaticamente il comando dpkg-reconfigure, presentando i pacchetti in un elenco organizzato per sotto-sezioni.

Se si verificano dei problemi durante l'installazione di un pacchetto[5] e il processo di installazione si interrompe senza essere portato a termine, inserite:

```
# aptitude -f install
```

per rimettere le cose a posto. Infine, se desiderate rimuovere un pacchetto inutile, inserite semplicemente:

```
# aptitude purge <nomepacchetto>
```

Prestate attenzione a quanto vi comunica il sistema: **spesso, a causa delle dipendenze, la cancellazione di un pacchetto ne comporta la cancellazione di numerosi altri**; il sistema però vi avverte sempre in modo che abbiate la possibilità di annullare il comando, se la cosa non vi piace.

APT deposita tutti i pacchetti del software installato in /var/cache/apt/archives. Potrebbe capitare che la dimensione di questa cartella, con il passare del tempo, divenga eccessiva; se necessario se ne può cancellare tutto il contenuto (vengono cancellati i pacchetti e **non** il software installato!) inserendo:

```
# aptitude clean
```

Da non dimenticare di istruire APT a caricare le chiavi crittografiche dei manutentori dei pacchetti Debian, in modo che esso possa verificare, quando si scarica un pacchetto da internet, che è proprio un pacchetto originale di Debian e non un "cavallo di troia" con un virus all'interno; allo scopo digitare pertanto:

```
# apt-key add /usr/share/apt/debian-archive.gpg
```

Per concludere accenniamo al concetto di pacchetto virtuale (*virtual package*), che è un nome generico per un gruppo di pacchetti che forniscono la medesima funzionalità. Per esempio, il pacchetto virtuale www-browser si riferisce ai pacchetti konqueror, dillo, lynx che forniscono tutti la funzionalità di navigatore web. Se si installa più di un pacchetto associato ad un pacchetto virtuale, Debian consente di selezionare quello che sarà usato in modo predefinito dal sistema. Provate:

```
# update-alternatives --display editor
```

[5] Tipicamente, un pacchetto non ufficiale; altro comando utile in simili occasioni è:

```
dpkg -i --force -overwrite <pacchetto.deb>
```

Tabella 3.3: Gradi di priorità di un pacchetto

Priorità	Disinstallazione	Pacchetto esempio
Obbligatoria	molto sconsigliata	login
Importante	sconsigliata	adduser
Normale	sconsigliata	perl
Facoltativa	a discrezione utente	xserver-xorg
Extra	a discrezione utente	flightgear

3.2 Attributi di un pacchetto

Quando il gestore dei pacchetti mostra il contenuto di un pacchetto fornisce una serie di informazioni su di esso tramite degli attributi caratteristici tra i quali, a giudizio dell'autore, si distinguono per importanza il grado di priorità (*priority*) e alcune etichette riguardanti il modo in cui un pacchetto dipende da altri pacchetti.
I gradi di priorità possibili, illustrati brevemente in tabella 3.3, sono:

Obbligatoria (*Required*)
Pacchetto non rimuovibile perchè necessario al funzionamento del sistema. Per esempio, disinstallando il pacchetto login non è più possibile accedere al sistema.

Importante (*Important*)
Pacchetto la cui rimozione causa inconvenienti da seri a gravi, giungendo anche a rendere instabile il sistema. Per esempio, disinstallando il pacchetto adduser diventa impossibile aggiungere nuovi utenti al sistema.

Normale (*Standard*)
Pacchetto sempre presente in ambiente Linux e non richiedente la presenza del sistema grafico. Un esempio è costituito dal pacchetto perl, contenente il linguaggio PERL (Larry Wall's Pratical Extraction and Report Language), usato in molti script di sistema.

Facoltativa (*Optional*)
Pacchetto che è ragionevolmente sensato installare. Un esempio è costituito dal pacchetto xserver-xorg, contenente una parte del sistema grafico.

Extra (*Extra*)
Pacchetto con software molto specialistico oppure con componenti software che hanno altri possibili candidati con la medesima funzionalità. Il pacchetto flightgear contenente un simulatore di volo ne è un esempio (Fig. 4.2 a pagina 48).

Bisogna sottolineare che l'installazione e rimozione dei pacchetti sono operazioni a carico dell'amministratore, che ha pieni poteri. Se egli vuole rimuovere un pacchetto con priorità, mettiamo, importante, il sistema non potrà far nulla per impedirlo; se però l'amministratore non sa quel che fa rischia di menomare seriamente il sistema, fino, nei casi più gravi, a renderlo inutilizzabile.

Il modo in cui un pacchetto dipende da altri pacchetti, ovvero la modalità delle dipendenze, si illustra di solito considerando un ipotetico pacchetto A ed analizzando i possibili modi

Tabella 3.5: Modalità delle dipendenze

Tipo	Traduzione	Modalità
Depends	Dipende	forte
Recommends	Raccomanda	debole
Suggests	Suggerisce	debole
Conflicts	Interferisce	forte
Replaces	Sostituisce	forte
Provides	Fornisce	debole

in cui può "dipendere" da un altro ipotetico pacchetto B. Naturalmente un pacchetto A può "dipendere" anche da numerosi pacchetti B. Nella discussione che segue sarà utile riferirsi anche alla tabella 3.5 riassuntiva. La modalità di dipendenza dall'ipotetico pacchetto B viene illustrata tramite una serie di etichette:

Dipende (*Depends*) Il pacchetto B deve essere installato affinché il pacchetto A funzioni

Raccomanda (*Recommends*) Il pacchetto B se installato aggiunge funzionalità accessorie al pacchetto A

Suggerisce (*Suggests*) Il pacchetto B è collegato alle funzionalità del pacchetto A e spesso le migliora

Interferisce (*Conflicts*) Il pacchetto B se installato non farà funzionare il pacchetto A

Sostituisce (*Replaces*) Il pacchetto B è rimosso se viene installato il pacchetto A

Fornisce (*Provides*) Le funzionalità del pacchetto B sono incorporate nel pacchetto A.

Alla luce di tutto ciò, osserviamo che per quanto riguarda le dipendenze aptitude consente di installare automaticamente anche i pacchetti sotto l'etichetta Raccomanda e di cancellare automaticamente tutti i file che, installati solo per soddisfare delle dipendenze, rimangono orfani del pacchetto da cui dipendevano. Inoltre, aptitude ha il vantaggio di ricordare quali sono i pacchetti installati su comando dell'utente piuttosto che installati automaticamente perché dipendenze. Per tali motivi è consigliabile usare aptitude al posto del vecchio apt-get, che infatti è stato dichiarato obsoleto[6] (*deprecated*).

Alla fine di tutto questo discorso, facciamo un esempio utilizzando aptitude dalla riga di comando; andiamo ad osservare il contenuto del pacchetto lilo:

```
$ aptitude show lilo
Package: lilo
State: installed
Automatically installed: no
Version: 1:22.6.1-9.3
Priority: optional
```

[6]Da notare che kpackage utilizza apt-get per eseguire le operazioni sui pacchetti.

```
Section: admin
Maintainer: Andreas Roldan <aroldan@debian.org>
Uncompressed Size: 1115k
Depends: mbr, debconf (>= 1.2.9),libc6 (>= 2.3.6-6),libdevmapper1.02
Suggests: lilo-doc
Conflicts: manpages (< 1.29-3)
Description: [omessa]
```

Il sistema ci dice che questa versione di lilo per funzionare ha bisogno (*Depends*) di determinate versioni della libreria libc6 ed è collegata (*Suggests*) al pacchetto lilo-doc, che sarebbe molto utile installare visto che ne contiene la documentazione; inoltre veniamo informati che lilo è incompatibile (*Conflicts*) con certe versioni delle pagine Unix man (*manpages*), probabilmente perchè esse si riferirebbero a versioni obsolete di lilo.

3.3 Un maledetto pomeriggio da cani

Come accennato, potrebbe capitare di installare, con motivazioni più o meno valide, dei pacchetti Debian non ufficiali; questi pacchetti di solito sono preparati dai medesimi sviluppatori Debian che lavorano ai pacchetti ufficiali, quindi in genere non danno problemi, anche se il rischio che qualcosa non funzioni a dovere bisogna metterlo in conto (altrimenti sarebbero pacchetti ufficiali!). Una tale spiacevole evenienza si può illustrare attraverso lo studio del seguente ipotetico caso.

Supponiamo che un pomeriggio vi mettiate di buzzo buono alla vostra fida tastiera, per installare certi nuovi *driver* che sostituiscono alcuni componenti critici del sistema grafico X11, per sfruttare meglio la scheda video con *chipset* Mach64 di cui è dotato il vostro computer. I nuovi driver sono contenuti in certi pacchetti che vi siete procurati in rete:

```
xlibmesa-dri-mach64 xserver-xorg-dri-mach64
```

e non esitate a prontamente installarli dopo aver fatto il *login* da amministratore:

```
# aptitude install xlibmesa-dri-mach64
# aptitude install xserver-xorg-dri-mach64
```

Dopo il riavvio della macchina, però, l'amara sorpresa: il sistema X11 non parte più; tutta la grafica è completamente andata e l'unica cosa che vi rimane è la riga di comando della shell! Forse il pacchetto aveva qualcosa di sbagliato oppure forse Voi avete dimenticato di fare qualcosa di importante (a proposito, avete letto eventuale documentazione a corredo?), fatto sta che rivolete indietro il vostro GNOME/KDE. Ebbene, perbacco, siete o non siete l'amministratore? Vi accingete pertanto a cancellare per sempre dal vostro amatissimo disco codesti maledetti pacchetti:

```
# aptitude remove xlibmesa-dri-mach64
# aptitude remove xserver-xorg-dri-mach64
```

Ma, accidenti, questo pomeriggio la sfiga deve aver fatto gli straordinari, perchè il sistema vi informa (se potesse credo che vi punterebbe il dito contro) che per disinstallare questi pacchetti occorrerà disinstallare l'intero GNOME/KDE!

Per uscire da questo pasticcio di dipendenze è utile ricorrere al vecchio dpkg, che a volte è in grado di fornire più complete informazioni. Infatti nel nostro ipotetico caso il tentativo di rimozione tramite dpkg:

```
# dpkg -r xlibmesa-dri-mach64
```

rivela l'importante dettaglio che ciò non può essere fatto perchè manca il pacchetto xlibmesa-dri della Debian stabile, evidentemente rimosso installando i pacchetti precedenti. Questa è proprio l'occasione buona per scatenare tutta la potenza di "preferenze" (*preferences*), una caratteristica di APT. Se non ci ricordiamo la pagina di manuale, possiamo farci dare dei suggerimenti dal sistema:

```
$ apropos preferences
apt_preferences (5) - Preference control file for APT
```

e digitare quindi:

```
$ man apt_preferences
```

Questa caratteristica di APT permette di degradare alla versione stabile di Debian dei singoli pacchetti. In sintesi, per risolvere il nostro problema di dipendenze occorrerà creare **da root** il file /etc/apt/preferences scrivendoci dentro:

```
Package: xlibmesa-dri
Pin: release a=stable
Priority: 1001
```

e salvare il file col nome indicato. A questo punto il comando:

```
# aptitude install xlibmesa-dri
```

avrà successo e forzerà (dato l'elevato 1001 di priorità) ad installare il componente grafico xlibmesa degradando a cascata tutto il sistema grafico X11 alla versione della distribuzione stabile e rimuovendo i pacchetti "alieni" xlibmesa-dri-mach64 ed xserver-xorg-dri-mach64, ripristinando in tal modo lo stato originale. Da tutto ciò si evince che fareste bene ad evitare i pacchetti non ufficiali se non avete ottime ragioni, se non avete intenzione di fronteggiare complicati conflitti di dipendenze, se non avete sufficiente dimestichezza con il sistema, se non amate gli imprevisti.

3.4 Una provincia dell'impero

A volte può capitare di poter produrre, più o meno automaticamente (MPlayer ne costituisce un esempio notevole), un pacchetto debian a partire da un codice in formato sorgente. Siccome il pacchetto in questione manca dalla cartella /var/cache/apt/archives, l'unico modo per installarlo consiste nell'usare dpkg, ma volendo è possibile creare un archivio personale, il quale, essendo visibile ad APT al pari della cartella `/var/cache/apt/archives`, consentirà di usare aptitude per installare il pacchetto; ciò costituirà inoltre un comodo riferimento in cui andare a sbirciare, per verificare quali e quanti sono i pacchetti autoprodotti.

Vediamo allora come fare per aggiungere una provincia all'impero di APT. Una possibile procedura è la seguente:

1. selezionare la cartella in cui mettere i pacchetti autoprodotti; scegliamo in questo esempio /opt/debs/archive;
2. dopo aver eventualmente creato la cartella prescelta, posizionare al suo interno i pacchetti autoprodotti che si vogliono rendere visibili ad APT;
3. Inserire:

```
# cd /opt/debs
# touch override
# dpkg-scanpackages archive override|gzip>archive/Packages.gz
```

4. configurare APT, inserendo nel file /etc/apt/sources.list la seguente riga:

```
deb file:/opt/debs archive/
```

5. salvare il sources.list modificato e digitare:

```
# aptitude update
```

Se tutto è andato liscio ora possiamo installare qualsiasi pacchetto posto in /opt/debs/archive inserendo:

```
# aptitude install <nomepacchetto>
```

Se aggiungete altri pacchetti in /opt/debs/archive dovrete ripetere quanto descritto nei punti tre e cinque (APT sarà già configurato nel modo giusto).

3.5 Installare un pacchetto non ufficiale

Chi usa la distribuzione stabile a volte può avere la necessità di reperire un componente software non disponibile nei pacchetti della distribuzione. Infatti, nonostante in Debian siano presenti migliaia di pacchetti, non tutto il software libero giornalmente prodotto nel mondo è pacchettizzato in Debian! Alcuni programmi, anzi, non lo saranno mai: se siamo fortunati, però, riusciremo a reperire un *backport*. Tipicamente, si tratta di software presente nella distribuzione instabile, che qualche sviluppatore, servendosi del codice sorgente, compila per la distribuzione stabile e successivamente lo rende disponibile confezionando il relativo pacchetto Debian. In questo caso, si potrebbe prelevare dalla rete in qualche modo il pacchetto, per poi installarlo con dpkg; ma conviene avvalersi della potenza di APT, che è in grado di automatizzare il processo: esso infatti può prelevare pacchetti da qualsiasi fonte.

Per prima cosa, dunque, occorre localizzare in rete il pacchetto: http://www.backports.org oppure http://www.apt-get.org sono i siti più frequentati a tale scopo. Successivamente, configurare opportunamente APT aggiungendo nel file di configurazione /etc/apt/sources.list una riga con la fonte ove APT andrà a scaricare il pacchetto; di solito in questi siti presso l'indicazione del pacchetto compare sempre la scritta "*download as text*": cliccando lì dal navigatore web si visualizza la riga con l'indicazione della fonte che bisogna poi inserire in /etc/apt/sources.list, facendo eventualmente un copia-incolla. A questo punto farà tutto APT; digitare:

```
# aptitude update
# aptitude install <nomepacchetto>
```

per far sì che APT prelevi il pacchetto da internet e lo installi, risolvendo automaticamente anche gli eventuali problemi di dipendenze. Controllate solo che il pacchetto in questione sia per la distribuzione stabile, altrimenti non si installerà perché le dipendenze non saranno quelle giuste. Una volta poi effettuata l'installazione, è consigliabile disattivare la riga aggiunta in /etc/apt/sources.list commentandola oppure cancellandola, perchè altrimenti nel caso vi troviate a digitare eventuali comandi di aggiornamento[7] del sistema, vi potrebbe capitare di aggiornare a versioni differenti i pacchetti installati dal medesimo sito, senza che ne abbiate intenzione: ovviamente se questa è proprio la vostra intenzione allora... ignorate il consiglio.

3.6 Aggiornare la distribuzione

Debian cerca sempre di garantire un facile aggiornamento della vecchia distribuzione stabile alla versione successiva; poter effettuare l'aggiornamento dell'intero sistema, senza dover ripartire da zero, è uno dei vantaggi più distintivi di Debian che viene offerto a chi ha speso, come spesso capita, parecchio tempo per configurare a dovere tutto l'hardware ed ovviamente non vuole ricominciare tutto da capo. Comunque, si tenga presente che durante un aggiornamento qualcosa può sempre andare storto, per cui bisogna essere preparati a qualche intervento manuale di aggiustamento fine.

Per aggiornare l'intero sistema alla nuova versione stabile inserire:

```
# aptitude update
# aptitude -f dist-upgrade
```

Storicamente questo comando non si è potuto digitare spesso, visti i lunghi tempi intercorsi tra una versione stabile e la successiva[8]. Leggere prima sempre le note di rilascio (*Release Notes*), ricavabili, tra l'altro, da tutti i CD/DVD Debian, anche se pensate che la cosa sia superflua.

E' consigliabile fare prima una copia di sicurezza dei vostri dati più importanti, per esempio almeno della cartella /home. Fate attenzione ad installare sempre software non pacchettizzato esclusivamente nelle cartelle /opt oppure /usr/local, altrimenti durante l'aggiornamento rischiate che i vostri file vengano sovrascritti dal sistema. Notate che l'aggiornamento indolore non è più garantito se avete reso impuro il sistema aggiungendo pacchetti esterni: ciò è da tenere presente quando si installano pacchetti non ufficiali o si decide di passare a sistemi misti[9].

L'aggiornamento via rete dell'intero sistema, da non tentare se non si dispone di una connessione a banda larga, richiede diverse ore; in alternativa, è sempre possibile effettuare

[7] Si veda il paragrafo 8.2 a pagina 67.

[8] Vi è una certa confusione nell'uso di dist-upgrade, che si può osservare venire usato quotidianamente: in realtà chi lo usa in tal modo sta usando altre versioni di Debian; poichè dist-upgrade rimuove pacchetti per installare i medesimi pacchetti di versione **superiore**, non avrebbe evidentemente senso il suo uso quotidiano nella distribuzione Debian stabile, visto che in essa non vi sono **mai** nuovi pacchetti di versione **superiore**.

[9] Stable/Testing, Stable/Unstable, Testing/Unstable etc. Naturalmente fanno eccezione i pacchetti di debian-volatile.

Tabella 3.7: Server distributori (*Mirrors*)

Nazione	Sito	Tipo
Italia	ftp.it.debian.org	Primario
Italia	ftp.eutelia.it	Secondario
Italia	ftp.bonomia.it	Secondario
Italia	debian.fastbull.org	Secondario
Italia	freedom.dicea.unifi.it	Secondario
Italia	mirror.units.it	Secondario
Italia	ftp.unina.it	Secondario
Italia	debian.dynamica.it	Secondario
Italia	mi.mirror.garr.it	Secondario
Italia	debian.fastweb.it	Secondario

l'aggiornamento dai CD/DVD della nuova versione. E' richiesta una certa cura nel rispondere alle domande di debconf durante l'installazione dei pacchetti: a tale proposito conviene valutare attentamente se sostituire i vecchi file di configurazione con quelli nuovi oppure no; talvolta tenere i vecchi file di configurazione può significare la disattivazione di nuove caratteristiche implementate nelle rinnovate versioni dei programmi, per cui di solito si aggiornano con quelli nuovi. Successivamente si può sempre decidere su dove orientarsi, visto che i vecchi file di configurazione sono comunque conservati in file con estensione **.dpkg-old**. Nel caso si voglia fare l'aggiornamento via rete, basta inserire in /etc/apt/sources.list uno dei server distributori di Debian (*Debian mirrors*) sparsi nel mondo (per un esempio si veda il paragrafo 1.2); nella tabella 3.7 sono elencati quelli attivi nel nostro paese[10]. Come si noterà, essi si differenziano in due tipologie: primari e secondari. Un server distributore primario è dotato di larga banda, è attivo 24 ore su 24, è aggiornato automaticamente, offre entrambi i protocolli ftp e http; un server distributore secondario può essere soggetto a limitazioni varie, ma non è detto che abbia caratteristiche inferiori a quelli primari.

Se necessario non abbiate timore di eseguire anche numerose volte il comando aptitude dist-upgrade; non è raro che l'utente lasci il sistema parzialmente aggiornato, credendo di aver terminato l'aggiornamento. Malfunzionamenti fastidiosi sono il tipico sintomo di un sistema solo parzialmente aggiornato; per essere sicuri che l'aggiornamento è stato completato dovete vedere un messaggio del tipo:

```
Reading Package Lists...Done
Building Dependency Tree...Done
Reading extended state information...Done
Initializing packages states...Done
Reading task descriptions...Done
0 packages upgraded, 0 newly installed,0 to remove and 0 not upgraded.
```

Giunti a questo punto potete eventualmente aggiornare anche il kernel ad una **serie** superiore. A volte però, nel caso di alcune piattaforme, viene raccomandato l'esatto contrario (aggiornare cioè prima il kernel e poi tutto il resto).

[10]Un elenco sempre aggiornato è reperibile presso http://www.debian.org/mirror/list

3.7 Localizzazione italiana del sistema

L'autore usa la localizzazione inglese[11] ma è facile rendersi conto che molti nel nostro paese usano la localizzazione italiana. Per localizzare un sistema Debian nella lingua italiana sono necessarie alcune operazioni che in verità sono eseguite anche in fase di installazione del sistema. La procedura di installazione si può richiamare in ogni momento con:

```
# dpkg-reconfigure locales
```

Controllare che la localizzazione italiana sia stata generata nel file /etc/locale.gen che deve contenere non commentata la seguente riga:

```
it_IT@euro ISO-8859-15
```

Assegnare a questo punto la lingua italiana a tutto il sistema aggiungendo al file /etc/profile le seguenti righe:

```
export LANG=it_IT@euro
export LC_CTYPE=it_IT@euro
```

Per controllare che la localizzazione linguistica sia coerente con queste impostazioni digitare il comando:

```
$ locale
```

che dovrà mostrare:

```
LANG=it_IT@euro
LC_CTYPE="it_IT@euro"
LC_NUMERIC="it_IT@euro"
LC_TIME="it_IT@euro"
LC_COLLATE="it_IT@euro"
LC_MONETARY="it_IT@euro"
LC_MESSAGES="it_IT@euro"
LC_PAPER="it_IT@euro"
LC_NAME="it_IT@euro"
LC_ADDRESS="it_IT@euro"
LC_TELEPHONE="it_IT@euro"
LC_MEASUREMENT="it_IT@euro"
LC_IDENTIFICATION="it_IT@euro"
LC_ALL=
```

Se la configurazione della lingua non è corretta di solito l'interprete Perl[12] mostra un avvertimento (*warning*) durante l'installazione dei pacchetti software; altrimenti il gestore dei pacchetti automaticamente installerà il pacchetto nella lingua italiana, se esiste la traduzione. Molti programmi sono stati tradotti in italiano da schiere di volontari; nella tabella 3.9

Tabella 3.9: Pacchetti traduttori a parte

Pacchetto	**Note**
openoffice.org-l10n-it	Traduzione italiana di OpenOffice
openoffice.org-help-it	Traduzione italiana dell'aiuto in linea
kde-i18n-it	Traduzione italiana del KDE
koffice-i18n-it	Traduzione italiana di KOffice

sono elencati i pacchetti più comuni delle traduzioni italiane da installare a parte, come a volte può essere necessario.

Un'interessante possibilità è offerta dal pacchetto localepurge, che in fase di installazione di un pacchetto permette di cancellare automaticamente dal disco tutte le localizzazioni linguistiche non usate:

```
# aptitude install localepurge
```

Esiste anche una traduzione italiana delle pagine Unix man, ma la sua installazione è sconsigliabile perchè la traduzione è incompleta e obsoleta; dovrete pertanto sorbirvi le pagine originali in inglese ma, se proprio ci tenete così tanto, perché non provate a tradurle voi?

[11] Se, come me, volete l'inglese ma avete bisogno di leggere e scrivere in italiano, vi consiglio il locale en_IE@euro, appartenente all'interlingua, secondo quanto dice l'ISO 639.

[12] Perl è un linguaggio di programmazione per calcolatori; si veda il paragrafo 8.4 a pagina 70.

4 Ambiente grafico

4.1 X Window System

Il sistema grafico X11 (che d'ora in poi chiameremo semplicemente X) è un insieme di librerie che fornisce un ambiente grafico basato su mouse e finestre, che si è imposto come standard de facto su tutti i sistemi tipo Unix, Linux incluso.

Un concetto fondamentale che deve essere chiaro nella mente è che grafica e console (qui per console si intende il terminale raggiungibile premendo per esempio contemporaneamente i tasti Ctrl-Alt-F1, vedere paragrafo 2.2) sono alquanto indipendenti tra loro; cito a caso il fatto che se la tastiera funziona perfettamente in grafica, non è affatto detto che funzioni altrettanto perfettamente in console, perchè grafica e console gestiscono ognuno per proprio conto la configurazione della tastiera. Questo stato di cose è inefficiente, a dire il meno, ed è senz'altro auspicabile in futuro una maggiore integrazione tra console e grafica.

Detto ciò, per quanto riguarda X, Debian utilizza l'implementazione X.org versione 7.1. Nella struttura del filesystem tutti i file statici di X (binari, librerie) sono al di sotto di /usr/bin/X11 e /usr/lib/X11, mentre il file di configurazione è /etc/X11/xorg.conf, che in verità è di piuttosto ostica interpretazione.

I componenti di X, nella sua terminologia invertita rispetto al normale significato dei termini *client/server*, sono:

server X
Programma che gestisce le funzionalità grafiche e le mette a disposizione degli altri programmi (denominati client). Il server X rende disponibile una specie di lavagna, dove poi vengono disegnate le forme geometriche per l'utilizzo delle applicazioni grafiche, che compongono per esempio KDE e GNOME.

client X
Programma che esegue l'elaborazione inviando il risultato al server X per la visualizzazione. In pratica tale programma deve essere su un computer ove girerà il programma da eseguire, ovvero su una macchina *server*.

protocollo X
Il protocollo attraverso il quale il server X e il client X comunicano tra loro.

Xlib
Librerie di funzioni grafiche specifiche a basso livello.

Come si vede X implementa uno schema client/server: il server fornisce delle risorse (lo spazio sullo schermo) a degli utilizzatori, detti client (le applicazioni che usano lo spazio offerto sullo schermo). Questo schema permette all'utente di lanciare un programma su un computer e di interagire con esso su uno schermo che può essere anche differente da quello del computer originario: lo schermo di un computer di una rete locale (LAN) o addirittura lo schermo di un computer che sta da qualche parte in internet. Questo supporto per la rete

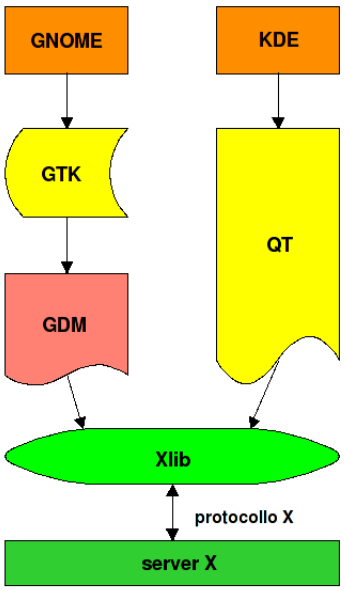

Figura 4.1: Relazione tra KDE e GNOME.

è fondamentale nel settore commerciale, sia per le applicazioni che per l'amministrazione remota di sistemi, sebbene sia di scarsa rilevanza per l'utente domestico su un singolo computer. Nella maggior parte dei casi, pertanto, server X e client X si troveranno sulla stessa macchina, ma useranno comunque l'architettura client/server.

Lo strumento di base per creare un'applicazione grafica si chiama, abbiamo accennato, Xlib. Essendo Xlib troppo a basso livello, risulta molto laborioso creare un'applicazione grafica usando interamente Xlib. Per ovviare a questa difficoltà sono stati creati strumenti di programmazione di livello di astrazione più elevato: le moderne librerie grafiche QT e GTK, usate per costruire gli ambienti grafici KDE e GNOME, rispettivamente (Fig.4.1).

Le librerie QT, scritte in linguaggio C++, sono state sviluppate dalla norvegese Trolltech (fondata nel 1994) che le rende disponibili con licenza GPL a partire dalla versione 2.2. L'ambiente di sviluppo integrato di riferimento è KDevelop, che aiuta a conformarsi agli standard di KDE.

Le librerie GTK (Gimp ToolKit), nate in contrasto con la mancanza di una licenza libera per le librerie QT, sono disponibili con licenza LGPL. Scritte in linguaggio C, hanno l'obiettivo di emergere come grande strumento di programmazione a se stante. L'ambiente di sviluppo integrato di riferimento è Anjuta.

4.2 Pratica di X

La panoramica sull'architettura di X e i suoi strumenti di sviluppo, peraltro interessante per un programmatore o aspirante tale, ha lo scopo di rendere il lettore più consapevole di quanto accade sotto il cofano del luccicante ambiente grafico. All'atto pratico, sarà sufficiente scegliere l'ambiente grafico tra i due più importanti KDE e GNOME (se avete scelto qualche

altro ambiente grafico probabilmente non avete bisogno di leggere questo manuale). Inserite eventualmente:

```
# aptitude install kde
```

oppure

```
# aptitude install gnome
```

La configurazione standard di Debian prevede all'avvio di eseguire il *runlevel* due (paragrafo 2.8.2 a pagina 29), che fa partire il server X ed attiva automaticamente il *login* grafico che lancerà KDE o GNOME. Alcuni utenti potrebbero preferire di restare in console e lavorare mediante la shell, senza avviare automaticamente il server X. In questo caso, possiamo poi in qualunque momento avviare il server X con il comando:

```
$ startx
```

che in generale prende la seguente sintassi:

```
$ startx -- :<n> vt<g>
```

ove:

<n> numero di schermo

<g> numero della console grafica

Provate:

```
$ startx -- :1 vt8
```

Facendo però partire il server X da console, dobbiamo specificare noi quale ambiente grafico lanciare, modificando il file ~/.xinitrc inserendovi la seguente riga:

```
startkde
```

per far partire KDE oppure:

```
gnome-session
```

per far partire GNOME. Per ottenere tale risultato, potremmo per esempio personalizzare il *runlevel* tre, commentando nel file /etc/inittab la riga:

```
id:2:initdefault:
```

ed impedendo l'avvio del *login* grafico nel *runlevel* tre; nel caso per esempio si usi il *session manager* kdm del KDE, occorrerà inserire:

```
# update-rc.d -f kdm remove
# update-rc.d kdm start 99 2 4 5 . stop 01 0 1 6 .
```

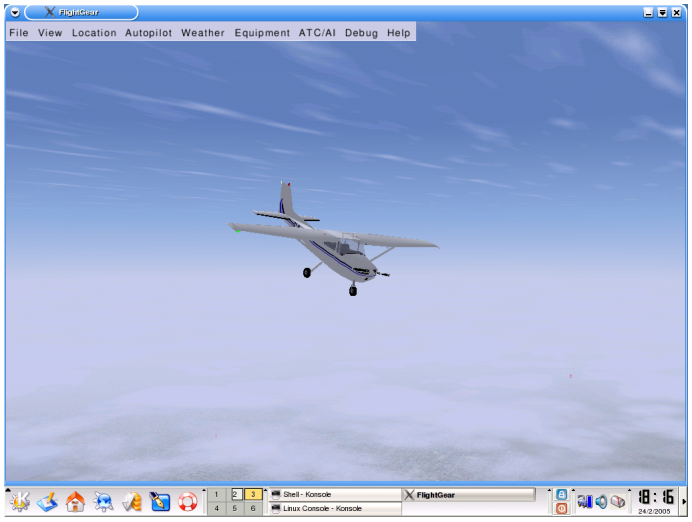

Figura 4.2: Flightgear: Cessna C-172 in volo a diecimila piedi

Con questa personalizzazione il computer si fermerà durante l'avvio a chiedere il valore di *runlevel* da utilizzare; inserendo il numero tre avremo il *login* che ci porterà direttamente in console.

Spesso gli utenti incontrano problemi a far funzionare correttamente la scheda video del computer; ciò dipende dal fatto che i produttori mantengono segrete le specifiche tecniche e nel migliore dei casi rilasciano *driver* proprietari, che non possono essere inclusi in Debian (e che spesso funzionano anche male). Un certo numero di schede video è supportato dal relativo *driver* all'interno di X.org; ma non aspettatevi miracoli: sono pochi i programmatori che hanno la capacità di scrivere *driver* video in *reverse-engineering*. E' sempre possibile tornare sul menu di configurazione della scheda video, inserendo:

```
# dpkg-reconfigure xserver-xorg
```

Attenzione, perché purtroppo debconf non dà alcuna indicazione sul modello di scheda video, richiedendo direttamente di specificare il *driver*. Le note si fanno ancora più dolenti per la grafica tridimensionale accelerata in hardware, che X.org realizza usando DRI (*Direct Rendering Infrastructure*), indispensabile per far funzionare molti giochi e programmi che fanno un uso intenso della grafica, come modellatori molecolari o simulatori di volo e spaziali (Fig.4.2). Se non sono tante le schede video supportate direttamente da X.org, ancora meno sono quelle capaci di usare DRI; per sapere se la vostra scheda ha l'accelerazione hardware attiva, inserite in una console **grafica**, ovvero sotto X:

```
$ glxinfo
```

Se nel copioso *output* leggete la riga:

```
direct rendering: no
```

allora non avete l'accelerazione grafica hardware attiva. Solo una maggiore apertura dei produttori delle schede video ed un impegno delle distribuzioni Linux a dedicare allo sviluppo di X la stessa attenzione e lo stesso numero di programmatori che forniscono al kernel potrà eliminare l'aspetto - la mancanza di *driver* video - che causa problemi all'utente finale.

5 Il problema della documentazione

5.1 Organizzare il proprio sito

In un sistema Debian è facile superare i centomila file residenti su disco fisso. Se pensate che molti di essi sono programmi, ognuno con le sue pagine di manuale in linea e magari svariati altri manuali in html a corredo, è facile immaginare come in breve tempo il problema principale diventi non la scarsità di informazione bensì il suo rapido reperimento. La cosa migliore, a mio avviso, consiste nel cercare di installare ogni tipo di documentazione in formato html (al limite va bene pure il testo puro, che può essere facilmente trasformato in html) per poi creare nel proprio sistema un motore di ricerca mediante i seguenti programmi:

Apache web server

Dwww interfaccia web per tutta la documentazione di un sistema Debian

Htdig completo sistema di indicizzazione e ricerca (Fig.5.1) per un sito web: in questo caso (*local-only indexing*) il sito web sarà il vostro disco fisso!

In prima approssimazione basterà digitare:

```
# aptitude install apache dwww htdig
```

Per poter utilizzare appieno le potenzialità di **dwww** è necessario anche installare **swish++** per creare un archivio indicizzato della documentazione Debian:

```
# aptitude install swish++
# dwww-index++
```

In particolare l'uso di dwww è sicuramente più efficiente dell'andare a sbirciare a mano in /usr/share/doc, che è il posto in cui Debian pone le FAQ, gli Howto e quant'altro (libri, manuali, etc.). Per accedere all'interfaccia grafica di dwww occorre inserire in un qualsiasi navigatore web:

```
http://localhost/dwww/
```

mentre per accedere all'interfaccia grafica di ricerca di Htdig inserire:

```
http://localhost/search.html
```

Se decidete di installare dpkg-www, già citato, potrete accedere via web anche alla gestione dei pacchetti. Per accedere all'interfaccia grafica di ricerca di dpkg-www inserire in un qualsiasi navigatore web:

```
http://localhost/cgi-bin/dpkg
```

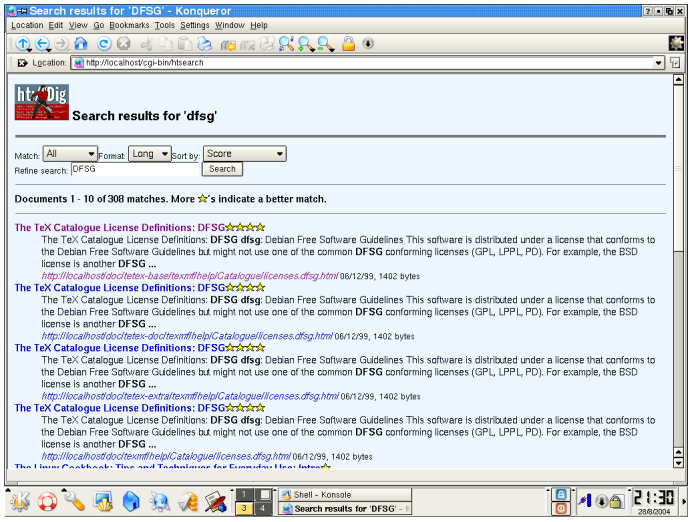

Figura 5.1: Htdig in azione nel KDE.

5.2 Scrivere un documento multiformato

Scrivere oggigiorno un documento multiformato significa poterlo generare in formato pdf (*Adobe Portable Document Format*) e html (*Hyper Text Markup Language*). Questi formati infatti danno la garanzia che un documento sarà leggibile nell'ambito di un qualsiasi sistema operativo (in particolare il formato pdf ci garantisce che il documento verrà correttamente stampato). Quando ci si orienta sulla composizione di un documento compatibile con diversi tipi di modalità di fruizione e formato (libro stampato, guida interna di sistema, ipertesto in rete) bisogna pianificare fin dall'inizio i formati di uscita e rinunciare a pretese stilistiche particolari.

Cosa offre il panorama Debian per soddisfare questa esigenza? Come spesso accade nel software libero, sono disponibili numerosi strumenti; è pertanto giocoforza selezionarne solo alcuni: qui si privilegerà la facilità d'uso e la potenza del software, cercando comunque di non suggerire un numero eccessivo di candidati.

In base a questi criteri la scelta si riduce sostanzialmente a tre prodotti:

Scribus

OpenOffice

LyX

Scribus, scritto avvalendosi delle librerie QT, è un programma di impaginazione professionale, ideale per la produzione di pubblicazioni anche complesse come giornalini, manifesti, riviste, libri su piccola scala. Il programma è basato sull'uso dei *frame* - bisogna cioè creare uno spazio per gli elementi della pagina prima di importarli. Scribus possiede un supporto estensivo al formato pdf, in particolare può produrre documenti nel formato PDF/X-3, basato su Postscript Language 3. Lo sviluppo del programma continua a ritmi sostenuti ma già adesso è ampiamente usabile e molto promettente:

```
# aptitude install scribus scribus-doc
```

Un grave difetto del programma è la scarsità di manuali d'uso gratuiti.
OpenOffice ha una lunga storia di sviluppo e si presenta come la passerella ideale per chi proviene dal software proprietario di automazione d'ufficio e vuole la soluzione meno traumatica possibile, senza perdere molto in termini di sofisticazione, facilità d'uso e compatibilità con i formati proprietari di una nota multinazionale americana. Da Ottobre 2003 è possibile sostenere l'esame ECDL (Patente europea del computer) interamente su Linux/OpenOffice:

```
# aptitude install openoffice.org
```

LyX è, come si dice, una categoria a parte. Esso è, in sintesi, un'interfaccia grafica molto curata (nella versione QT), anche se non facilissima da usare, al linguaggio LaTeX, un potente formattatore di testi. LaTeX a sua volta è un'estensione di un altro linguaggio di formattazione più primitivo chiamato TeX, creato dal matematico Donald Knuth e risalente al 1978. LaTeX esiste quindi da oltre venti anni, è usato da studenti e ricercatori di tutto il mondo e funziona molto bene, soprattutto nella gestione di documenti molto complessi come libri, tesi, oppure pubblicazioni scientifiche che contengano molta matematica.

Lo svantaggio principale di LaTeX è che esso è un linguaggio di "programmazione" del testo, pertanto un documento si presenta nella forma di un vero e proprio codice sorgente, che ha bisogno della compilazione per essere visualizzato (LyX però è in grado di operare una parziale visualizzazione del documento già in fase di scrittura). LaTeX non è adatto per scrivere documenti, magari semplici, in cui è necessario controllare manualmente la posizione degli elementi sul foglio, perchè questo lo fa il programma secondo le regole tipografiche standard. Per contro, LaTeX elabora il testo nel suo insieme e può quindi avere una visione globale e capacità di impaginazione e indicizzazione automatiche; per esempio è molto facile produrre indici generali.

Per usare LyX è necessario il supporto completo a LaTeX inserendo:

```
# aptitude install lyx latex2html
```

che installa la distribuzione TeTeX, cioè il supporto a LaTeX con i suoi programmi di servizio così come più o meno accorpati da Thomas Esser. TeTeX è presente in tutti i moderni sistemi Linux. Una guida in italiano a LaTeX (Impara Latex) è reperibile presso:
http://www.manuali.it
che è una traduzione dal francese a cura di Alessandro Cannarsi del centro Cefriel di Milano. Altre guide sono contenute nella distribuzione TeTeX stessa, accessibili sul vostro sistema digitando in una console grafica:

```
$ /usr/bin/texdoctk
```

Una volta ottenuto da LyX il sorgente LaTeX del documento (che è sempre un file con estensione .tex) si possono usare due ulteriori programmi di servizio per generare i formati pdf e html del documento. Per generare il formato pdf inserire:

```
$ pdflatex nomefile.tex
```

mentre per generare il formato html inserire:

```
$ latex2html nomefile.tex
```

che in verità nell'esperienza dell'autore ha funzionato correttamente solo con file semplici e di piccole dimensioni.
Per ottenere i migliori risultati personalizzati è necessario studiare i relativi manuali dei programmi citati. Per esempio, se aggiungete un indice analitico, per far farlo includere nel documento pdf dovrete passare attraverso diverse compilazioni del documento stesso:

```
$ pdflatex nomefile.tex
$ makeindex nomefile.idx
$ pdflatex nomefile.tex
$ pdflatex nomefile.tex
```

E' probabile, se non altro allo stato attuale, che LaTeX non sia molto adatto all'utente comune, poco avvezzo a linguaggi di programmazione e compilazioni, sia pure oscurati da interfacce grafiche più o meno attraenti. Se però siete arrivati fin qui a leggere, potreste trovarvi a vostro agio in LaTeX, apprezzandone nel contempo l'enorme potenza.

5.3 I file di registrazione

Ogni moderno programma è istruito per interfacciarsi con un demone di sistema che si occupa di salvare su disco tutti i messaggi di vario tipo (autorizzazioni, notifiche di malfunzionamenti, etc.) eventualmente emessi. Questi messaggi sono memorizzati in file su disco la cui ubicazione è definita in /etc/syslog.conf, che è il file di configurazione del demone, personalizzabile dall'utente.

Si può ben dire che una delle cose in cui differiscono un principiante da un utente esperto è che il primo ignora i file di registrazione (*log*), mentre il secondo li consulta regolarmente. Il file di registrazione principale è visualizzabile con:

```
# less /var/log/syslog
```

Consultare i file di registrazione non vi consentirà di risolvere un problema, ma almeno sarete informati di un problema del quale magari avreste ignorato l'esistenza; inoltre così sarete in una posizione migliore per porre le vostre domande agli esperti. E' possibile attivare un demone di registrazione dei messaggi d'avvio modificandone il suo file di configurazione /etc/default/bootlogd: la scrittura dei messaggi avverrà in /var/log/boot; è normalmente disattivato, specialmente perchè blocca l'avvio di alcune macchine non-x86 dotate di console seriali.

Nella cartella /var/log troverete molti altri file di registrazione potenzialmente interessanti da leggere.

6 Comunicare con il computer

6.1 Liste di posta

Una buona fonte di informazioni, utile anche per chiarire numerosi dubbi, è rappresentata dalle liste di posta (*mailing lists*), da non confondere con i forum, i gruppi Usenet oppure i canali *chat*. Nell'universo Debian una grossa parte della comunicazione utilizza questo mezzo, per cui è importante che vi abituiate ad usarlo. Ovviamente, se non lo avete ancora fatto, dovete installare un programma capace di inviare e leggere la posta elettronica, un programma ad esempio come **Kmail** oppure **Evolution**[1]. I forum sono più immediati da usare, ma Debian preferisce le liste di posta per una serie di ottime ragioni:

I. in un forum tutti gli utenti devono usare la medesima interfaccia mentre nelle liste di posta ognuno può scegliersi il programma con cui leggerle;

II. l'archivio di un forum è più ostico da consultare rispetto a quello delle liste di posta;

III. in un forum non è possibile effettuare la lettura dei messaggi in differita (*offline*).

Ogni lista è caratterizzata da un indirizzo univoco a cui tutti gli iscritti mandano i propri messaggi di posta elettronica (*email*), i quali a sua volta tutti possono leggere comodamente a casa. Le liste mantenute da Debian sono elencate in `/usr/share/doc/debian/mailing-lists.txt`, ove è anche descritto il metodo di iscrizione e cancellazione. Le liste Debian hanno una pagina internet:

`http://lists.debian.org`

che illustra come iscriversi e varie altre cose. In alternativa, si può usare esclusivamente la posta elettronica. E' importante notare in quest'ultimo caso che l'indirizzo a cui mandare l'email di iscrizione è leggermente diverso da quello della lista prescelta. Se per esempio volete iscrivervi alla lista:

`debian-italian@lists.debian.org`

dovete mandare un'email all'indirizzo:

`debian-italian-request@lists.debian.org`

con nel campo oggetto (*subject*) la parola ''*subscribe*". Riceverete un messaggio di risposta per cui è sufficiente attivare la funzione "rispondi"(*reply*) per confermare l'iscrizione: questa è una misura di sicurezza per evitare che altri vi iscrivano alla lista a vostra insaputa. Dopo quest'ultimo passo potete inviare e leggere i messaggi della lista. Controllate la lingua dei

[1] Se siete degli scafati linuxiani potreste prendere in considerazione il camaleontico **Mutt**.

messaggi ammessi in lista: molte liste Debian ammettono solo la lingua inglese[2]. Come forse avete indovinato, se volete cancellare l'iscrizione alla lista dovete mandare un'email all'indirizzo:

```
debian-italian-request@lists.debian.org
```

con nel campo oggetto (*subject*) la parola "*unsubscribe*". Per mandare l'email di *subscribe* o di *unsubscribe* dovete avere un valido indirizzo di posta elettronica nel vostro campo "*From*", ma ciò non è necessario, anzi è sconsigliabile, per i messaggi inviati alla lista, perchè attualmente ci sono programmi software malevoli che setacciano le liste di posta in rete per carpire gli indirizzi di posta elettronica delle persone, che poi diventano destinatari di email non richieste (il cosiddetto *spam*: il nome deriva dalla marca di una carne in scatola a buon mercato venduta negli Stati Uniti). Per difendersi da questa invasione dell'intimità personale bisogna mettere indirizzi fasulli nel campo "*From*" dei messaggi inviati alle liste: il vostro vero indirizzo potete piazzarlo in qualche altra parte del messaggio, oppure non metterlo affatto.

Oltre alle liste Debian, nel mondo esistono tantissime liste di posta, ognuna con le sue modalità di iscrizione e cancellazione; spesso la lista ha una pagina internet, come visto per quelle Debian. Un caso tipico è quando si usa un programma che si conosce poco: iscriversi alla lista dedicata al programma, frequentata da utenti e sviluppatori, rappresenta un'ottima occasione per ottenere un minimo di assistenza tecnica; per esempio se avete problemi col popolare **MPlayer** potreste iscrivervi alla lista ad esso dedicata presso la pagina internet:
`http://www.mplayerhq.hu`
Qualunque sia la vostra lista però, ci sono dei codici di comportamento che ogni utente di posta elettronica deve rispettare, la cosiddetta "*Netiquette*". Eccone alcuni:

1. E' responsabilità dell'utente informarsi sulle modalità di iscrizione e cancellazione nelle liste di posta
2. Non mandate in lista messaggi in html: i messaggi devono essere in formato testo puro
3. Riempite il campo Oggetto (*Subject*) in modo che rispecchi il contenuto del messaggio
4. Non usate lettere maiuscole nei messaggi: USARE LETTERE MAIUSCOLE E' COME URLARE
5. Non mandate in lista il contenuto di messaggi ricevuti in privato senza il consenso di chi ve li ha mandati
6. Cercate di essere brevi: un messaggio di oltre cento righe è considerato lungo; se non potete farne a meno mettete nell'oggetto (*subject*) la parola "Long" o altro avvertimento equivalente
7. Cercate di non rispondere alle provocazioni
8. Evitate di rispondere ad un messaggio riproducendone il contenuto per intero (*full quoting*): riproducete solo le parti più importanti e appropriate

[2]Ovviamente la lista debian-italian ammette solo messaggi in lingua italiana.

Tabella 6.1: Gruppi Usenet tradizionali

Gruppo	Note
humanities	Materie umanistiche
misc	Tematiche varie
news	Discussioni su Usenet
rec	Tematiche poco impegnative
sci	Materie scientifiche
soc	Scienze sociali
talk	Varietà

9. Prima di chiedere un'informazione cercate di accertarvi se la risposta non sia già contenuta in qualche FAQ o documentazione disponibile

Il testo originale di riferimento è l'RFC 1855, che potrete facilmente recuperare dalla rete cercandolo in `/usr/share/doc/RFC/rfc-informational.html.gz` dopo aver installato il pacchetto *non-free* doc-rfc-std:

```
# aptitude install doc-rfc-std
```

6.2 Articoli Usenet

Col termine Usenet si intende oggi un'entità che gestisce la distribuzione di pubblici messaggi, chiamati **articoli**. Gli articoli di Usenet sono organizzati in gruppi di discussione (*newsgroups*). Questi gruppi di discussione sono classificati secondo una gerarchia che si sviluppa da sinistra a destra, come per la notazione all'interno di un filesystem e in modo simile ad un sistema di scatole cinesi, usando però il punto per delimitarne i vari pezzi. All'inizio dei tempi (informatici) Usenet era organizzata secondo otto gruppi di discussione fondamentali, elencati in tabella 6.1.

Per esempio, nel gruppo "rec", troveremo sia il gruppo di discussione:

```
rec.boats.cruising
```

dedicato alle crociere in barca, sia il gruppo di discussione:

```
rec.arts.books.reviews
```

dedicato alle recensioni di libri. Nel corso del tempo la gerarchia dei gruppi si è enormemente espansa; quelli in tabella 6.1 si ritengono gli originali gruppi Usenet (o "gruppo degli otto"); oggi ne esistono tanti altri: la gerachia "alt" con i frequentatissimi alt.binaries, la gerarchia "gnu" del progetto omonimo, la gerarchia "bit" con la versione *news* delle liste di posta di Bitnet, la gerarchia "g" di gmane, la gerarchia "it" con gruppi di discussione italiani...

Leggere gli articoli Usenet storicamente ha comportato l'avere a che fare con software non proprio alla portata di chiunque; oggi le cose sono più semplici. Arrivare a leggere gli articoli Usenet comporta la risoluzione di tre problemi fondamentali:

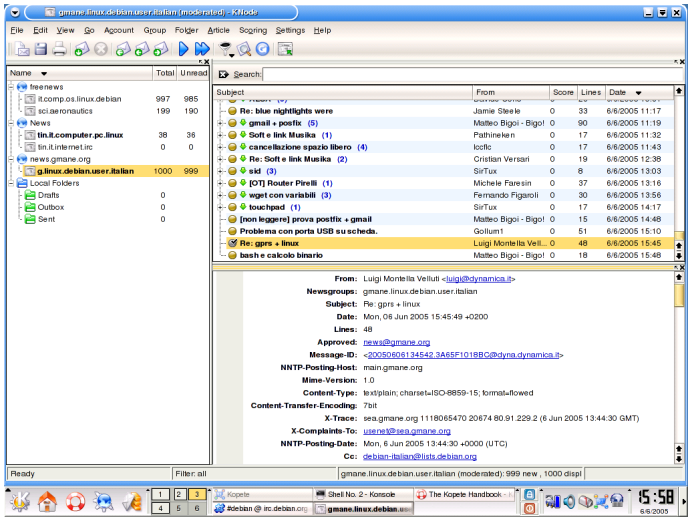

Figura 6.1: KNode: lettura di g.linux.debian.user.italian

1. installare e configurare un lettore di articoli Usenet (*newsreader*);
2. decidere tra i centinaia di gruppi disponibili quelli da leggere;
3. reperire i server che li distribuiscono.

Il primo problema è risolvibile installando un programma come **KNode** o **pan**. Tali programmi possono leggere gli articoli Usenet solo mentre sono connessi a internet; perfino gli articoli letti durante la connessione non sono leggibili in differita a connessione inattiva, a meno che non vengano esplicitamente salvati. Il secondo problema è relativamente semplice e dipende in larga misura dai propri interessi personali. Infine per risolvere il terzo problema si può cominciare a puntare il proprio *newsreader* al server **news.gmane.org** che gestisce alcuni gruppi che contengono la versione *news* delle principali liste di posta di Debian (Fig. 6.1); per esempio la lista:

```
debian-italian@lists.debian.org
```

diventa il gruppo di discussione:

```
g.linux.debian.user.italian
```

Bisogna notare che i server Usenet gratuiti sono pochi; di solito è il fornitore della connessione internet che garantisce ai propri abbonati l'accesso ad un tale server. Come avrete intuito dalle denominazioni la maggior parte dei gruppi di discussione è in lingua inglese. Le regole di comportamento in generale sono quelle descritte nella *Netiquette* del paragrafo precedente.

6.3 Messaggeria istantanea

I servizi di messaggeria istantanea (IM) consentono, con rapidità pari a quella del teletrasporto dell'astronave di Star Trek, di scambiare messaggi in tempo reale con altri utenti. La

Tabella 6.3: Reti di Messaggeria Istantanea (IM)

Rete	Note
ICQ	http://go.icq.com
IRC	Internet Relay Chat
Yahoo	http://edit.yahoo.com
Jabber	Protocollo basato su XML
AOL	http://aim.aol.com
MSN	http://messenger.msn.com

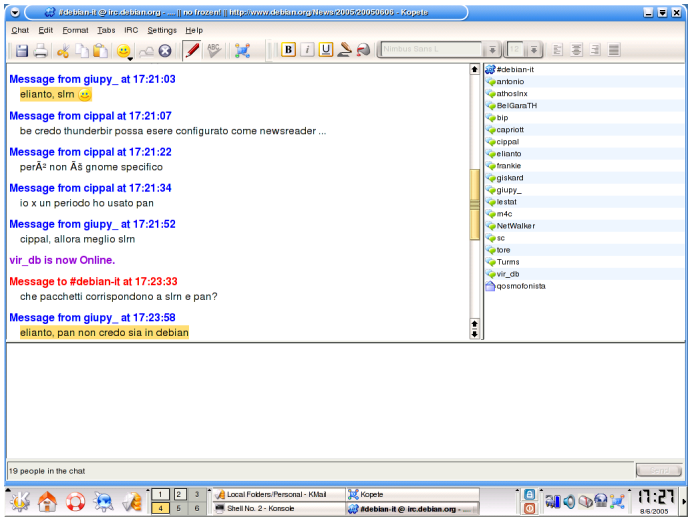

Figura 6.2: Kopete: collegamento al canale #debian-it

differenza con la posta elettronica sta nel fatto che si sa in ogni istante se gli utenti sono collegati o meno alla rete, facendo diventare il recapito dei messaggi uno scambio in tempo reale. Le principali reti di IM sono quelle elencate in tabella 6.3.

Gli utenti sono individuati tramite un soprannome (*nickname*) e di solito per collegarsi ogni rete richiede di usare il proprio specifico programma, rendendo così estremamente scomodo comunicare con utenti di reti diverse[3]; se però installate **Kopete** oppure **Gaim**, rispettivamente per KDE e Gnome, godrete del vantaggio di riunire in una sola applicazione la compatibilità di accesso alle reti più diffuse (Fig.6.2).

Naturalmente uno degli impieghi più utili di IM potrebbe essere quello di collegarsi con utenti Debian in grado di fornire assistenza. Debian utilizza la rete IRC ove la comunicazione avviene tramite **canali** (ce ne sono tanti). I canali gestiti da Debian sono sulla sottorete OFTC (*Open and Free Technology Community*) di IRC. Per collegarvi potete usare il server **irc.oftc.net**, su cui troverete in particolare i canali ufficiali #debian e #debian-it, quest'ultimo frequentato da utenti italiani. Anche se, per collegarvi, usate i programmi ricchi di grafica prima citati,

[3]Ciò è emblematico di come spesso, purtroppo, le aziende non si fanno scrupolo di applicare politiche commerciali che fanno gli interessi di profitto dell'azienda ma non quelli, più importanti e generali, della società.

potrebbero farvi comodo alcuni **comandi IRC**:

```
/nick gimpo
/msg giovanni ciao
```

che negli esempi di cui sopra rispettivamente cambiano il soprannome corrente a “gimpo”, mandano il messaggio privato “ciao” a colui con il soprannome “giovanni”. Su OFTC è facoltativa la registrazione del soprannome da voi scelto: se per esempio come soprannome avete scelto “carmilla” allora, una volta collegati come carmilla, digitate in sequenza i seguenti comandi IRC (la *password* la scegliete voi) per procedere alla registrazione:

```
/msg nickserv register <password>
/msg nickserv set hide email on
/msg nickserv set email <vostro_indirizzo_email>
/nick carmilla_
/msg nickserv register <password>
/msg nickserv link carmilla <password>
```

ove l’ultimo comando associa alla vostra *password* anche il soprannome alternativo “carmilla_”, per evitare che qualcuno se ne possa appropriare, ingenerando confusione. Oltre alla registrazione, che è permanente, ogni volta che ci si collega bisogna identificarsi così:

```
/msg nickserv identify <password>
```

altrimenti OFTC non ha modo di sapere se siete realmente voi a collegarvi con quel determinato soprannome. Sarebbe il caso di fare in modo che il programma IM che usiamo mandi automaticamente la stringa di identificazione all’atto del collegamento, per evitare di doverla inserire ogni volta. In assenza di identificazione, il soprannome risulta non usato e dopo un certo tempo decade e può essere assegnato ad un’altra persona. L’identificazione è una misura contro i **bot**, programmi automatici (robot, appunto) connessi a IRC che appaiono come un utente connesso, ma senza che in realtà ci sia dietro un essere umano che sta comunicando; spesso i bot sono usati per scopi malevoli.

L’impressione dell’autore è che la messaggeria istantanea, al di là di piacevoli discussioni dal tono leggero, non sia molto adatta a discussioni di natura tecnica, ove è spesso necessario fornire lunghe e meditate risposte; essa invece si rivela ottima per veicolare soluzioni rapide (*quick fix*), sfruttando il suo grande vantaggio: l’immediatezza della comunicazione.

7 Compilazione dei programmi

7.1 Compilazione del kernel

Il kernel, come ogni programma per elaboratore elettronico, è scritto in un linguaggio, detto anche codice sorgente, che deve essere tradotto da un altro programma, detto compilatore, in un linguaggio comprensibile al computer, che dopo può eseguire le istruzioni in esso contenute. La sequenza di istruzioni che viene fuori dal processo di compilazione viene denominata, non a caso, codice binario, perché è fatta di zeri e uno, la sola lingua che il computer è in grado di capire. Per semplicità, normalmente i programmi sono forniti già compilati e pronti per essere utilizzati, ma il kernel, cuore pulsante del nostro mostro da scrivania, merita un trattamento speciale.

Compilare il codice sorgente del kernel anzichè installare uno dei pacchetti pre-compilati assicura certamente alcuni vantaggi, tra cui:

- ➢ avvio più veloce perché i moduli da caricare sono solo quelli della vostra macchina e non ce ne sono di inutili
- ➢ sistema operativo più veloce perché potete ottimizzare il codice sul vostro processore
- ➢ attivare opzioni che sono inattive nei kernel pre-compilati
- ➢ stupire gli amici (le amiche un po' meno)

ma soprattutto, considerate che non potrete mai definirvi un vero linuxiano se non avete mai compilato un kernel! Nel caso decidiate di non usare i kernel pre-compilati messi a disposizione da Debian e vogliate compilare il vostro kernel personalizzato, per tenere il passo con gli aggiornamenti di sicurezza (paragrafo 8.2) avete due possibilità:

a. usare i kernel di Debian nel formato sorgente e ricompilarli ogni volta che essi incorporano dei rimedi contro bachi conosciuti (la versione del kernel non cambia);
b. usare i kernel standard di Linus ricompilando nuove versioni ogni volta che esse incorporano dei rimedi contro bachi (pararagrafo 8.3) conosciuti (la versione del kernel cambia).

Debian mette a disposizione dei comandi che rendono più semplice la procedura di compilazione del kernel; tale procedura verrà illustrata (quasi) passo-passo. Assicuratevi di aver installato i pacchetti[1] elencati in tabella 7.1.

Se siete su piattaforma x86 (in pratica tutti i comuni computer con processore Intel/AMD) dovrete avere installato anche il pacchetto **bin86**.

[1] Dato un pacchetto esiste spesso un pacchetto omonimo con estensione -dev, necessario solo per esigenze di compilazione; un esempio ne sono proprio i pacchetti **dpkg** e **dpkg-dev**.

Tabella 7.1: Pacchetti per compilazione kernel

kernel-package	make	libncurses-dev
binutils	gcc	dpkg-dev
build-essential	patch	libc-dev

Primo passo: ottenere il sorgente del kernel
Diversamente da come alcuni credono, come accennato sopra non ci sono controindicazioni nell'usare in Debian un kernel vanilla, curato da Linus Torvalds in persona, prelevato da kernel.org: l'unica differenza con i kernel Debian è che essi sono per comodità disponibili anche già compilati e contenenti delle specifiche *patch*. L'ultima versione stabile del kernel (denominata "vanilla") è sempre disponibile su internet; provate per esempio da:

```
http://ftp.it.kernel.org/pub/linux/kernel/v2.6/
```

Il sorgente è tipicamente contenuto in un file compresso: supponiamo di prelevare il file linux-2.6.19.tar.bz2

Secondo passo: scompattare il file sorgente
Posizionarsi in /usr/src dopo avervi posto il file linux-2.6.19.tar.bz2

```
# cd /usr/src
```

e digitare

```
# tar -xjf linux-2.6.19.tar.bz2
```

che creerà in /usr/src/linux-2.6.19 un albero di file sorgenti (*source-tree*); a questo punto creare il collegamento simbolico

```
# ln -s /usr/src/linux-2.6.19 linux
```

in modo da operare con comodità su /usr/src/linux invece che su /usr/src/linux-2.6.19. In alternativa, se si vuole usare un kernel Debian, digitare:

```
# aptitude install linux-source
```

Da notare che nel primo caso al termine della scompattazione ci si potrebbe trovare nell'avere degli strani numeri al posto dei soliti nomi utente e gruppo:

```
drwxrwxr-x 16 573 573 4096 2007-09-30 11:33 linux-2.6.19
```

ciò perché Debian inizia ad assegnare lo *user Id* a partire dal numero 1000 mentre il file scompattato assume che lo *user Id* si assegni a partire dal numero 500; l'utente *root* invece sarà sempre corretto, perché esso assume sempre l'identificativo utente e di gruppo pari al numero zero. Comunque, nel caso dei file sorgenti del kernel, una simile difformità non genera alcun problema e non necessita pertanto di alcuna correzione.

Terzo passo: accertarsi della compatibilità del sistema

Prima di iniziare la compilazione occorre verificare che le versioni del software di sistema siano compatibili con il nuovo kernel. A tale scopo, basta selezionare in un qualunque *file manager* la cartella "*Documentation*" in /usr/src/linux; lì apriamo il file "*Changes*" e in esso cerchiamo la sezione "*Current Minimal Requirements*": qui controlliamo che le versioni del nostro software di sistema siano accettabili (inutile controllare i programmi di servizio di filesystem che non usiamo, come per esempio di ReiserFS se noi usiamo ext3).

Quarto passo: applichiamo le eventuali patch

Questo passo è facoltativo perchè non è strettamente necessario applicare delle *patch*[2] e viene aggiunto solo per completezza. Se però applicate delle *patch*, assicuratevi di farlo nel giusto ordine. Qui si supporrà di avere un'unica super-patch da applicare, contenuta nel file linux-2.6.19-ck2.patch (già eventualmente scompattato); posizionare il file della *patch* in /usr/src/linux e digitare:

```
# cd /usr/src/linux
# patch -p1 < linux-2.6.19-ck2.patch
```

E' superfluo osservare che le *patch* devono riferirsi esattamente alla versione del kernel da compilare.

Quinto passo: configurare il kernel

Lanciamo il menu di configurazione:

```
# cd /usr/src/linux
# make menuconfig
```

ove compariranno anche le voci introdotte dalle *patch*, se le avete applicate. La configurazione del kernel è il passo più delicato: una scelta errata ne può pregiudicare gravemente il funzionamento. E' in questa sede che dovete decidere se volete un kernel monolitico o modulare[3]. Fare comunque attenzione al fatto che alcuni pezzi del kernel sono compilabili solo come moduli[4].

Esiste numerosa documentazione, inclusa quella in linea del menù di configurazione del kernel, che è prodiga di consigli sulle impostazioni da dare; inoltre il kernel stesso già contiene molte impostazioni predefinite di base (nel menù di configurazione le trovate già selezionate). Nel seguito si daranno comunque un minimo di indicazioni sulle impostazioni da dare (tenete anche presente che il modo in cui è strutturato il menù di configurazione cambia abbastanza spesso), ma il consiglio principe è quello di studiare l'hardware della vostra macchina e valutare attentamente le spiegazioni dell'aiuto in linea delle varie voci del menù di configurazione del kernel. Nel seguito per risposta "yes" si intende la selezione (come modulo o direttamente compilato nel kernel) della corrispondente voce del menu di configurazione, mentre per risposta "no" si intende la medesima voce lasciata o resa non selezionata.

[2] Un file *patch* contiene delle modifiche da applicare al sorgente; *applicare una patch* significa applicare tali modifiche all'albero dei file sorgenti del kernel.

[3] In un kernel modulare una parte delle funzionalità del kernel è dirottata su moduli esterni.

[4] I moduli sono anche talvolta denominati *driver*, soprattutto in altri sistemi operativi.

1. Nella sezione “Enable loadable module support” la risposta “yes” definisce un kernel modulare, mentre la risposta “no” un kernel monolitico.

2. Se, come probabile, avete una macchina moderna, dovete sicuramente attivare il supporto PCI nella sezione “Bus options”:

   PCI Support –> yes
   PCI device name database –> yes

3. Nella sezione “Networking Options” assicurarsi che siano attive:

   Networking Packet Filtering –> yes
   TCP/IP networking –> yes

4. Cercate ed attivate tutte le opzioni riguardanti ATA/IDE per far funzionare il disco fisso.

5. Nella sezione “File Systems” cercate ed attivate il supporto al tipo di filesystem che avete sul disco fisso, tipicamente ext3.

Prendetevi il vostro tempo: la configurazione viene salvata nel file nascosto `/usr/src/linux/.config` e può essere richiamata un numero infinito di volte.

Fate dei tentativi: questa procedura che si illustra permette, se qualcosa non funziona, di riavviare la macchina col vecchio kernel di sempre; considerate inoltre che la mostruosa potenza dei computer moderni permette di compilare il kernel in una manciata di minuti.

Sesto passo: compilare il sorgente

Una volta soddisfatti di come è configurato il kernel si può passare alla compilazione; è adesso che si usano gli strumenti Debian; digitare:

```
# cd /usr/src/linux
# make-kpkg --revision=prova.1.0 kernel_image
```

Questo comando produrrà un pacchetto Debian nuovo fiammante contenente il codice binario del nostro kernel personalizzato:

linux-image-2.6.19_prova.1.0_i386.deb

con “i386” se siamo su piattaforma x86. L’opzione *revision* è importante per impedire che il gestore dei pacchetti vada a sostituire il nostro kernel personalizzato in occasione di un dist-upgrade (pararagrafo 3.6 a pagina 41) del sistema. A questo punto non abbiamo fatto ancora nulla perché il nuovo kernel è stato compilato ma non ancora installato!

Settimo passo: installare il nuovo kernel

Per installare il kernel inserire:

```
# cd /usr/src
# dpkg -i linux-image-2.6.19_prova.1.0_i386.deb
```

Da notare che dpkg con l’opzione di installazione richiede di specificare il nome **completo** del pacchetto, come nell’esempio di cui sopra. Non resta ora che riavviare il sistema per far caricare automaticamente il nuovo kernel. Per essere sicuri che è in funzione il nuovo kernel inserire:

```
$ uname -a
```

Se per qualsiasi motivo si vuole caricare il vecchio kernel basta riavviare il sistema tenendo poi premuto il tasto *shift*: comparirà un menù da cui potrete selezionare con i tasti cursore l'etichetta "LinuxOld", che corrisponde al vecchio kernel. Questa procedura può variare un poco se per avviare il kernel usate GRUB, installato in modo predefinito da Debian, invece di usare LILO preferito dall'autore.

La procedura di compilazione descritta, conducendo alla generazione di un pacchetto Debian del kernel personalizzato, contiene il vantaggio della facile portabilità del kernel compilato, che si traduce in:

- ⋆ possibilità di compilare il kernel di una data macchina su macchine più veloci (utile soprattutto in passato)
- ⋆ possibilità di compilare il kernel di una data macchina su un'altra macchina di diversa piattaforma (*cross compilation*)

La prima possibilità è stata sfruttata con successo dall'autore nel caso di un portatile che non ne voleva sapere di avviare il kernel dell'installazione Debian: il problema è stato risolto compilando un kernel personalizzato (specificando l'hardware ed il processore del portatile) su un altro computer, passando poi al portatile il corrispondente pacchetto Debian, che una volta installato si è avviato regolarmente.

7.1.1 Ricompilazione del kernel

Dopo aver compilato il kernel, in caso di successiva compilazione si possono presentare due casi:

a. compilazione di una versione superiore del kernel in uso;

b. compilazione della medesima versione del kernel in uso.

Nel primo caso bisogna ricordarsi, prima di procedere alla compilazione, di **cancellare** il collegamento a /usr/src/linux; digitare pertanto:

```
# cd /usr/src
# rm linux
```

perché altrimenti si continuerebbe ad operare sui sorgenti del kernel di versione inferiore. Potrà altresì sorgere l'esigenza di analizzare solo le nuove voci del menù di configurazione introdotte dalla nuova versione; in questo caso, supponendo che il vecchio kernel sia la versione 2.6.10, inserire:

```
# cd /usr/src/linux
# cp /boot/config-2.6.19/.config .config
# make oldconfig
```

che copia la configurazione del vecchio kernel nell'albero di file sorgenti di quello di versione superiore. Tenete presente però che l'aiuto in linea non sarà disponibile e che questo non è possibile se si cambia la serie del kernel (per esempio passando da kernel 2.4.x a 2.6.x). Dopodichè si procede come illustrato nei passi precedenti.

Se invece non siamo soddisfatti di come funziona il kernel che abbiamo compilato e vogliamo ricompilarlo, siamo allora nel secondo caso e dobbiamo fare più attenzione, in modo da preservare il vecchio e provato kernel che costituisce la garanzia di poter ripristinare lo stato del sistema preesistente alle nostre compilazioni. La prima cosa da fare è riavviare la macchina facendo caricare il vecchio kernel, come spiegato in precedenza; in caso contrario verrebbe sovrascritto dagli script di kernel-package (ci ritroveremmo con due kernel della stessa versione compilati in momenti diversi): è sempre il vecchio e fidato kernel che deve compilare i nostri kernel sperimentali!

Se poi avete attivato la funzione di modularità del kernel, dovete rinominare la cartella in /lib/modules che contiene i moduli del kernel; nel nostro esempio sarà:

```
# mv /lib/modules/2.6.19 /lib/modules/2.6.19.old
```

oppure potete drasticamente cancellarli

```
# rm -R /lib/modules/2.6.19
```

e successivamente procedere alla nuova configurazione:

```
# cd /usr/src/linux
# make menuconfig
```

e quindi alla compilazione con un *revision number* maggiorato:

```
# make-kpkg clean
# make-kpkg --revision=prova.2.0 kernel_image
```

che consentirà al gestore di pacchetti di installare il kernel di medesima versione ma di diversa configurazione come se fosse di versione superiore:

```
# cd /usr/src
# dpkg -i linux-image-2.6.19_prova.2.0_i386.deb
```

Potete iterare questo procedimento un numero infinito di volte, mantenendo a portata di mano il vecchio e fidato kernel.

7.2 Installare un programma dai sorgenti: WINE

L'installazione di un programma in formato binario pacchettizzato è sempre preferibile, per facilità e rapidità, all'installazione del medesimo programma partendo dal formato sorgente da compilare. E' anche opportuno non esagerare con il numero di programmi compilati da sorgente, perchè essi sono virtualmente invisibili al gestore dei pacchetti (questo non vale se dopo la compilazione viene creato un pacchetto Debian); al crescere della presenza

nel sistema di programmi non pacchettizzati si perde via via il vantaggio principale dei pacchetti, cioè la facilità di gestione del software installato.

In un documento del tipo *howto* dedicato alla compilazione dei programmi si afferma che bisogna essere preparati al fatto che il 50% dei sorgenti non si compilerà correttamente, ma nella mia esperienza, forse a causa dell'imperizia, tale percentuale deve essere aumentata. Se, come appare, non è davvero cosa banale la compilazione corretta di un sorgente, nondimeno si prenderà in esame un caso, che dovrebbe risultare semplice.

Considereremo la compilazione di WINE, un simulatore del sistema operativo Windows di Microsoft (per le versioni di Windows maggiormente assistite riferirsi alla documentazione del programma), specificamente sviluppato per la piattaforma x86 (**non** funziona sulle altre piattaforme). Il progetto, cominciato nel lontano 1992, ha totalizzato più di un milione di righe di codice e oltre cinquecento sviluppatori.

Il programma è disponibile in forma pacchettizzata in Debian, ma essendo in stadio "beta", cioè ancora in sviluppo (esce una nuova versione ogni pochi mesi), rappresenta un ottimo esempio in cui è conveniente installare l'ultima versione prelevata da internet, partendo dalla compilazione del codice sorgente.

Per compilare e installare il programma (eventualmente scompattato) portarsi nella cartella del sorgente ed inserire:

```
$ ./tools/wineinstall
```

e rispondere alle domande dello script: attenzione a quando si chiede se avete un computer con Windows presente oppure no[5]. Lo script inoltre ad un certo punto chiederà la password di root per installare WINE (eseguendo automaticamente il comando make install).

Prima di installare una nuova versione di WINE è opportuno cancellare la precedente; è sufficiente inserire:

```
# cd /usr/local/src/wine
# make uninstall
```

ove /usr/local/src/wine è la cartella ove si suppone sia il sorgente della precedente versione di WINE.

In generale i sorgenti ben confezionati dai propri autori hano una funzionalità "make uninstall" che provvede ad eliminare dal sistema tutto quello che è stato precedentemente installato, proprio come in WINE.

Spesso il comando make uninstall non funziona, semplicemente perché l'autore del programma non lo ha implementato. In questi casi non resta che cancellare manualmente tutte le tracce del programma.

[5] Ricordatevi che i **credenti** piallano senza ritegno altri sistemi operativi, mentre i **nuovi adepti** fanno convivere Linux con Windows o MacOS.

8 Perfezionamenti di Debian

8.1 L'immagine più bella del reame

Chi non ricorda la favola di Biancaneve ove la strega cattiva si guarda allo specchio chiedendo qual'è la più bella del reame? Linux però si occupa di altre immagini ovvero delle cosiddette immagini ISO. Si tratta di particolari file in formato ISO che contengono "l'immagine", ovvero un file unico con l'esatto contenuto di un CD-ROM; pertanto memorizzano un intero filesystem di tipo ISO9660.

Tutti i CD di Debian reperibili in rete sono distribuiti in forma di file ISO, che per poter essere utilizzati devono essere masterizzati in modo da "ricostruire" un CD con filesystem ISO9660 a partire da un supporto vergine come un CD-R o un CD-RW. Bisogna cioè sottolineare che non è sufficiente masterizzare la pura copia del file unico ISO; nell'operazione di masterizzazione bisogna selezionare l'opzione specifica per un'immagine ISO, a pena la creazione di un CD inservibile, buono solo a conservare a sua volta una copia del file ISO. Per esempio in k3b, famoso software di masterizzazione del KDE, occorre selezionare la voce "Burn CD ISO image" (in realtà questo programma la attiva automaticamente non appena si clicca sopra un'icona che rappresenta un'immagine ISO).

Sono distribuiti mediante immagine ISO anche molti *live*-DVD, cioè distribuzioni Linux che si avviano da DVD e funzionano senza che sia necessaria l'installazione sul disco fisso del computer, a pena di un certo rallentamento generale nelle fasi di caricamento dei programmi. Knoppix, basato su Debian e sviluppato da Klaus Knopper, ne costituisce un esempio famoso. Questi *live*-DVD, avendo eccellenti programmi di autoconfigurazione, sono utilissimi per provare la compatibilità hardware con Linux di un computer, anche per esempio direttamente nella vetrina del negozio prima dell'acquisto, senza minimamente andare a toccare il disco fisso della macchina. Basta solo inserire il DVD e riavviare la macchina per avere in pochi minuti un sistema Linux perfettamente funzionante, se almeno le principali periferiche vengono riconosciute dal processo di autoconfigurazione.

8.2 Suid root: l'importanza di essere sicuri

Vi sarà forse capitato di ascoltare l'oscura dizione di "bit suid root" riferita ad un determinato programma. L'argomento riguarda i permessi associati ad un file (si veda il paragrafo 2.1 a pagina 15); a questi permessi si aggiungono altre informazioni definibili globalmente come modalità dei permessi, nelle quali rientra la modalità detta "suid''.

La modalità suid attribuisce al file in esecuzione i privilegi dell'utente cui appartiene; se però l'utente cui appartiene è l'amministratore di sistema, ecco che la modalità suid **attribuisce al file in esecuzione i privilegi di root**. Ciò sarà più chiaro con un semplice esempio.

Facciamo per prima cosa una copia dell'eseguibile cp nella nostra cartella personale:

```
$ cd ~
```

```
$ cp /bin/cp .
```

e non dimenticate il punto finale, mi raccomando! Assumiamo i privilegi di root:

```
$ su
Password: ****
```

e cambiamo l'utente proprietario del file cp:

```
# chown root.root cp
```

attribuendo al file la modalità suid:

```
# chmod u+s cp
```

Infatti se ora facciamo elencare il file:

```
# ls -l cp
-rwsr-xr-x root root 51212 2004-08-24 17:48 cp
```

osserviamo che al posto della normale "x" compare una "s", che sta ad indicare la modalità suid root.

Torniamo dunque allo stato precedente lasciando i privilegi di root:

```
# exit
```

e creiamo il solito file vuoto "prova":

```
$ touch prova
```

e copiamolo adesso col nostro cp modificato (non quello di sistema) nel file "prova2":

```
$ ./cp prova prova2
```

Infine facciamoci fare un elenco per vedere il risultato ottenuto; vedremmo qualcosa del tipo:

```
$ ls -l prova prova2
-rw-r--r-- 1 morena morena 0 Aug 23 12:56 prova
-rw-r--r-- 1 root morena  0 Aug 23 12:57 prova2
```

Sorpresa! Il file copiato prova2 appartiene a root **pur essendo stato creato da un utente normale**. Le implicazioni sulla sicurezza del sistema derivanti dall'uso di file eseguibili di proprietà di root in modalità suid sono enormi e pericolose, perchè la funzionalità suid non consente un uso selettivo sulla base dell'utente che esegue il comando stesso: in altre parole, chiunque può usare il comando con i privilegi di root. Infatti uno dei grandi problemi di Linux è che l'utente root non ha restrizioni, rendendo in pratica Linux un sistema a "*single point of failure*".

La sicurezza del sistema dovrebbe essere tenuta presente sin dalla fase del partizionamento dei dischi[1] durante l'installazione. Data la vastità e importanza dell'argomento, si consiglia caldamente di leggere il *Securing Debian Manual*, disponibile con:

[1] Si veda il paragrafo 2.6 a pagina 23.

```
# aptitude install harden-doc
```

Intanto assicuratevi che nel vostro /etc/apt/sources.list sia presente la seguente riga:

deb http://security.debian.org/debian-security stable/updates main contrib non-free

Potrete così almeno accedere agli aggiornamenti fatti alla distribuzione per migliorarne la sicurezza, digitando saltuariamente:

```
# aptitude update
# aptitude upgrade
```

Questi comandi aggiorneranno solo i pacchetti che effettivamente risultano installati sul vostro computer. La distribuzione Debian è caratterizzata da una cura particolare verso tutti gli aspetti concernenti la sicurezza e si aspetta che l'amministratore di sistema faccia altrettanto.

8.3 Uso corretto del Debian BTS

In gergo informatico un errore di programmazione che dà luogo ad un malfunzionamento di un programma che lo contiene è chiamato baco, italianizzazione dell'originale inglese *bug* che significa insetto.

Se vi state chiedendo cosa c'entrano gli insetti con gli errori di programmazione, sappiate che il primo baco informatico della storia pare si verificò nel 1945 su un calcolatore della serie Mark presso l'università di Harward (USA). Il calcolatore funzionava a relè elettromagnetici ed un insetto, una farfalla notturna rimasta intrappolata nei meccanismi di commutazione, causò il blocco del calcolatore su cui lavorava la programmatrice Grace Brewster Murray Hopper, la quale fu costretta ad intervenire manualmente, estraendo la farfalla dalle viscere del calcolatore, per ripristinare il normale funzionamento della macchina.

Non è affatto facile scrivere software esente da bachi; ancora più difficile è mettere insieme distribuzioni Linux che funzionino in modo armonioso e senza errori nel tempo. Debian nello sforzo di produrre distribuzioni stabili almeno prive di bachi critici si è dotata di un sistema di individuazione dei bachi (*BTS, Bug Tracking System*) che archivia i dettagli dei bachi segnalati dagli utenti e dagli sviluppatori, consultabile presso:
http://bugs.debian.org
anche da riga di comando con:

```
$ lynx http://bugs.debian.org
```

ove è possibile anche segnalare un baco. In alternativa si può usare il programma **reportbug**, che è progettato per segnalare i bachi via posta elettronica; esso guida alla creazione di una *email* predefinita e la invia al BTS presso l'indirizzo submit@bugs.debian.org con i dettagli relativi al baco segnalato. Ad ogni baco viene sempre assegnato un numero univoco e distintivo, più altre informazioni per facilitare l'eliminazione del baco stesso.
Prima di segnalare un baco si leggano attentamente le modalità di segnalazione presso:
http://www.debian.org/Bugs/Reporting
oppure direttamente sul vostro sistema con:

```
$ less /usr/share/doc/debian/bug-reporting.txt
$ less /usr/share/doc/debian/bug-maint-info.txt
```

Ci sono altri BTS come ad esempio bugzilla (usato dal progetto Mozilla) e jitterbug (usato da Samba, FreeCiv) ma non sono compatibili con reportbug, che è specifico per il Debian BTS.

Un baco viene chiuso solo quando il problema che l'ha causato è risolto. Il problema si considera risolto quando un pacchetto contenente la soluzione del baco entra negli archivi Debian. Un baco può essere chiuso solo da:

1. colui che ha segnalato il baco
2. il manutentore del pacchetto con il baco

ma ci sono eccezioni a questa regola ed il manutentore ha la precedenza, nel senso che può chiudere il baco anche se colui che l'ha segnalato non è d'accordo. Infine alcune raccomandazioni:

- ➢ non mischiare in un'unica segnalazione bachi di pacchetti diversi
- ➢ informarsi, prima di segnalare un baco, se è già stato segnalato da qualcun altro: in questo caso si possono eventualmente aggiungere ulteriori commenti
- ➢ chiedere aiuto agli esperti su debian-italian oppure, se conoscete l'inglese, su debian-user, se non si riesce a collegare un baco ad un determinato pacchetto

Anche se la segnalazione dei bachi è aperta agli utenti oltre che agli sviluppatori, si consiglia di usare Linux per un congruo periodo di familiarizzazione (diciamo almeno un anno) prima di avventurarsi nella segnalazione di un baco; infatti non è facile per un principiante distinguere tra il frutto di un errore di programmazione ed un malfunzionamento, causato magari da un errore di configurazione commesso dall'utente medesimo. In ogni caso, se siete in dubbio chiedete agli esperti.

La segnalazione dei bachi è sempre auspicabile e gradita, anche se usate la distribuzione stabile, perchè essa quando viene revisionata per incorporare eventuali aggiornamenti di sicurezza a volte incorpora anche aggiornamenti che risolvono bachi particolarmente gravi, che erano sfuggiti durante il periodo di prova. E' chiaro però che la segnalazione di bachi è un'attività idealmente concepita per gli utenti delle distribuzioni *testing* e *unstable*.

8.4 CPAN: Comprehensive Perl Archive Network

CPAN è un archivio liberamente consultabile su internet all'indirizzo `http://www.cpan.org` di software scritto in Perl. Il linguaggio Perl, creato da Larry Wall nel 1987, usatissimo negli script dei sistemi Linux, viene usato anche per scrivere veri e propri programmi. Presso CPAN si possono trovare numerosi moduli, script, e documentazione sul linguaggio Perl suddivisi per aree tematiche: grafica, sicurezza, manipolazione di file, etc.

Tutto ciò non è molto interessante per chi non sia un programmatore Perl, ma potrebbe tornare utile nel caso si voglia installare un programma in Perl che necessita di moduli non disponibili in Debian. Normalmente è possibile scaricarli automaticamente da CPAN; se per esempio ci manca il modulo "Set::Scalar", per scaricarlo dalla rete ed installarlo inserire:

```
# perl -MCPAN -e 'install "Set::Scalar"
```

ma è possibile far di meglio installando dh-make-perl di Paolo Molaro, che permette di pacchettizzare il modulo, se esso non è troppo complesso, subito dopo averlo scaricato dalla rete:

```
# aptitude install dh-make-perl
# dh-make-perl Set::Scalar --cpan module --build
```

Il pacchetto così ottenuto si può poi installare sul sistema nei modi soliti. Alcuni moduli Perl richiedono un minimo di interattività per rispondere ad alcune domande per la loro corretta configurazione.

8.5 Approfondire Debian

Se quello che avete letto vi sembra semplice e vi trovate a vostro agio davanti alla quieta riga di comando, qualcosa di più impegnativo per la vostra tastiera è "La guida Debian" di Osamu Aoki, a portata di dito con:

```
# aptitude install debian-reference-it
```

ove troverete anche infiniti suggerimenti sull'enorme mole di documentazione tecnica disponibile per Debian e Linux. Tenendo presente la non numerosa documentazione in italiano, vale la pena citare, anche se non direttamente ed esclusivamente riguardanti Debian:

- Linux Facile di Daniele Medri
- Appunti di Informatica Libera di Daniele Giacomini et. al.
- Documentazione *howto*

rispettivamente installabili nel vostro computer con:

```
# aptitude install linuxfacile
# aptitude install appunti-informatica-libera
# aptitude install doc-linux-it
```

e consultabili facilmente mediante i menù di **dwww** (paragrafo 5.1 a pagina 49).

Indice analitico