# Linux-Mandrake

## Manuale di riferimento

**MandrakeSoft**

**Novembre 2000**

**http://www.linux-mandrake.com/**

**Linux-Mandrake : Manuale di riferimento**
MandrakeSoft

# Sommario

**Prefazione ..... 15**

Informazioni legali ..... 15

1. ..... 15

Autori e traduttori ..... 16

1 ..... 16

Strumenti usati per la stesura di questo manuale ..... 16

Nota del curatore ..... 17

Convenzioni usate in questo manuale ..... 17

Convenzioni tipografiche ..... 18

Convenzioni di tipo generale ..... 19

**1. Introduzione ..... 23**

**2. Concetti base di `Unix` ..... 25**

Utenti e gruppi ..... 25

21 ..... 25

Principi di base riguardo i file ..... 28

22 ..... 30

23 ..... 31

24 ..... 31

25 ..... 32

Processi ..... 32

Breve introduzione alla linea di comando ..... 33

`cd`: *`Change Directory`* ..... 35

Alcune variabili di ambiente e il comando `echo` ..... 36

`cat`: visualizza il contenuto di uno o più file sullo schermo 37

`less`: un visualizzatore a pagine ..... 38

`ls`: elencare file (*`LiSt`*) ..... 39

Scorciatoie utili per la tastiera ..... 40

**3. Introduzione alla linea di comando ..... 43**

Comandi per la gestione dei file ..... 43

`mkdir`, `touch`: creazione di directory e file vuoti (*`MaKe DIRectory`*) ..... 43

rm: cancellazione di file o directory (*ReMove*) ..... 44
mv: spostare o rinominare dei file (*MoVe*) ..... 45
cp: copiare file e directory (*CoPy*) ..... 46
Gestione degli attributi dei file ..... 47
chown, chgrp: cambia il proprietario o il gruppo di uno o più file ..... 47
chmod: modifica dei permessi di file e directory (*CHange MODe*) ..... 49
I caratteri speciali (meta-caratteri) e le ***espressioni regolari*** nella *shell* ..... 51
312. ..... 51
313. ..... 52
La redirezione e le pipe ..... 52
Ancora a proposito dei processi ..... 52
La redirezione ..... 53
Le pipe ..... 55
Completamento automatico ..... 56
Esempio ..... 56
Altri metodi di completamento ..... 57
Avviare e gestire i processi in background: il controllo dei job 58
Conclusione ..... 59
**4. Elaborazione testi: Emacs e VI ..... 61**
*Emacs* ..... 61
Una breve introduzione ..... 61
I primi passi ..... 62
La gestione dei buffer ..... 63
Copia, taglia, incolla, cerca ..... 64
Uscire da *Emacs* ..... 67
*VI*: l'antenato ..... 67
Modalità inserimento, modalità comandi, modalità ex ... 67
La gestione dei buffer ..... 69
Elaborazione del testo e comandi di movimento ..... 69
Taglia, copia, incolla ..... 71
Uscire da *VI* ..... 73
Un'ultima parola... ..... 74

**5. La stampa ..... 75**
Installazione e gestione delle stampanti ..... 75
Installazione di *CUPS* e uso della sua interfaccia *web* ..... 75
Configurazione di una nuova stampante ..... 76
Una nota riguardo la sicurezza ..... 83
Gestione della coda di stampa ..... 83
La stampa di documenti ..... 85
Semplice stampa di un file ..... 85
Configurazione avanzata ..... 88
**6. Samba ..... 95**
Che cos'è *Samba*? ..... 95
Che cosa può fare *Samba*? ..... 95
61 ..... 95
Installazione di *Samba* ..... 96
Descrizione del file `/etc/smb.conf` ..... 96
La sezione `[global]` ..... 96
Impostare una condivisione ..... 98
La sezione `[homes]` ..... 99
La sezione `[printers]` ..... 100
Test e attivazione ..... 102
Alcuni strumenti ..... 102
`smbclient` ..... 102
`smbfs` ..... 103
`smbtar` ..... 103
*SWAT*, lo strumento grafico di configurazione ..... 104
**7. MSEC – Mandrake Security tools ..... 107**
Introduzione a *MSEC* ..... 107
71 ..... 107
Impostazione del livello di sicurezza ..... 107
Livello 0 ..... 108
Livello 1 ..... 109
Livello 2 ..... 109
Livello 3 ..... 109
Livello 4 ..... 109

Livello 5 ..... 110
Caratteristiche dei livelli di sicurezza ..... 110
73 ..... 110
75 ..... 113
“controllo generale di sicurezza” ..... 113
“*umask* per gli utenti” ..... 113
“*umask* per root” ..... 114
“*shell* senza password” ..... 114
“autorizzati a connettersi al display *X*” ..... 114
“utenti nel gruppo audio” ..... 114
“. in $PATH” ..... 115
“avvisi nel file /var/log/security.log” ..... 115
“avvisi direttamente su *tty*” ..... 115
“avvisi in *syslog*” ..... 115
“avvisi inviati via *e-mail* a root” ..... 115
“controllo dei file *suid* root” ..... 115
“controllo MD5 dei file *suid* root” ..... 115
“controllo file scrivibili” ..... 116
“controllo permessi” ..... 116
“controllo file gruppo *suid*” ..... 116
“controllo file senza proprietario” ..... 116
“controllo promiscuità” ..... 116
“controllo porte in ascolto” ..... 117
“controllo integrità file passwd” ..... 117
“controllo integrità file shadow” ..... 117
“controllo sicurezza del sistema ogni giorno a mezzanotte” ..... 117
“i servizi sconosciuti sono disabilitati” ..... 117
“password di avvio” ..... 118
“accetta connessioni da” ..... 118
**8. Organizzazione della struttura del filesystem ..... 121**
Dati condivisibili e non, statici e variabili ..... 121
La directory radice: / ..... 122
81 ..... 122
/usr: quella grande ..... 123

82 ..... 123
/var: dati modificabili durante l'uso ..... 124
83 ..... 124
/etc: file di configurazione ..... 125
84 ..... 125
**9. Filesystem e punti di mount ..... 127**
Principi ..... 127
91 ..... 129
Partizionamento di un disco rigido e formattazione di una partizione ..... 129
I comandi mount e umount ..... 129
Il file /etc/fstab ..... 131
92 ..... 131
93 ..... 132
Una nota sull'opzione supermount ..... 133
**10. Il filesystem di GNU/Linux: ext2fs (EXTended 2 FileSystem) ..... 135**
Tutto è un file ..... 135
101. ..... 135
Collegamenti ..... 137
101 ..... 137
Pipe "anonime" e pipe con nome ..... 139
File "speciali": file in modalità a caratteri e file in modalità a blocchi ..... 141
I link simbolici e le limitazioni degli "hard" link ..... 143
102 ..... 144
Attributi dei file ..... 145
102. ..... 145
**11. Il filesystem /proc ..... 147**
Informazioni sui processi ..... 147
111. ..... 148
Informazioni sull'hardware ..... 150
111 ..... 152
La sotto-directory /proc/sys ..... 152

112. ..... 153
**12. I file di avvio del sistema: init “System V” ..... 155**
In principio fu init ..... 155
I runlevel ..... 156
121 ..... 157
**13. Controllo dei processi ..... 159**
Ancora sui processi ..... 159
L’albero dei processi ..... 159
Segnali ..... 159
Informazioni sui processi: i comandi ps e pstree ..... 160
ps ..... 160
pstree ..... 161
Inviare segnali ai processi: kill, killall e top ..... 161
Scorciatoia in *X*: xkill ..... 161
kill, killall ..... 162
top ..... 163
**14. Compilazione e installazione di nuovi kernel ..... 165**
Dove trovare i sorgenti del kernel ..... 165
Decomprimere i sorgenti, applicare le patch al kernel (se necessario) ..... 166
Configurazione del kernel ..... 167
141 ..... 168
142 ..... 169
Compilazione del kernel e dei moduli, installazione dei moduli 178
Installazione del nuovo kernel ..... 179
Aggiornamento di *GRUB* ..... 179
Aggiornamento di *LILO* ..... 182
**15. La compilazione e l’installazione di software libero ..... 189**
Introduzione ..... 189
Prerequisiti ..... 190
Compilazione ..... 191
Struttura di una distribuzione ..... 192
Decompressione ..... 193

Archivi `tar.gz` ..... 193
Uso di *GNU Tar* ..... 194
`bzip2` ..... 196
Fatelo e basta! ..... 196
Configurazione ..... 198
155 ..... 198
*Autoconf* ..... 198
*Imake* ..... 202
Vari script da *shell* ..... 202
Alternative ..... 203
Compilazione ..... 203
`make` ..... 204
Regole ..... 204
Attenti, pronti, via! ..... 205
Spiegazioni ..... 206
Cosa fare... se non funziona? ..... 207
Installazione ..... 215
Con `make` ..... 215
Problemi ..... 216
Supporto ..... 217
Documentazione ..... 217
Supporto tecnico ..... 217
Come reperire il software libero ..... 218
Ringraziamenti ..... 219
1522 ..... 219
**16. Strumenti da linea di comando ..... 221**
`grep`: *General Regular Expression Parse* ..... 221
161. ..... 221
`find`: cerca file in base a determinati criteri ..... 222
162. ..... 223
163. ..... 223
164. ..... 224
165. ..... 225
`crontab`: analizzare o modificare il file `crontab` ..... 227
166. ..... 227

`at`: programmare un comando, ma per una sola volta ..... 229
167. ..... 229
168. ..... 230
`tar`: *`Tape ARchiver`* ..... 230
169. ..... 231
`bzip2` e `gzip`: programmi per la compressione di dati ..... 233
1610. ..... 234
Tanti, tanti altri... ..... 235
**A. La Licenza Pubblica Generica GNU ..... 237**
Preambolo ..... 237
Termini e condizioni per la copia, la distribuzione e la modifica ..... 239
A1. ..... 239
**Glossario ..... 247**
**Indice ..... 291**

# Lista delle Figure

2-1. Schermata di login in modalità grafica ..... 26
2-2. Schermata di login in modalità console ..... 27
2-3. L'icona del terminale nel pannello di *KDE* ..... 34
4-1. *Emacs*, modifica simultanea di due file ..... 62
4-2. *Emacs*, prima di copiare il blocco di testo ..... 64
4-3. *Emacs*, dopo la copia del blocco di testo ..... 65
4-4. Situazione iniziale in *VIm* ..... 67
4-5. *VIm*, prima di copiare il blocco di testo ..... 72
4-6. *VIm*, dopo la copia del blocco di testo ..... 73
5-1. La pagina di benvenuto di *CUPS* ..... 76
5-2. La lista priva di stampanti di *CUPS* ..... 77
5-3. La finestra di login di *CUPS* ..... 77
5-4. Installazione di una nuova stampante, passo 1 ..... 78
5-5. Installazione di una nuova stampante, passo 2 ..... 79
5-6. Installazione di una nuova stampante, passo 3 ..... 81
5-7. Installazione di una nuova stampante, passo 4 ..... 81
5-8. La pagina relativa allo stato della stampante ..... 84
5-9. La finestra principale di *XPP* ..... 86
5-10. Scelta di un file con *XPP* ..... 86
5-12. La scheda delle opzioni base di *XPP* ..... 89
5-13. La scheda relativa alle opzioni per i file di testo di *XPP* ..... 91
5-14. La scheda relativa alle opzioni avanzate di *XPP* ..... 91
6-1. Connessione a *Samba* ..... 104
6-2. La pagina di benvenuto di *SWAT* ..... 105
9-1. Un filesystem non ancora montato ..... 127
9-2. Il filesystem ora è montato ..... 127
13-1. L'icona di `xkill` ..... 162
13-2. Esempio di esecuzione di `top` ..... 163

# Prefazione

## Informazioni legali

| | Original Copyright | License |
|---|---|---|
| *La compilazione e l'installazione di software libero, pag. 189* | Benjamin Drieu, **APRIL (www.april.org/)** | *La Licenza Pubblica Generica GNU, pag. 237* |

## Autori e traduttori

Le persone qui elencate hanno contribuito alla stesura dei manuali di **Linux-Mandrake**:

- Yves Bailly
- Camille Bégnis
- Francis Galiègue
- Hinrich Göhlmann
- Carsten Heiming
- Fabian Mandelbaum
- Roberto Rosselli Del Turco
- Stefan Siegel

Inoltre hanno contribuito anche: Nicolas Berdugo, Sébastien Blondeel, Paolo Calisse, Vincent Danen, Marco De Vitis, Hoyt Duff, Jürgen Grojer, Andrea Monni, Matt Sherer.

## Strumenti usati per la stesura di questo manuale

Questo manuale è stato impaginato con *DocBook*. Sono stati utilizzati il linguaggio *Perl* e *GNU Make* per gestire i file relativi. I sorgenti in SGML sono stati elaborati con `openjade` e `jadetex`, facendo uso dei

fogli di stile di Norman Walsh. Le immagini sono state catturate con `xwd` e *GIMP*, e convertite con `convert` (quest'ultimo programma fa parte del pacchetto *ImageMagick*). I file `PostScript` sono stati generati con il programma `dvips`. Tutti questi programmi sono presenti nella vostra distribuzione **Linux-Mandrake**, e sono tutti liberamente distribuibili.

## Nota del curatore

Come potrete notare passando da un capitolo all'altro, questo libro è un documento composito, frutto del lavoro di vari autori. Per quanto sia stata esercitata la massima cura nell'assicurare una omogeneità sul piano tecnico e lessicale, lo stile di ogni autore è stato ovviamente mantenuto.

Alcuni degli autori, inoltre, hanno scritto in inglese malgrado questa non sia la loro lingua madre. Per questo motivo, se notate delle strane costruzioni sintattiche non esitate a segnalarcele.

Per finire, in pieno accordo con la filosofia del software libero, eventuali contributi saranno molto apprezzati! Potete fornire un valido aiuto a questo progetto di documentazione in molti modi: se avete molto tempo a disposizione, potete scrivere un capitolo intero; se parlate una lingua straniera, potete contribuire all'internazionalizzazione di questo libro. Se avete qualche idea su come migliorare il contenuto, fateci sapere: anche la correzione di un errore di battitura sarà ben accolto!

Per informazioni in merito al progetto di documentazione **Linux-Mandrake** per favore contattate l'amministratore della documentazione (mailto:documentation@mandrakesoft.com).

# Convenzioni usate in questo manuale

## Convenzioni tipografiche

Sono state utilizzate diverse forme di evidenziazione del testo, al fine di rendere immediatamente evidenti e distinte rispetto al testo normale alcune parole di tipo speciale. La tabella che segue vi propone un esempio per ciascun tipo o gruppo di parole speciali, con la sua speciale formattazione grafica e il relativo significato.

| Esempio formattato | Significato |
|---|---|
| ***inode*** | Questo tipo di formattazione ha lo scopo di mettere in evidenza un termine tecnico spiegato nel *Glossario*. |
| `ls -lta` | Usato per comandi o argomenti necessari a tali comandi. Questa formattazione è applicata a comandi impartiti da linea di comando, alle loro opzioni e ai nomi di file. Si veda anche la sezione riguardo la "*Sintassi dei comandi, pag. 19*" |
| `$ ls *.pid imwheel.pid $` | Usata per frammenti di testo che potreste (o dovreste) vedere sul vostro schermo. Include esempi di interazione con il computer, listati di programmi, etc. |
| `localhost` | In questo caso si tratta di qualche tipo di dato letterale che, in genere, non rientra in nessuna delle categorie definite in precedenza. Un esempio potrebbe essere costituito da una parola chiave di un file di configurazione. |
| *Apache* | Questa formattazione è usata per indicare i nomi di applicazioni. Non è il caso dell'esempio che abbiamo usato, ma in particolari contesti il nome dell'applicazione e il nome di un comando che fa parte di tale applicazione potrebbero coincidere, la loro formattazione provvederà a tenerli distinti. |

| Esempio formattato | Significato |
|---|---|
| Files | Questa viene usata per le voci di menu e, più in generale, per il testo degli elementi di interfacce grafiche. La lettera sottolineata indica la scorciatoia da tastiera, se presente. |
| *SCSI-Bus* | Indica un componente del computer, o il computer stesso. |
| *Le petit chaperon rouge* | Le parole evidenziate appartengono a un lingua diversa rispetto a quella in cui è scritto il manuale. |
| **Attenzione!** | Questa formattazione, come avrete intuito, è riservata ad avvertimenti particolari, e ha la funzione di enfatizzare ogni singola parola (come se fossero gridate ;-) |

## Convenzioni di tipo generale

### Sintassi dei comandi

L'esempio che segue vi mostra il tipo di caratteri che troverete in questo manuale quando descriveremo gli argomenti di un comando:

```
command <non literal argument> [-option={arg1,arg2}] [optional argument ...]
```

Queste convenzioni seguono lo standard e le troverete identiche in altre parti della documentazione (elettronica), come ad esempio le pagine `man`.

I caratteri "<" (meno di) e ">" (più grande di) indicano un argomento che non dev'essere digitato così com'è, ma che dev'essere stabilito in base ai vostri bisogni. Ad esempio, `<nome_di_un_file>` si riferisce al nome di un file effettivamente esistente: se il nome in questione

è `pippo.txt`, dovreste digitare `pippo.txt`, e non `<pippo.txt>` o `<nome_di_un_file>`.

Le parentesi quadre ("[ ]") indicano argomenti opzionali, che potreste voler includere nella linea relativa al comando, oppure no.

I puntini di sospensione ("...") significano che in questo punto è possibile inserire un numero di elementi arbitrario.

Le parentesi graffe ("{ }") contengono gli argomenti che possono essere inseriti in questo punto: uno di loro va inserito nella riga di comando.

## Notazioni particolari

Di tanto in tanto vi verrà chiesto di premere i tasti Ctrl+R. Questo significa che dovete premere e mantenere premuto il tasto Ctrl mentre allo stesso tempo premete anche il tasto R. Lo stesso principio vale per il tasto Alt.

Lo stesso per i menu, scegliere la voce di menu File→Ricarica configurazione utente (**Ctrl+R**) significa: cliccate sul testo File sulla barra dei menu (in genere in posizione orizzontale nella parte superiore della finestra), poi, una volta comparso il menu verticale relativo a File, cliccate sulla voce Ricarica configurazione utente. Come informazione supplementare, sapete che per ottenere lo stesso risultato potete usare la combinazione di tasti Ctrl+R, come descritto in precedenza.

## Utenti generici del sistema

Tutte le volte che è stato possibile abbiamo usato due utenti generici nei nostri esempi:

| | | |
|---|---|---|
| Queen Amidala |  | Questo utente viene creato al momento dell'installazione |
| Darth Vader |  | Questo utente è creato successivamente dall'amministratore del sistema |

# Capitolo 1. Introduzione

Benvenuto, e grazie per aver scelto **Linux-Mandrake**! Questo libro è rivolto a quelle persone che desiderano tuffarsi nelle profondità del loro sistema *GNU/Linux*, sfruttando le sue enormi potenzialità

Questa prima parte vi mostra i meccanismi interni del sistema e come utilizzarli. Comincia con un capitolo che servirà da introduzione a *Unix* e, in particolare, alle caratteristiche di *GNU/Linux*. È necessario comprendere bene i concetti presentati in questo capitolo prima di affrontare quello successivo, dedicato alla linea di comando. Questo capitolo vi spiegherà l'uso dei programmi standard per la manipolazione dei file e alcune caratteristiche utili fornite dalla *shell*.

Un altro capitolo è dedicato alla gestione dei testi. Dato che la maggior parte dei file di configurazione *Unix* sono file di testo, è probabile che dovrete modificarli con un ***editor di testo***. Imparerete a usare i due editor di testo più famosi nel mondo *Unix*: il poderoso *Emacs* e il moderno **:-)** *VI*.

A questo punto dovreste essere in grado di compiere la manutenzione ordinaria del vostro sistema. I tre capitoli successivi vi presenteranno altrettante caratteristiche utili del vostro sistema **Linux-Mandrake**: come gestire le stampanti e stampare i documenti, la connessione con sistemi basati su *Windows* e, infine, il sistema di sicurezza di **Linux-Mandrake**: *MSEC*.

Subito dopo ci addentreremo in profondità nel sistema *GNU/Linux*. Vedrete come è organizzato l'albero dei file e come ogni file abbia il suo posto in una specifica directory, anche se i sistemi *Unix* tendono a crescere fino a raggiungere dimensioni considerevoli. Dopo aver letto questo capitolo, saprete dove cercare un file a seconda del suo ruolo all'interno del sistema.

Un'altra sezione riguarda il ***filesystem*** e i ***punti di mount***. Qui imparerete cosa significano questi termini e vedrete un caso pratico.

Un capitolo sarà dedicato al filesystem di *GNU/Linux*, ext2fs. In questo capitolo imparerete qualcosa in più sui tipi di file e su alcuni con-

cetti che potrebbero risultarvi nuovi. Il capitolo successivo vi introdurrà al filesystem speciale di *GNU/Linux*: `/proc`.

In seguito, imparerete a conoscere la procedura di avvio di **Linux-Mandrake** e come utilizzarla in modo efficace.

A questo punto incontriamo tre capitoli dedicati in particolar modo alle persone che desiderano diventare esperti di *GNU/Linux*: il primo spiega come compilare e installare un nuovo kernel.

Seguirà il capitolo dedicato alla compilazione e all'installazione del software liberamente distribuibile; infine vi sarà presentato (ancora) un altro set di programmi da linea di comando.

# Capitolo 2. Concetti base di Unix

Il nome "*Unix*" potrebbe essere già noto a qualcuno dei lettori. Probabilmente alcuni di voi utilizzano già un sistema *Unix*, nel qual caso questo capitolo non vi sarà di grande aiuto.

Per chi non lo ha mai utilizzato, invece, questa lettura è un passo obbligato: la conoscenza dei concetti che descriveremo qui fornisce la risposta a un numero sorprendentemente elevato di domande poste da chi si avvicina per la prima volta a GNU/Linux . Queste nozioni, inoltre, potranno esservi utili per risolvere molti dei problemi in cui potreste imbattervi in futuro.

## Utenti e gruppi

I concetti di utente e di gruppo sono estremamente importanti, poiché sono direttamente in relazione con tutti gli altri concetti che esporremo qui.

*GNU/Linux* è un vero sistema ***multiutente***, pertanto per poter utilizzare la vostra macchina *GNU/Linux* dovete disporre di un ***account*** su di essa. Creando un utente durante l'installazione avete già creato, di fatto, un account utente. Forse ricorderete che vi sono state chieste queste informazioni:

- il "vero nome" dell'utente (o qualsiasi altro nome scegliate);
- un nome di ***login***;
- una ***password*** (ne avete scelta una, vero? **:-)** ).

I dati veramente importanti qui sono il nome di login (spesso abbreviato semplicemente in login) e la password. Sono quelli che userete per accedere al sistema.

Un'altra operazione effettuata durante l'installazione del sistema è la creazione di un gruppo. Come opzione predefinita, il programma di installazione crea un gruppo per ciascun utente. Come vedremo, i gruppi

tornano utili quando si devono condividere file fra utenti diversi. Un gruppo può contenere tutti gli utenti che volete, ed è una caratteristica utilizzata spesso sui grossi sistemi per separare i diversi tipi di utenti. In una università, ad esempio, può esserci un gruppo per ciascuna facoltà, uno per i docenti, e così via. Ma è vero anche il contrario: un utente può far parte di diversi gruppi allo stesso tempo. Un professore, ad esempio, può essere membro del gruppo dei docenti, e, contemporaneamente, di quello dei suoi amati studenti.

Detto questo, non abbiamo ancora chiarito come ci si collega al sistema: vediamolo ora.

Se durante l'installazione avete scelto di avviare automaticamente l'interfaccia grafica, la vostra schermata di avvio sarà pressappoco così (Figura 2-1):

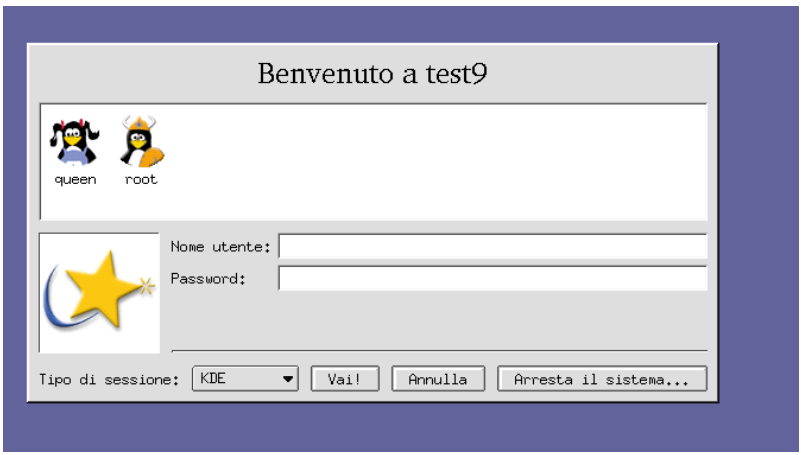


**Figura 2-1. Schermata di login in modalità grafica**

Per accedere, dovete digitare il vostro login nel campo Utente:, poi immettere la vostra parola d'accesso nel campo password. Notate che dovrete digitare la vostra password alla cieca: non vedrete nessun carattere di ***echo*** nel campo di testo relativo.

Se invece il vostro sistema si avvia in modalità console (solo testo), la schermata sarà simile a questa (Figura 2-2):


**Figura 2-2. Schermata di login in modalità console**

Qui dovrete digitare il vostro nome di login al prompt di Login: e premere il tasto `Invio`, dopodiché il programma di accesso (che si chiama, guarda caso, *`login`*) visualizzerà un prompt di richiesta della Password: per questo account. Dato che il programma di login della console non produce nessuna "echo" dei caratteri che compongono la password, fate attenzione nel digitare la password – anche qui alla cieca.

Notate che potete accedere al sistema più volte con lo stesso account, facendo uso di ulteriori ***console*** e di sessioni *`X`*. Ogni sessione che aprite è indipendente, ed è persino possibile avere più sessioni *`X`* contemporaneamente. Come opzione predefinita, **Linux-Mandrake** dispone di sei ***console virtuali*** oltre a quella riservata all'interfaccia grafica. Potete passare a una qualsiasi di queste digitando la combinazione `Ctrl-Alt-F<n>`, dove `<n>` è il numero della console alla quale volete accedere. Come regola generale, l'interfaccia grafica si trova sulla console numero 7.

Oltre alla creazione degli account utente, avrete notato che durante l'installazione *DrakX* (o il programma che avete utilizzato) vi ha richiesto la password di un utente molto particolare: `root`. Questo account è speciale per un motivo molto particolare: è l'utenza normalmente usata dall'amministratore del sistema (che molto probabilmente siete voi). Per la sicurezza del vostro sistema, è importante che l'account `root` sia protetto da una buona password.

Se vi collegate spesso come `root` è facile fare degli errori che possono rendere il sistema inutilizzabile: basta un solo errore perché ciò avvenga. In particolare, se non avete protetto l'account `root` con una password, qualsiasi utente potrà utilizzarlo e causare danni al sistema: una situazione davvero poco piacevole!

È bene notare che, internamente, il sistema non vi identifica con il vostro nome di login, ma con un numero univoco: l'*user ID* (meglio noto come UID). Allo stesso modo, a ogni gruppo corrisponde un *group ID* (GID) univoco, e non il suo nome.

## Principi di base riguardo i file

I file sono un altro campo in cui *GNU/Linux* è sensibilmente diverso rispetto a *Windows* e a molti altri ***sistemi operativi***. Qui vedremo le differenze più evidenti, per informazioni più approfondite consultate il capitolo Il filesystem di *GNU/Linux*: *ext2fs* (*EXTended 2 File-System*) nel **Manuale di riferimento**, che descrive questo argomento più in dettaglio.

La prima e più evidente differenza è legata alla presenza degli utenti. In un sistema Linux, ciascun utente ha una sua cartella (nota come la sua ***directory home***), ma questo non è tutto: su un sistema *Unix*, ogni file è di **proprietà esclusiva** di un utente e di un gruppo. Pertanto, non solo un utente ha la sua home directory, ma è anche il ***proprietario*** dei suoi file nel senso più stretto del termine.

A ogni file, inoltre, sono associati dei permessi che solo il proprietario può modificare. Questi permessi sono distinti in tre diverse categorie:

permessi del proprietario del file, di qualsiasi utente membro del gruppo associato al file (il cosiddetto ***owner group***) escluso l'utente che ne è il proprietario, e gli altri, ovvero qualsiasi utente che non rientri nei primi due casi.

Ci sono tre diversi permessi:

1. Permesso *Read* (`r`) "lettura". Per un file, questo permesso consente di leggerne il contenuto; per una directory, di elencare i file che essa contiene, se e solo se per la stessa directory è assegnato anche il permesso *execute* (esecuzione).
2. Permesso *Write* (`w`) "scrittura". Per un file, questo permesso consente di modificarne il contenuto. Per una directory, consente di modificare e cancellare i file contenuti in essa: questo è possibile anche se non si è il proprietario della directory, basta esserlo del file che si vuole modificare o cancellare;
3. Permesso *eXecute* (`x`) "esecuzione". Per un file (ad esempio per un programma o un file batch), questo permesso consente di lanciarlo in esecuzione; di conseguenza, solo i file eseguibili dovrebbero avere questo permesso. Per una directory, questo permesso consente di ***attraversare*** la stessa, cioè spostarsi dentro o attraverso di essa.

È possibile combinare a volontà questi permessi: come proprietari di un file, ad esempio, potete consentire solo a voi stessi di leggerlo, proibirlo a tutti gli altri utenti, e impedire ogni altro uso del file; potete anche fare l'opposto, anche se a prima vista non è molto logico **:-)** Se siete proprietari del file, potete anche cambiarne il gruppo di appartenenza, purché siate membri del nuovo gruppo, e addirittura privarvi della proprietà del file (ovvero indicare un nuovo proprietario). Naturalmente, se lo fate vi priverete anche di ogni diritto sul file...

Facciamo un esempio pratico con un file e una directory. La schermata qui sotto mostra i risultati del comando `ls -l` impartito da una ***linea***

***di comando***:

```
$ ls -l
total 1
-rw-r-----   1 queen    users           0 Jul  8 14:11 un_file
drwxr-xr--   2 darth    users        1024 Jul  8 14:11 una_directory/
$
```

Da sinistra a destra, il significato delle informazioni ottenute digitando `ls -l` è il seguente:

- i primi dieci caratteri rappresentano, in sequenza, il tipo di file e i permessi associati. Il primo carattere è il tipo di file; questo contiene un trattino (`-`) se è un file normale, o una `d` se è una directory. Ci sono anche altri tipi di file, che tratteremo nel **Manuale di riferimento**. I nove caratteri successivi rappresentano i permessi associati a quel file. Qui si notano le differenze relative alle diverse categorie di utenti: i primi tre caratteri rappresentano i diritti associati al proprietario del file, i tre caratteri successivi riguardano tutti i membri del gruppo (ma non l'utente proprietario del file), e gli ultimi tre valgono per tutti gli altri. Un trattino (`-`) indica che il permesso corrispondente non è assegnato;
- quindi viene il numero di link per il file. Nel **Manuale di riferimento** vedremo che per identificare un file non conta il suo nome, ma un numero (il ***numero di inode***); pertanto in *Unix* è possibile che uno stesso file sul disco abbia più di un nome. Per una directory, il numero di link ha un significato particolare; torneremo anche su questo argomento nel **Manuale di riferimento**;
- di seguito sono indicati il nome del proprietario e il nome del gruppo di appartenenza;
- infine, la dimensione del file (in ***byte***) e la data e ora della sua ultima modifica, seguite da una ripetizione del nome del file o della directory.

Diamo un'occhiata più da vicino ai permessi associati a ciascuno di questi file: prima di tutto, saltiamo il primo carattere relativo al tipo

di file; per il file `un_file` abbiamo i seguenti diritti: `rw-r------`. Questi caratteri vanno interpretati come segue:

- i primi tre (`rw-`) sono i diritti del proprietario del file, che in questo caso è `francis`. L'utente `francis`, quindi, ha il diritto di leggere il file (`r`) e modificarne il contenuto (`w`), ma non di lanciarlo in esecuzione (`-`);
- i tre caratteri successivi (`r--`) riguardano tutti gli utenti che non sono `queen`, ma sono membri del gruppo `users`: questi utenti sono in grado di leggere il file (`r`), ma non potranno né modificarlo, né lanciarlo in esecuzione (`--`);
- gli ultimi tre caratteri (`---`) riguardano qualsiasi utente che non è `queen` e non è membro del gruppo `users`: tutti questi utenti non hanno alcun diritto sul file.

Per la directory `una_directory`, i diritti sono `rwxr-xr--`, e pertanto:

- `gael`, come proprietario della directory, può elencare i file contenuti in essa (`r`), aggiungere o rimuovere file in essa (`w`), e può attraversarla (`x`);
- tutti gli utenti che non sono `gael`, ma sono membri del gruppo `users`, potranno elencare i file in questa directory (`r`), ma non potranno rimuovere né aggiungere file (`-`), e potranno attraversarla (`x`);
- tutti gli altri utenti non avranno diritti su questa directory. Come già sapete, il solo permesso di lettura su una directory non basta per consentire a un utente di elencarne il contenuto, poiché anche se questo permesso è presente, in questo caso il permesso di esecuzione (`r--`) non lo è.

Ricordate, c'è un'eccezione a questa regola: l'account `root` può modificare gli attributi (permessi, proprietario e gruppo di appartenenza) di tutti i file, anche se non ne è il proprietario. Questo significa che può

anche diventarne proprietario. `root` può leggere i file per cui non ha il permesso di lettura, attraversare le directory che normalmente non potrebbe attraversare, e così via. E se gli manca un permesso, non deve far altro che attribuirselo.

Infine, dobbiamo menzionare un'ultima particolarità relativa ai nomi dei file. Infatti questi sono molto meno limitati e molto più flessibili rispetto a quanto avviene sotto *Windows*:

- possono contenere qualsiasi carattere (eccettuato il carattere ***null*** e il carattere /), anche quelli non stampabili. La conseguenza principale è che Unix fa differenza tra maiuscole e minuscole, fatto cui bisogna prestare attenzione: i file `readme` e `Readme` sono diversi, perché `r` e `R` sono due **caratteri** del tutto diversi;
- come avrete notato, un nome di file non deve necessariamente avere un'estensione, a meno che voi non preferiate così. In *GNU/Linux*, a differenza di quello che avviene in altri sistemi, le estensioni non identificano il contenuto di un file.

## Processi

Un ***processo*** definisce un'istanza di un programma in esecuzione e il suo ***ambiente***. Come per i file, qui citeremo solo le differenze più importanti, potrete consultare il **Manuale di riferimento** per una trattazione più approfondita di questo argomento.

La differenza più importante è, ancora una volta, legata al concetto di utente: ogni processo viene eseguito con i diritti dell'utente che lo ha lanciato. Quindi, tornando all'esempio del file `un_file` che abbiamo appena visto, un processo lanciato dall'utente `darth` potrà aprire questo file in modalità ***read-only*** (sola lettura), ma non in modalità ***read-write*** (lettura e scrittura), perché i permessi associati al file lo impediscono. Ancora una volta, `root` fa eccezione a questa regola...

Come avete appreso più sopra, uno dei parametri di ambiente di un processo sono i valori di UID e GID dell'utente che lo ha lanciato. Questo permette al sistema di sapere se le richieste che il processo effettua sono "legali", ovvero consentite.

Una conseguenza di questa gestione dei processi è che *GNU/Linux* è praticamente immune ai virus. Per causare danni, i virus devono poter infettare dei file eseguibili. Come semplici utenti, normalmente voi non avete accesso ai file critici per il sistema, quindi il rischio è assai ridotto. A questo, aggiungete il fatto che i virus sono, in generale, poco diffusi nel mondo *Unix*. Fino a oggi sono stati individuati solo tre virus su *GNU/Linux*, e tutti erano praticamente innocui se attivati da un utente comune. C'è un solo utente che può danneggiare il sistema avviando questi virus, ed è - ancora una volta - `root`.

Esistono dei programmi anti-virus anche per *GNU/Linux*, ma servono solo per i file *DOS*/*Windows* ... Questo perché si trovano sempre più spesso file server *GNU/Linux* che lavorano per client *Windows* tramite il software *Samba* (consultate il capitolo *Samba* del **Manuale di riferimento**).

Internamente il sistema identifica i processi, ancora una volta, con un numero univoco detto *process ID*, o PID. Inoltre, ciascun processo può ricevere dei segnali tramite i quali potete controllarlo, ma questo vale solo per i processi che voi stessi avete lanciato, e non per quelli di un altro utente (l'unica eccezione a questa regola è costituita, di nuovo, da `root`). Potete fermare un processo, "ucciderlo" se sta causando dei problemi, e così via. In uno dei prossimi capitoli imparerete come tenere traccia di un PID e inviargli dei segnali. Questo è utile per terminare o sospendere i processi che presentano dei problemi.

## Breve introduzione alla linea di comando

La linea di comando è il modo più diretto di impartire dei comandi al sistema. Se utilizzate la linea di comando di *GNU/Linux* vi accorgerete presto che è molto più potente e flessibile di qualsiasi prompt dei

comandi che possiate aver utilizzato in passato. Il prompt dei comandi di *GNU/Linux* offre l'accesso a migliaia di programmi di utilità che non hanno un equivalente grafico. Tramite il prompt dei comandi, infatti, avrete l'accesso non solo a centinaia di applicazioni *X*, ma anche a migliaia di programmi che operano in modalità console (invece che in modalità grafica) e che non hanno un equivalente grafico, o le cui centinaia di comandi e di possibili combinazioni di funzioni non potranno mai essere visualizzate sotto forma di menu o pulsanti.

Ma, ammettiamolo, per iniziare ci vuole un po' di aiuto. Questo è lo scopo del presente capitolo. La prima cosa da fare, se utilizzate *KDE*, è lanciare un emulatore di terminale. C'è un'icona che lo indica chiaramente nel pannello (Figura 2-3).

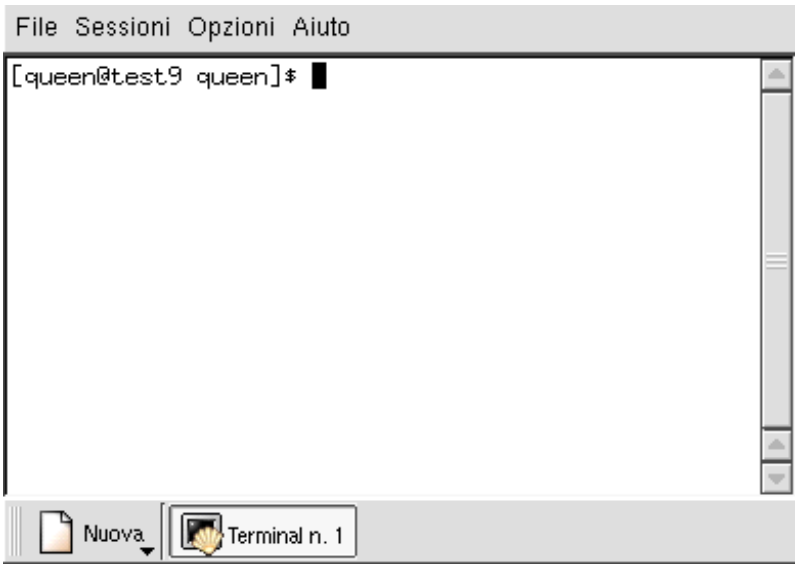


**Figura 2-3. L'icona del terminale nel pannello di** KDE

Ciò che vedete in questo emulatore di terminale quando lo lanciate è una *shell*. Questo è il nome del programma con cui interagite. Vi troverete davanti al ***prompt***:

```
[mario@localhost] ~ $
```

Supponiamo che il vostro nome utente sia `mario` e che il nome del vostro computer sia *localhost* (il che corrisponde alla realtà, se il vostro PC non fa parte di una rete). A questo punto, tocca a voi digitare uno o più comandi. Notate che quando siete `root`, il `$` del prompt diventa un `#` (tutto questo fa parte della configurazione standard del sistema, che può essere modificata a vostro piacimento). Il comando per "diventare" `root` dopo aver lanciato una shell come utente normale è `su`:

```
# Digitate la password di root; non comparirà sullo schermo
[mario@localhost] ~ $ su
Password:
# digitando "exit" tornate al vostro account normale
[root@localhost] mario # exit
[mario@localhost] ~ $
```

In tutto il resto del manuale, il prompt verrà rappresentato da un `$`, sia che siate `root`, sia che siate un utente comune. Vi diremo quando dovrete essere `root`, quindi ricordatevi di usare il comando `su` secondo quanto visto in precedenza. Un `#` all'inizio di una linea di codice rappresenta invece un commento.

Quando ***lanciate*** una *shell* per la prima volta, normalmente vi trovate nella vostra home directory. Per visualizzare la directory in cui vi trovate, digitate il comando `pwd` (che sta per *Print Working Directory*):

```
$ pwd
/home/mario
```

Adesso vedremo alcuni comandi di cui, come scoprirete, non si può proprio fare a meno.

## `cd`: Change Directory

Il comando `cd` è proprio come quello del *DOS*, solo con qualcosa in più. Esso fa tipicamente quello che dice il suo nome, cambia la di-

rectory di lavoro. Potete usare `.` e `..`, che indicano, rispettivamente, la directory attuale e la directory immediatamente superiore (*parent directory*). Digitando `cd` senza argomenti, ritornerete alla vostra home directory. Digitando `cd -` tornerete indietro all'ultima directory che avete attraversato. Infine, potete specificare la home directory di un utente, ad esempio `mario` digitando `~mario` (~ da solo o seguito da / indica la vostra home directory). Notate che, come utente normale, di solito non potete andare nelle home directory di altri utenti, a meno che questi non lo autorizzino o che questa non sia la regola generale sul vostro sistema; oppure, a meno che non siate `root`, quindi diventiamo `root` e facciamo un po' di pratica:

```
$ pwd
/root
$ cd /usr/doc/HOWTO
$ pwd
/usr/doc/HOWTO
$ cd ../FAQ
$ pwd
/usr/doc/FAQ
$ cd ../../lib
$ pwd
/usr/lib
$ cd ~mario
$ pwd
/home/mario
$ cd
$ pwd
/root
```

Adesso ritorniamo allo status di utente normale.

## Alcune variabili di ambiente e il comando `echo`

I processi hanno delle loro ***variabili di ambiente*** e la *shell* vi consente di visualizzarle direttamente con il comando `echo`. Ecco alcune variabili interessanti:

1. `HOME` Questa variabile contiene una stringa che rappresenta il percorso della vostra home directory.

2. `PATH` Questa variabile contiene la lista di tutti i percorsi in cui la shell cerca gli eseguibili quando digitate un comando. A differenza del *DOS*, come opzione predefinita una *shell* non cercherà i programmi nella directory corrente!
3. `USERNAME` Questa variabile contiene il vostro nome di login.
4. `UID` Contiene il vostro user ID.
5. `PS1` Contiene il valore del vostro prompt. Spesso è una combinazione di sequenze speciali, per avere maggiori informazioni potete leggere la ***pagina di manuale*** `man 1 bash`.

Affinché la *shell* visualizzi il contenuto di una variabile, dovete aggiungere un `$` prima del suo nome. Ecco come usare il comando `echo`:

```
$ echo Ciao
Ciao
$ echo $HOME
/home/mario
$ echo $USERNAME
mario
$ echo Ciao $USERNAME
Ciao mario
$ cd /usr
$ pwd
/usr
$ cd $HOME
$ pwd
/home/mario
```

Come potete vedere, la shell sostituisce il valore della variabile prima di eseguire il comando. Altrimenti, il nostro `cd $HOME` non avrebbe funzionato. Infatti la shell ha prima di tutto sostituito `$HOME` con il suo valore, `/home/mario`, quindi la linea è diventata `cd /home/mario`, ovvero il comando che volevamo. Lo stesso vale per `echo $USERNAME`, e così via.

## `cat`: visualizza il contenuto di uno o più file sullo schermo

Non c'è molto da aggiungere: questo comando visualizza semplicemente sullo schermo il contenuto di uno o più file:

```
$ cat /etc/fstab
/dev/hda5 / ext2 defaults 1 1
/dev/hda6 /home ext2 defaults 1 2
/dev/hda7 swap swap defaults 0 0
/dev/hda8 /usr ext2 defaults 1 2
/dev/fd0 /mnt/floppy auto sync,user,noauto,nosuid,nodev 0 0
none /proc proc defaults 0 0
none /dev/pts devpts mode=0620 0 0
/dev/cdrom /mnt/cdrom auto user,noauto,nosuid,exec,nodev,ro 0 0
$ cd /etc
$ cat conf.modules shells
alias parport_lowlevel parport_pc
pre-install plip modprobe parport_pc ; echo 7 > /proc/parport/0/irq
#pre-install pcmcia_core /etc/rc.d/init.d/pcmcia start
#alias car-major-14 sound
alias sound esssolo1
keep
/bin/zsh
/bin/bash
/bin/sh
/bin/tcsh
/bin/csh
/bin/ash
/bin/bsh
/usr/bin/zsh
```

## `less`: un visualizzatore a pagine

Il nome di questo comando è un gioco di parole tratto dal nome del primo programma di questo tipo per *Unix*, che si chiamava `more`. Un ***pager*** è un programma che permette di visualizzare lunghi file una pagina alla volta (o meglio, uno schermo alla volta). Parliamo di `less` piuttosto che di `more` perché è molto più intuitivo nell'uso. Usate `less` per visualizzare file lunghi, che non stanno in una sola schermata. Per esempio:

```
less /usr/doc/HOWTO/PCMCIA-HOWTO
```

Per scorrere il file, usate le frecce sù/giù. Premete **q** per uscire. `less`, tuttavia, può fare molto più di questo: digitate `h` per l'aiuto, e guar-

date. Lo scopo di questo paragrafo, in ogni caso, era solo quello di permettervi di leggere file lunghi, e l'abbiamo raggiunto **:-)**

## `ls`: elencare file (`LiSt`)

Questo comando è l'equivalente di `dir` del *DOS*, ma può fare molto di più. In verità, questo è ampiamente dovuto al fatto che i file possono fare molto di più **:-)** La sintassi di `ls` è come segue:

```
ls [opzioni] [file|directory] [file|directory...]
```

Se non si specifica un file o una directory, `ls` visualizza la lista dei file nella directory corrente. Le opzioni di `ls` sono moltissime, qui ne vediamo solo alcune:

1. `-a` elenca tutti i file, inclusi i ***file nascosti*** (in *Unix* i file nascosti sono quelli il cui nome comincia con `.`); l'opzione `-A` elenca "quasi" tutti i file, il che significa: tutti i file che l'opzione `-a` visualizzerebbe tranne "." e "..";
2. `-R` elenca in modo ricorsivo, ovvero elenca tutti i file e le sotto-directory della directory indicata sulla linea di comando;
3. `-s` visualizza la dimensione in kilobyte di seguito a ciascun file;
4. `-l` visualizza informazioni aggiuntive sui file;
5. `-i` visualizza il numero di inode (il numero identificativo di un file nel filesystem, consultate in proposito il capitolo Il filesystem di *GNU/Linux*: *ext2fs* (*EXTended 2 FileSystem*)) accanto a ciascun file;
6. `-d` visualizza le directory come file normali invece di visualizzare il loro contenuto.

Alcuni esempi:

1. `ls -R` visualizza il contenuto della directory attuale in modo ricorsivo;
2. `ls -is images/ ..` elenca i file nella directory `images/` e nella directory superiore, e visualizza per ciascun file il numero di inode e la dimensione in kilobyte;
3. `ls -al images/*.gif` elenca tutti i file (compresi quelli nascosti) nella directory `images/` i cui nomi terminano in `.gif`. Notate che questo include anche il file `.gif` se ne esiste uno.

## Scorciatoie utili per la tastiera

Ci sono molte utili combinazioni di tasti che vi possono far risparmiare parecchio lavoro, e in questo paragrafo vedremo alcune tra le più comuni. Questa sezione presuppone che stiate utilizzando la *shell* predefinita fornita con **Linux-Mandrake**, *Bash*, ma queste scorciatoie dovrebbero funzionare anche in altre *shell*. In questa sezione, `C-<x>` significa `Ctrl+<x>` (tenere premuto il tasto `Ctrl`, premere il tasto `<x>`, rilasciare entrambi i tasti).

Per prima cosa, i tasti cursore. *Bash* mantiene una storia dei comandi che avete impartito, e potete esplorarla usando i tasti cursore (le frecce di direzione) sù e giù. Potete risalire per un numero di righe definito dalla variabile di ambiente `HISTSIZE`. Questo registro, inoltre, è persistente da una sessione all'altra, quindi rimane disponibile anche dopo aver chiuso e riaperto una sessione.

I tasti destra e sinistra spostano il cursore a destra e a sinistra nella linea di comando attuale, quindi potete modificare i vostri comandi in questo modo. Ma potete anche fare altre cose: `C-a` e `C-e` vi porteranno, rispettivamente, all'inizio e alla file della linea attuale. I tasti `Backspace` e `Canc` funzioneranno come potete immaginare, un equivalente di `Backspace` è `C-h` e un equivalente di `Canc` è `C-d`. `C-k` cancellerà tutti i caratteri dalla posizione del cursore alla fine della linea, e `C-w` cancellerà la parola prima del cursore.

Digitando `C-d` su una linea vuota chiuderete la sessione attuale, il che è una scorciatoia per il comando `exit`. `C-c` interromperà il comando attualmente in corso di esecuzione, a meno che non stiate modificando la riga di comando: in tal caso, questa combinazione di tasti cancellerà la modifica e vi riporterà al prompt. `C-l` pulisce lo schermo.

Infine, `C-s` e `C-q`: questi due comandi servono rispettivamente a sospendere e a riattivare il flusso di caratteri su un terminale. Vengono usati molto raramente, ma potrebbe accadere che digitiate `C-s` per errore. Per cui, se premete dei tasti ma non vedete apparire niente sullo schermo, provate prima `C-q` e attenzione: tutti i caratteri che avete digitato tra `C-s` e `C-q` verranno scritti sullo schermo tutti in una volta.

# Capitolo 3. Introduzione alla linea di comando

Nel capitolo *Concetti base di Unix*, pag. 25 vi abbiamo spiegato come aprire una *shell*. In questo capitolo vi mostreremo come utilizzarla.

La principale dote della shell è il numero di programmi di utilità di cui dispone: ce ne sono migliaia, e ciascuno può svolgere un compito particolare. Qui ne vedremo soltanto alcuni. Una delle migliori caratteristiche di *Unix* è la sua capacità di combinare questi programmi, come vedremo.

## Comandi per la gestione dei file

In questo caso, la gestione dei file consiste nel copiarli, spostarli e rinominarli. Più avanti vedremo anche come fare per modificarne gli attributi (proprietario, permessi).

### `mkdir`, `touch`: creazione di directory e file vuoti (`MaKe DIRectory`)

`mkdir` viene utilizzato per creare le directory. La sua sintassi è semplice:

```
mkdir [opzioni] <directory> [directory ...]
```

Solo un'opzione merita di essere commentata: l'opzione `-p`. Se la si utilizza, `mkdir` creerà le directory superiori se in precedenza non esistevano. Se non la si specifica, e le directory superiori non esistono, `mkdir` visualizzerà un errore. Esempi:

1. `mkdir foo` crea la directory `foo` nella directory corrente;
2. `mkdir -p images/misc docs` crea la directory `misc` nella directory `images` creando prima quest'ultima, se non esiste già, insieme alla directory `docs`.

Il comando `touch` non è nato con la funzione di creare dei file, ma quella di aggiornare le date di accesso e di modifica dei file[1]. Se il file prima non esisteva, tuttavia, `touch` provvederà a creare i file desiderati come file di dimensione 0. La sintassi è:

```
touch [opzioni] file [file...]
```

Quindi lanciando il comando

```
touch file1 images/file2
```

verranno creati un file di dimensione pari a 0 chiamato `file1` nella directory corrente, e un file di dimensione 0 `file2` nella directory `images`.

## `rm`: cancellazione di file o directory (`ReMove`)

Questo comando è l'equivalente dei comandi *DOS* `del` e `deltree`, rispetto ai quali dispone di più opzioni. Ecco la sua sintassi:

```
rm [opzioni] <file|directory> [file|directory...]
```

Le sue opzioni includono:

1. `-r`, o `-R` Cancella ricorsivamente. Questa opzione è **necessaria** per cancellare una directory, vuota o meno. Tuttavia, si può usare anche il comando `rmdir` per cancellare directory vuote.

1. In *Unix* vi sono tre diverse date registrate assieme ai file: la data dell'ultimo accesso al file (`atime`), vale a dire l'ultima volta che il file è stato aperto in lettura o scrittura; la data dell'ultima volta che sono stati modificati gli attributi dell'inode (`mtime`) e, infine, la data dell'ultima volta che il contenuto del file è stato modificato (`ctime`).

2. `-i` Richiede una conferma prima di ogni cancellazione. Si raccomanda di creare un ***alias*** per la parola `rm` che punti al comando `rm -i` nella vostra shell; lo stesso vale per i comandi `cp` e `mv`.
3. `-f` È l'opposto dell'opzione `-i`: forza la cancellazione di file e directory, anche se l'utente non ha i diritti di scrittura sui file[2].

Alcuni esempi:

1. `rm -i images/*.jpg file1` Cancella tutti i file il cui nome termina con `.jpg` nella directory `images` e il file `file1` nella directory corrente, richiedendo una conferma per ciascun file. Rispondete **y** per confermare la cancellazione, **n** per annullarla.
2. `rm -Rf images/misc/ file*` Cancella senza chiedere conferma l'intera directory `misc/` nella directory `images/` insieme a tutti i file che si trovano nella directory corrente e il cui nome comincia con `file`.

> **Attenzione**
>
> Un file cancellato usando `rm` è perduto **in maniera definitiva**. Non c'è alcun modo per recuperare il file! Non esitate a utilizzare l'opzione `-i` in modo da esser sicuri di non cancellare nulla per errore...

## `mv`: spostare o rinominare dei file (`MoVe`)

La sintassi del comando `mv` è la seguente:

```
mv [opzioni] <file|directory> [file|directory ...] <destinazione>
```

2. È sufficiente per un utente possedere i diritti di scrittura su una directory per poter cancellare i file che contiene, anche se non è il proprietario di tali file.

Alcune opzioni:

1. `-f` Forza lo spostamento dei file – non avvisa se qualche file preesistente viene sovrascritto dall'operazione.
2. `-i` L'opposto – richiede all'utente una conferma prima di sovrascrivere un file esistente.
3. `-v` Modalità ***prolissa*** (*`Verbose`*), riporta tutte le modifiche effettuate.

Alcuni esempi:

1. `mv -i /tmp/pics/*.gif .` Sposta tutti i file nella directory `/tmp/pics/` il cui nome finisce con `.gif` nella directory corrente (`.`), richiedendo una conferma prima di sovrascrivere qualsiasi file.
2. `mv foo bar` Rinomina il file `foo` come `bar`.
3. `mv -vf file* images/ trash/` Sposta dalla directory corrente, senza chiedere conferma, tutti i file il cui nome comincia per `file` e l'intera directory `images/` nella directory `trash/`, e mostra tutte le operazioni effettuate.

## `cp`: copiare file e directory (`CoPy`)

`cp` è l'equivalente dei comandi *`DOS`* `copy`, `xcopy`, rispetto ai quali dispone di più opzioni. Questa è la sua sintassi:

```
cp [opzioni] <file|directory> [file|directory ...] <destinazione>
```

`cp` ha moltissime opzioni. Queste sono le più comuni:

1. `-R` Copia ricorsiva; **obbligatorio** se si vuole copiare una directory, anche se vuota.
2. `-i` Richiede una conferma prima di sovrascrivere qualsiasi file preesistente.
3. `-f` È l'opposto dell'opzione `-i`, sostituisce qualsiasi file esistente senza chiedere conferma.
4. `-v` Modalità prolissa, visualizza tutte le azioni compiute dal comando `cp`.

Alcuni esempi:

1. `cp -i /tmp/images/* images/` Copia tutti i file dalla directory `/tmp/images` nella directory `images/` che si trova all'interno della directory attuale, richiedendo una conferma se un file deve essere sovrascritto.
2. `cp -vR docs/ /shared/mp3s/* mystuff/` Copia l'intera directory `docs` nella directory corrente e tutti i file della directory `/shared/mp3s` alla directory `mystuff` situata nella directory corrente.
3. `cp foo bar` Crea una copia del file `foo` con il nome `bar` nella directory corrente.

## Gestione degli attributi dei file

I comandi elencati qui sotto servono a modificare il proprietario o il gruppo proprietario di un file, o i suoi permessi. Abbiamo visto in cosa consistono i permessi nel capitolo `Concetti base di` *`Unix`* del **Manuale utente**.

## `chown`, `chgrp`: cambia il proprietario o il gruppo di uno o più file

La sintassi del comando `chown` è la seguente:

```
chown [opzioni] <utente[.gruppo]> <file|directory> [file|directory...]
```

Le opzioni includono:

1. `-R` Modalità ricorsiva: modifica il proprietario di tutti i file e le directory in una directory.
2. `-v` Modalità prolissa: descrive tutte le operazioni compiute dal comando `chown`; riporta quali file hanno cambiato proprietario e quali sono rimasti invariati.
3. `-c` Simile all'opzione `-v`, ma riporta solo quali file sono cambiati.

Alcuni esempi:

1. `chown nobody /shared/book.tex` cambia il proprietario del file `/shared/book.tex` in `nobody`.
2. `chown -Rc darth.music *.mid concerts/` attribuisce la proprietà di tutti i file nella directory corrente il cui nome termina con `.mid` e di tutti i file e le sotto-directory nella directory `concerts/` all'utente `darth` e al gruppo `music`, elencando solo i file modificati dal comando.

Il comando `chgrp` vi consente di cambiare il gruppo proprietario di uno o più file; la sua sintassi è molto simile a quella di `chown`:

```
chgrp [opzioni] <gruppo> <file|directory> [file|directory...]
```

Le opzioni di questo comando sono le stesse di `chown`, e anche il suo utilizzo è molto simile. Pertanto, il comando:

```
chgrp disk /dev/hd*
```

attribuisce al gruppo `disk` tutti i file nella directory `/dev/` il cui nome comincia con `hd`.

## `chmod`: modifica dei permessi di file e directory (`CHange MODe`)

Il comando `chmod` ha una sintassi molto particolare. La sintassi generica è:

```
chmod [opzioni] <cambio modo> <file|directory> [file|directory...]
```

ma ciò che lo distingue sono i diversi modi in cui la modifica dei permessi può essere specificata. In particolare, si possono utilizzare due formati:

- in ottale: in questo caso i permessi dell'utente proprietario sono indicati nella forma `<x>00`, dove `<x>` corrisponde al permesso assegnato: 4 per il permesso di lettura, 2 per quello di scrittura e 1 per quello di esecuzione; allo stesso modo i permessi del gruppo proprietario vengono espressi come `<x>0` e i permessi per gli altri utenti ("others") nella forma `<x>`. Pertanto tutto quello che dovete fare è sommare insieme i permessi per ottenere la cifra corretta. Quindi i permessi `rwxr-xr--` corrispondono a 400+200+100 (diritti del proprietario, `rwx`) +40+10 (diritti del gruppo, `r-x`) +4 (diritti degli altri, `r--`) = 754; se utilizzate `chmod` in questo modo, i permessi vengono espressi in forma assoluta e i permessi assegnati in precedenza vengono sovrascritti;

- con delle espressioni: in questo caso i permessi vengono indicati da una sequenza di espressioni separate da virgole, da specificare nel formato `[categoria]<+|-><permessi>`. La categoria può essere indicata da una o più lettere `u` (*User*, permessi del proprietario), `g` (*Group*, permessi per il gruppo proprietario) o `o` (*Others*, permessi per "gli altri"). Se non viene specificata una categoria, il cambiamento si applica a tutte. Un segno `+` assegna un permesso, un segno `-` lo rimuove. Per finire, i permessi vanno indicati da una o più lettere `r` (*Read*), `w` (*Write*) o `x` (*eXecute*).

Le opzioni principali sono molto simili a quelle dei comandi `chown` o `chgrp`:

1. `-R` Cambia i permessi in modo ricorsivo.
2. `-v` Modalità prolissa, riporta le operazioni effettuate per ciascun file.
3. `-c` Simile all'opzione `-v`, ma mostra solo i file per cui sono stati modificati i permessi.

Esempi:

1. `chmod -R o-w /shared/docs` Rimuove ricorsivamente il permesso di scrittura per gli "altri" su tutti i file e le sotto-directory della directory `/shared/docs/`.
2. `chmod -R og-w,o-x private/` Rimuove ricorsivamente il permesso di scrittura per il gruppo e gli "altri" sull'intera directory `private/`, e rimuove il permesso di esecuzione per gli altri.
3. `chmod -c 644 miscellaneous/file*` modifica i permessi per tutti i file nella directory `miscellaneous/` il cui nome comincia per `file`, impostandoli come `rw-r--r--` (ovvero permesso di lettura per tutti e permesso di scrittura solo per il proprietario) visualizzando solo i file i cui permessi sono stati effettivamente modificati.

## I caratteri speciali (meta-caratteri) e le *espressioni regolari* nella `shell`

Probabilmente usate già i caratteri speciali (noti anche come ***meta-caratteri***, o anche caratteri jolly) senza saperlo. Quando scrivete o cercate un file in *Windows* potete utilizzare il carattere `*` al posto di una sequenza di caratteri qualsiasi. Per esempio, scrivendo `*.txt` identificate tutti i file il cui nome finisce per `.txt`. Lo abbiamo anche utilizzato spesso nell'ultimo paragrafo. Ma `*` non è l'unico carattere speciale di cui potete servirvi.

Quando digitate un comando come `ls *.txt` e premete `Invio`, il compito di individuare i file che corrispondono all'espressione `*.txt` non viene svolto dal comando `ls`, ma dalla shell stessa. Vediamo più in dettaglio in che modo la shell interpreta una linea di comando quando questa viene digitata. Se scrivete:

```
$ ls *.txt
readme.txt  ricette.txt
```

prima di tutto la linea di comando viene suddivisa in parole (`ls` e `*.txt` in questo esempio). Quando la shell trova un carattere `*` in una parola, interpreterà la parola intera come un ***modello di meta-espansione*** e la sostituirà con i nomi di tutti i file che corrispondono al modello. Pertanto la linea di comando che viene effettivamente eseguita sarà `ls readme.txt recipe.txt`, che darà i risultati che abbiamo mostrato. Ci sono altri caratteri che vengono interpretati in modo simile dalla shell:

1. `?` sostituisce un carattere qualsiasi, ma soltanto uno;
2. `[...]` stabilisce una corrispondenza con qualsiasi carattere si trovi fra le parentesi quadre; in questo caso è possibile definire un am-

bito di caratteri (ad es. `1-9`) o dei valori ***discreti***, o addirittura entrambi. Ad esempio: `[a-zBE5-7]` comprende tutti i caratteri da `a` a `z`, una `B`, una `E`, un `5`, un `6` o un `7`;

3. `[!...]` stabilisce una corrispondenza con qualsiasi carattere che **non** sia indicato tra le parentesi quadre. `[!a-z]`, ad esempio, verrà sostituito da qualsiasi carattere che non sia una lettera minuscola;
4. `{c1,c2}` viene sostituito da `c1` o `c2`, dove `c1` e `c2` sono a loro volta delle espressioni jolly.

Ecco alcuni esempi con il relativo significato:

1. `/etc/*conf` Tutti i file nella directory `/etc` il cui nome finisce con `conf`. Può rappresentare `/etc/inetd.conf`, ma vale anche per `/etc/conf.linuxconf`, come pure per `/etc/conf` se esiste un file con questo nome. Ricordate che `*` può rappresentare anche una stringa vuota.
2. `image/cars,space[0-9]`/*.jpg Tutti i file il cui nome termina con `.jpg` nelle directory `image/cars`, `image/space0`, ... , `image/space9`, se queste directory esistono.
3. `/usr/doc/*/README` Tutti i file di nome `README` in tutte le directory immediatamente sotto `/usr/doc`. Questa espressione vale per `/usr/doc/mandrake/README`, ad esempio, ma non per `/usr/doc/myprog/doc/README`.
4. `*[!a-z]` Tutti i file nella directory attuale il cui nome non termina con una lettera minuscola.

## La redirezione e le pipe

### Ancora a proposito dei processi

Per comprendere i concetti di redirezione e di pipe, è necessario spiegare alcune nozioni sui processi ancora non discusse. In *Unix* ciascun processo (incluse le applicazioni grafiche) apre un minimo di tre diversi descrittori del file: lo standard input, lo standard output e lo standard error. I numeri assegnati a questi tre sono rispettivamente 0, 1 e 2. Generalmente i tre descrittori vengono associati al terminale dal quale è stato lanciato il processo, e, in particolare, lo standard input è associato alla tastiera. Lo scopo della redirezione e delle pipe è, appunto, di redirigere questi tre canali. Gli esempi che seguono ve lo mostreranno chiaramente.

## La redirezione

Immaginate di voler visualizzare una lista di tutti i file che terminano con `.gif`[3] nella directory `images`. Visto che l'elenco è molto lungo, volete salvarlo in un file per consultarlo con comodo in seguito. Per farlo potete usare questo comando:

```
$ ls images/*.gif 1>file_list
```

Questo significa che lo standard output di questo comando (`1`) viene rediretto (`>`) verso il file chiamato `file_list`. Il carattere `>` è l'operatore di redirezione dell'output. Se il file specificato nella redirezione non esiste, viene creato al momento, mentre se esiste il suo contenuto viene sovrascritto. Se in un comando di redirezione che usa questo operatore non viene specificato un descrittore, la shell assume che vo-

3. Potreste pensare che sia strano parlare di "file il cui nome termina per `.gif`" piuttosto che "immagini GIF". Tuttavia, ricordiamolo ancora, i file in un sistema *Unix* non sono necessariamente identificati dalla loro estensione: in questo ambiente le estensioni non definiscono il tipo di file. Un file il cui nome termina per `.gif` potrebbe tranquillamente essere un'immagine JPEG, un programma, un file di testo o di qualsiasi altro tipo.

gliate utilizzare lo standard output; volendo quindi potreste scrivere, più semplicemente:

```
$ ls images/*.gif >file_list
```

e otterreste il medesimo risultato. Poi potreste leggere il contenuto del file con un comando come `less`.

Ora immaginate di voler contare quanti file di quel tipo ci sono. Piuttosto che contarli a occhio, potreste utilizzare il comando `wc` (*Word Count*) con l'opzione `-l`, che scrive sullo standard output il numero di parole contenute in un file. Una soluzione potrebbe quindi essere:

```
wc -l 0<file_list
```

e in questo modo raggiungereste lo scopo. Il carattere `<` è l'operatore di redirezione dell'input, e il descrittore su cui questo agisce per opzione predefinita è lo standard input, ovvero lo `0`; quindi potreste digitare anche

```
wc -l <file_list
```

Ora immaginiamo che vogliate prendere questo file, rimuovere tutte le "estensioni" e scrivere i risultati dell'operazione in un altro file. Uno strumento adeguato per questo scopo è `sed`, ovvero *Stream EDitor*. Vi basterà redirigere lo standard input di `sed` verso il file `file_list` e redirigere invece l'output verso il file `the_list`:

```
sed -e 's/\.gif$//g' <file_list >the_list
```

questo creerà con un'unica operazione una lista pronta per essere consultata con un qualsiasi programma adatto allo scopo.

Spesso può essere utile anche redirigere gli standard error, ovvero il canale in cui i processi riversano i loro messaggi di errore. Per esempio, se volete scoprire a quali directory contenute in `/shared` non potete accedere, potreste elencare ricorsivamente queste directory e redirigere gli errori in un file, senza visualizzare lo standard output:

```
ls -R /shared >/dev/null 2>errors
```

che significa che lo standard output verrà rediretto (`>`) verso `/dev/null`, un file che ha la caratteristica di eliminare qualsiasi cosa vi si scriva (in questo caso, redirigere lo standard output verso null ne impedisce la visualizzazione); allo stesso tempo il canale standard error (2) viene rediretto (`>`) verso il file `errors`.

## Le pipe

Le pipe sono una sorta di combinazione di redirezione dell'input e dell'output. Il principio è quello di una tubazione, da cui il nome (che significa appunto 'tubo, condotto'). L'operatore di pipe è `|`. Riprendiamo l'esempio della lista di file, supponiamo che vogliate trovare direttamente quanti sono i file corrispondenti, senza immagazzinare il risultato in un file temporaneo: per farlo potreste usare il comando

```
ls images/*.gif | wc -l
```

che significa che lo standard output del comando `ls` (ovvero l'elenco dei file) viene rediretto verso lo standard input del comando `wc`. Questo produce il risultato che cercate.

Potete anche creare una lista dei file "senza estensioni" usando questo comando:

```
ls images/*.gif | sed -e 's/\.gif$//g' >the_list
```

o, se volete consultare la lista direttamente, senza salvarla in un file:

```
ls images/*.gif | sed -e 's/\.gif$//g' | less
```

L'uso delle pipe e della redirezione non è limitato ai file di testo, leggibili dall'uomo. Ad esempio, lanciando questo comando da una console di *xterm*:

```
xwd -root | convert - ~/my_desktop.gif
```

viene creata un'istantanea del vostro desktop che viene salvata nel file `my_desktop.gif`[4] nella vostra directory personale.

## Completamento automatico

Il ***completamento automatico*** è una funzione utilissima che tutte le shell moderne (inclusa la *Bash*) possiedono. Il suo ruolo è di fare in modo che gli utenti debbano scrivere il meno possibile. Il modo migliore per illustrare la funzione di completamento automatico è fare un esempio.

### Esempio

Immaginate che la vostra directory personale contenga un file il cui nome è `file_dal_nome_troppo_lungo_per_digitarlo`, e che vogliate leggerlo. Supponiamo che nella vostra directory ci sia anche un file chiamato `file_text`. Vi trovate nella vostra directory personale, e digitate questo comando:

```
$ less fi<TAB>
```

(ovvero digitate `less fi` e poi premete il tasto `TAB`). La shell a questo punto completerà la linea di comando per voi:)

```
$ less file_
```

e vi darà anche l'elenco delle scelte possibili (nella sua configurazione predefinita, che potreste modificare). Quindi digitate questa sequenza di caratteri:

```
less file_w<TAB>
```

e la shell completerà la vostra linea di comando in questo modo:

4. E in questo caso - avete indovinato - sarà veramente un'immagine GIF **:-)**

```
less file_dal_nome_troppo_lungo_per_digitarlo
```

A questo punto vi basterà premere `Invio` per confermare e leggere il file.

## Altri metodi di completamento

Il tasto `TAB` non è il solo modo di attivare il completamento, anche se è il più utilizzato. In linea di massima, la parola da completare sarà il nome di un comando se è la prima parola della linea di comando (`nsl<TAB>` verrà espanso in `nslookup`), e un nome di file per le altre parole, a meno che la parola non sia preceduta da un carattere "magico" da scegliere tra `~`, `@` o `$`; in questo caso, la shell cercherà di completare la parola assumendo che sia, rispettivamente, il nome di un utente, il nome di una macchina o di una variabile di ambiente [5]. Esiste anche un carattere magico per completare il nome di un comando (`!`) o il nome di un file (`/`).

Gli altri due modi per attivare il completamento sono le sequenze `Esc-<x>` e `C-x <x>` (`Esc` è il tasto `Escape`, e `C-x` significa `Control+<x>`), dove `<x>` è uno dei caratteri magici che abbiamo visto. `Esc-<x>` cercherà di eseguire il completamento in modo univoco, altrimenti completerà la parola con la sequenza di caratteri più lunga tra quelle possibili. Un ***beep*** indicherà che la scelta non è univoca, o semplicemente che non esiste una scelta corrispondente. La sequenza `C-x <x>` visualizza la lista delle possibili scelte senza tentare il completamento. Premere il tasto `TAB`, quindi, ha lo stesso effetto che premere in successione `Esc-<x>` e `C-x <x>`, dove il carattere magico dipende dal contesto.

Come conseguenza, un sistema per visualizzare tutte le variabili di ambiente definite è digitare `C-x $` su una linea vuota. Un altro esempio:

5. Ricordate: *Unix* fa differenza fra caratteri maiuscoli e minuscoli. La variabile di ambiente `HOME` e `home` non sono la stessa variabile.

se volete visualizzare la pagina di manuale del comando `nslookup`, potete digitare `man nsl` seguito da `Esc-!`, e la shell completerà la linea di comando in `man nslookup`.

## Avviare e gestire i processi in background: il controllo dei job

Avrete già notato che quando lanciate un comando da un terminale normalmente dovete attendere che il comando abbia terminato l'esecuzione prima di riavere il controllo della shell: questo avviene quando eseguite i comandi in ***foreground*** (ovvero in primo piano). Ci sono però delle occasioni in cui questo non è desiderabile.

Immaginate ad esempio di voler copiare ricorsivamente il contenuto di una grossa directory in un'altra. Volete anche ignorare gli eventuali errori, quindi redirigete l'output del canale di errore verso `/dev/null`:

```
cp -R images/ /shared/ 2>/dev/null
```

Questo comando potrebbe richiedere anche diversi minuti. A questo punto avete due possibilità: la prima, più distruttiva, consiste nell'arrestare l'esecuzione (kill) e riavviare il comando quando avrete il tempo di aspettare; per farlo dovreste digitare `C-c` (`Control+c`): questo comando vi riporta al prompt.

Ma supponiamo che vogliate che il comando venga eseguito mentre voi fate qualcos'altro. La soluzione consiste nello spostare il processo in ***background***. Per farlo, digitate `C-z` per sospendere il processo:

```
$ cp -R images/ /shared/ 2>/dev/null
  # Adesso premete C-z
[1]+  Stopped                 cp -R images/ /shared/ 2>/dev/null
$
```

e vi ritroverete di nuovo al prompt. A questo punto il processo è in standby, in attesa che voi lo riattiviate (come risulta dalla dicitura `Stopped`). Ed è ciò che vorrete fare, ma in modo tale che il proces-

so riparta in background. Digitate `bg` (che sta per *BackGround*) per ottenere questo risultato:

```
$ bg
[1]+ cp -R images/ /shared/ 2>/dev/null &
$
```

In questo modo il processo riprenderà l'esecuzione, ma questa volta in background, il che viene indicato dal simbolo `&` (e commerciale) alla fine della linea di comando. Subito dopo ritornerete al prompt e potrete continuare a lavorare. Un processo che viene eseguito in background viene indicato con il termine ***job***.

Ovviamente potete anche lanciare i processi in modo tale che vengano eseguiti in background, aggiungendo una **&** alla fine della linea di comando. Ad esempio, potete far eseguire fin dall'inizio la procedura di copia della directory in background digitando:

```
cp -R images/ /shared/ 2>/dev/null &
```

Se lo desiderate, potete anche riportare il processo in primo piano e attendere che venga completato usando il comando `fg` (*ForeGround*). Per riportarlo in background, usate la sequenza `C-z`, `bg`.

Potete anche lanciare diversi job con questo sistema; in questo caso a ciascun job verrà assegnato un numero consecutivo. Il comando `jobs` elenca tutti i job associati alla shell corrente. Il job contrassegnato da un carattere `+` è l'ultimo processo che è stato avviato in background. Potete riportare un qualsiasi processo in primo piano digitando il comando `fg <n>` dove `<n>` è il numero del job, ad esempio `fg 5`.

Notate che con questi comandi potete anche avviare o fermare applicazioni ***full-screen*** (ovvero scritte in modo tale da girare a tutto schermo), come ad esempio `less` o un editor di testo come *VI*, e riportarle in primo piano quando volete.

## Conclusione

Come potete vedere, la *shell* è uno strumento molto complesso, e utilizzarla con profitto richiede una certa pratica. In questo capitolo, per quanto lungo, abbiamo potuto illustrare solo alcuni dei comandi disponibili; **Linux-Mandrake** dispone di migliaia di programmi di utilità, e anche gli utenti più esperti non li usano tutti.

Esistono programmi di utilità per svolgere ogni sorta di funzione: da quelli per la gestione delle immagini (come `convert`, che abbiamo visto di sfuggita, o anche come *GIMP* o altri programmi per l'utilizzo in modalità ***batch*** o per la gestione delle ***pixmap***), programmi per la gestione di suono e musica (compressori MP3 riproduttori audio e per CD), per la scrittura di CD, per l'*e-mail*, client FTP e persino navigatori *web* (`lynx` o `w3m`), per non parlare di tutti gli strumenti utili per l'amministrazione del sistema.

Anche nel caso in cui esistano applicazioni grafiche con funzionalità equivalente, molto spesso si tratta di interfacce grafiche che si basano su questi stessi programmi; i programmi da linea di comando, inoltre, presentano il vantaggio di poter operare in modo non interattivo: potete cominciare a masterizzare un CD e poi uscire dal sistema con la certezza che il processo di masterizzazione verrà portato a termine (si veda la pagina di manuale `nohup`).

# Capitolo 4. Elaborazione testi: Emacs e VI

Come scritto nell'introduzione, l'elaborazione di testi[1] è una caratteristica fondamentale per l'uso di un sistema *Unix*. I due editor dei quali stiamo per spiegare brevemente le modalità d'uso risultano inizialmente un po' ostici, ma una volta imparate le nozioni di base si rivelano entrambi strumenti molto potenti.

## Emacs

*Emacs* è probabilmente il più potente programma di elaborazione testi esistente; può veramente fare qualsiasi cosa ed è espandibile all'infinito grazie al suo linguaggio di programmazione incorporato basato sul *Lisp*. Con *Emacs* potete andare in giro per il *web*, leggere la vostra posta, prendere parte a gruppi di discussione, fare il caffè, etc... ma quello che sarete in grado di fare dopo aver letto questa sezione sarà semplicemente: avviare *Emacs*, modificare uno o più file, salvarli e uscire da *Emacs*. Il che non è un cattivo inizio.

### Una breve introduzione

Avviare *Emacs* è relativamente semplice:

```
emacs [file] [file...]
```

1. "Elaborare un testo" significa modificare il contenuto di un file costituito principalmente da lettere, cifre e segni di punteggiatura; questi file possono essere messaggi di posta elettronica, codice sorgente di programmi, documenti, o file di configurazione.

*Emacs* aprirà in un buffer ogni file immesso come argomento, con un massimo di due buffer visibili contemporaneamente, e aprirà invece il buffer `*scratch*` se non specificate alcun file. Se state usando *X* avrete a disposizione anche dei menu, ma noi qui ci occuperemo delle operazioni da tastiera. `C-x` rappresenta la combinazione `Control+x`, `M-s` rappresenta la combinazione `Alt+s`.

## I primi passi

È il momento di passare alla pratica. Per fare un esempio apriamo due file, `file1` e `file2`. Se questi due file non esistono, saranno creati non appena scriverete qualcosa al loro interno.

```
$ emacs file1 file2
```

per ottenere la finestra mostrata in Figura 4-1.

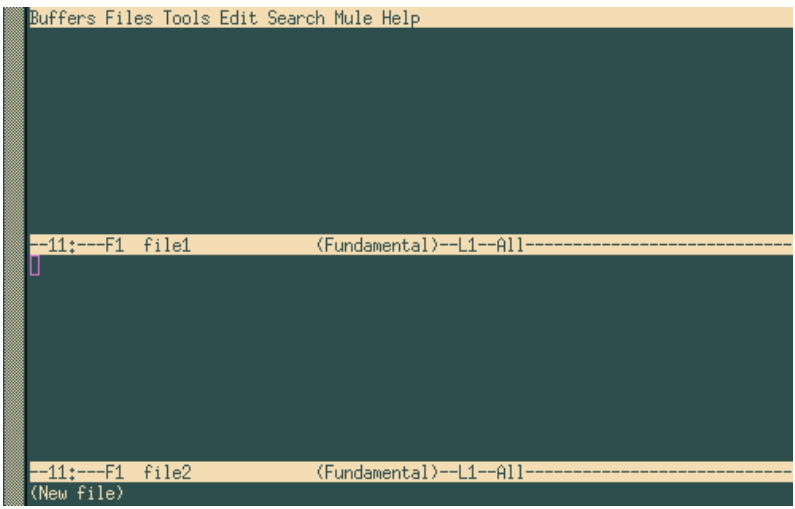


**Figura 4-1. `Emacs`, modifica simultanea di due file**

Come potete vedere, sono stati creati due buffer: uno per ciascun file. Ne è visibile anche un terzo, in fondo allo schermo (dove è scritto `(New file)`): questo è il mini-buffer; non potete andare di vostra iniziativa in questo buffer, dovete essere invitati a farlo da *Emacs* stesso durante le operazioni interattive. Per cambiare buffer digitate `C-x o`. Potete scrivere del testo come in un editor "normale", e cancellarlo con i tasti `Canc` o `Backspace`.

Per spostarvi all'interno del documento potete usare i tasti cursore, ma anche altre combinazioni di tasti: `C-a` per andare all'inizio della riga, `C-e` per andare alla fine della riga, `M-<` per andare all'inizio del buffer e `M->` per andare alla fine del buffer. Esistono molte altre combinazioni, anche per simulare tutti i tasti cursore[2].

Quando volete salvare i cambiamenti effettuati in un file digitate `C-x C-s`, se invece volete salvare il contenuto del buffer in un altro file digitate `C-x C-w`, e *Emacs* vi chiederà il nome del file in cui deve essere salvato il contenuto del buffer. Per far questo potete aiutarvi con il ***completamento automatico***.

## La gestione dei buffer

Se lo desiderate, potete visualizzare sullo schermo un solo buffer. Ci sono due modi per farlo:

- vi trovate nel buffer che volete nascondere: digitate `C-x 0`;
- vi trovate nel buffer che volete mantenere sullo schermo: digitate `C-x 1`.

2. *Emacs* è stato progettato per funzionare su tutti i sistemi possibili, e ancora oggi esistono terminali sprovvisti dei tasti cursore. Questo è ancor più vero per *VI*.

Esistono poi due modi per richiamare sullo schermo il buffer che volete:

- digitate `C-x b` e scrivete il nome del buffer che desiderate,
- digitate `C-x C-b` e sarà aperto un nuovo buffer, denominato `*Buffer List*`; potete muovervi all'interno di esso tramite la combinazione `C-x o`, quindi selezionate il buffer desiderato e premete il tasto `Invio`, oppure potete digitarne il nome all'interno del mini-buffer. Quando avrete fatto la vostra scelta, il buffer `*Buffer List*` tornerà sullo sfondo.

Se avete finito di modificare un file e volete togliere di mezzo il relativo buffer, digitate `C-x k`; *Emacs* vi chiederà quale buffer deve essere chiuso. Come opzione predefinita verrà indicato il nome del buffer in cui vi trovate attualmente; se volete eliminare un buffer diverso da quello indicato scrivetene direttamente il nome, oppure premete `TAB`: in questo caso *Emacs* aprirà (ancora) un altro buffer denominato `*Completions*`, contenente la lista delle scelte possibili. Confermate la vostra scelta con il tasto `Invio`.

Potete anche ripristinare in qualsiasi momento la visualizzazione di due buffer contemporaneamente; per farlo, digitate `C-x 2`. Il nuovo buffer aperto sarà inizialmente una copia del buffer attuale (il che vi permette, ad esempio, di modificare un file lungo in più punti "allo stesso tempo"), e potrete poi procedere come già descritto per spostarvi in un altro buffer.

Potete aprire altri file in qualsiasi momento, tramite `C-x C-f`; *Emacs* vi chiederà il nome del file e potrete utilizzare nuovamente il completamento automatico.

## Copia, taglia, incolla, cerca

Supponiamo di essere nella situazione di Figura 4-2.

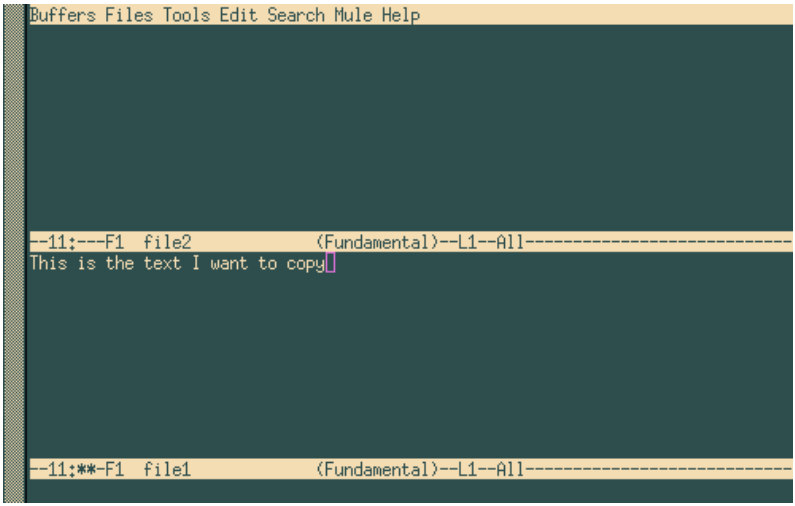


**Figura 4-2. Emacs, prima di copiare il blocco di testo**

Innanzitutto dovete selezionare il testo che volete copiare. In *X* potete farlo usando il mouse, e l'area selezionata sarà anche evidenziata; ma ora siamo in modo testo **:-)**. In questo caso vogliamo copiare l'intera frase. Prima di tutto dovete posizionare un marcatore per marcare l'inizio dell'area; supponendo che il cursore sia nella posizione mostrata nell'immagine qui sopra, digitate innanzitutto `C-SPAZIO` (`Control` + barra spaziatrice): *Emacs* mostrerà il messaggio `Mark set` nel minibuffer. Quindi spostatevi a inizio riga con `C-a`: l'area da copiare o tagliare è rappresentata da tutto il testo compreso tra il marcatore e la posizione attuale del cursore, quindi in questo caso è tutta la riga. Digitate poi `M-w` (per copiare) o `C-w` (per tagliare). Nel caso della copia, *Emacs* vi riporterà temporaneamente alla posizione del marcatore, in modo che possiate vedere l'area selezionata.

Posizionatevi poi sul buffer nel quale volete copiare il testo, e premete `C-y` per ottenere quello che potete vedere in Figura 4-3.

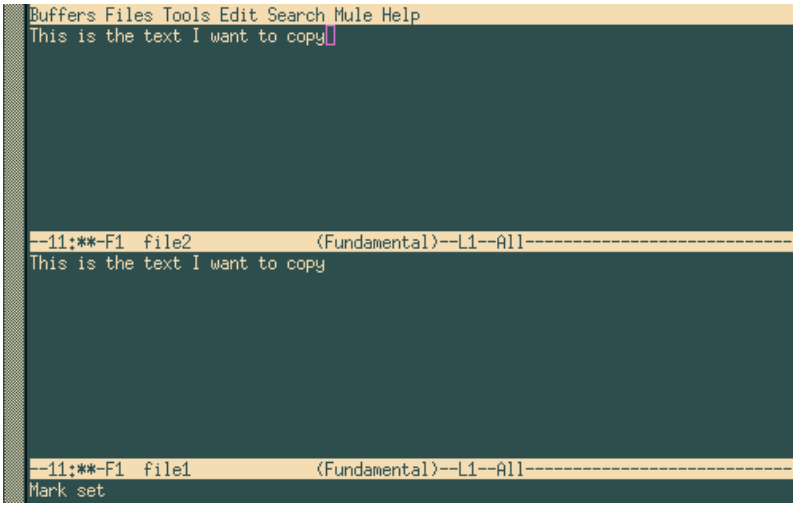


**Figura 4-3. Emacs, dopo la copia del blocco di testo**

In effetti, ciò che avete appena fatto è stato copiare il testo nel ***"kill ring"*** di *Emacs*: questo "kill ring" contiene tutti i blocchi di testo copiati o tagliati da quando *Emacs* è stato avviato. **Qualsiasi** blocco, appena copiato o tagliato, viene inserito in cima al kill ring. La combinazione `C-y` "incolla" solo il blocco che si trova in cima: se volete accedere agli altri blocchi premete `C-y` e poi `M-y` finché non arrivate al testo desiderato.

Per cercare del testo, andate nel buffer in cui volete effettuare la ricerca e premete `C-s`: *Emacs* vi chiederà quale stringa deve cercare. Per avviare una ulteriore ricerca per la stessa stringa, sempre nello stesso buffer, premete nuovamente `C-s`. Quando *Emacs* arriva alla fine del buffer e non trova più occorrenze della stringa, potete premere ancora `C-s` per far ripartire la ricerca dall'inizio del buffer. Premendo il tasto `Invio` la ricerca sarà conclusa.

Per effettuare una ricerca con sostituzione, premete `M-%`: *Emacs* vi chiederà quale stringa deve cercare, con cosa deve sostituirla, e vi chiederà conferma per ogni occorrenza che troverà.

Per finire, una cosa molto utile: `C-x u` annulla l'ultima operazione effettuata. Potete annullare un numero di operazioni a vostro piacimento.

### Uscire da Emacs

La scorciatoia per farlo è `C-x C-c`. Se non avete salvato le modifiche apportate ai buffer, *Emacs* vi chiederà se volete farlo ora.

## VI: l'antenato

*VI* è stato il primo editor a tutto schermo esistente. Rappresenta una delle principali obiezioni mosse dai detrattori di *Unix*, ma anche uno dei principali argomenti dei suoi difensori: sebbene sia complicato da imparare, è anche uno strumento estremamente potente una volta che ci si abitua a usarlo. Con poche pressioni dei tasti un utente di *VI* può spostare le montagne e, a parte *Emacs*, di pochi editor di testo può essere detta la stessa cosa.

La versione fornita con **Linux-Mandrake** è in realtà *VIm*, che sta per *VI iMproved*, ma in questo capitolo lo chiameremo comunque *VI*.

### Modalità inserimento, modalità comandi, modalità ex...

Prima di tutto, come avviarlo: esattamente come *Emacs*. Quindi riprendiamo i nostri due file e digitiamo:

```
$ vi file1 file2
```

A questo punto vi troverete di fronte a una finestra simile a Figura 4-4.

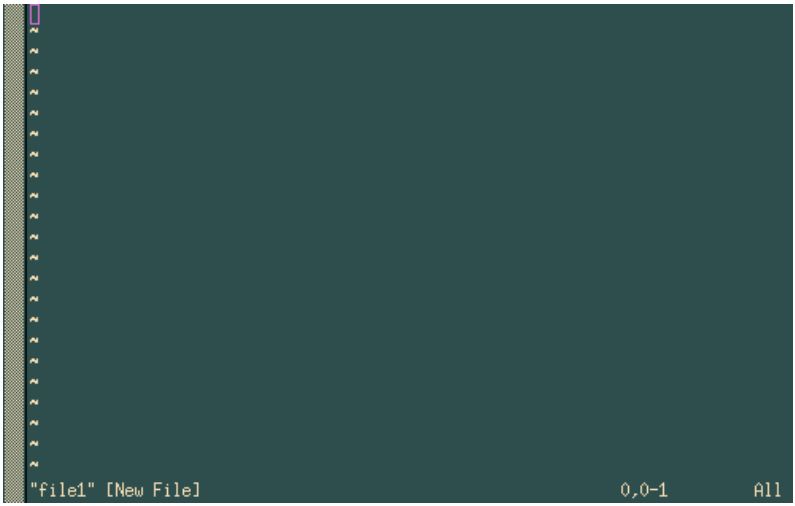


**Figura 4-4. Situazione iniziale in VIm**

Vi trovate ora di fronte al primo file aperto, in ***modalità comandi***. In modalità comandi non potete inserire testo in un file... Per farlo, dovete andare in ***modalità inserimento***, e perciò dovete digitare uno dei comandi che vi permettono di fare questo passaggio:

- **a** e **i**: per inserire testo rispettivamente dopo e prima del cursore (**A** e **I** inseriscono il testo alla fine e all'inizio della riga attuale);
- **o** e **O**: per inserire testo rispettivamente sotto e sopra la riga attuale.

In modalità inserimento vedrete comparire in fondo allo schermo la scritta `--INSERT--` (affinché possiate sapere in che modalità vi trovate); potete inserire testo solo e unicamente in questa modalità. Per tornare in modalità comandi premete il tasto `Esc`.

In modalità inserimento potete usare i tasti `Backspace` e `Canc` per cancellare il testo durante la digitazione. Per spostarvi all'interno del testo, sia in modalità comandi che in modalità inserimento, potete usare i tasti cursore. In modalità comandi sono disponibili anche altre combinazioni di tasti di cui ci occuperemo in seguito.

Alla modalità ex si accede premendo il tasto **:** in modalità comandi: lo stesso carattere **:** comparirà in fondo allo schermo, e il cursore sarà posizionato su di esso; tutto ciò che scriverete in seguito, fino alla pressione di Invio, sarà considerato da *VI* come un comando ex. Se cancellate il comando e il **:** sarete riportati in modalità comandi, e il cursore tornerà nella sua posizione originale.

Per salvare i cambiamenti effettuati in un file digitate `:w` in modalità comandi. Se volete salvare il contenuto del buffer in un file differente scrivete `:w <nome_file>`.

## La gestione dei buffer

Come in *Emacs*, potete avere diversi buffer visualizzati sullo schermo; per ottenere questo risultato usate il comando `:split`.

Per spostarvi da un file all'altro, quando siete in un buffer, scrivete `:next` per andare al file successivo e `:prev` per andare al precedente. Potete anche usare il comando `:e <nome_file>`, che vi permette di spostarvi sul file desiderato se esso è stato già aperto o di aprire un nuovo file; anche in questo caso è disponibile il completamento automatico.

Per cambiare buffer digitate `C-w j`, per andare al buffer inferiore, o `C-w k` per andare a quello superiore; potete anche usare i tasti cursore su e giù invece di **j** e **k**. Il comando `:close` nasconde un buffer, mentre il comando `:q` lo chiude.

Fate attenzione, *VI* è pignolo: se provate a nascondere o chiudere un buffer senza aver salvato i cambiamenti, il comando non sarà eseguito e comparirà questo messaggio:

```
No write since last change (use! to override)
```

In questo caso fate come vi viene detto e digitate `:q!` o `:close!`.

## Elaborazione del testo e comandi di movimento

Oltre ai tasti `Backspace` e `Canc` in modalità inserimento, *VI* ha molti altri comandi per cancellare, copiare, incollare e sostituire testo – in modalità comandi. In questa sede ci occuperemo di alcuni di essi. Tutti i comandi mostrati qui sono in realtà divisi in due parti: l'azione da compiere e l'area interessata. L'azione può essere:

- **c**: sostituire (*Change*); l'editor cancella il testo indicato, e dopo questo comando torna in modalità inserimento;
- **d**: cancellare (*Delete*);
- **y**: copiare (*Yank*), ce ne occuperemo nella prossima sezione.
- **.**: ripete l'ultima azione.

L'area interessata indica su quali gruppi di caratteri deve agire il comando. Questi stessi comandi di area interessata immessi da soli, così come sono, in modalità comandi corrispondono a movimenti:

- **h**, **j**, **k**, **l**: un carattere a sinistra, in giù, in su, a destra[3];
- **e**, **b**, **w**: fino alla fine e fino all'inizio della parola attuale, fino all'inizio della parola successiva;
- **^**, **0**, **$**: fino al primo carattere non vuoto della riga attuale, fino all'inizio della riga attuale, fino alla fine della riga attuale;
- `f<x>`: fino alla prossima occorrenza del carattere `<x>`; ad esempio, `fe` sposta il cursore sulla prossima occorrenza del carattere **e**;
- `/<stringa>`, `?<stringa>`: fino alla successiva occorrenza della stringa o espressione regolare `<stringa>`, e la stessa cosa a ritroso nel file; ad esempio, `/foobar` sposta il cursore fino alla prossima occorrenza della parola `foobar`;
- **{**, **}**: fino all'inizio e fino alla fine del paragrafo attuale;

3. Una scorciatoia per `dl` (cancella un carattere in avanti) è **x**; una scorciatoia per `dh` è **X**; `dd` cancella la riga attuale.

- **G**, **H**: fino alla fine del file, fino all'inizio dello schermo.

Ciascuno di questi caratteri di area interessata o comandi di movimento può essere preceduto da un numero di ripetizioni; nel caso di **G**, invece, esso indica un numero di riga all'interno del file. Partendo da queste basi, potete ottenere tutte le possibili combinazioni. Alcuni esempi:

- `6b`: si sposta di 6 parole indietro;
- `c8fk`: cancella tutto il testo fino alla ottava occorrenza del carattere **k**, quindi va in modalità inserimento;
- `91G`: va alla linea 91 del file;
- `d3$`: cancella fino alla fine della riga attuale, più le due righe successive.

È vero che questi comandi non sono molto intuitivi, ma come sempre il miglior metodo è fare pratica. Comunque potete vedere che l'espressione "spostare le montagne con pochi tasti" non era poi così esagerata **:-)**

## Taglia, copia, incolla

*VI* possiede un comando per copiare testo che abbiamo già incontrato: il comando **y**. Per tagliare il testo è sufficiente usare il comando **d**. Avete a disposizione 27 memorie per memorizzare il testo: una memoria anonima e 26 memorie denominate come le 26 lettere minuscole dell'alfabeto inglese.

Per usare la memoria anonima dovete digitare il comando semplice. Perciò il comando `y12w` copia nella memoria anonima le 12 parole che si trovano dopo il cursore[4]. Usate invece `d12w` se volete tagliare lo

4. ...sempre se il cursore si trova all'inizio della prima parola!

stesso blocco di testo.

Per utilizzare una delle 26 memorie associate alle lettere dell'alfabeto, digitate la sequenza `"<x>` prima del comando, dove `<x>` rappresenta il nome della memoria. Perciò dovreste scrivere `"ky12w` per copiare le stesse 12 parole nella memoria **k**, e `"kd12w` per tagliarle.

Per incollare il contenuto della memoria anonima dovete usare i comandi **p** o **P** (per *Paste*), per inserire il testo rispettivamente dopo o prima del cursore. Per incollare il contenuto di una memoria con un nome usate `"<x>p` o `"<x>P` nello stesso modo (ad esempio, `"dp` inserirà il contenuto della memoria `d` dopo il cursore).

Diamo un'occhiata all'esempio di Figura 4-5.

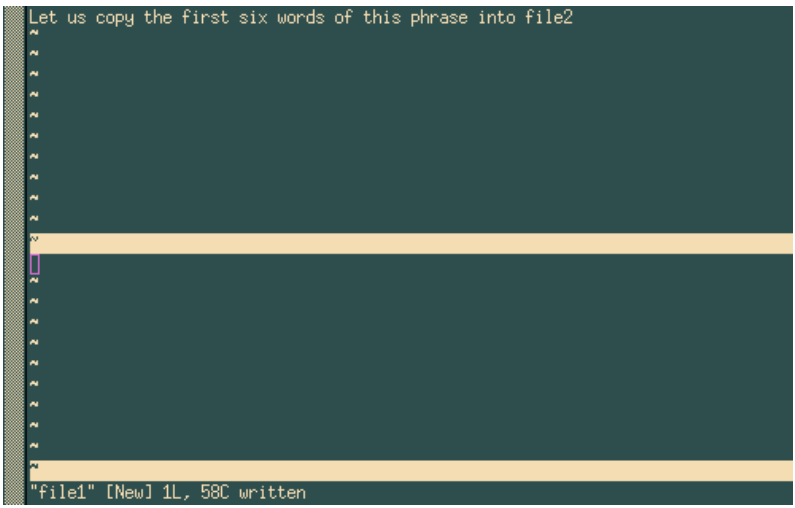


**Figura 4-5. `VIm`, prima di copiare il blocco di testo**

Per compiere questa operazione seguiremo questi passi:

- copiamo le prime 6 parole della frase nella memoria **r** (per esempio): `"ry6w`[5];

- andiamo nel buffer `file2`, che si trova sotto: `C-w j`;
- incolliamo il contenuto della memoria **r** prima del cursore: `"rp`.

Otteniamo così il risultato che ci aspettavamo, come mostrato in Figura 4-6.

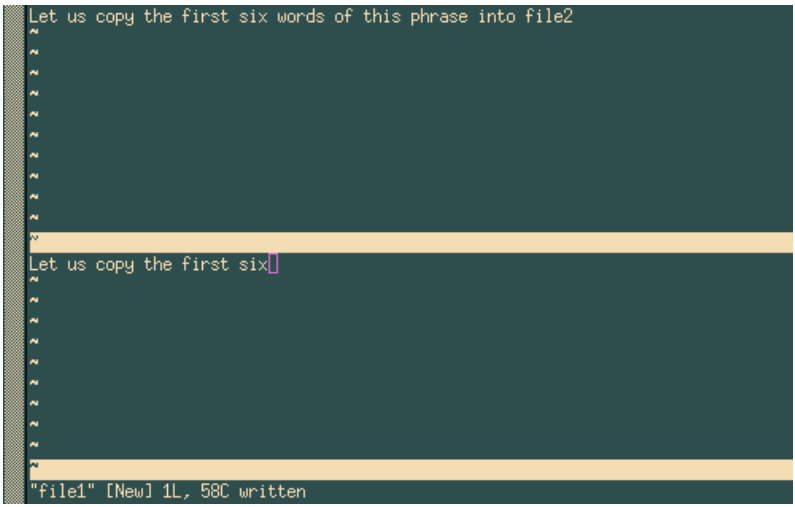


**Figura 4-6. `VIm`, dopo la copia del blocco di testo**

Cercare del testo è molto semplice: in modalità comandi è sufficiente digitare / seguito dalla stringa che deve essere cercata, e quindi premere il tasto `Invio`. Per esempio, con `/party` faremo una ricerca della stringa `party` a partire dalla posizione attuale del cursore. Premendo **n** venite portati all'occorrenza successiva, e se siete arrivati alla fine del file la ricerca ripartirà dall'inizio. Per cercare all'indietro usate **?** invece di /.

5. `y6w` significa letteralmente: "*`Yank 6 words`*".

### Uscire da VI

Il comando per uscire è `:q` (in realtà questo comando chiude il buffer attuale, come abbiamo già visto, ma se questo è l'unico buffer esistente uscirete da *VI*). Esiste una scorciatoia: nella maggior parte dei casi voi modificherete un solo file per volta. Quindi per uscire potrete usare:

- `:wq` per salvare le modifiche e uscire (un metodo più veloce è `ZZ`), oppure
- `:q!` per uscire senza salvare.

Avrete già notato che, se avete diversi buffer aperti, `:wq` salverà il buffer attivo per poi chiuderlo.

## Un'ultima parola...

Naturalmente abbiamo detto molto più di quanto fosse necessario (dopo tutto lo scopo principale era modificare un file di testo), ma è servito anche a mostrarvi alcune delle possibilità di ciascuno di questi editor. Ci sarebbero moltissime altre cose da dire su di essi, come testimoniato dalla gran quantità di libri dedicati all'uno o all'altro.

Prendetevi il tempo per assimilare tutte queste informazioni, sceglietene uno, oppure imparate soltanto ciò che vi sembra necessario. Ma almeno ora sapete che, quando vorrete andare oltre, potrete farlo **:-)**.

# Capitolo 5. La stampa

Questo capitolo è diviso in due parti: *Installazione e gestione delle stampanti*, pag. 75, per chi provvede alla normale amministrazione del proprio sistema; e *La stampa di documenti*, pag. 85, che spiega come configurare e utilizzare uno strumento di stampa avanzato: *XPP*.

## Installazione e gestione delle stampanti

A partire dalla versione 7.2, **Linux-Mandrake** fa uso di un nuovo sistema di stampa basato su *CUPS* (http://www.cups.org/)[1]. Si tratta di uno strumento molto potente basato su una gestione e configurazione decentralizzata, tale da rendere tutte le stampanti connesse in una rete locale disponibili per tutti gli utenti.

### Installazione di CUPS e uso della sua interfaccia web

Dato che *CUPS* adesso è il programma predefinito di **Linux-Mandrake** per quanto riguarda la stampa, tutti i pacchetti necessari dovrebbero essere stati installati automaticamente. Se così non fosse, assicuratevi che siano stati installati per lo meno i pacchetti `cups`, `cups-drivers` e `xpp`.

1. *Common Unix Printing System*, o "Sistema di stampa comune per Unix"

> **Nota:** Potete gestire le vostre stampanti usando *CUPS* in due modi diversi: tramite un'interfaccia *web*, oppure con un'applicazione per *KDE* chiamata *Kups*. Abbiamo già parlato di quest'ultima nel **Manuale utente**, adesso invece ci concentreremo sull'interfaccia *web*, alla quale si può accedere da qualunque piattaforma.

Una volta lanciato il vostro navigatore *web* preferito, digitate `http://localhost:` nel campo di testo relativo al percorso o all'URL. Vi verrà mostrato il menu *CUPS* principale (Figura 5-1).

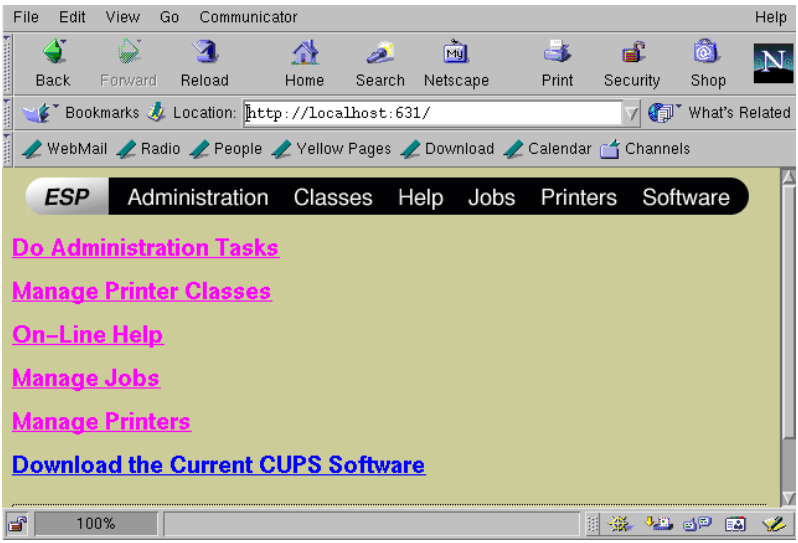


**Figura 5-1. La pagina di benvenuto di** `CUPS`

Adesso potete sfogliare l'interfaccia di configurazione come se si trattasse di un sito *web*.

## Configurazione di una nuova stampante

Nel caso la vostra LAN disponga già di macchine sulle quali è stato installato, ed è utilizzato, *CUPS*, potrete vedere una lista di stampanti seguendo il collegamento Manage Printers. Per il momento si presuppone che voi intendiate installare una stampante connessa a un computer isolato da qualsiasi rete. Per configurazioni più complesse consultate l'On-Line Help ("aiuto in linea").

La pagina Manage Printers (Figura 5-2) dovrebbe quindi risultare vuota, per il momento.

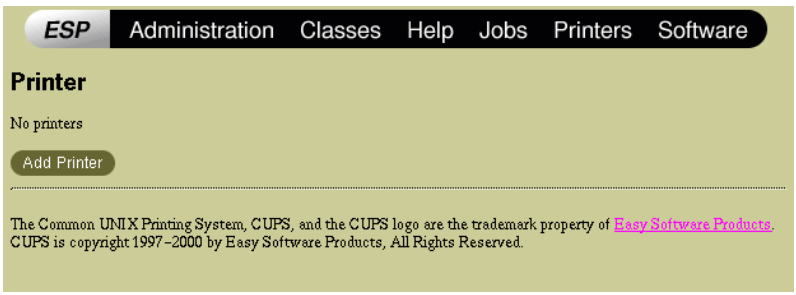


**Figura 5-2. La lista priva di stampanti di** `CUPS`

Adesso, per configurare una nuova stampante, cliccate sul pulsante Add Printer in fondo alla pagina: questo darà inizio a una procedura di configurazione in quattro passi. Per avanzare da un passo all'altro cliccate sul pulsante Continue dopo aver riempito tutti i campi richiesti sulla pagina.

> **Nota:** La prima volta che vorrete portare a termine una funzione amministrativa, come installare una nuova stampante, con *CUPS*, vi verrà richiesta la password di `root` (Figura 5-3). Per continuare, dovete inserire il login e la password di root.

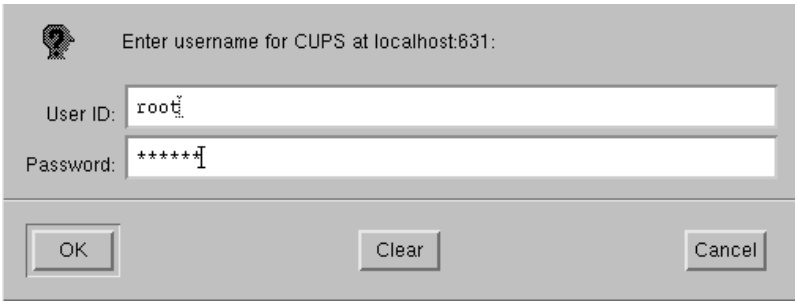


**Figura 5-3. La finestra di login di** `CUPS`

## Informazioni informali riguardo la stampante

Questa prima scheda contiene tre campi che potete riempire a vostro piacimento per aiutare altri utenti a capire con che tipo di stampante hanno a che fare. Queste informazioni non hanno nessun effetto sul comportamento della stampante, ma è comunque consigliabile inserirle in maniera accurata, in modo da evitare possibili confusioni in seguito.

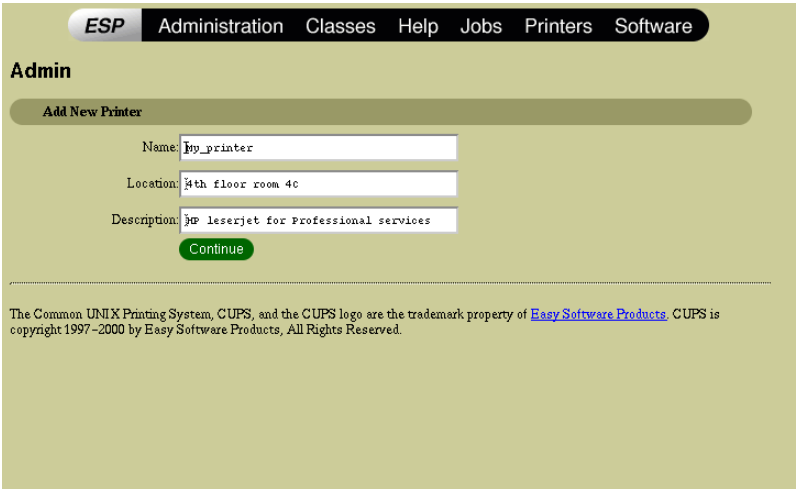


**Figura 5-4. Installazione di una nuova stampante, passo 1**

L'unico campo indispensabile è il nome della stampante.

## Connessione della stampante

È necessario comunicare a *CUPS* dove si trova fisicamente la stampante. Per una stampante connessa direttamente al vostro computer, dovete scegliere Parallel Port #1, Serial Port, o USB, a seconda del tipo di connessione.


**Figura 5-5. Installazione di una nuova stampante, passo 2**

Sono disponibili molti tipi di connessione:

Ethernet

Per stampanti che sono direttamente connesse a una rete locale.

LPD/LPR

Per stampanti che implementano in modo nativo questo protocollo, o stampanti che sono servite da questo tipo di coda di stampa. I sistemi operativi *Unix*, in genere, offrono questo tipo di connessione.

Samba

Per stampanti che dipendo da server *Windows*. Si noti che per accedere a questo tipo di stampanti è necessario installare anche il pacchetto *Samba*.

## Inserimento della marca della stampante

Adesso è arrivato il momento di comunicare a *CUPS* che tipo di stampante state installando. Dovete semplicemente scegliere il nome del produttore nella lista a scorrimento.

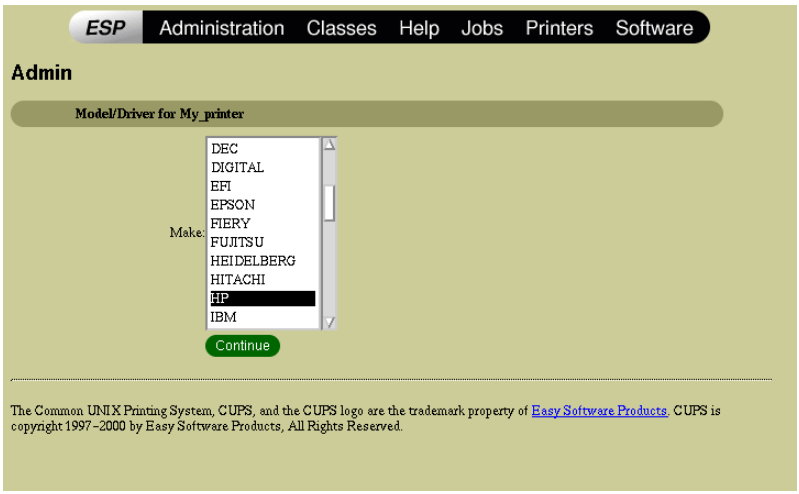


**Figura 5-6. Installazione di una nuova stampante, passo 3**

## Inserimento del modello della stampante

Questo è l'ultimo passo: in base alla vostra scelta precedente, la lista adesso vi mostrerà tutti i modelli venduti dal produttore specificato. Scegliete con attenzione il modello della vostra stampante.

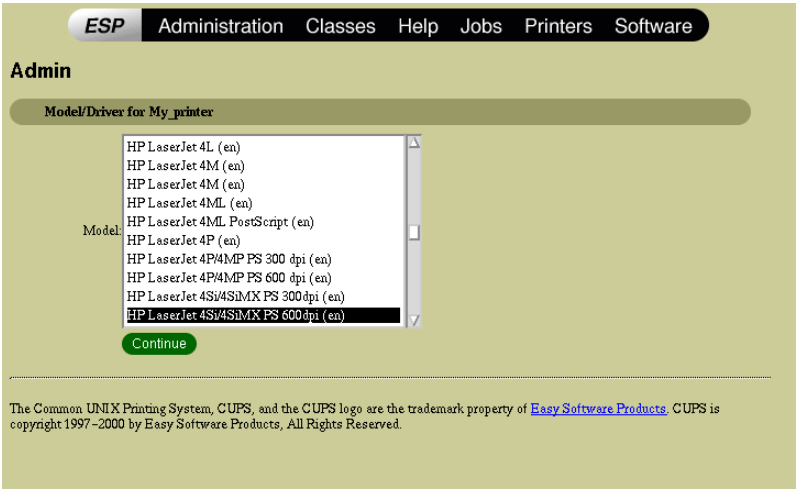


**Figura 5-7. Installazione di una nuova stampante, passo 4**

Se tutto è andato bene, dovreste poter vedere la vostra nuova stampante nella pagina raggiungibile cliccando sul collegamento Printers.

## Configurazione finale e test della stampante

Prima di effettuare un test della stampante, dovete assicurarvi che la configurazione delle dimensioni della carta per quest'ultima sia corretta. Recatevi sulla pagina della stampante, e cliccate sul pulsante Configure Printer. Una volta raggiunta la pagina dei parametri relativi alla stampante, spostatevi alla sezione `General` e scegliete le dimensioni della carta (Page Size) appropriate. Alcune stampanti si rifiutano di funzionare se la carta del formato appropriato non è presente.

**Attenzione**

Una nota riguardo la pagina dei parametri: ogni volta che modificate un parametro della stampante in una qualsiasi delle sezioni, dovete cliccare sul pulsante Continue corrispondente a tale sezione perché i vostri cambiamenti vengano accettati.

## Una nota riguardo la sicurezza

Come impostazione predefinita, ogni volta che configurate una stampante sulla vostra macchina essa diviene disponibile per tutti gli altri utenti della rete locale a cui quest'ultima è connessa. Se desiderate che nessun altro possa utilizzare la vostra stampante, dovete modificare a mano il file di configurazione di *CUPS*: `/etc/cups/cupsd.conf`. Tutto quello che dovete fare è sostituire la riga

```
#BrowseInterval 30
```

con

```
BrowseInterval 0
```

Questo file, inoltre, contiene un gran numero di opzioni che vi permettono di configurare con grande precisione il vostro server di stampa. In particolare, potete limitare l'accesso alla stampante da parte di specifiche macchine o intere sotto-reti. Per maggiori informazioni sull'argomento consultate l'aiuto in linea dell'interfaccia *web*.

**Suggerimento:** Tutte le volte che apportate modifiche al file di configurazione non dimenticatevi di far ripartire il demone del server *CUPS* impartendo il comando:

```
/etc/rc.d/init.d/cups restart
```

## Gestione della coda di stampa

Questa caratteristica è particolarmente utile per stampanti caratterizzate da un elevato carico di lavoro, ma potreste comunque averne bisogno in maniera occasionale, ad esempio per annullare un comando di stampa di 10.000 pagine dato per errore. Quando inviate un lavoro alla stampante potete controllare se ne avete altri in attesa e, se siete l'amministratore della macchina da cui dipende la stampante, anche quelli di tutti gli utenti: è sufficiente collegarsi alla pagina relativa alla stampante in questione (Figura 5-8).

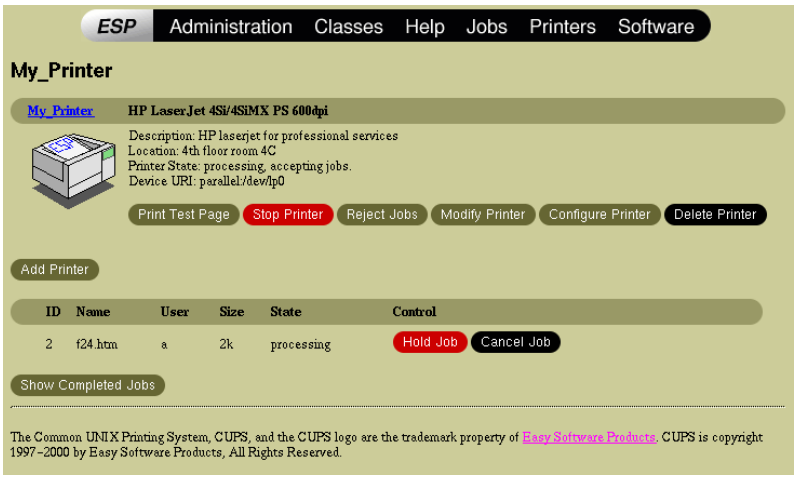


**Figura 5-8. La pagina relativa allo stato della stampante**

Potete effettuare due azioni diverse su un particolare lavoro di stampa:

- Hold Job ("sospendi lavoro"): per mettere il lavoro in una lista d'attesa, verrà mandato in stampa soltanto quando tornerete alla

pagina relativa e cliccherete sul pulsante verde Release Job ("riprendi lavoro").

- Cancel Job ("annulla lavoro"): per cancellare questo lavoro in modo definitivo, depennandolo dalla coda di stampa.

Se volete rendere temporaneamente impossibile l'accesso alla vostra stampante (per cambiare il toner, ad esempio) potete semplicemente cliccare sul pulsante Reject Jobs ("rifiuta lavori"). In seguito, quando la stampante sarà di nuovo pronta ad accettare lavori, premete il pulsante Accept Jobs ("accetta lavori").

> **Suggerimento:** Se siete interessati a funzionalità avanzate per quanto riguarda la gestione della coda di stampa, prendete in considerazione il programma `kups`.

## La stampa di documenti

Oggi la distribuzione **Linux-Mandrake** offre un'applicazione basata su una gradevole interfaccia grafica, *`XPP`*, in sostituzione dei tradizionali comandi per stampare documenti, `lpr` e `lp`; questo strumento permette agli utenti di stampare file e di configurare i parametri di stampa per tutti i documenti mandati in stampa. Inoltre può essere usata come "comando di stampa" in altre applicazioni[2], in modo da poter accedere comodamente a tutte le stampanti disponibili anche da altri programmi.

2. A questo scopo potete usare anche il comando `qtcups`.

## Semplice stampa di un file

Supponiamo che abbiate salvato sul vostro disco rigido un'immagine scaricata da un sito *web*, e che desideriate stamparla. Per prima cosa lanciate *XPP* dal menu Applicazioni+Publishing→X Printing Panel, vedrete comparire la finestra principale (Figura 5-9).

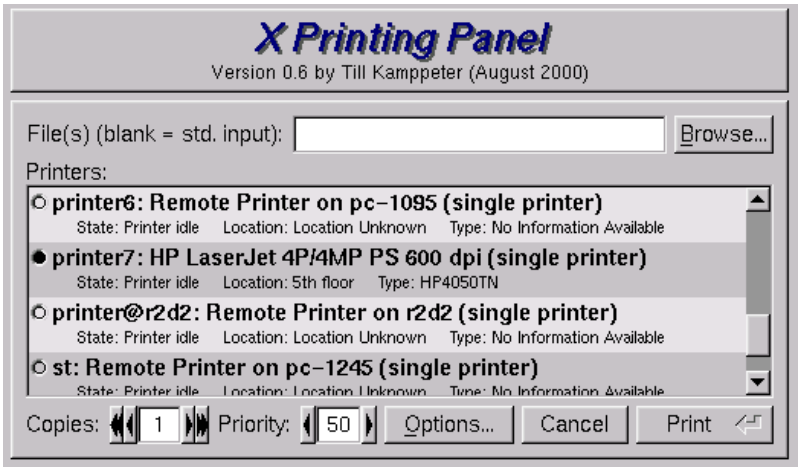


**Figura 5-9. La finestra principale di** `XPP`

La finestra è divisa in tre parti principali: un campo di testo per indicare dove si trova il file (o i file) da stampare, la lista di stampanti disponibili e una serie di opzioni nella parte inferiore.

Per scegliere il file da stampare, cliccate sul pulsante Browse..., comparirà una finestra di dialogo (Figura 5-10) grazie alla quale potrete navigare all'interno del vostro disco rigido e scegliere il file che desiderate stampare. Notate che, se sapete già dove si trova tale file, potete anche digitare il percorso completo nel campo di immissione testo accanto al pulsante.

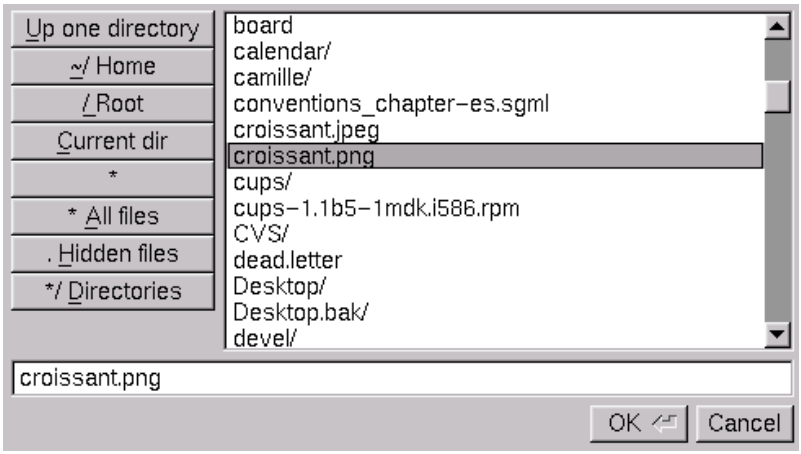


**Figura 5-10. Scelta di un file con** `XPP`

Successivamente accertatevi di aver scelto la stampante voluta (quella corrente presenta un punto nero sulla sinistra) e infine premete il pulsante Print.

> **Suggerimento:** Se desiderate usare *`XPP`* come programma (filtro) di stampa per documenti mandati in stampa da altre applicazioni (come *`Netscape`* ad esempio, ma sono molti quelli che lo consentono), tutto quello che dovete fare è cambiare il comando di stampa: in genere nella finestra di dialogo relativa alle opzioni di stampa è presente un campo di testo che contiene la stringa lpr, sostituitela con `xpp`, e questo è tutto.

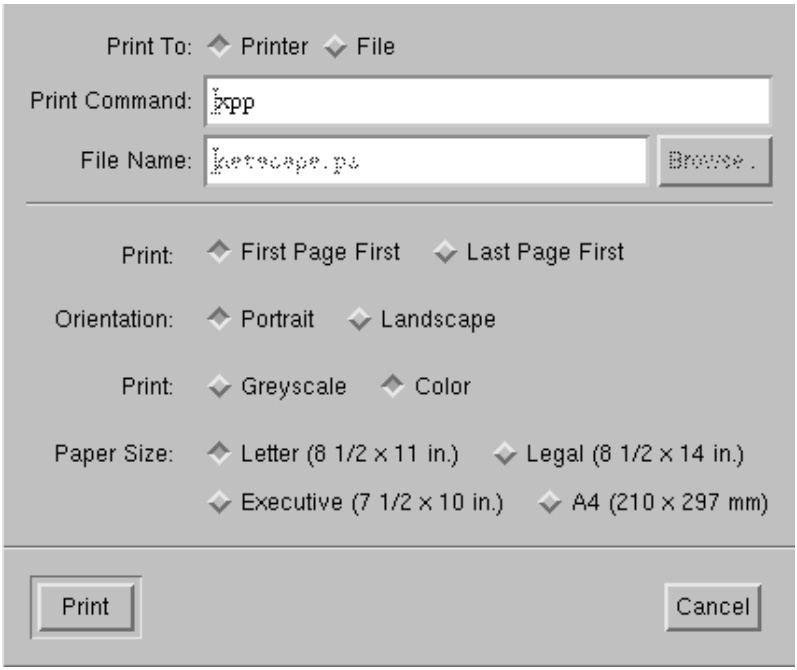


**Figura 5-11. La finestra di stampa di `Netscape`**

A questo punto, quando cliccherete sul pulsante OK, comparirà la finestra di *XPP*. Semplicemente cambiate le opzioni come desiderate, senza preoccuparvi del file.

## Configurazione avanzata

*XPP* vi permette di configurare con precisione le vostre stampe. In primo luogo, nella finestra principale è presente l'opzione di stampare più copie dello stesso file (campo Copies), e di cambiare la priorità dei lavori nella vostra coda di stampa. Quest'ultima caratteristica è utile quando le stampanti disponibili sono impiegate in maniera pesante da molti utenti: se avete necessità che un documento venga stampato

il prima possibile, aumentate il numero di priorità; se, invece, mandate in stampa un documento che non vi serve immediatamente, abbassate il numero di priorità.

In secondo luogo, grazie al pulsante Options... potete accedere a una finestra di dialogo che presenta diverse schede di configurazione dei parametri di stampa (Figura 5-12).

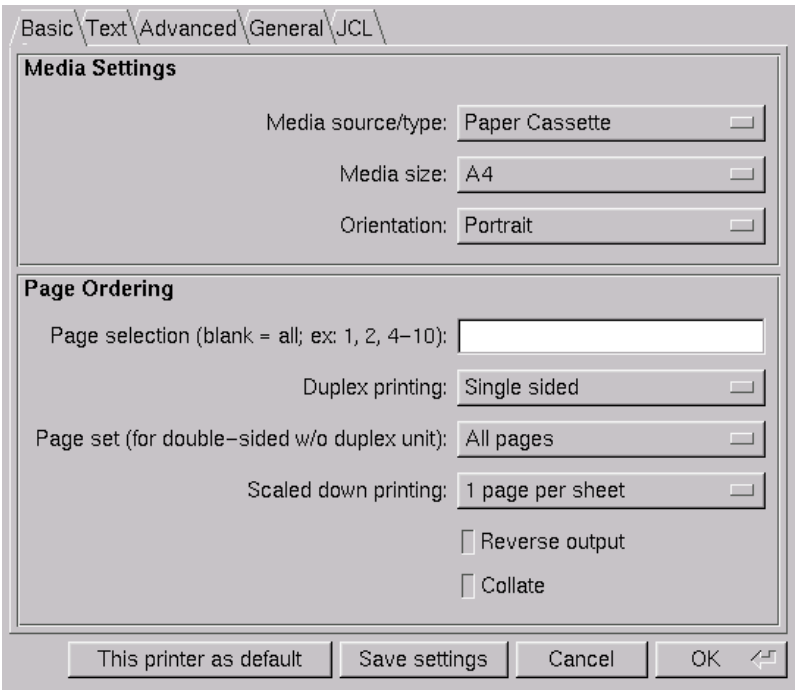


**Figura 5-12. La scheda delle opzioni base di `XPP`**

La prima scheda è divisa in due parti:

Media Settings

> In questa sezione potete scegliere i valori di tre parametri, per quanto, a seconda delle caratteristiche della stampante, non è

certo che siano tutti o anche solo in parte applicabili. Il primo consiste nella scelta tra il vassoio per la carta e l'alimentazione manuale. Quindi abbiamo le dimensioni della carta, e infine l'orientamento della stampa.

Page Ordering

Queste opzioni sono molto utili per stabilire in che modo il documento viene stampato, e quali parti di esso.

- Grazie al campo page selection è possibile stampare solo le pagine desiderate: nell'esempio proposto verrebbero stampate le pagine 1, 2, 4, 5, 6, 7, 8, 9 e 10 del documento.
- L'opzione Duplex Printing è utile soltanto se la vostra stampante la supporta. Viene chiamata anche "stampa sui due lati".
- Come afferma l'etichetta relativa, l'opzione page set vi permette di stampare sui due lati anche su stampanti che non hanno il supporto per il Duplex Printing. Cominciate con lo stampare le pagine pari, poi rimettete i fogli nel vassoio (prestate attenzione all'orientamento!) e stampate le pagine dispari sul lato opposto.
- Scaled down printing è un'opzione gradita agli ecologisti, in quanto consente la stampa di più pagine in un unico foglio di carta. Se usata contemporaneamente al duplex printing dovrebbe contribuire a salvare le foreste pluviali **:-)**
- L'opzione reverse output fa in modo che le pagine vengano stampate a partire dall'ultima. È utile per stampanti che impilano i fogli a faccia in sù, usando questa opzione i documenti che consistono di più pagine sono stampati nell'ordine corretto.
- Per finire, con l'opzione Collate viene modificato l'ordine in cui sono stampate le pagine quando si stampano più copie di uno stesso documento. Se attivate questa opzione, l'ordine di stampa di un documento di tre pagine sarà 1,2,3,1,2,3. In caso

contrario, le pagine usciranno in quest'ordine: 1,1,2,2,3,3.

La scheda Text (Figura 5-13) offre opzioni ancora più raffinate per cambiare il modo in cui vengono stampati i file di testo.

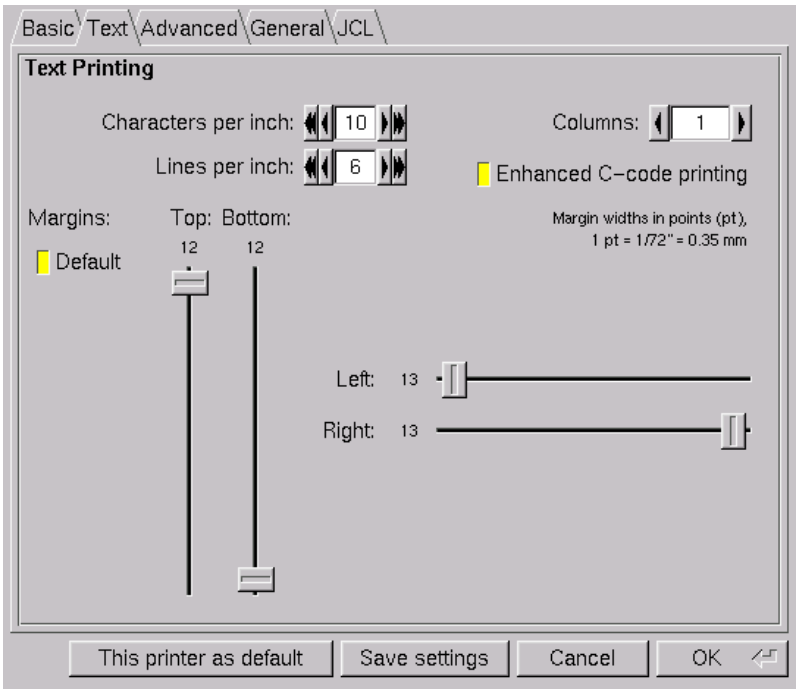


**Figura 5-13. La scheda relativa alle opzioni per i file di testo di** `XPP`

Le opzioni sono molte, ma è piuttosto facile capire il significato di ciascuna di esse. Vi segnaliamo in particolare l'opzione Enhanced C-Code printing, che provvede a stampare un'intestazione per ogni pagina e a mettere in evidenza la sintassi per i listati di programmi in `C`.

La scheda Advanced (Figura 5-14) offre opzioni finalizzate a modificare l'aspetto della pagina.

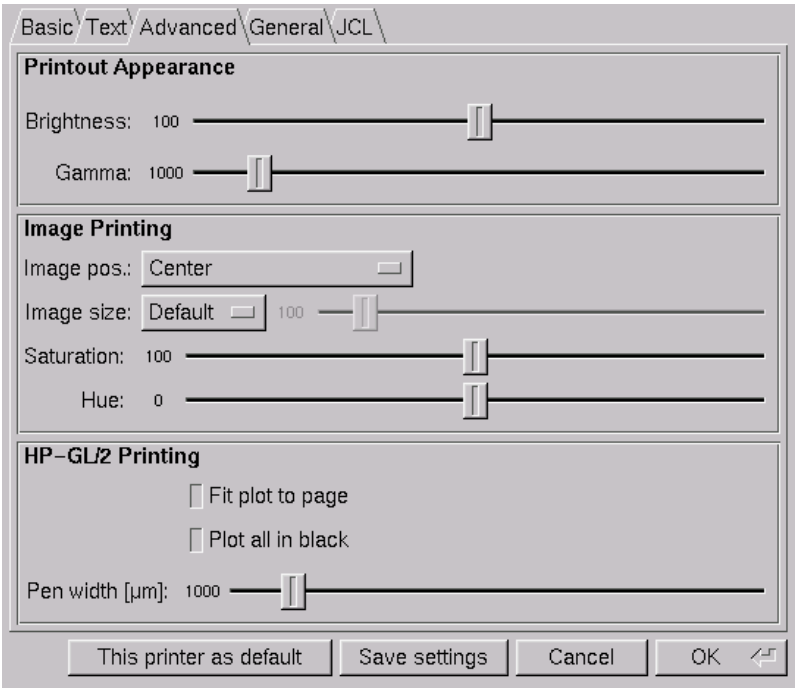


**Figura 5-14. La scheda relativa alle opzioni avanzate di `XPP`**

Questa scheda è divisa in tre parti:

Printout Appearance

I parametri che più influenzeranno la stampa di parti a mezzitoni e di immagini (chiaro-scuri), sono questi due: Brightness e Gamma.

Image Printing

I primi due parametri cambiano la posizione relativa e le dimen-

sioni di un'immagine in rapporto alla pagina; gli ultimi due modificano il modo in cui vengono corretti i colori.

HP-GL/2 Printing

Tre parametri:

- Fit plot to page: Adatta le dimensioni dell'immagine in maniera che corrispondano perfettamente alle dimensioni della carta.
- Plot all in black: Stampa soltanto in bianco e nero.
- Pen width: Modifica la larghezza del pennino "virtuale" che disegna il documento.

Le schede successive contengono opzioni specifiche per le diverse stampanti: colori, risoluzione, dithering, etc.

**Attenzione**

Nell'eventualità che scegliate opzioni incompatibili fra di loro, quelle che causano problemi vengono evidenziate in rosso in modo da permettervi di individuare facilmente il conflitto. In una situazione di questo tipo, quando cliccherete sul pulsante Save settings o OK il programma vi invierà un messaggio di errore - prima di salvare (o usare) la configurazione dovete risolvere il problema.

Quando modificate le opzioni potete scegliere fra due possibilità: salvare la configurazione utilizzando il pulsante Save settings, in maniera tale da poterla utilizzare automaticamente per le prossime stampe, o cliccare sul pulsante OK se tali opzioni sono valide unicamente per la stampa corrente.

# Capitolo 6. Samba

## Che cos'è Samba?

Bene, potreste pensare che *Samba* sia un ballo sudamericano per *GNU/Linux*: che altro potrebbe voler dire? In realtà *Samba* non ha niente a che vedere con il ballo brasiliano, è un demone che offre servizi per client che utilizzano il protocollo SMB (ovvero *Server Message Block*) o il suo successore CIFS (*Common Internet File System*).

## Che cosa può fare Samba?

Un server SMB offre alcuni servizi di condivisione di file e stampanti, esattamente come farebbero *Windows NT* o *LAN Manager*, a client quali *Windows 9x*, *Warp Server*, *Unix* e molti altri. Ma i servizi messi a disposizione da *Samba* non si limitano a questo, ecco una lista di quanto offre:

- un server di nomi NetBIOS (1001/1002);
- un componente client SMB simile all'FTP, che consente di accedere a risorse condivise su *PC* (dischi e stampanti) da macchine *Unix*, *Netware* o che eseguono altri sistemi operativi;
- un'estensione `tar` per i client, che consente di effettuare backup remoti;
- uno strumento di amministrazione da linea di comando, piuttosto limitato, che supporta alcune caratteristiche di *Windows NT* e può essere utilizzato con *Samba*, *Windows NT Workstation* o *Server*;
- l'implementazione di un gateway per la sincronizzazione delle password e di un gateway di stampa fra *Unix* e *Windows NT*;

- la possibilità di accedere a file residenti su sistemi *Windows NT* da sistemi *Unix*.

Per maggiori informazioni su queste caratteristiche, potete consultare il sito *web* di *Samba* (http://samba.org/samba/).

## Installazione di Samba

Se *Samba* è già installato e attivo, digitate `/etc/rc.d/init.d/smb stop`. In caso contrario, sul CDROM di installazione troverete il pacchetto `samba` che potete installare, dopo esservi collegati come `root`, utilizzando il comando `urpmi samba`.

Il comando `rpm -ql samba|less` vi consentirà poi di visualizzare la lista dei file installati con la relativa posizione, il che vi permetterà di vedere dove si trovano i file della sezione `doc` o altri che potrebbero interessarvi.

## Descrizione del file /etc/smb.conf

Questo è il file di testo che vi permette di configurare il server *Samba*.

Il file è suddiviso in sezioni il cui titolo è indicato fra parentesi quadre `[ ]`. Ogni riga di testo che inizia con un **;** o un **#** viene ignorata quando *Samba* viene avviato: questa caratteristica viene spesso usata per aggiungere osservazioni e spiegazioni riguardo le varie sezioni. In tal modo sarà più facile rivedere il file in un secondo momento.

Durante l'installazione del pacchetto viene creato un file `/etc/smb.conf`. Lo utilizzeremo come modello.

Ci sono tre sezioni speciali: `[global]`, `[printers]`, e `[homes]`.

## La sezione `[global]`

Qui sono indicate le impostazioni che si applicano al server nel suo complesso, o che vengono utilizzate come parametri predefiniti per le altre sezioni.

```
[global]
netbios name =          Zeus
netbios aliases =       creation
# se mancano queste due voci, viene utilizzata
# la prima parte del nome DNS

workgroup =             DESIGN
# Samba può far parte di un solo gruppo di lavoro
# per volta

server string =         File server [%v]
# questa voce indica il nome e il numero di versione
# che devono essere mostrati

deadtime =              15
# durata massima del tempo di inattività

auto services =         mario
# questo servizio, sebbene presente nella lista di
# esplorazione, non è disponibile prima che mario si
# sia connesso al server

security = user

# permette la stampa con cups
printing = cups
printcap name = lpstat
load printers = yes

security = user
```

Quest'ultimo parametro può assumere uno di questi quattro valori: `share`, `user`, `server` o `domain`.

Modalità `share`:

> In questa modalità il client invia una password quando richiede una connessione, ma non specifica un nome utente. Questa è la modalità di accesso a file e stampanti predefinita in *Windows 95*.

Può essere cambiata sotto *Windows 95* nella scheda Rete del Pannello di controllo, agendo sulla voce Utenti.

Modalità `user`:

Questa è la modalità di sicurezza raccomandata. Se la scegliete, alla richiesta di una connessione dovrete immettere un nome utente valido e la password relativa.

Modalità `server`:

Questa modalità deriva da quella `user`. Il server *Samba* invia una richiesta di apertura di una sessione a un server delle password.

```
password server = NT_passerv
# dove NT_passerv rappresenta il nome del server NT delle
# password. Possono essere elencati più server.
```

Modalità `domain`:

Questa modalità è quasi identica a quella precedente.

Per usare le ultime tre modalità è necessario “creare” l’utente sul server *Samba*.

```
hosts allow = mario.design.org 192.168.2.
```

Grazie a questa riga, tutti gli utenti che accedono dalla macchina *mario.design.org* e dalla rete IP *192.168.2.* saranno autorizzati a connettersi.

```
hosts deny = 192.168.2.5
```

Questa riga nega l’accesso alla sola macchina *192.168.2.5*. Si potrebbe ottenere lo stesso risultato aggiungendo una voce `EXCEPT` nella sezione `hosts allow`.

```
guest account = pcguest
# se volete aggiungere un account per ospiti ('guest'); deve
# essere inserito in /etc/passwd
```

## Impostare una condivisione

Prima di tutto, create una directory utilizzando il comando `mkdir /home/shared`, e assegnatele i permessi appropriati con `chmod` e `chown`. Ad esempio, `chmod 0777` assegna tutti i diritti su questa directory dal lato del filesystem *Unix*. A questo punto, tuttavia, bisogna assegnare i permessi anche all'interno di *Samba*.

```
[share]
  comment = accesso condiviso garantito a tutti
  path = /home/shared
  browsable = yes
  writable = yes
  create mask = 0750
# significa che il creatore del file ha diritti rwx
# su di esso, il gruppo r-x e tutti gli altri r--

  directory mask = 0750
# stesso significato, ma per le directory

  mangled names = yes
# converte i nomi secondo il formato DOS/Windows,
# cioè otto caratteri per il nome e tre per
# l'estensione.

  preserve case = no
# non conservare maiuscole/minuscole.
```

Un altro esempio di condivisione. Per prima cosa, assicuratevi di caricare il modulo ppa con il comando `modprobe ppa.o`.

```
[zip]
comment = monta e 'smonta' automaticamente il lettore zip
  browsable = yes
  path = /mnt/zip
  root preexec = /bin/mount /dev/sda4 /mnt/zip
  root postexec = /bin/unmount /mnt/zip
```

## La sezione [homes]

Impostando questa sezione potete rendere le home directory di tutti gli utenti definiti sul sistema *GNU/Linux* raggiungibili da una macchina *Windows*, usando il proprio nome utente e la password. Questo è un tipo particolare di condivisione.

```
[homes]
  comment = directory home (le cartelle personali degli utenti)
  browseable = no
  writable = yes
  path = /export/homes/%U
  valid users = %S
```

## La sezione [printers]

Anche questo è un tipo particolare di condivisione.

```
[printers]
   comment = Tutte le stampanti
   browseable = yes
   printable = yes
   public = yes
   writable = no
   create mode = 0700
   print command = lpr-cups -P 0
   lpq command = lpstat -o 0
   lprm command = cancel 0-%j
```

Potete anche stampare da *GNU/Linux* usando una stampante connessa a un *PC* che esegue *Windows*. Per farlo, utilizzate *printtool* e configurate la stampante. Questo creerà una voce nel file `/etc/printcap`. Assicuratevi che la stampante sia stata condivisa da *Windows*.

Ci sono molti altri parametri di configurazione, per maggiori informazioni consultate la pagina di manuale relativa (`man smb.conf`).

Un file di esempio `smb.conf`:

```
[global]
  workgroup = IlMioGruppoDiLavoro
```

```
  server string = (Samba %v)
  #server string = Samba Server
  browseable = yes
  printing = bsd
  printcap name = /etc/printcap
  load printers = yes
  log file = /var/log/samba/log.%m
  max log size = 100
  lock directory = /var/lock/samba
  locking = no
  strict locking = no
  share modes = yes
  security = user
  null passwords = yes
  socket options = TCP_NODELAY

  # Permette al server Samba di diventare il server
  # principale del dominio
  #os level = 33
  #domain master = yes

  # Facilita il riconoscimento dei nomi lunghi
  preserve case = no
  short preserve case = no
  character set = iso8859-1

[homes]
  comment = directory home (le cartelle personali degli utenti)
  preexec = /bin/sh -c 'echo /usr/bin/smbclient -M %m -I %I'&
  browseable = yes
  readonfiltered= no
  writable = yes
  create mode = 0750
  public = yes

["public"]
  comment = "Public"
  path = /public/
  browseable = yes
  hide dot files = yes
  readonfiltered= no
  public = yes
  printable = yes
  #create mode = 0775
  printable = no

["printers"]
  comment = tutte le stampanti
  browseable = yes
  printable = yes
```

```
public = yes
writable = no
create mode = 0700
print command = lpr-cups -P 0
lpq command = lpstat -o 0
lprm command = cancel 0-%j
```

## Test e attivazione

Prima di lanciare i demoni *SMB* e *nmb* , eseguite il comando `testparm`. Questo leggerà il file `/etc/smb.conf` e visualizzerà quali opzioni verranno effettivamente utilizzate.

Quindi, digitate semplicemente `/etc/rc.d/init.d/smb start`. Dovrebbero venire visualizzati questi messaggi: `Starting SMB services: [OK] Starting NMB services: [OK]` Per verificare che il demone sia effettivamente in esecuzione, eseguite `ps aux | grep smbd` o `/etc/rc.d/init.d/smb status`

## Alcuni strumenti

`smbclient`

Questo è un programma simile a *ftp* che vi permette di connettervi a un *PC* che esegue *Windows*.

- Per visualizzare l'elenco delle risorse condivise dal *PC win* attraverso SMB, eseguite `smbclient -L win -N`.
- Lanciate `smbclient //win/work` per connettervi alla directory condivisa `work` sul *PC* chiamato *win*. Potete usare anche il comando `smbclient \\\\win\\work`.
- Per creare un archivio `tar` della directory condivisa `work`, digitate `smbclient //win/work -Tc work.tar`.
- Per stampare un file `lettera.txt` sulla stampante `nome_stampante` connessa al *PC win*, digitate

```
cat lettera.txt | smbclient //win/nome_stampante la_mia_password \
  -N -c "put - mario"
```

Per avere maggiori informazioni, lanciate `man smbclient`.

### smbfs

Per essere in grado di montare/smontare delle cartelle condivise da un *PC Windows* sul vostro sistema *GNU/Linux*, il vostro kernel deve supportare il filesystem `smbfs` (il kernel distribuito con **LinuxMandrake** lo supporta). Potrete quindi utilizzare i programmi `smbmount` e/o `smbumount`. Per esempio: `smbmount "\\win\work" -c 'mount /mnt -u 123 -g 456'`. Questo comando monterà localmente la directory condivisa `work` usando l'UID locale 123 e la GID locale 456.

### smbtar

Permette di creare degli archivi `tar` in modalità remota. Leggete la relativa pagina di manuale per ulteriori informazioni sulle sue varie opzioni.

## `SWAT`, lo strumento grafico di configurazione

*`SWAT`* è un programma di configurazione basato su interfaccia grafica distribuito con il pacchetto *`Samba`*. Per poterlo utilizzare dovete prima attivare (cancellando i caratteri `#` all'inizio della riga) questa riga: `swat stream tcp nowait.400 root /usr/sbin/swat swat` nel file `/etc/inet.conf`, e poi riavviare i servizi `inet` eseguendo `/etc/rc.d/init.d/inetrestart`.

Dal vostro browser *`web`* preferito (possibilmente uno che supporti la grafica), digitate questa URL: http://localhost:901. A questo punto dovrebbe apparire una richiesta di autenticazione (Figura 6-1) dove dovrete immettere il vostro nome utente e la vostra password. L'account `root` dovrebbe essere abbastanza sicuro.

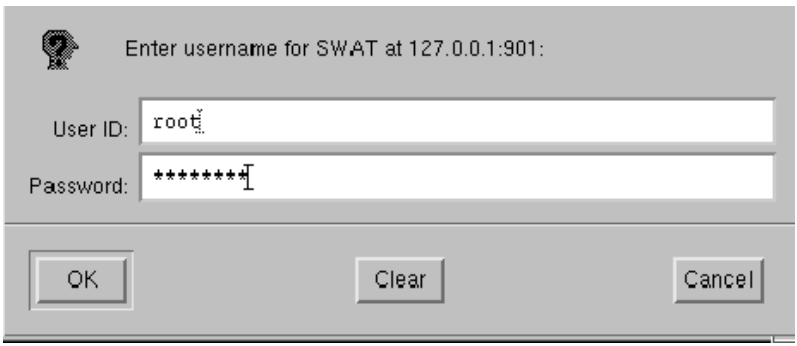


**Figura 6-1. Connessione a `Samba`**

Questa è l'interfaccia grafica per l'amministrazione dei parametri del file `smb.conf`. Dovreste essere in grado di utilizzarla senza problemi,

ma come sempre se avete bisogno di maggiori informazioni potete consultare la relativa pagina di manuale.

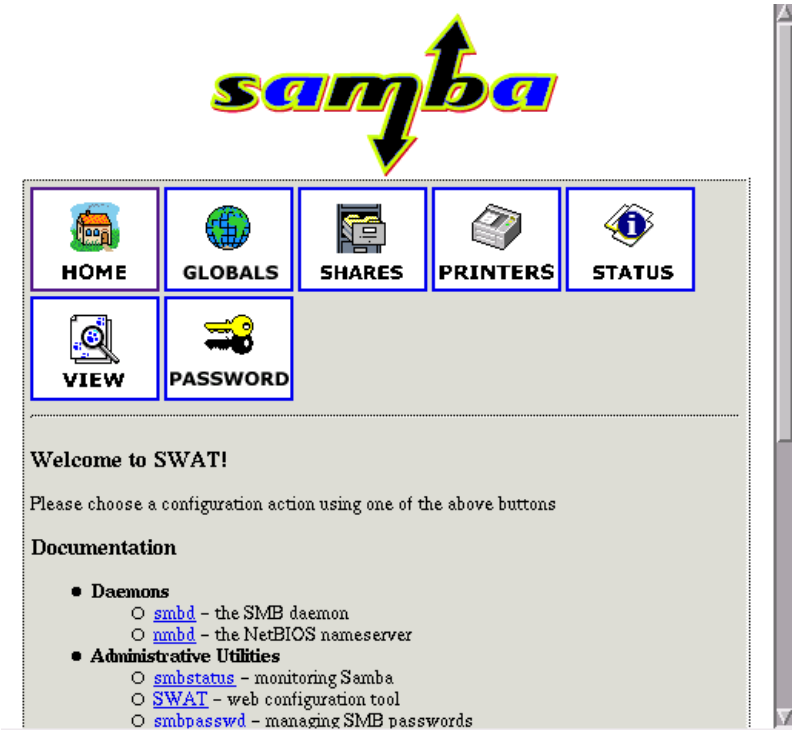


**Figura 6-2. La pagina di benvenuto di `SWAT`**

# Capitolo 7. MSEC – Mandrake Security tools

## Introduzione a MSEC

Essendo *GNU/Linux* ormai utilizzato in moltissimi campi, dai più semplici lavori d'ufficio ai server ad alta accessibilità, si è sentita la necessità di avere a disposizione diversi ***livelli di sicurezza***. Ovviamente le limitazioni necessarie su server altamente protetti non coincidono con i bisogni di una segretaria d'ufficio: dopo tutto, un grosso server pubblico è più esposto ai malintenzionati rispetto a una macchina *GNU/Linux* isolata.

È a questo scopo che è stato progettato il pacchetto *MSEC*. Esso è composto da due parti:

- Script che modificano l'intero sistema in modo da portarlo in uno dei sei livelli di sicurezza forniti da *MSEC*. Questi livelli vanno da una sicurezza assolutamente minima, associata a una grande facilità d'uso, a configurazioni estreme adatte ad applicazioni molto delicate e gestibili solo da esperti.
- Comandi a tempo (*cron jobs*) che eseguono controlli periodici sull'integrità del sistema, in base alle impostazioni del livello di sicurezza, per scoprire eventuali intrusioni o falle di sicurezza nel sistema stesso e avvertirvi di conseguenza.

Va sottolineato che l'utente può anche definire un livello di sicurezza personalizzato, adattando i parametri alle proprie esigenze personali.

## Impostazione del livello di sicurezza

*MSEC* è un RPM di base. Questo significa che, se avete già installato **Linux-Mandrake**, *MSEC* è già presente nel vostro sistema.

La procedura di installazione crea una directory `msec` all'interno della directory `/etc/security`, contenente tutto ciò di cui avete bisogno per proteggere il vostro sistema.

*MSEC* è dotato di una interfaccia grafica, denominata *draksec*: essa è raggiungibile tramite *DrakConf*, e vi permette di cambiare il livello di sicurezza del vostro sistema. Si veda il capitolo "*draksec*: impostazione del livello di sicurezza" nel **Manuale utente**.

È disponibile anche uno strumento da linea di comando, che consente una configurazione più accurata. Entrate nel sistema come `root` e digitate `msec <x>`, dove `<x>` è il livello di sicurezza desiderato, oppure `custom` se volete crearne uno personalizzato. Lo script inizierà a rimuovere tutte le modifiche effettuate dal cambio di livello precedente, e applicherà al vostro sistema le caratteristiche del livello di sicurezza selezionato. Se avete invece scelto `custom`, vi verrà posta una serie di domande per tutte le caratteristiche di sicurezza offerte da *MSEC*; alla fine, le opzioni scelte saranno applicate al vostro sistema.

Ricordatevi che, qualunque sia il livello scelto, la vostra configurazione sarà memorizzata nel file `/etc/security/msec/security.conf`.

## Livello 0

Questo livello deve essere usato con cautela, poiché rende il vostro sistema più facile da usare, ma estremamente vulnerabile. In particolare, non dovreste usare questo livello di sicurezza se rispondete "sì" ad almeno una delle seguenti domande:

- Il mio computer è connesso a *Internet*?
- Il mio computer è connesso in rete con altri computer?
- Questo computer sarà usato da altre persone?
- Sul mio computer ci sono file riservati ai quali non devono accedere altre persone?

- È possibile che io danneggi il sistema a causa della mia incompleta conoscenza di *GNU/Linux*?

Come potete vedere, questo livello di sicurezza non deve essere impostato come livello predefinito, poiché potrebbe risultare molto pericoloso per i vostri dati.

## Livello 1

Il miglioramento principale rispetto al livello 0 è che l'accesso ai dati di qualsiasi utente avviene ora attraverso ***nome utente*** e ***password***. Pertanto il sistema può essere usato da più persone ed è più al sicuro da errori. Comunque questo livello non dovrebbe essere usato su un computer connesso a un modem o a una LAN (*Local Area Network*, ovvero una rete locale).

## Livello 2

Questo livello di sicurezza presenta poche differenze rispetto al precedente; esso introduce principalmente ulteriori controlli e messaggi di avviso. È più affidabile nel caso di sistemi multi-utente.

## Livello 3

Questo è il livello di sicurezza standard, consigliato per un computer che sarà usato per la connessione a *Internet* come client. Viene avviata periodicamente la maggior parte dei controlli di sicurezza, in particolare uno che controlla eventuali porte aperte del sistema. Tuttavia queste porte vengono lasciate aperte e chiunque può accedervi.

Dal punto di vista dell'utente, il sistema in questo caso è un po' più chiuso, e saranno necessarie delle nozioni di base su *GNU/Linux* per compiere alcune operazioni particolari. La sicurezza di questo livello

è paragonabile a quella offerta da una distribuzione **Red Hat** standard o da una qualsiasi **Linux-Mandrake** precedente.

### Livello 4

Con questo livello di sicurezza diventa possibile l'uso del sistema come server; la protezione è ora sufficientemente alta da poter usare il computer come un server che accetti connessioni da molti client. Come impostazione predefinita, saranno ammesse solo connessioni dal computer stesso. I servizi avanzati vengono infatti disabilitati, e l'amministratore di sistema dovrà attivare manualmente, usando i relativi file di configurazione, quelli da lui desiderati. Egli dovrà anche specificare a chi sarà consentito l'accesso al sistema.

I controlli di sicurezza avviseranno l'amministratore di sistema di possibili falle di sicurezza e di eventuali intrusioni nel sistema stesso.

### Livello 5

Le caratteristiche del livello 4 vengono rafforzate, e ora il sistema è completamente chiuso. La sicurezza è al suo massimo. L'amministratore di sistema deve attivare le porte, e accettare le connessioni, per poter dare ad altri computer l'accesso ai servizi offerti dalla propria macchina.

## Caratteristiche dei livelli di sicurezza

Quella che segue è la descrizione delle varie caratteristiche di sicurezza che ogni livello introduce nel sistema. Queste caratteristiche sono di diversi tipi:

- permessi sui file,

- messaggi di avviso,
- controlli di sicurezza periodici:
  - sui file: *suid* `root`, scrivibili, senza proprietario,
  - controllo delle porte attive, NIC in modalità ***promiscua***,
  - file delle password

- connessioni a *X*,
- controllo delle porte in ascolto,
- servizi disponibili,
- password di avvio,
- client autorizzati.

| **Caratteristica \ Livello** | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| controllo generale di sicurezza | | | sì | sì | sì | sì |
| *umask per gli utenti* | 002 | 002 | 002 | 002 | 077 | 077 |
| *umask per* `root` | 002 | 002 | 002 | 002 | 002 | 077 |
| *shell senza password* | sì | | | | | |
| autorizzati a connettersi al display *X* | tutti | locale | locale | nessuno | nessuno | nessuno |
| utenti nel gruppo **audio** | sì | sì | sì | | | |
| `. in $PATH` | sì | sì | | | | |
| avvisi nel file `/var/log/security.log` | | sì | sì | sì | sì | sì |
| avvisi direttamente su *tty* | | | sì | sì | sì | sì |
| avvisi in `syslog` | | | sì | sì | sì | sì |

| **Caratteristica \ Livello** | **0** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|---|
| avvisi inviati via *e-mail* a root | | | sì | sì | sì | sì |
| controllo dei file *suid* root | | | sì | sì | sì | sì |
| controllo MD5 dei file *suid* root | | | sì | sì | sì | sì |
| controllo file scrivibili | | | | sì | sì | sì |
| controllo permessi | | | | sì | sì | sì |
| controllo file gruppo *suid* | | | | sì | sì | sì |
| controllo file senza proprietario | | | | sì | sì | sì |
| controllo promiscuità | | | | sì | sì | sì |
| controllo porte in ascolto | | | | sì | sì | sì |
| controllo integrità file passwd | | | | sì | sì | sì |
| controllo integrità file shadow | | | | sì | sì | sì |
| controllo sicurezza del sistema ogni giorno a mezzanotte | | | | sì | sì | sì |
| registra tutti gli eventi del sistema anche su /dev/tty12 | | | | sì | sì | sì |
| solo root può usare ctrl-alt-del | | | | | sì | sì |
| i servizi sconosciuti sono disabilitati | | | | | sì | sì |
| password di avvio (*GRUB/LILO*) | | | | | sì | sì |
| accetta connessioni da | tutti | tutti | tutti | tutti | locale | nessuno |

Notate che sei dei dieci controlli periodici possono individuare cambiamenti nel sistema. Essi memorizzano la configurazione del sistema al momento dell'ultimo controllo (un giorno prima) in alcuni file nella directory `/var/log/security/`, e vi avvisano di eventuali cambiamenti avvenuti nel frattempo. Questi controlli sono:

- controllo file *suid* `root`
- controllo MD5 dei file *suid* `root`
- controllo file scrivibili
- controllo file gruppo *suid*
- controllo file senza proprietario
- controllo porte in ascolto

## "controllo generale di sicurezza"

1. "NFS filesystems globally exported" ("filesystem NFS esportati globalmente"): viene considerato un'impostazione poco sicura, dato che non esistono restrizioni su chi può montare quei filesystem.
2. "NFS mounts with missing `nosuid`" ("mount NFS senza `nosuid`"): questi filesystem sono esportati senza l'opzione `nosuid`, che impedisce ai programmi *suid* di funzionare sul sistema.
3. "Host trusting files contain + sign" ("i file dei permessi degli host contengono il segno +"): significa che uno dei file `/etc/hosts.equiv`, `/etc/shosts.equiv` e `/etc/hosts.lpd` contiene nomi di computer a cui è consentita la connessione senza che venga effettuata un'autenticazione completa.
4. "Executables found in the aliases files" ("trovati eseguibili nei file degli alias"): è un avviso che riporta i nomi di eventuali es-

eguibili incontrati nei due file `/etc/aliases` e `/etc/postfix/aliases`.

### "`umask` per gli utenti"

Imposta semplicemente *`umask`* per gli utenti normali al valore corrispondente al livello di sicurezza.

### "`umask` per `root`"

La stessa cosa, ma per `root`.

### "`shell` senza password"

L'accesso alle console è consentito senza richiedere una password.

### "autorizzati a connettersi al display `X`"

1. `tutti`: chiunque da qualsiasi luogo può aprire una finestra *`X`* sul vostro schermo.
2. `locale`: solo le persone connesse a *`localhost`* possono aprire una finestra *`X`* sul vostro schermo.
3. `nessuno`: nessuno può farlo.

### "utenti nel gruppo `audio`"

Tutti gli utenti sono membri dei gruppi `audio`, `urpmi` e `cdrom`. Questo significa che tutti gli utenti hanno accesso ad alcuni privilegi speciali riguardanti la scheda audio, i pacchetti, etc.

### “. in $PATH”

L’elemento . viene aggiunto nella variabile d’ambiente $PATH, agevolando l’esecuzione di programmi situati nella directory di lavoro attuale (ciò rappresenta, entro certi limiti, anche una falla di sicurezza).

### “avvisi nel file /var/log/security.log”

Tutti gli avvisi generati da *MSEC* vengono registrati nel file di nome /var/log/security.log.

### “avvisi direttamente su tty”

Tutti gli avvisi generati da *MSEC* vengono stampati direttamente sulla console attuale.

### “avvisi in syslog”

Gli avvisi di *MSEC* vengono redirezionati verso il servizio *syslog*.

### “avvisi inviati via e-mail a root”

Gli avvisi generati da *MSEC* vengono anche inviati via *e-mail* a root.

### “controllo dei file suid root”

Controlla se nel sistema sono stati cancellati o creati nuovi file *suid* root. Se questo è avvenuto, viene generato un avviso con la lista dei file in questione.

### “controllo MD5 dei file `suid root`”

Controlla la firma MD5 di ogni file *`suid`* `root` presente nel sistema. Se la firma è cambiata, significa che è stata fatta una modifica al programma, forse una back door. Viene quindi generato un avviso.

### “controllo file scrivibili”

Controlla se i file del sistema sono modificabili da chiunque. Se è così, genera un avviso contenente la lista dei file a rischio.

### “controllo permessi”

Controlla i permessi di alcuni file speciali come `.netrc` o i file di configurazione degli utenti. Controlla anche i permessi delle directory base degli utenti. Se questi permessi sono troppo deboli, o se i proprietari sono insoliti, viene generato un avviso.

### “controllo file gruppo `suid`”

Controlla se nel sistema sono stati cancellati o creati nuovi file del gruppo *`suid`*. Se questo è avvenuto, viene generato un avviso con la lista dei file in questione.

### “controllo file senza proprietario”

Questo controllo cerca i file i cui utenti o gruppi proprietari sono sconosciuti al sistema. Se simili file vengono trovati, viene automaticamente impostato come proprietario l’utente/gruppo `nessuno`.

### “controllo promiscuità”

Questo controllo esamina tutte le schede *Ethernet* per sapere se esse sono in modalità "promiscua". Questa modalità permette alla scheda di intercettare tutti i pacchetti da essa ricevuti, anche quelli non diretti a lei. Questo può significare che uno ***sniffer*** è in funzione sulla vostra macchina. **Va sottolineato che questo controllo è impostato per essere eseguito a intervalli di un minuto.**.

## "controllo porte in ascolto"

Genera un avviso con un elenco di tutte le porte in ascolto.

## "controllo integrità file `passwd`"

Controlla che ogni utente abbia una password (e non una vuota o facile da indovinare), e verifica che essa sia mascherata.

## "controllo integrità file `shadow`"

Controlla che ogni utente contenuto nel file `shadow` abbia una password (e non una vuota o facile da indovinare).

## "controllo sicurezza del sistema ogni giorno a mezzanotte"

Tutti i controlli precedenti saranno effettuati ogni giorno a mezzanotte. Questo meccanismo si basa sull'aggiunta di uno script *cron* nel file `crontab`.

## "i servizi sconosciuti sono disabilitati"

Tutti i servizi non compresi in `/etc/security/msec/init-sh/server.4` per il livello 4 o `server.5` per il livello 5 saranno disabilitati. Non saranno eliminati, ma, semplicemente, non vengono avviati durante il caricamento di un *runlevel*. Se doveste avere bisogno di alcuni di essi, vi basterà aggiungerli nuovamente con il programma *chkconfig* (potreste anche doverli riavviare con gli script `init` in `/etc/rc.d/init.d`).

## "password di avvio"

### Si hanno due diversi comportamenti, in base al bootloader da voi utilizzato:

*GRUB*

> All'avvio, *GRUB* vi chiederà la password solo se passate manualmente dei parametri al kernel. Questo permette al vostro sistema di riavviarsi da solo, senza necessità di intervento da parte di un operatore, impedendo comunque a persone non autorizzate di riavviare la macchina in modo insolito (ad esempio in modalità `fail safe`).

*LILO*

> Vi permette di impostare una password per *LILO*. Impedisce ad altre persone (inesperte) di riavviare il sistema ma, d'altra parte, il sistema non sarà in grado di riavviarsi da solo.

## "accetta connessioni da"

1. `tutti`: qualsiasi computer può connettersi alle porte aperte.
2. `locale`: solo *localhost* può connettersi alle porte aperte.

3. `nessuno`: nessun computer può connettersi alle porte aperte.

# Capitolo 8. Organizzazione della struttura del filesystem

Al giorno d'oggi, un sistema *Unix* può raggiungere dimensioni notevoli, davvero notevoli. Questo è particolarmente vero con *GNU/Linux*: la grande quantità di software disponibile lo renderebbe un sistema ingestibile, se non ci fossero delle linee guida per l'allocazione dei file nella struttura gerarchica del filesystem.

Lo standard riconosciuto in questo campo è il FHS (*Filesystem Hierarchy Standard*), che ha raggiunto la versione 2.1 al momento della stesura di questo manuale. Il documento che descrive lo standard è disponibile su *Internet* in vari formati all'URL pathname (http://www.pathname.com/fhs/). Questo capitolo ne costituisce solo un breve riassunto, ma dovrebbe essere sufficiente per capire in quale directory cercare (o collocare) un dato file.

## Dati condivisibili e non, statici e variabili

I dati su un sistema *Unix* possono essere classificati secondo questi due criteri. Probabilmente avrete già capito cosa significano: i dati condivisibili sono i dati che possono essere gestiti in comune da un numero di macchine distribuite in una rete, mentre i dati non condivisibili non possono esserlo. I dati statici non devono essere modificati durante l'uso normale, mentre i dati variabili possono essere modificati. Mentre esploreremo la struttura gerarchica del filesystem, classificheremo le varie directory secondo queste due categorie.

Si noti che queste classificazioni sono solamente consigliate. Non siete obbligati ad adottarle, tuttavia adottarle semplificherà notevolmente la gestione del vostro sistema. Si noti ancora che che la distinzione statico/variabile si applica solo all'uso di un sistema non alla sua configurazione. Nell'installazione di un programma si dovranno necessariamente modificare directory "normalmente" statiche, ad esempio `/usr`.

## La directory radice: /

La directory radice (ingl. *root*) contiene l'intera gerarchia del sistema. Non può essere classificata dal momento che le sue sotto-directory possono contenere dati condivisibili oppure no, statici o variabili. Ecco una lista delle principali directory e sotto-directory, con la relativa classificazione:

- `/bin`: programmi essenziali. Questa directory contiene i comandi di base che saranno utilizzati da tutti gli utenti e che sono necessari per le normali operazioni del sistema: `ls`, `cp`, *login*, etc. Statica, non condivisibile.
- `/boot`: contiene i file richiesti dal programma che gestisce il boot di *GNU/Linux* (*GRUB* o *LILO* per le piattaforme **Intel**). Questa directory può contenere il kernel oppure no, se non si trova qui deve trovarsi nella directory `root`. Statica, non condivisibile.
- `/dev`: contiene i file dei dispositivi di sistema (`dev` per *DEVices*). Statica, non condivisibile.
- `/etc`: questa directory contiene tutti i file di configurazione specifici per la macchina. Statica, non condivisibile.
- `/home`: contiene le directory personali degli utenti del sistema. Questa directory può essere condivisibile oppure no (alcuni network molto grandi la rendono condivisibile attraverso NFS). Variabile, condivisibile.
- `/lib`: questa directory contiene le librerie essenziali al funzionamento del sistema e, in `/lib/modules`, i moduli del kernel. Tutte le librerie richieste dai programmi nelle directory `/bin` e `/sbin` devono trovarsi in questa directory, come pure il linker `ld.so`. Statica, non condivisibile.
- `/mnt`: directory contenente i punti di mount per i filesystem temporanei. Variabile, non condivisibile.

- `/opt`: questa directory contiene i pacchetti non necessari alle operazioni di sistema. Si raccomanda di mettere i file statici (programmi, librerie, documentazione, ecc) di questi pacchetti in `/opt/nome_del_pacchetto`, e i loro file di configurazione specifici per la macchina in `/etc/opt`.
- `/root`: directory personale dell'Onnipotente. Variabile, non condivisibile.
- `/usr`: vedi la prossima sezione. Statica, non condivisibile.
- `/sbin`: contiene i programmi di sistema necessari all'avvio del sistema, utilizzabile solo da `root`. Un utente normale li può eseguire, ma non andrà molto lontano. Statica, non condivisibile.
- `/tmp`: directory che contiene i file temporanei creati da alcuni programmi. Variabile, non condivisibile .
- `/var`: directory per i dati che possono essere modificati in tempo reale dai programmi (ad esempio, il server della posta elettronica, programmi di controllo, il server di stampa, etc.). Tutti i `/var` sono variabili, ma le sue varie sotto-directory possono essere condivisibili o meno.

## `/usr`: quella grande

La directory `/usr` è la directory destinata ad accogliere la maggior parte dei programmi installati sul sistema. Tutti i programmi in questa directory non devono essere necessari all'avvio del sistema o alla sua manutenzione, dal momento che la gerarchia `/usr` molto spesso è un filesystem separato. Date le sue dimensioni, spesso notevoli, `/usr` ha una propria gerarchia di directory e sotto-directory. Ne citiamo solo alcune:

- `/usr/X11R6`: l'intera gerarchia di *X Window System*. Tutti i programmi richiesti per il funzionamento di *X* (inclusi i server *X*) e le librerie relative devono trovarsi in questa directory. La directory

`/usr/X11R6/lib/X11` contiene tutte le impostazioni di *X* che non variano da una macchina all'altra. Le impostazioni specifiche per ogni macchina si trovano invece in `/etc/X11`.

- `/usr/bin`: questa directory contiene la maggior parte dei programmi installati sul vostro sistema. **Qualsiasi** programma che non è necessario alla manutenzione e/o all'amministrazione del sistema deve trovarsi in questa directory, tranne i programmi personali che andrebbero installati in `/usr/local`.
- `/usr/lib`: questa directory contiene tutte le librerie necessarie all'esecuzione dei programmi che si trovano in `/usr/bin` e `/usr/sbin`. C'è anche collegamento simbolico alla directory `/usr/lib/X11` che si riferisce alla directory che contiene le librerie di *X Window System*: `/usr/X11R6/lib` (posto che *X Window System* sia installato!).
- `/usr/local`: questa è la directory dove andrebbero installate le applicazioni personali. Il programma di installazione creerà la gerarchia necessaria: `lib/`, `man/`, etc.
- `/usr/share`: questa directory contiene tutti i dati richiesti dalle applicazioni che si trovano in `/usr` e quelli indipendenti dall'architettura della macchina. Fra le altre cose, in questa directory si trovano le informazioni sul fuso orario e sulla localizzazione del sistema (`zoneinfo` e `locale`).

Ci sono anche le directory `/usr/share/doc` e `/usr/share/man` che contengono, rispettivamente, la documentazione del sistema e le pagine di manuale del sistema.

## `/var`: dati modificabili durante l'uso

La directory `/var` contiene tutti i dati dei programmi in esecuzione sul sistema. Al contrario dei dati di lavoro che si trovano in `/tmp`, questi dati devono essere mantenuti integri nell'eventualità di un riavvio. Ci sono molte sotto-directory, e alcune sono molto utili:

- `/var/log`: contiene i file registro del sistema (*system log files*);
- `/var/spool`: contiene i file di lavoro dei demoni di sistema. Ad esempio: `/var/spool/lpd` contiene i file di lavoro del server di stampa e `/var/spool/mail` i file di lavoro del server di posta elettronica (cioè tutti i messaggi in arrivo e in partenza dal sistema).
- `/var/run`: questa directory è usata per tenere traccia di tutti i processi usati dal sistema, in maniera che si possa agire su questi processi nell'eventualità di un cambio nel ***runlevel*** di sistema (si veda il capitolo *I file di avvio del sistema: init "System V"*, pag. 155).

## `/etc`: file di configurazione

La directory `/etc` è una della directory essenziali in qualunque sistema *Unix*. Contiene tutti i principali file di configurazione del sistema. Non cancellatela per risparmiare spazio! Un altro suggerimento: se si vuole estendere la struttura gerarchica del filesystem su più partizioni, si ricordi che la directory `/etc` non deve essere messa su una partizione separata, è necessaria all'inizializzazione del sistema.

Alcuni file importanti sono:

- `passwd` e `shadow`: questi due file sono file di testo che contengono tutti gli utenti del sistema e le loro parole d'ordine in forma criptata. `shadow` è presente solo nel caso che si utilizzino le password di tipo *shadow*, che è l'opzione predefinita del programma di installazione;
- `inittab`: è il file di configurazione del programma `init`, che gioca un ruolo fondamentale nell'avvio del sistema, come si vedrà più avanti;
- `services`: questo file contiene una lista dei servizi di rete esistenti;
- `profile`: è il file di configurazione della *shell*, ma alcune *shell* ne usano altri. Ad esempio, *Bash* usa `bashrc`;

- `crontab`: file di configurazione del comando `cron`, il comando responsabile dell'esecuzione periodica di determinati programmi.

Esiste anche un certo numero di sotto-directory per programmi che hanno bisogno di un gran numero di file per la loro configurazione. È il caso di *X Window System*, che utilizza la directory `/etc/X11`.

# Capitolo 9. Filesystem e punti di mount

Il modo migliore per comprendere “come funziona” è fare riferimento a un caso pratico, cosa che faremo in questa sezione. Supponiamo che abbiate acquistato un disco rigido nuovo di zecca, ancora senza partizioni. La vostra partizione **Linux-Mandrake** è piena fino a scoppiare e, invece di ricominciare da capo, decidete di spostare un’intera sezione dell’albero delle directory nel nuovo disco rigido. Dato che il nuovo disco è molto grande, decidete di spostare la directory di maggiori dimensioni: `/usr`. Ma prima serve un po’ di teoria.

## Principi

Come abbiamo già detto nella **Guida all’installazione**, ogni disco rigido è suddiviso in diverse partizioni, e ognuna di queste contiene un filesystem; *Windows* assegna una lettera a ognuno di questi filesystem (o meglio: solo a quelli che riconosce), invece *GNU/Linux* ha un’unica struttura ad albero delle directory e ogni filesystem è ***montato*** in un punto della struttura dell’albero delle directory.

*Windows* ha bisogno di un “disco C:”, *GNU/Linux* invece ha bisogno di montare la radice del suo albero di file e directory (`/`) e lo fa sulla partizione che contiene il ***filesystem di root***. Una volta montata la radice, è possibile montare altri filesystem su altri ***punti di mount*** nella struttura ad albero. Qualunque directory sotto la directory radice può fungere da punto di mount.

Questo consente una grande flessibilità nelle configurazioni possibili. Generalmente, ad esempio, nei server *web* si dedica un’intera partizione alla directory che ospita i dati del server *web*, la directory che li ospita normalmente è `/home/httpd` che di conseguenza funge da punto di mount per la partizione. In Figura 9-1 e in Figura 9-2 potete vedere la situazione del sistema prima e dopo aver montato il filesystem.

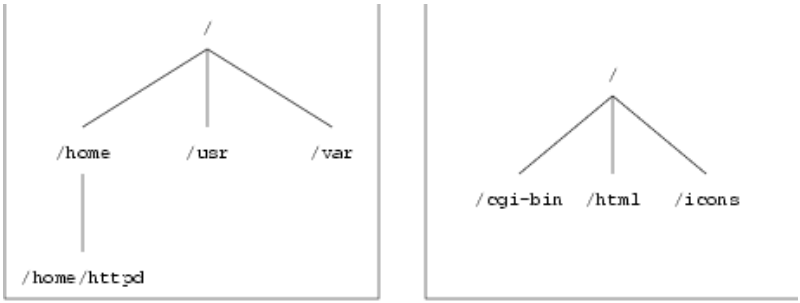


**Figura 9-1. Un filesystem non ancora montato**

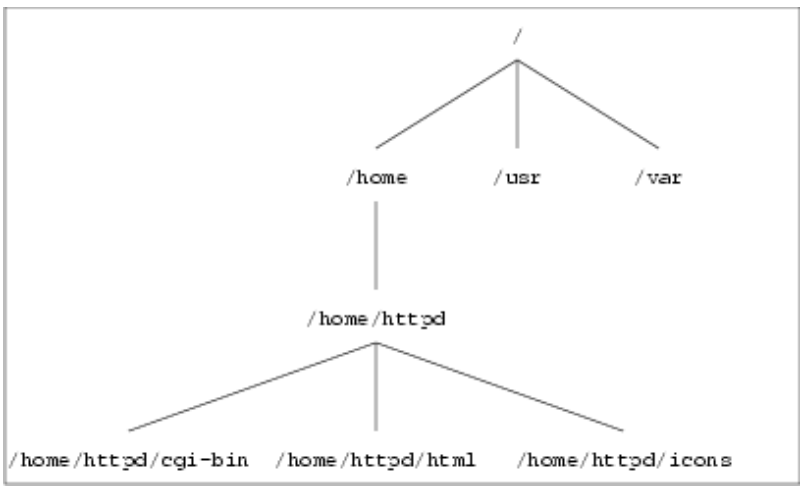


**Figura 9-2. Il filesystem ora è montato**

Come potete immaginare, questa caratteristica offre diversi vantaggi: la struttura ad albero rimane sempre la stessa, sia che stia su un solo filesystem, sia che si estenda su svariate dozzine[1]. Quando lo spa-

zio comincia a mancare, è sempre possibile spostare fisicamente parti importanti della struttura delle directory, cosa che stiamo per fare in questa sezione.

Ci sono due cose che dovete sapere sui punti di mount:

- la directory che funge da punto di mount deve esistere,
- questa directory **dovrebbe essere vuota**: se la directory scelta come punto di mount contiene già dei file, questi verranno "nascosti" dal filesystem appena montato, ma non cancellati.

## Partizionamento di un disco rigido e formattazione di una partizione

Per quanto riguarda i principi cui abbiamo fatto riferimento sopra, e nei limiti di quanto ci interessa in questa sede, ci sono due cose da notare: un disco rigido è diviso in partizioni e ognuna di esse ospita un filesystem. Al momento il vostro disco rigido non ha nessuna di queste due cose, per cui dovete cominciare col partizionarlo. Per fare ciò dovete essere `root`.

Come prima cosa, dovete sapere il "nome" del vostro disco rigido, cioè quale file lo designa. Se supponiamo che lo installiate come slave sulla vostra interfaccia IDE primaria, il nome sarà `/dev/hdb`[2].

## I comandi `mount` e `umount`

1. *GNU/Linux* può gestire fino a 64 filesystem montati contemporaneamente.

2. Come trovare il nome di un disco è spiegato nella **Guida all'installazione**.

Adesso che il filesystem è stato creato potete montare la partizione. Inizialmente, e ovviamente, sarà vuota. Il comando per montare il filesystem è `mount` , la sua sintassi è come segue:

```
mount [opzioni] <-t tipo> [-o opzioni di mount] <dispositivo> <punto di mount>
```

In questo caso, vogliamo montare la nostra partizione sulla directory `/mnt` (o su qualunque altro punto di mount abbiate scelto – ma non dimenticate che deve esistere!); il comando per montare la partizione che abbiamo appena creato è:

```
$ mount -t ext2 /dev/hdb1 /mnt
```

L'opzione `-t` è usata per specificare il tipo di filesystem che la partizione deve ospitare. I filesystem che si incontrano più spesso sono: `ext2` (il filesystem di *GNU/Linux*), `vfat` (per tutte le partizioni *DOS*/*Windows*: FAT 12, 16 o 32) e `iso9660` (un filesystem per i CDROM).

L'opzione `-o` è usata per specificare una o più opzioni per montare la partizione; queste opzioni dipendono dal tipo di filesystem usato. Si faccia riferimento alla pagina di manuale `man 8 mount` per avere maggiori informazioni.

Una volta montata la nuova partizione dovete copiarci l'intera directory `/usr`:

```
$ (cd /usr && tar cf - .) | (cd /mnt && tar xpvf -)
```

Ora che i file sono stati copiati potete smontare la partizione. Il comando per compiere questa operazione è `umount`. La sintassi è facile:

```
umount <mount point|device>
```

Quindi, per smontare la partizione, digitate:

```
$ umount /mnt
```

oppure:

```
$ umount /dev/hdb1
```

Dal momento che la partizione deve "diventare" la directory `/usr`, il sistema deve venirne a conoscenza. Per far ciò, dobbiamo modificare

## Il file `/etc/fstab`

Il file `/etc/fstab` rende possibile montare automaticamente, all'avvio del sistema, alcuni filesystem. Contiene una serie di righe che descrivono i vari filesystem, i loro punti di mount e altre opzioni. Quello che segue è un esempio di file `/etc/fstab`:

```
/dev/hda1   /           ext2    defaults        1 1
/dev/hda5   /home       ext2    defaults        1 2
/dev/hda6   swap        swap    defaults        0 0
/dev/fd0    /mnt/floppy auto    sync,user,noauto,nosuid,nodev,unhide 0 0
/dev/cdrom  /mnt/cdrom  auto    user,noauto,nosuid,exec,nodev,ro 0 0
none        /proc       proc    defaults        0 0
none        /dev/pts    devpts  mode=0622       0 0
```

Ogni riga contiene, nell'ordine:

- il dispositivo che ospita il filesystem,
- il punto di mount,
- il tipo di filesystem,
- le opzioni per montare il dispositivo,
- il ***flag*** di `dump`, un programma di backup,
- l'ordine in cui `fsck` (*`FileSystem CheckK`*) controllerà i filesystem.

E, sorpresa sorpresa, c'è sempre una voce per il filesystem root. Le partizioni di *`swap`* sono speciali perché non sono visibili nella struttura delle directory, il loro campo nel punto di mount presenta la parola chiave `swap`. Torneremo a parlare di `/proc` in maggior dettaglio.

Ma torniamo al nostro compito. Avete spostato l'intera gerarchia `/usr` nella directory `/dev/hdb1`, e quindi volete che sia montata all'avvio, quello che dovete fare è aggiungere una voce al file:

```
/dev/hdb1 /usr ext2 defaults 1 2
```

Adesso la partizione verrà montata all'avvio e, se necessario, verrà anche controllata.

Ci sono due opzioni speciali: `noauto` e `user`. L'opzione `noauto` specifica che il filesystem non deve essere montato all'avvio, quindi dovrà essere montato con un comando esplicito. L'opzione `user` specifica che qualunque utente può montare e smontare il filesystem. Come avrete notato queste due opzioni sono usate per i drive CDROM e per i floppy, ma valgono anche per i dischi rigidi. Ci sono anche altre opzioni, infatti il file `/etc/fstab` è corredato da una sua pagina di manuale: `man 5 fstab`.

L'ultimo, ma non meno importante, vantaggio nell'uso di questo file è che semplifica la sintassi del comando `mount`: per montare un filesystem descritto nel file basta rifarsi al punto di mount oppure al dispositivo. Ad esempio, per montare un floppy disk si può digitare:

```
$ mount /mnt/floppy
```

oppure:

```
$ mount /dev/fd0
```

Ora portiamo a termine il compito di spostare una partizione: avete copiato la gerarchia della directory `/usr` e modificato il file `/etc/fstab` in maniera tale che la nuova partizione sia montata all'avvio. Al momento, però, i file della vecchia directory `/usr` sono ancora lì! Dovrete, quindi, eliminarli per liberare un po' di spazio (cosa che, d'altronde, era il vostro scopo iniziale!), per far ciò è necessario:

- cancellare tutti i file della directory `/usr` (della "vecchia" /usr, dato che la "nuova" non è stata ancora montata!): `rm -Rf /usr/*`;
- montare la "nuova" `/usr`: `mount /usr`

e avete finito. Non c'è più lavoro amministrativo da fare e dovreste uscire dall'account di `root`.

## Una nota sull'opzione `supermount`

I kernel più recenti, come quelli forniti con **Linux-Mandrake**, hanno una caratteristica molto interessante per gli utenti che usano frequentemente floppy disk e CDROM. Comunque sia stato installato nel corso del processo di installazione, `supermount` provvede a montare e smontare automaticamente i supporti al momento in cui sono inseriti o rimossi. Questo comportamento è molto comodo, poiché non è più necessario digitare il comando `mount`, o `umount`, ogni volta che si inserisce o si estrae un supporto rimuovibile.

Nel caso vogliate cambiare questo comportamento, tornando al caro vecchio montaggio manuale dei supporti, dovete dare il comando, da `root`

```
supermount -i disable
```

Se, invece, volete attivare questa opzione date il comando:

```
supermount -i enable
```

Per maggiori informazioni sul comando `supermount` consultate la sua pagina di manuale (`man supermount`).

# Capitolo 10. Il filesystem di GNU/Linux: ext2fs (EXTended 2 FileSystem)

Se avete letto il **Manuale utente** probabilmente avrete già incontrato i concetti di proprietà dei file e di permessi di accesso, ma per capire davvero il ***filesystem*** di *GNU/Linux* è necessario ridefinire il concetto stesso di file. Uno dei motivi è che:

## Tutto è un file

Qui "tutto" significa **davvero** tutto: un disco rigido, una partizione sul disco, una porta parallela, una connessione a un sito *web*, una scheda *Ethernet*, tutte queste cose sono dei file. Persino le directory sono file. *GNU/Linux* è in grado di riconoscere molti tipi di file oltre ai file e alle directory standard. Notate che con tipo di file qui non ci riferiamo al **contenuto** di un file: per *GNU/Linux* e qualsiasi altro sistema *Unix*, un file, che si tratti di un'immagine GIF, di un file binario o di qualsiasi altro tipo, non è altro che una sequenza di byte. L'operazione di distinzione dei file in base al loro contenuto viene lasciata alle applicazioni.

Come ricorderete, quando impartite il comando `ls -l`, il carattere subito prima dei diritti di accesso identifica il tipo di file. Abbiamo già visto due tipi di file: file normali (`-`) e directory (`d`). Se cominciate a esplorare la struttura delle directory del vostro sistema, ed elencate il contenuto delle directory, troverete anche qualcuno dei seguenti:

1. ***File in modalità a caratteri*** Questi sono file di sistema speciali (come `/dev/null`, che abbiamo già incontrato), o periferiche (porte parallele o seriali), che hanno in comune il fatto che il loro contenuto (se ne hanno) non è ***bufferizzato*** (in altre parole, non sono conservati in memoria). File di questo tipo sono identificati per mezzo della lettera `'c'`.

2. ***File in modalità a blocchi*** Questi file sono periferiche, e, a differenza dei file carattere, il loro contenuto **è** bufferizzato. File che rientrano in questa categoria sono, ad esempio, dischi rigidi, partizioni su questi ultimi, drive floppy, lettori CDROM e così via. I file `/dev/hda`, `/dev/sda5` sono altrettanti esempi di File in modalità a blocchi. Nella visualizzazione dei risultati di `ls -l`, questi file sono identificati con la lettera `'b'`.

3. ***Collegamenti simbolici*** (noti anche come "link simbolici") Questi file sono molto diffusi, e ampiamente utilizzati nella procedura di avvio del sistema in **Linux-Mandrake** (si veda in proposito il capitolo *I file di avvio del sistema: init "System V"*, pag. 155). Come suggerisce il nome, il loro scopo è quello di collegare i file in modo simbolico, il che significa che file di questo tipo possono riferirsi a un file esistente oppure no; questo argomento verrà spiegato più avanti in questo capitolo. Molto spesso sono chiamati ***"soft link"***, e sono identificati con una `'l'`.

4. ***Pipe con nome*** Nel caso ve lo stiate chiedendo, sì, queste pipe sono molto simili a quelle usate con i comandi *shell*, ma la loro particolarità è quella di avere dei nomi. Continuate a leggere per saperne di più. Sono molto rare, comunque, ed è assai improbabile che voi ne incontriate una durante le vostre esplorazioni del sistema; in caso questo succedesse, la lettera che le identifica è la `'p'`. Per saperne di più, date uno sguardo a *Pipe "anonime" e pipe con nome*, pag. 139.

5. ***Socket*** Questo è il tipo di file usato per tutte le connessioni di rete. Soltanto alcuni di questi hanno un nome, tuttavia. Soprattutto, esistono diversi tipi di socket, ma solo un tipo può avere un nome nel filesystem. Tuttavia, questo argomento va ben oltre gli obiettivi di questo manuale. File di questo tipo vengono identificati con la lettera `'s'`.

Ecco un esempio di ciascun file:

```
$ ls -l /dev/null /dev/sda  /etc/rc.d/rc3.d/S20random /proc/554/maps \
  /tmp/ssh-queen/ssh-510-agent
crw-rw-rw-     1 root     root       1,   3 mag  5  1998 /dev/null
brw-rw----     1 root     disk       8,   0 mag  5  1998 /dev/sda
lrwxrwxrwx     1 root     root           16 dic  9 19:12 /etc/rc.d/rc3.d/S20random
-> ../init.d/random*
pr--r--r--     1 queen    queen           0 dic 10 20:23 /proc/554/maps|
srwx------     1 queen    queen           0 dic 10 20:08 /tmp/ssh-queen/ssh-
510-agent=
$
```

Si consideri, inoltre, che *ext2fs*, come tutti gli altri filesystem *Unix*, archivia i file, qualunque sia il loro tipo, in una tabella ***inode***. . Una caratteristica particolare di *ext2fs* è che un file non è identificato per mezzo del suo nome, ma con un numero di inode. Non tutti i file, in effetti, hanno un nome. I nomi sono soltanto la conseguenza di un principio più generale:

## Collegamenti

Il modo migliore per capire cosa sta alla base del concetto di collegamento è quello di fare un esempio. Creiamo perciò un file di tipo normale:

```
$ pwd
/home/queen/example
$ ls
$ touch a
$ ls -il a
  32555 -rw-rw-r--    1 queen       queen              0 dic 10 08:12 a
```

L'opzione `-i` del comando `ls` visualizza il numero di inode, che si trova sul primo campo dell'output. Come potete vedere, prima che noi creassimo il file `a`, non esistevano altri file nella directory. L'altro campo che ci interessa è il terzo, quello che riporta il contatore di link del file.

Il comando `touch a`, in effetti, può essere scomposto in due azioni separate:

- la creazione di un inode, al quale il sistema ha assegnato il numero 32555, e il cui tipo è quello di un file normale,
- la creazione di un collegamento a tale inode, il cui nome è `a`, all'interno della directory corrente, `/home/queen/example`. Il file `/home/queen/example/a`, pertanto, è un collegamento all'inode numero 32555 e, al momento, è l'unico collegamento esistente: il contatore mostra uno.

Ma adesso, se digitiamo:

```
$ ln a b
$ ls -il a b
  32555 -rw-rw-r--    2 queen       queen             0 dic 10 08:12 a
  32555 -rw-rw-r--    2 queen       queen             0 dic 10 08:12 b
$
```

abbiamo creato un altro collegamento allo stesso inode. Come potete vedere, non abbiamo creato un file col nome `b`, abbiamo soltanto, invece, aggiunto un altro collegamento all'inode numero 32555 nella stessa directory, e lo abbiamo chiamato `b`. Potete infatti vedere dall'output del comando `ls -l` che il contatore di collegamenti per questo inode adesso ne riporta due, e non più uno soltanto.

Ora, se noi digitiamo:

```
$ rm a
$ ls -il b
  32555 -rw-rw-r--    1 queen       queen             0 dic 10 08:12 b
$
```

notiamo che, malgrado il "file originale" sia stato cancellato, l'inode esiste ancora. Ma ora l'unico collegamento che fa riferimento a quest'ultimo è il file di nome `/home/queen/example/b`.

Un inode, pertanto, è collegato (***linked***) al filesystem se, e soltanto se, esiste un nome in almeno una directory[1] che vi faccia riferimento.

1. Se tale directory si trova sullo stesso filesystem dell'inode.

Anche le directory vengono archiviate in inode, ma il loro contatore di collegamenti, a differenza di tutti gli altri tipi di file, è il numero di sotto-directory contenuto al loro interno. Esistono almeno due collegamenti per ogni directory: la directory stessa (`.`) e quella immediatamente superiore (`..`).

Le connessioni di rete costituiscono esempi tipici di file che non hanno collegamenti (cioè, che non hanno nome): non vedrete mai il file corrispondente alla vostra connessione al sito linux-mandrake (www.linux-mandrake.com) nell'albero delle directory, qualunque sia la directory che state utilizzando. Allo stesso modo, quando usate una ***pipe*** in una *shell*, il file che corrisponde alla pipe esiste, ma non è collegato.

## Pipe "anonime" e pipe con nome

Torniamo all'esempio delle pipe, poiché è molto interessante e costituisce anche un buon esempio del concetto di collegamento. Quando usate una pipe in una riga di comando, la *shell* crea la pipe per voi e agisce in maniera tale che il comando che precede la pipe scrive su di essa, mentre il comando che segue legge da essa: Tutte le pipe, che siano anonime (come quelle impiegate dalla *shell*) o provviste di nome (si veda più avanti), si comportano come delle FIFO (*First In, First Out*). Abbiamo già visto esempi in merito a come usare delle pipe nella *shell*, adesso prendiamone una in considerazione per quanto riguarda la nostra dimostrazione:

```
$ ls -d /proc/[0-9] | head -6
/proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
```

Un fatto che non potrete osservare in questo esempio (in quanto avviene troppo rapidamente per essere visto) è che le azioni di scrittura sulle pipe creano un blocco. Questo significa che quando il comando `ls` scrive su una pipe, rimane bloccato finché un processo dal lato oppo-

sto legge dalla pipe. Per poter visualizzare questo effetto, potete creare delle pipe con nome, che, a differenza delle pipe usate dalle *shell*, hanno un proprio nome (cioè, sono visibili nel filesystem, mentre le pipe della *shell* non lo sono). Il comando per creare questo tipo di pipe è `mkfifo`:

```
$ mkfifo una_pipe
$ ls -il
total 0
    169 prw-rw-r--    1 queen       queen              0 dic 10 14:12 una_pipe|
  #
  # Come potete vedere il contatore di link counter è 1, e l'output mostra
  # che il file è un pipe ('p').
  #
  # Qui potete anche usare ln:
  #
$ ln una_pipe la_stessa_pipe
$ ls -il
total 0
    169 prw-rw-r--    2 queen       queen              0 dic 10 15:37 una_pipe|
    169 prw-rw-r--    2 queen       queen              0 dic 10 15:37 la_stessa_pipe|
$ ls -d /proc/[0-9] >a_pipe
  #
  # Il processo è bloccato, dato che non c'è un lettore dall'altro lato.
  # Digitate C-z per sospendere il processo...
  #
zsh: 3452 suspended  ls -d /proc/[0-9] > una_pipe
  #
  # ...quindi mettetelo in background:
  #
$ bg
[1]  + continued  ls -d /proc/[0-9] > una_pipe
 #
  # adesso leggete dalla pipe...
  #
$ head -6 <la_stessa_pipe
  #
  # ...il processo di scrittura termina
  #
[1]  + 3452 done       ls -d /proc/[0-9] > una_pipe
/proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
#
```

Allo stesso modo, anche la lettura da una pipe provoca un blocco. Se eseguiamo i comandi elencati sopra al contrario, noteremo che `head` si blocca, in quanto resta in attesa che qualche processo gli dia qualcosa da leggere:

```
$ head -6 <una_pipe
  #
  # Il programma si blocca, sospendiamo l'esecuzione: C-z
  #
zsh: 741 suspended  head -6 < una_pipe
  #
  # Mettiamolo in background...
  #
$ bg
[1]  + continued  head -6 < una_pipe
  #
  # ...e diamogli da mangiare :)
  #
$ ls -d /proc/[0-9] >la_stessa_pipe
$ /proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
[1]  + 741 done       head -6 < una_pipe
$
```

Nell'esempio precedente potete anche vedere un effetto non desiderato: il comando `ls` ha terminato l'esecuzione prima che subentrasse il comando `head`. Come conseguenza, il prompt vi viene restituito immediatamente, ma `head` verrà eseguito solo successivamente. Perciò il suo output è stato prodotto solo dopo che voi siete tornati al prompt.

## File "speciali": file in modalità a caratteri e file in modalità a blocchi

Come abbiamo detto in precedenza, si tratta di file creati dal sistema oppure da periferiche collegate alla vostra macchina. Abbiamo anche menzionato il fatto che il contenuto di file in modalità a blocchi è bufferizzato, mentre quello di file in modalità a caratteri non lo è. Per

meglio illustrare questo concetto, inserite un floppy nel drive e digitate il comando che segue due volte di seguito:

```
$ dd if=/dev/fd0 of=/dev/null
```

Potrete notare quanto segue: la prima volta che avete lanciato il comando è stato letto l'intero contenuto del floppy, ma la seconda volta non c'è stato nessun accesso a quest'ultimo. Questo succede perché, molto semplicemente, il contenuto del dischetto è stato bufferizzato la prima volta che avete lanciato il comando – e non avete cambiato dischetto nel frattempo.

Ma ora, se volete stampare un file di grandi dimensioni con questo metodo (sì, funzionerà):

```
$ cat /un/grande/file/da/stampare >/dev/lp0
```

il comando impiegherà esattamente la stessa quantità di tempo se lo lanciate una volta soltanto, due oppure cinquanta volte. Questo è dovuto al fatto che `/dev/lp0` è un file file in modalità a caratteri, e il suo contenuto non è bufferizzato.

Il fatto che i file file in modalità a blocchi siano bufferizzati ha un effetto collaterale positivo: sono bufferizzate non solo le operazioni di lettura, ma anche quelle di scrittura. Ciò consente alle operazioni di scrittura su disco di essere asincrone: quando scrivete un file sul disco, l'operazione non viene effettuata immediatamente: viene portata a termine solo quando *GNU/Linux* decide che è il momento.

Ciascun file speciale, infine, ha un numero primario (***major***) e uno secondario (***minor***). Visualizzando il risultato di un comando `ls -l`, questi numeri compaiono al posto delle dimensioni, poiché le dimensioni sono irrilevanti per file di questo tipo:

```
 ls -l /dev/hda /dev/lp0
brw-rw----    1 root     disk       3,   0 May  5  1998 /dev/hda
crw-rw----    1 root     daemon     6,   0 May  5  1998 /dev/lp0
```

Qui il numero primario e il secondario di `/dev/hda` sono, rispettiva-

mente, 3 e 0, mentre per `/dev/lp0` sono rispettivamente 6 and 0. Notate che questi numeri sono unici per categoria di file, il che significa che può esserci un file file in modalità a caratteri con numero primario 3 e secondario 0 (questo file esiste effettivamente: `/dev/ttyp0`), e, allo stesso modo, può esistere solo un file file in modalità a blocchi con primario 6 e secondario 0. Questi numeri esistono per un motivo molto semplice: permettono a *GNU/Linux* di associare questi file alle operazioni appropriate (alle periferiche, cioè, cui si riferiscono questi file). Un lettore floppy non viene gestito nello stesso modo di un disco rigido SCSI, ad esempio.

## I link simbolici e le limitazioni degli “hard” link

Dobbiamo adesso affrontare un errore molto comune, anche fra gli utenti *Unix*, dovuto al fatto che i link come li abbiamo visti fino a questo momento (chiamati erroneamente “hard” link) sono associati soltanto a file di tipo normale (e abbiamo visto che le cose non stanno così – soprattutto se si considera che anche i link simbolici sono “linked”). Ma questo richiede che prima spieghiamo cosa sono i collegamenti simbolici (“soft” link, o, anche più spesso, “symlink”).

I collegamenti simbolici  sono file di un tipo particolare, il cui unico contenuto è una stringa arbitraria, che può riferirsi a un nome di file esistente oppure no. Quando utilizzate un collegamento simbolico dalla riga di comando o in un programma, di fatto accedete al file al quale si riferisce, se esiste. Ad esempio:

```
$ echo Ciao >ilmiofile
$ ln -s ilmiofile ilmiolink
$ ls -il
total 4
    169 -rw-rw-r--    1 queen       queen              6 dic 10 21:30 ilmiofile
    416 lrwxrwxrwx    1 queen       queen              6 dic 10 21:30 ilmi-
olink -> ilmiofile
$ cat ilmiofile
Ciao
$ cat ilmiolink
```

```
Ciao
```

Come potete vedere il tipo di file per `ilmiofile` è `'l'`, che sta per *Link* di tipo simbolico. I diritti di accesso per un collegamento simbolico non sono significativi: sono sempre impostati a `rwxrwxrwx`. Noterete anche che **è** un file differente rispetto a `ilmiofile`, poiché il suo numero di inode è diverso. Ma si riferisce a quest'ultimo in maniera simbolica, per cui quando digitate `cat ilmiolink`, in realtà ottenete come risultato la visualizzazione del contenuto di `ilmiofile`. Per dimostrare che un collegamento simbolico contiene una stringa arbitraria possiamo fare quanto segue:

```
$ ln -s "Io non esisto" unaltrolink
$ ls -il unaltrolink
    418 lrwxrwxrwx    1 queen       queen             20 dic 10 21:43 unal-
trolink ->;
Io non esisto
$ cat unaltrolink
cat: unaltrolink: No such file or directory
$
```

Ma i collegamenti simbolici esistono perché permettono di superare molte limitazioni degli ("hard") link:

- non è possibile collegare due file l'uno all'altro se questi file risiedono su filesystem diversi, per un semplice motivo: il contatore di link è conservato all'interno dell'inode stesso, e gli inode non possono essere condivisi fra filesystem. I collegamenti simbolici, invece, permettono questa operazione;
- non è possibile collegare due directory, poiché abbiamo visto che il contatore di link per una directory ha un uso particolare. Ma potete creare un collegamento a una directory e usarlo come se fosse effettivamente una directory.

I collegamenti simbolici, dunque, si rivelano utili in molte situazioni e, molto spesso, vengono usati per collegare file anche quando potrebbe essere usato un collegamento normale. Un vantaggio dei collegamenti normali, comunque, è dato dal fatto che, se cancellate "il

file originale", il file non viene perso, ma rimane accessibile tramite il link.

Se ci avete seguito con attenzione, infine, sapete quali sono le dimensioni di un collegamento simbolico: si tratta semplicemente delle dimensioni della stringa.

## Attributi dei file

Com'è noto, la FAT dispone di attributi dei file (archivio, file di sistema, nascosto): anche *ext2fs* presenta la stessa caratteristica, ma sono differenti. Ne parliamo qui per ragioni di completezza, ma sono usati molto raramente. Se davvero desiderate un sistema sicuro, comunque, proseguite nella lettura.

Ci sono due comandi per modificare gli attributi dei file: `man 1 lsattr` e `man 1 chattr`. Come avrete probabilmente indovinato, `lsattr` elenca (*LiSt*) gli attributi, mentre `chattr` li cambia (*CHange*). Gli attributi di cui stiamo parlando riguardano soltanto le directory e i file normali, sono possibili i seguenti tipi:

1. `A` ((*no Access time* "non aggiornare la data di accesso") Se un file o una directory presenta questo attributo, tutte le volte che vi si accede, sia per leggerlo sia per scrivervi, l'indicazione cronologica relativa all'ultimo accesso resta inalterata. Questo può tornare utile, ad esempio, nel caso di file o directory che vengono richiesti in lettura molto spesso, soprattutto se si considera che questo parametro è l'unico tra quelli di un inode che cambia quando il file è aperto solo in lettura.
2. `a` (*append only* "aggiungi soltanto") Se un file presenta questo attributo ed è accessibile in scrittura, l'unica operazione possibile sarà quella di aggiungere dati al suo precedente contenuto. Per una directory, invece, questo significa che è possibile soltanto aggiungervi file, ma non rinominare o cancellare uno dei file già presenti. Solo `root` può impostare o rimuovere questo attributo.

3. `d` (*no dump* "niente dump") `dump (8)` è il programma standard per effettuare backup dei dati sotto *Unix*. Effettua una copia di qualsiasi filesystem per il quale sia impostato 1 come valore del *dump counter* nel file `/etc/fstab` (si veda il capitolo *Filesystem e punti di mount*, pag. 127). Ma se un file o una directory presentano questo attributo, a differenza degli altri non verranno presi in considerazione la prossima volta che verrà usato il comando dump. Si noti che, nel caso delle directory, questo riguarda anche tutte le sotto-directory e i file contenuti da queste ultime.

4. `i` (*immutable* "non modificabile") Un file o una directory che presentino questo attributo non possono in nessun modo essere modificati: non possono essere rinominati, nessun ulteriore collegamento può essere creato[2] e non possono essere cancellati. Soltanto `root` può impostare o disabilitare questo attributo. Notate che, come conseguenza, anche il la data di accesso non può essere modificata, perciò quando questo attributo è attivo non è necessario ricorrere all'attributo `A`.

5. `s` (*secure deletion* "cancellazione sicura") Quando viene cancellato un file o una directory che presenta questo attributo i blocchi che occupava su disco vengono riscritti con degli zero.

6. `S` (*Synchronous mode* "modo sincrono") Se un file o una directory presenta questo attributo, tutte le modifiche apportate sono sincrone e vengono immediatamente registrate su disco.

Potrebbe essere una buona idea, ad esempio, impostare l'attributo `'i'` sui file di sistema più importanti al fine di evitare brutte sorprese. Un altro utilizzo potrebbe essere l'impiego dell'attributo `'A'` sulle pagine di manuale: questo eviterebbe un gran numero di operazioni sul disco e, in particolare, permetterebbe di risparmiare un po' di carica delle batterie sui portatili.

---

2. A questo punto dovreste sapere bene cosa significa "aggiungere un link", sia per un file sia per una directory **:-)**

# Capitolo 11. Il filesystem `/proc`

Il filesystem `/proc` è una caratteristica specifica di *GNU/Linux*. È un filesystem virtuale, e come tale non occupa spazio su disco. Esso rappresenta un mezzo molto comodo per ottenere informazioni sul sistema, anche perché gran parte dei file nella relativa directory sono facilmente leggibili (magari con un po' d'aiuto). Molti programmi in effetti prelevano informazioni dai file in `/proc`, le formattano a modo loro e poi le visualizzano; è il caso, ad esempio, di tutti i programmi che mostrano informazioni sui processi, alcuni dei quali abbiamo visto in precedenza (`top`, `ps` e simili). `/proc` rappresenta anche una buona fonte di informazioni riguardo al vostro hardware e, anche in questo caso, parecchi programmi non sono altro che interfacce verso le informazioni contenute in `/proc`.

Esiste anche una sotto-directory speciale, `/proc/sys`. Essa permette di visualizzare o modificare in tempo reale alcuni parametri del kernel.

## Informazioni sui processi

Se osservate il contenuto della directory `/proc` vedrete molte directory il cui nome è un numero; quelle sono le directory che contengono informazioni su tutti i processi attualmente in corso di esecuzione nel sistema:

```
$ ls -d /proc/[0-9]*
/proc/1/    /proc/302/  /proc/451/  /proc/496/  /proc/556/  /proc/633/
/proc/127/  /proc/317/  /proc/452/  /proc/497/  /proc/557/  /proc/718/
/proc/2/    /proc/339/  /proc/453/  /proc/5/    /proc/558/  /proc/755/
/proc/250/  /proc/385/  /proc/454/  /proc/501/  /proc/559/  /proc/760/
/proc/260/  /proc/4/    /proc/455/  /proc/504/  /proc/565/  /proc/761/
/proc/275/  /proc/402/  /proc/463/  /proc/505/  /proc/569/  /proc/769/
/proc/290/  /proc/433/  /proc/487/  /proc/509/  /proc/594/  /proc/774/
/proc/3/    /proc/450/  /proc/491/  /proc/554/  /proc/595/
```

Notate, però, che come utenti voi potete (giustamente) visualizzare solo le informazioni relative ai vostri processi, e non a quelli di altri

utenti. Quindi proviamo a diventare `root` e a vedere che informazioni possiamo ottenere dal processo 127:

```
$ su
Password:
$ cd /proc/127
$ ls -l
total 0
-r--r--r--    1 root     root            0 dic 14 19:53 cmdline
lrwx------    1 root     root            0 dic 14 19:53 cwd -> //
-r--------    1 root     root            0 dic 14 19:53 environ
lrwx------    1 root     root            0 dic 14 19:53 exe -> /usr/sbin/apmd*
dr-x------    2 root     root            0 dic 14 19:53 fd/
pr--r--r--    1 root     root            0 dic 14 19:53 maps|
-rw-------    1 root     root            0 dic 14 19:53 mem
lrwx------    1 root     root            0 dic 14 19:53 root -> //
-r--r--r--    1 root     root            0 dic 14 19:53 stat
-r--r--r--    1 root     root            0 dic 14 19:53 statm
-r--r--r--    1 root     root            0 dic 14 19:53 status
$
```

Tutte le directory contengono gli stessi elementi; ecco una breve descrizione di alcuni di essi:

1. `cmdline` Questo (pseudo-)file contiene l'intera linea di comando che è stata utilizzata per avviare il processo. Non è formattata: non ci sono spazi tra il nome del programma e i suoi argomenti, e non c'è neanche il ritorno a capo a fine riga.
2. `cwd` Questo collegamento simbolico punta all'attuale directory di lavoro ("current working directory", da qui il nome) del processo.
3. `environ` Questo file contiene tutte le variabili d'ambiente definite per il processo, nella forma `VARIABLE=valore`. Come per `cmdline`, il contenuto non è minimamente formattato: nessun ritorno a capo, né per separare le diverse variabili, né alla fine.
4. `exe` Questo è un link simbolico che punta al file eseguibile corrispondente al processo in esecuzione.
5. `fd` Questa sotto-directory contiene la lista dei descrittori di file attualmente aperti dal processo. Si veda più avanti.

6. `maps` Stampando (ad esempio tramite `cat`) il contenuto di questa pipe denominata (o "pipe con nome") potrete vedere le sezioni dello spazio di indirizzamento del processo che sono attualmente assegnate a un file. I campi da sinistra a destra sono: lo spazio di indirizzamento relativo all'assegnazione, i permessi associati all'assegnazione, lo scostamento (ingl. "offset") da cui parte l'assegnazione rispetto all'inizio del file, il dispositivo su cui si trova il file a cui è assegnato lo spazio, il numero di inode del file, e infine il nome del file stesso. Si veda `man 2 mmap`.
7. `root` Questo è un collegamento simbolico che punta alla directory considerata come radice dal processo. Di solito questa sarà `/`, ma si veda anche `man 2 chroot`.
8. `status` Questo file contiene varie informazioni sul processo: il nome dell'eseguibile, il suo stato attuale, i suoi PID e PPID, i suoi UID e GID reali ed effettivi, il suo utilizzo di memoria, e altro ancora.

Visualizzando il contenuto della directory `fd` otterremo questo:

```
$ ls -l fd
total 0
lrwx------    1 root     root           64 dic 16 22:04 0 -> /dev/console
l-wx------    1 root     root           64 dic 16 22:04 1 -> pipe:[128]
l-wx------    1 root     root           64 dic 16 22:04 2 -> pipe:[129]
l-wx------    1 root     root           64 dic 16 22:04 21 -> pipe:[130]
lrwx------    1 root     root           64 dic 16 22:04 3 -> /dev/apm_bios
lr-x------    1 root     root           64 dic 16 22:04 7 -> pipe:[130]
lrwx------    1 root     root           64 dic 16 22:04 9 ->
/dev/console
$
```

Questa, infatti, è la lista dei descrittori di file aperti dal processo. Ogni descrittore aperto è indicato da un collegamento simbolico, il cui nome è il numero del descrittore stesso, e il quale punta al file aperto tramite il descrittore[1]. Notate anche i permessi dei collegamenti sim-

1. Se ricordate quello che abbiamo detto nella sezione *La redirezione e le pipe*, pag. 52 nel *Introduzione alla linea di comando*, pag. 43, sapete già

bolici: questo è l'unico caso in cui hanno senso, poiché rappresentano i permessi con i quali è stato aperto il file corrispondente al descrittore.

## Informazioni sull'hardware

Oltre alle directory associate ai diversi processi, `/proc` contiene anche una miriade di informazioni sull'hardware del vostro sistema. Quello che segue è un elenco dei file della directory `/proc`:

```
$ ls -d [a-z]*
apm       dma          interrupts  loadavg  mounts      rtc       swaps
bus/      fb           ioports     locks    mtrr        scsi/     sys/
cmdline   filesystems  kcore       meminfo  net/        self/     tty/
cpuinfo   fs/          kmsg        misc     partitions  slabinfo  uptime
devices   ide/         ksyms       modules  pci         stat      version
$
```

Ad esempio, se esaminiamo il contenuto di `/proc/interrupts`, possiamo vedere che esso contiene l'elenco degli interrupt attualmente usati dal sistema, insieme alla periferica che li gestisce. Allo stesso modo, `ioports` contiene l'elenco dei range di indirizzi di ingresso/uscita attualmente occupati, e infine `dma` fa la stessa cosa per i canali DMA. Pertanto, per trovare la causa di un conflitto, esaminate il contenuto di questi tre file:

```
$ cat interrupts
           CPU0
  0:     127648          XT-PIC  timer
  1:       5191          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  5:       1402          XT-PIC  xirc2ps_cs
  8:          1          XT-PIC  rtc
 10:          0          XT-PIC  ESS Solo1
 12:       2631          XT-PIC  PS/2 Mouse
 13:          1          XT-PIC  fpu
 14:      73434          XT-PIC  ide0
 15:      80234          XT-PIC  ide1
NMI:          0
```

cosa rappresentano i descrittori 0, 1 e 2.

```
$ cat ioports
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0070-007f : rtc
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
0300-030f : xirc2ps_cs
0376-0376 : ide1
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
1050-1057 : ide0
1058-105f : ide1
1080-108f : ESS Solo1
10c0-10cf : ESS Solo1
10d4-10df : ESS Solo1
10ec-10ef : ESS Solo1
$ cat dma
 4: cascade
$
```

Oppure, più semplicemente, usate il comando `lsdev`, che raccoglie le informazioni da questi tre file e le ordina per periferica, il che è senza dubbio più comodo[2]:

```
$ lsdev
Device            DMA   IRQ  I/O Ports
----------------------------------
cascade             4     2
dma                             0080-008f
dma1                            0000-001f
dma2                            00c0-00df
ESS                             1080-108f 10c0-10cf 10d4-10df 10ec-10ef
fpu                          13 00f0-00ff
ide0                         14 01f0-01f7 03f6-03f6 1050-1057
ide1                         15 0170-0177 0376-0376 1058-105f
keyboard                      1 0060-006f
```

2. `lsdev` fa parte del pacchetto `procinfo`.

```
Mouse                   12
pic1                        0020-003f
pic2                        00a0-00bf
rtc                      8  0070-007f
serial                      03f8-03ff
Solo1                   10
timer                    0  0040-005f
vga+                        03c0-03df
xirc2ps_cs               5  0300-030f
$
```

Un elenco completo dei file sarebbe troppo lungo, ma ecco la descrizione di alcuni di essi:

- `cpuinfo` Questo file contiene, come dice il nome stesso, informazioni sul processore (o sui processori) di cui dispone il vostro sistema.
- `modules` Questo file contiene un elenco dei moduli attualmente usati dal kernel, insieme a un contatore di utilizzi per ciascuno. In effetti, queste sono le stesse informazioni mostrate dal comando `lsmod`.
- `meminfo` Questo file contiene informazioni sull'uso della memoria al momento in cui ne viene visualizzato il contenuto. Si può ottenere una versione delle stesse informazioni formattata in modo più chiaro tramite il comando `free`.
- `apm` Se possedete un portatile, potete controllare lo stato della batteria esaminando il contenuto di questo file. Potete sapere se è collegato alla presa di corrente, la carica attuale della batteria e, se il *BIOS* APM del vostro portatile lo supporta (sfortunatamente non è così per tutti), la carica residua della batteria in minuti. Questo file non è molto chiaro, quindi fareste meglio a usare il comando `apm`, che fornisce le stesse informazioni in un formato leggibile.
- `bus` Questa sotto-directory contiene informazioni su tutte le periferiche che sono collegate ai vari bus del vostro sistema. Queste informazioni sono raramente leggibili, e nella maggior parte dei casi

vengono gestite e formattate da programmi esterni: `lspcidrake`, `lspnp`, etc.

## La sotto-directory `/proc/sys`

Questa sotto-directory ha lo scopo di mostrare vari parametri del kernel e di permettere la modifica in tempo reale di alcuni di essi. A differenza di tutti gli altri file contenuti in `/proc`, alcuni file di questa directory possono essere modificati, ma solo da `root`.

Un elenco di tutti i file e le directory sarebbe troppo lungo; oltretutto la loro presenza o meno dipende in gran parte dal vostro particolare sistema, e la maggior parte di essi è utile solo per applicazioni molto particolari. In ogni caso, ecco tre usi classici per questa sotto-directory:

1. Permettere il routing: sebbene il kernel predefinito di **Linux-Mandrake** sia in grado di farlo, è necessario abilitare esplicitamente questa funzione. Per abilitarla è sufficiente digitare il seguente comando da `root`:

   ```
   $ echo 1 >/proc/sys/net/ipv4/ip_forward
   ```

   Sostituite 1 con uno 0 se volete disabilitare il routing.

2. Impedire il mascheramento di IP: il mascheramento di IP consiste nel far credere al sistema che un pacchetto arrivato dal mondo esterno provenga invece dall'interfaccia tramite la quale è stato ricevuto. Questa tecnica è comunemente usata dai ***cracker***[3], ma potete fare in modo che il kernel impedisca questo tipo di intrusione. Dovete solo digitare:

3. E non dagli ***hacker***!

```
$ echo 1 >/proc/sys/net/ipv4/conf/all/rp_filter
```

e questo tipo di attacco diverrà impossibile.

3. Aumentare la dimensione della tabella dei file aperti e della tabella degli inode: le dimensioni di queste due tabelle in *GNU/Linux* sono dinamiche. I valori predefiniti generalmente sono sufficienti per un uso normale, ma potrebbero essere troppo bassi se il vostro sistema è un grosso server (ad esempio un server di database). In effetti, l'ostacolo principale può essere il fatto che i processi non possono più aprire alcun file perché la tabella è piena, e quindi dovrete aumentarne la dimensione. Contemporaneamente, dovrete anche aumentare la dimensione della tabella degli inode. Queste due righe risolveranno il problema:

   ```
   $ echo 8192 >/proc/sys/fs/file-max
   $ echo 16384 >/proc/sys/fs/inode-max
   ```

   Perché esse siano eseguite ad ogni avvio del sistema potreste aggiungere queste due righe nel file `/etc/rc.d/rc.local`, in modo da evitare di doverle digitare ogni volta.

# Capitolo 12. I file di avvio del sistema: init "System V"

Nella tradizione *Unix* esistono due diversi "stili" di avvio del sistema: lo stile BSD e lo stile "*System V*"; entrambi devono il loro nome al sistema *Unix* che li ha utilizzati per primo (lo *Unix Berkeley Software Distribution* e lo *Unix System V* della **AT&T**, rispettivamente). Lo stile di avvio BSD è il più semplice, ma quello *System V*, sebbene sia meno facile da comprendere (fatto che cambierà grazie alla lettura di questo capitolo), è estremamente più flessibile.

## In principio fu init

All'avvio del sistema, subito dopo aver configurato tutti i componenti hardware e aver montato il filesystem della partizione root, il kernel lancia il programma `/sbin/init`[1]. `init` è il padre di tutti i processi del sistema, e ha il compito di portare il sistema al ***runlevel*** desiderato. Nel prossimo paragrafo daremo uno sguardo al concetto di runlevel (livello di esecuzione).

Il file di configurazione di `init` è `/etc/inittab`. C'è una pagina di manuale che ne descrive l'uso nei minimi dettagli (`man inittab`), ma qui vedremo solo alcune delle opzioni di configurazione.

La prima riga che dovrebbe attirare la vostra attenzione è la seguente:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

Questa istruzione dice a `init` che, al momento dell'inizializzazione del sistema, lo script `/etc/rc.d/rc.sysinit` deve essere esegui-

1. Ora sapete perché mettere la directory `/sbin` su un filesystem diverso da root è una pessima idea **:-)**

to prima di ogni altra operazione. In seguito, `init` controlla la riga che contiene l'indicazione `initdefault` per determinare il runlevel predefinito:

```
id:5:initdefault:
```

In questo caso, `init` utilizzerà il 5 come runlevel predefinito. `init` inoltre sa che per portare il sistema al runlevel 5 dovrà eseguire questo comando:

```
l5:5:wait:/etc/rc.d/rc 5
```

Come potete notare, la sintassi di questi comandi è più o meno la stessa per tutti i runlevel.

`init` ha anche il compito di riavviare (`respawn`) determinati servizi, e solo lui può farlo. Questo avviene, ad esempio, per tutti i programmi di login che girano in ciascuna delle 6 console virtuali[2]. Questo, ad esempio, è il comando relativo alla seconda console virtuale:

```
2:2345:respawn:/sbin/mingetty tty2
```

## I runlevel

Tutti i file che controllano la procedura di avvio del sistema sono situati nella directory `/etc/rc.d` . Quello che segue è l'elenco dei file:

```
$ ls /etc/rc.d
init.d/  rc.local*    rc0.d/  rc2.d/  rc4.d/  rc6.d/
rc*      rc.sysinit*  rc1.d/  rc3.d/  rc5.d/
```

2. Modificando il file `/etc/inittab` potete, se lo desiderate, aumentare fino a un massimo di 64 il numero delle console virtuali, oppure ridurlo, seguendo sempre la stessa sintassi. Ma non dimenticate che anche *X* gira su una console virtuale, quindi lasciategliene almeno una!

Come abbiamo visto, per prima cosa all'avvio del sistema viene eseguito lo script `rc.sysinit` Questo è il file responsabile della configurazione iniziale della macchina: tipo di tastiera, configurazione di alcuni dispositivi, controllo dei filesystem, etc.

Subito dopo viene eseguito lo script `rc` con un numero di runlevel come argomento. Come sappiamo, il runlevel è indicato da un numero intero: per ogni runlevel `<x>` che è stato definito esiste una directory `rc<x>.d` corrispondente. In un'installazione tipica di **LinuxMandrake** troverete 6 runlevel predefiniti, che sono:

- 0: Arresto completo della macchina;
- 1: Modalità***monoutente***;, va utilizzata in caso di gravi problemi del sistema o per operazioni di recupero in seguito a un crash.
- 2: Modalità ***multiutente***, senza accesso alla rete;
- 3: Modalità multi-user, con accesso alla rete;
- 4: Inutilizzato;
- 5: Come il 3, ma con il lancio dell'interfaccia grafica;
- 6: Riavvio del sistema.

Tanto per fare un esempio, diamo un'occhiata al contenuto della directory `rc5.d`:

```
$ ls rc5.d
K15postgresql@  K60atd@      S15netfs@   S60lpd@         S90xfs@
K20nfs@         K96pcmcia@   S20random@  S60nfs@         S99linuxconf@
K20rstatd@      S05apmd@     S30syslog@  S66yppasswdd@   S99local@
K20rusersd@     S10network@  S40crond@   S75keytable@
K20rwhod@       S11portmap@  S50inet@    S85gpm@
K30sendmail@    S12ypserv@   S55named@   S85httpd@
K35smb@         S13ypbind@   S55routed@  S85sound@
```

Come potete vedere, tutti i file di questa directory sono link simbolici, e tutti hanno un nome in un formato particolare. Questo formato corrisponde allo schema `<S|K><ordine><nome_servizio>`. La lettera `S` sta per *`Start`*, "avvia" un servizio, e la `K` indica *`Kill`*, "ferma" il

servizio. Gli script vengono eseguiti secondo un ordine ascendente, e se due script hanno lo stesso numero vengono eseguiti in ordine alfabetico. Noterete anche che ciascun link simbolico punta agli script che si trovano nella directory `/etc/rc.d/init.d` (tranne quello indicato dal nome `local`), e sono questi ultimi a controllare effettivamente ogni singolo servizio.

Quando entra in un determinato runlevel, il sistema comincia a eseguire i link contrassegnati dalla lettera `K` secondo l'ordine predefinito: `rc` controlla dove punta il link, quindi esegue lo script relativo con `stop` come unico argomento. Subito dopo vengono eseguiti gli script di tipo `S` seguendo la stessa procedura, solo che stavolta vengono chiamati fornendogli l'argomento `start`.

Così, senza che sia necessario menzionare tutti gli script, quando il sistema entra nel runlevel 5 `rc` per prima cosa esegue `K15postgresql`, ovvero `/etc/rc.d/init.d/postgresql stop`, poi `K20nfs`, quindi `K20rstatd`, e così via, fino all'ultimo; subito dopo esegue tutti gli script di tipo `S`: per primo `S05ampd`, che chiama `/etc/rc.d/init.d/apmd start`, e via di seguito.

Con questi strumenti siete liberi di creare, con poca fatica, i vostri runlevel personalizzati, o di impedire a un servizio di partire o di fermarsi cancellando il corrispondente link simbolico (ci sono anche dei programmi dotati di interfaccia per fare questo, tra cui citiamo *DrakXServices* e `chkconfig`; il primo è un programma grafico).

# Capitolo 13. Controllo dei processi

## Ancora sui processi

Nel capitolo *Processi*, pag. 32 nel *Concetti base di Unix*, pag. 25 abbiamo visto che è possibile monitorare i processi: questo è l'argomento di cui ci occuperemo ora. Per comprendere meglio le operazioni che effettueremo, è opportuno introdurre qualche altra informazione su di essi.

### L'albero dei processi

Come per i file, tutti i processi che girano in un sistema *GNU/Linux* sono organizzati in una struttura ad albero, e ogni processo ha un numero (il suo PID, *Process ID*), insieme al numero del proprio processo padre (PPID, *Parent Process ID*).

Questo significa che c'è un processo, alla radice dell'albero, che è l'equivalente di `root` per il filesystem: `init` (consultate il **Manuale di riferimento** in proposito), che ha sempre il numero 1. Il prossimo paragrafo illustrerà due comandi, `ps` e `pstree`, che vi consentono di ottenere informazioni su un processo in esecuzione.

### Segnali

Ogni processo in un sistema *Unix* può reagire ai segnali inviatigli. Vi sono 31 differenti segnali. Per ciascuno di essi, il processo può definire una diversa reazione, fatta eccezione per due segnali: il segnale numero 9 (`KILL`), e il segnale numero 19 (`STOP`).

Il segnale 9 uccide un processo in modo irrevocabile, senza lasciargli il tempo di terminare l'esecuzione in maniera appropriata. Questo è il segnale che dovreste inviare a un processo bloccato, o che manifesta

altri problemi. Un elenco completo dei segnali è disponibile digitando il comando `kill -l`.

## Informazioni sui processi: i comandi `ps` e `pstree`

Questi due comandi visualizzano una lista dei processi presenti sul sistema, in modo coerente ai criteri dai voi stabiliti.

### `ps`

Digitando questo comando senza alcun argomento, verranno mostrati solo i processi avviati da voi e collegati al terminale che voi state utilizzando:

```
$ ps
  PID TTY          TIME CMD
 5162 ttya1    00:00:00 zsh
 7452 ttya1    00:00:00 ps
```

Vi è un gran numero di opzioni, ve ne segnaliamo alcune tra le più comuni:

- `a`: visualizza anche i processi avviati da altri utenti;
- `x`: visualizza anche i processi che non hanno un terminale di controllo (questo è vero per quasi tutti i server);
- `u`: visualizza per ciascun processo il nome dell'utente che lo ha avviato e l'ora in cui è partito.

Vi sono molte altre opzioni. Fate riferimento alla pagina di manuale relativa per ulteriori informazioni (`man 1 ps`).

L'output di questo comando è suddiviso in due diversi campi: quello che vi interesserà di più è il campo `PID`, che contiene il numero iden-

tificativo del processo. Il campo `CMD` contiene il nome del comando eseguito.

Un modo molto comune di lanciare `ps` è il seguente:

```
$ ps ax | less
```

Questo vi dà una lista di tutti i processi attualmente in esecuzione, in maniera tale da permettervi di identificare uno o più processi che stanno causando problemi e quindi di terminarli.

### `pstree`

Il comando `pstree` visualizza i processi sotto forma di struttura ad albero. Uno dei vantaggi è che potete immediatamente vedere qual è il processo padre di un altro processo: quando volete terminare un'intera serie di processi e tutti questi sono padre o figlio l'uno dell'altro, dovete semplicemente terminare il processo padre. Potete utilizzare l'opzione `-p`, che visualizza il PID di ciascun processo, e l'opzione `-u` che visualizza il nome dell'utente che ha avviato il processo. Poiché la struttura ad albero è generalmente lunga, può essere utile eseguire `pstree` in questo modo:

```
$ pstree -up | less
```

Questo vi fornisce una visione schematica dell'intero albero dei processi.

## Inviare segnali ai processi: `kill`, `killall` e `top`

## Scorciatoia in `X`: `xkill`

Se state utilizzando *KDE*, esiste una scorciatoia per terminare un processo *X* bloccato. C'è un'icona facilmente identificabile sul desktop, riprodotta qui sotto:

**Figura 13-1. L'icona di `xkill`**

Questa icona attiva il programma `xkill`, che potete eseguire anche da un terminale. Quando cliccate su quest'icona (o eseguite il programma su un terminale), il cursore del mouse cambia forma. Quindi potete cliccare con il tasto sinistro del mouse sulla finestra corrispondente al processo che intendete terminare.

## `kill`, `killall`

Questi due comandi vengono utilizzati per inviare segnali ai processi. Il comando `kill` richiede come argomento il numero di un processo, mentre `killall` richiede il nome di un comando.

I due comandi possono ricevere il numero di un segnale come argomento opzionale. Come opzione predefinita, entrambi inviano il segnale 15 (`TERM`) al processo interessato. Ad esempio, se intendete terminare il processo con il PID 785, digitate il comando:

```
$ kill 785
```

Se intendete inviargli il segnale 9, digitate:

```
$ kill -9 785
```

Supponete di voler terminale un processo di cui conoscete il nome del comando. Invece di trovare il numero del processo usando `ps`, potete terminare direttamente il processo:

```
$ killall -9 netscape
```

Qualunque cosa accada, terminerete solo i vostri processi (a meno che non siate `root`), quindi non preoccupatevi dei processi del "vicino" che hanno lo stesso nome: questi non verranno terminati.

## `top`

`top` è un programma onnicomprensivo: svolge contemporaneamente le funzioni di `ps` e `kill`. Il programma funziona in modalità console, quindi viene lanciato in un terminale, come viene mostrato qui (Figura 13-2).

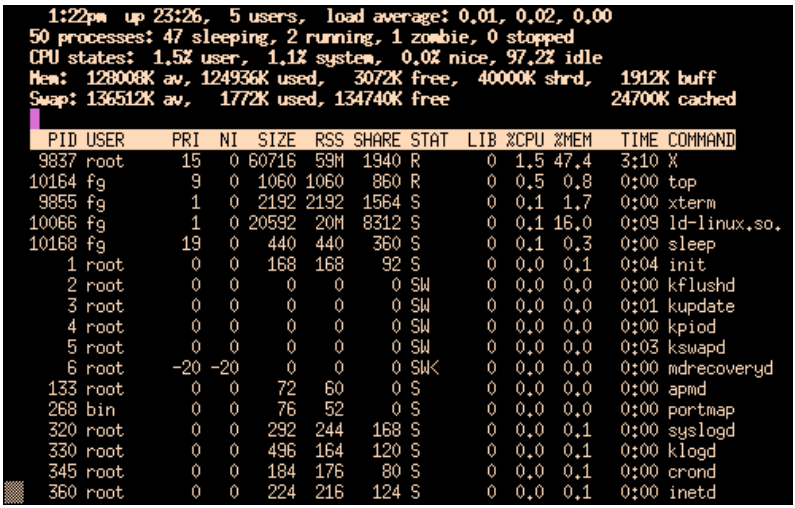


**Figura 13-2. Esempio di esecuzione di `top`**

Il programma è interamente controllato da tastiera. Potete visualizzare l'aiuto premendo **h**. Ecco alcuni dei comandi che potete usare:

- **k**: questo comando serve a inviare un segnale a un processo. `top` vi chiederà il PID del processo, seguito dal numero del segnale da inviare (15 come opzione predefinita);
- **M**: questo comando è usato per elencare i processi in base alla memoria utilizzata (campo `%MEM`);
- **P**: questo comando è usato per elencare i processi in base al tempo di CPU che utilizzano (campo `%CPU`; questo è l'ordinamento predefinito);
- **u**: questo comando serve a mostrare i processi di un utente in particolare, `top` vi chiederà quale. Dovrete inserire il **nome** dell'utente, non il suo UID. Se non fornite alcun nome, verranno visualizzati tutti i processi;
- **i**: questo comando opera come un interruttore: normalmente vengono visualizzati tutti i processi, anche quelli sospesi, questo comando fa in modo che vengano mostrati solo i processi attualmente in esecuzione (i processi il cui campo `STAT` indica `R`, *`Running`*), e non gli altri. Usando di nuovo il comando si ritorna alla precedente visualizzazione.

# Capitolo 14. Compilazione e installazione di nuovi kernel

Dopo la compilazione di sorgenti e le operazioni relative al filesystem, la compilazione del kernel è senza dubbio l'argomento che comporta maggiori problemi per i principianti. La compilazione di un nuovo kernel non è strettamente necessaria, in genere, poiché il kernel installato da **Linux-Mandrake** include il supporto per un numero significativo di dispositivi, come pure l'applicazione di molte patch e altre migliorie. Ma...

Può capitare che un giorno decidiate di compilarlo, forse per nessun'altra ragione eccetto che per vedere "che cosa succede": non molto, a parte il fatto che il vostro *PC* e la vostra caffettiera lavoreranno un po' più duramente del solito. Al di là della semplice curiosità, i motivi che potrebbero spingervi a compilare un vostro kernel possono essere i più vari: dalla (dis)attivazione di un'opzione, alla costruzione di un kernel sperimentale del tutto nuovo. Lo scopo di questo capitolo, comunque, è quello di far sì che la vostra caffettiera sia ancora in perfetta efficienza una volta terminata la compilazione.

Esistono altre valide ragioni per ricompilare il kernel. Avete appena letto, ad esempio, che il kernel che state usando presenta un ***bug*** che può compromettere la sicurezza del sistema, e che è stato corretto in una versione più recente; oppure è stato rilasciato un nuovo kernel che include il supporto per un dispositivo di cui avete bisogno. Naturalmente anche in questi ultimi casi potete scegliere di aspettare il rilascio di aggiornamenti in forma binaria, ma effettuare un aggiornamento dei sorgenti del kernel e ricompilarlo di persona costituisce una soluzione più rapida.

Qualunque cosa decidiate di fare, assicuratevi di avere a portata di mano una buona quantità di caffè.

## Dove trovare i sorgenti del kernel

Il sito principale che ospita i sorgenti del kernel è ftp.kernel.org, ma dispone di un buon numero di mirror, i cui nomi seguono lo schema ftp.xx.kernel.org, dove xx rappresenta il codice ISO della nazione. Se tenete d'occhio gli annunci ufficiali riguardo la disponibilità del kernel, rispetto ad essi dovrete aspettare un paio d'ore per dare il tempo ai mirror di effettuare l'aggiornamento.

Su tutti questi server FTP i sorgenti si trovano nella directory `/pub/linux/kernel`. Dopo di che, spostatevi nella directory che ospita la serie che più vi interessa, senza dubbio si tratterà della `v2.2`. Non c'è nulla che vi impedisca di provare i kernel della serie 2.3, ma ricordate che questi sono in fase sperimentale. Il file che contiene i sorgenti del kernel si chiama `linux-<kernel.version>.tar.gz`, ad esempio `linux-2.2.15.tar.gz`.

Sono anche disponibili delle patch da applicare ai sorgenti del kernel, in maniera da effettuare degli aggiornamenti incrementali: così, se avete già i sorgenti della versione 2.2.15 del kernel e volete aggiornarli alla 2.2.17, non è necessario scaricare il voluminoso archivio relativo a quest'ultima versione, ma potete semplicemente procurarvi e applicare le ***patch*** `patch-2.2.16.gz` e `patch-2.2.17.gz`. Come regola generale, questo è il metodo più conveniente, visto che i sorgenti al momento occupano più di 12 MB.

## Decomprimere i sorgenti, applicare le patch al kernel (se necessario)

I sorgenti del kernel dovrebbero trovarsi in `/usr/src`. Dunque spostatevi in questa directory e decomprimete gli archivi:

```
$ cd /usr/src
$ mv linux linux.old
$ tar xzf /path/to/linux-2.2.17.tar.gz
```

Il comando `mv linux linux.old` è necessario perché potrebbero essere presenti i sorgenti di un'altra versione del kernel. Con questo co-

mando sarete sicuri di mantenere questa versione meno recente, evitando di cancellarla scrivendoci sopra quella nuova. Una volta decompresso l'archivio, avrete a vostra disposizione una directory `linux` contenente i sorgenti del nuovo kernel.

E adesso, le patch: supponiamo, ad esempio, che vogliate ***applicare una patch*** per passare dalla versione 2.2.15 alla 2.2.17, e che abbiate già scaricato i file necessari allo scopo: spostatevi nella nuova directory `linux`, poi applicate le patch:

```
$ cd linux
$ gzip -dc /path/to/patch-2.2.16.gz | patch -p1 #
$ gzip -dc /path/to/patch-2.2.17.gz | patch -p1
$ cd ..
```

In genere, quando ci si sposta da una versione 2.2.x a una versione 2.2.y è necessario applicare tutte le patch numerate 2.2.x+1, 2.2.x+2, ..., 2.2.y in questo preciso ordine. Per "scendere" dalla 2.2.y alla 2.2.x, ripetete esattamente la stessa procedura, ma applicando le patch in ordine inverso e utilizzando l'opzione `-R` del comando `patch` (`R` sta per *`Reverse`*). Così, per tornare dal kernel 2.2.17 alla versione 2.2.15, dovreste digitare:

```
$ gzip -dc /path/to/patch-2.2.17.gz | patch -p1 -R
$ gzip -dc /path/to/patch-2.2.16.gz | patch -p1 -R
```

Poi, per amor di precisione (e per sapere esattamente con quale versione avete a che fare), potete rinominare la directory `linux` in maniera tale che indichi la versione del kernel, e creare un link simbolico che punti ad essa:

```
$ mv linux linux-2.2.11
$ ln -s linux-2.2.11 linux
```

Adesso è il momento di procedere alla configurazione. Per fare questo, dovete trovarvi nella directory che contiene i sorgenti:

```
$ cd linux
```

## Configurazione del kernel

Per configurare il kernel potete scegliere tra:

- `make xconfig` per usare un'interfaccia grafica,
- `make menuconfig` per usare un'interfaccia basata sulla libreria `ncurses`, oppure
- `make config` per usare l'interfaccia più rudimentale, riga per riga, sezione per sezione.

Procederemo a descrivere la configurazione sezione per sezione, ma se state usando `menuconfig` o `xconfig` potete anche saltare alcune sezioni e spostarvi direttamente a quelle che vi interessano. Le scelte possibili per ciascuna opzione sono, in genere, tre: **y** per Yes (la funzionalità in questione viene compilata come parte integrante del kernel), **m** per Module (la funzionalità in questione viene compilata come modulo), oppure **n** per No (non includere nel kernel).

Sia `make xconfig` sia `make menuconfig` raccolgono le opzioni in gruppi ordinati secondo una struttura gerarchica. Ad esempio, `Processor family` fa parte del gruppo `Processor type and features`.

Per quanto riguarda `xconfig`, se vi trovate all'interno di un gruppo gerarchico il pulsante Menu principale vi riporterà al menu principale, Next vi farà avanzare al gruppo di opzioni successivo e Prev vi farà tornare al gruppo precedente. Se invece avete lanciato `menuconfig`, usate il tasto `Invio` per selezionare una sezione, e i tasti **y**, **m** o **n** per cambiare le impostazioni; oppure premete `Invio` e fate la vostra scelta in base alle opzioni che vi vengono presentate. Exit vi porterà fuori da una sezione, e vi farà abbandonare la configurazione se vi trovate nel menu principale. Vi ricordiamo, infine, l'aiuto in linea: Help.

Potete anche consultare il file `/usr/src/linux/Documentation/Configure.help` per avere un testo di aiuto in merito a ciascuna opzione, secondo l'ordine in cui compaiono. Nella sua intestazione,

inoltre, troverete dei collegamenti che vi porteranno a numerose traduzioni.

Ecco dunque una lista, per quanto non esaustiva, delle opzioni e delle scelte raccomandate per tali opzioni, arricchite da spiegazioni quando necessario. Le opzioni non descritte qui di seguito sono a vostra discrezione. Lasciarle "come sono" è, in genere, una buona idea.

- Prompt for development and/or incomplete code/drivers: **y**.
- Processor family: come avrete indovinato, si riferisce al tipo di processore installato sulla vostra macchina. Indicate `PPro/6x86MX` se il vostro processore è un **Intel** *`Pentium Pro`*, *`Pentium II`*, *`Celeron`* o superiore, oppure un *`Cyrix 6x86`* o *`Cyrix MII`*.
- Maximum Physical Memory: se avete meno di un GB di RAM, indicate 1 GB, in caso contrario 2 GB.
- Math emulation: **n**.
- MTRR (Memory Type Range Register) support: **y**. Anche se il vostro processore non ha il supporto per questa caratteristica, non vi sono conseguenze negative nell'attivarla.
- Symmetric multi-processing support: scegliete **y** solo se disponete di una macchina multi-processore!
- Enable loadable module support: **y**.
- Set version information on all symbols for modules: **n**.
- Kernel module loader: **y**.
- Networking support: **y** – anche se non siete collegati a una rete! Infatti ne avrete bisogno almeno per l'interfaccia di ***loopback***.
- PCI support: **y** – a meno che sulla vostra macchina il bus PCI sia del tutto assente.
- PCI access mode: lasciate come Any.
- PCI quirks: **y**.

- Backward-compatible /proc/pci: **y**.
- MCA support: **n** – a meno che non abbiate un computer con questo tipo di bus (ad esempio macchine **IBM** della serie PS/2).
- SGI Visual Workstation support: **n** – a meno che non sappiate bene quello che state facendo!
- System V IPC: **y**.
- BSD Process Accounting: **y**.
- Sysctl support: **y**.
- Kernel support for a.out binaries: **m**.
- Kernel support for ELF binaries: **y**.
- Kernel support for MISC binaries: **m**.
- Kernel support for JAVA binaries (obsolete): come dichiarato, è del tutto obsoleto – **n**.
- Parallel port support: **y** o **m**, la scelta è a vostra discrezione.
- PC-style hardware: potete scegliere **y** solo se avete indicato **y** nella precedente opzione Parallel port support. In caso contrario, dovete rispondere **m** e aggiungere la riga `alias parport_lowlevel parport_pc` al file di configurazione `/etc/conf.modules`.
- Support foreign hardware: **n**.
- Advanced Power Management BIOS support: **y**, se la vostra scheda madre offre il supporto per questa caratteristica.
- Ignore USER SUSPEND: **n**.
- Enable PM at boot time: **n**.
- Make CPU Idle calls when idle: **n**.
- Enable console blanking using APM: **y**.
- Power off on shutdown: **y**.
- Ignore multiple suspend: **y**.

- Ignore multiple suspend/resume cycle: **y**.
- RTC stores time in GMT: **n** se il vostro *PC* segue l'ora locale, **y** se invece si basa sul GMT.
- Allow interrupts during APM BIOS calls: **n** (ma leggete il testo di aiuto!)
- Plug and Play support: **y** – ma forse sapete già che tutto quello che questa opzione fa è chiedere al *BIOS* PNP informazioni in merito alle schede PNP, se ne possedete qualcuna (ricordate che, sul piano puramente tecnico, il PNP non significa nulla per quanto riguarda i dispositivi PCI).
- Auto-probe for parallel devices: **m** se disponete di dispositivi collegati alla porta parallela, altrimenti **n**.
- Normal PC floppy disk support: **m**.
- Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support: se sulla vostra macchina sono presenti dispositivi IDE (com'è tipicamente il caso) rispondete **y**.
- Use old disk-only driver on primary interface: **n**.
- Include IDE/ATA-2 DISK support: se il vostro sistema esegue l'avvio a partire da un disco IDE rispondete **y**. Se possedete dischi IDE, ma eseguite il boot da un disco SCSI, potete indicare **m**.
- Include IDE/ATAPI CDROM support: **m** se possedete un lettore CDROM di tipo IDE.
- Include IDE/ATAPI TAPE support: se disponete di un dispositivo IDE di backup su nastro rispondete **y** o **m**.
- Include IDE/ATAPI FLOPPY support: indicate **y** o **m** se avete, ad esempio, un drive ZIP di tipo IDE.
- SCSI emulation support: rispondete **m** se possedete un masterizzatore IDE, altrimenti **n**.
- Generic PCI IDE chipset support: **y**.

- Generic PCI bus-master DMA support: **y**.
- Boot off-board chipsets first support: **n**.
- Use DMA by default when available: **y**.
- Loopback device support: **m**.
- Network block device support: **n**.
- Multiple devices driver support: **n** – a meno che non vogliate sperimentare la tecnologia RAID. In tal caso, consultate il `RAID-HOWTO`.
- RAM disk support: **n**.
- XT hard disk support: **n** – non vorrete forse dirci che avete dei dischi di questo tipo? **:-)**
- Parallel port IDE device support: scegliete **m** se avete dispositivi di questo tipo, **n** in caso contrario. Se rispondete **m** dovrete poi specificare quale tipo di dispositivo desiderate supportare, e quali protocolli. Fate riferimento al file di aiuto relativo per saperne di più. Non vi sono soluzioni generiche per questo punto, se non compilare tutto come moduli.
- Packet socket: **m**.
- Kernel/User netlink socket: **y**.
- Routing messages: **n**.
- Netlink device emulation: **m**.
- Network firewalls: **y**, a meno che non intendiate rinunciare completamente all'IP *masquerading* (più macchine nascoste dietro una singola connessione a *Internet*) o, più semplicemente, all'uso di un *firewall*, nel qual caso dovreste rispondere **n**.
- UNIX domain sockets: **y** – altrimenti *X* non potrà funzionare.
- TCP/IP networking: **y**.
- IP: multicasting: **n**.

- IP: advanced router: se sapete quello che state facendo, qui potete rispondere **y** – in caso contrario, indicate **n**.
- IP: firewalling: se avete intenzione di installare un *firewall* o un IP *masquerading*, rispondete **y**, altrimenti **n**. Perché il *masquerading* funzioni è necessario rispondere **y** anche alle seguenti opzioni: IP: always defragment (required for masquerading), IP: masquerading e IP: ICMP masquerading.
- IP: optimize as router not host: **n**, a meno che la macchina in questione non sia effettivamente impiegata come router.
- IP: TCP syncookie support: **y** se siete connessi a una rete – consultate anche l'aiuto relativo a questo argomento.
- IP: Allow large windows (not recommended if <16MB of memory): **y**.
- CPU is too slow to handle full bandwidth: **n**, a meno che non siate connessi per mezzo di un'interfaccia ad altissima velocità (*gigabit Ethernet*, FDDI, etc.)
- SCSI support: **y** se possedete uno (o più) controller e dispositivi SCSI, un drive ZIP su porta parallela oppure un masterizzatore IDE, in caso contrario indicate **n**. Scegliete **y** per l'opzione SCSI disk support se il ***boot*** viene effettuato a partire da un disco SCSI, e non **m**! Inoltre scegliete **m** per l'opzione SCSI generic support se possedete un masterizzatore (SCSI o IDE), e rispondete in maniera appropriata per quanto riguarda dispositivi di altro tipo. Quando verrà il momento di specificare la marca e il modello del vostro controller SCSI (o dei controller, se ne possedete più d'uno), consultate il file `/etc/conf.modules`: l'installazione **Linux-Mandrake** avrà già determinato quali sono i driver necessari.
- Probe all LUNs on each SCSI device: consultate l'aiuto relativo – come indicazione generale, **n**.
- Verbose SCSI error reporting (kernel size +=12K): **n**.
- SCSI logging facility: **n**.

- IOMEGA parallel port (ppa - older drives): **m** solo se disponete di un drive ZIP di vecchia generazione!
- IOMEGA parallel port (imm - newer drives): se avete un drive ZIP piuttosto recente, qui dovreste rispondere **m** .
- Network device support: **y** se nella vostra macchina è presente un'interfaccia di rete, e/o se desiderate connettervi a *Internet* per mezzo di un modem, **n** in caso contrario.
- Dummy net driver support: **m**.
- ETHERNET (10 or 100Mbit): **y** se disponete di una o più schede *Ethernet*. In seguito dovrete scegliere i driver appropriati per la vostra scheda (o schede) *Ethernet*.
- PPP (point-to-point) support: **y** o **m** se desiderate connettervi a *Internet* per mezzo di un modem.
- SLIP (serial line) support: **n**.
- IrDA subsystem support: **y** o **m** se sul vostro *PC* sono presenti dispositivi a infrarossi. In tal caso dovrete rispondere **y** o **m** in una delle opzioni che vengono subito dopo: IrLAN protocol se possedete un ricevitore/trasmettitore a infrarossi per comunicare con altri *PC* che presentano lo stesso tipo di interfaccia (emulazione di *Ethernet*), IrCOMM protocol se avete un dispositivo a infrarossi che emula una porta seriale, IrLPT protocol per dispositivi a infrarossi che emulano una porta parallela. Indicate **y** per le opzioni IrDA protocol options e Cache last LSAP, **n** per Fast RRs (ma consultate l'aiuto relativo a questo argomento), **n** per Debug information, **n** per IrLAP compression a meno che non intendiate sperimentare questa caratteristica (consultate l'aiuto in merito), **y** o **m** per IrTTY (uses Linux serial driver) e IrPORT (IrDA serial driver); segue la sezione relativa al supporto dei diversi tipi di chip per l'infrarosso, scegliete il tipo in vostro possesso (per determinare di quale chip si tratta fate riferimento alla documentazione che accompagna il vostro hardware).

- ISDN support: rispondete **y** se possedete un adattatore ISDN **interno**. Se vi collegate a *Internet* usando questo tipo di connessione, è indispensabile anche rispondere **y** all'opzione Support synchronous PPP. Dovrete poi chiedere al vostro fornitore di servizi *Internet* se supporta la compressione *Van Jacobson*, in maniera da poter fare la scelta corretta riguardo l'opzione Use VJ-compression with synchronous PPP. Rispondete **n** a Support generic MP (RFC 1717) (ma consultate anche l'aiuto in linea), **n** a Support audio via ISDN (anche in questo caso date uno sguardo al file di aiuto) e a Support ISDN diversion services. Segue la scelta del driver per la vostra scheda ISDN: fate riferimento alla documentazione che accompagna il vostro hardware.
- Support non-SCSI/IDE/ATAPI CDROM drives: **n**, a meno che non possediate un lettore di CDROM con un'interfaccia proprietaria. Molto rari al giorno d'oggi.
- Virtual terminal: **y**.
- Support for console on virtual terminal: **y**.
- Standard/generic (dumb) serial support: **y**.
- Support for console on serial port: **n**.
- Extended dumb serial driver options: **n**.
- Non-standard serial port support: **n**.
- Unix98 PTY support: **y** – non modificate il valore predefinito (256) dell'opzione Maximum number of Unix98 PTYs in use (0-2048) .
- Parallel printer support: **m** se al vostro computer è collegata una stampante su porta parallela. In tal caso, rispondete **y** anche per l'opzione Support IEEE1284 status readback.
- Mouse Support (not serial mice): Come suggerito dal nome. Se il vostro mouse non è collegato a una porta COM qui dovreste rispondere **y**, e poi **y** o **m** per l'opzione che specifica il tipo di mouse di cui disponete. In caso di dubbi, consultate il file di aiuto relativo

a ciascuna di queste opzioni. Com'è specificato nell'aiuto, per ogni tipo inusuale di mouse, che non appartiene al tipo seriale né a quello PS/2, fate riferimento al `Busmouse-HOWTO`. Prestate particolare attenzione per quanto riguarda i computer portatili.

- QIC-02 tape support: **y** se avete questo tipo particolare, non SCSI, di unità a nastro.
- Watchdog Timer Support: **n**.
- /dev/nvram support: **n**.
- Enhanced Real Time Clock Support: **y**.
- Video For Linux: se possedete una scheda con funzioni TV, una scheda radio o una *QuickCam*, rispondete **y**. Poi indicate **y** o **m**, se preferite, per le opzioni relative al vostro dispositivo. Anche in questo caso la documentazione che accompagna il vostro hardware può rivelarsi molto utile.
- Joystick support: **y** o **m** se avete un joystick e volete usarlo. Dovrete poi scegliere il driver appropriato per il vostro joystick: consultate il testo di aiuto e la documentazione che accompagna il vostro hardware.
- Ftape (QIC-80/Travan) support: **y** se possedete una unità a nastro connessa al controller del disco floppy. Quindi consultate il testo di aiuto riguardo le varie opzioni.
- Quota support: **n** – se qui rispondete **y**, significa che sapete di cosa stiamo parlando **:-)**
- Kernel automounter support: **n**.
- DOS FAT fs support: **y** o **m**, a meno che proprio non abbiate nessuna necessità di accedere a floppy o partizioni con filesystem MS-DOS/*Windows* da *GNU/Linux*.
- MSDOS fs support: **m**.
- UMSDOS: Unix-like filesystem on top of standard MSDOS filesystem: **n**.

- VFAT (Windows-95) fs support: **m** – include il supporto per FAT32.
- ISO 9660 CDROM filesystem support: **m**.
- Microsoft Joliet CDROM extensions: **y**.
- /proc filesystem support: **y**.
- /dev/pts filesystem for Unix98 PTYs: **y**.
- Second extended fs support: **y**.
- NFS filesystem support: **y** se la vostra macchina è un **client** NFS. In caso contrario, **n**.
- NFS server support: **y** se la vostra macchina avrà il compito di fungere da **server** NFS.
- SMB filesystem support (to mount WfW shares etc.): **y** se volete accedere alle partizioni di un server sul quale gira *Windows* (*9x* o *NT*); in caso contrario, scegliete **n**. Questa opzione può essere ignorata (scegliendo **n**) se intendete installare un **server** SMB.
- Codepage 437 (United States, Canada): **m**.
- Codepage 850 (Europe): **m**.
- NLS ISO 8859-1: **m**.
- NLS ISO 8859-15: **m**.
- VGA text console: **y**.
- Video mode selection support: se intendete usare il ***framebuffer***, scegliete **y**. Il ***framebuffer*** vi permette di usare delle console virtuali molto più gradevoli alla vista, e in più offre anche un logo carino al momento dell'avvio del sistema **:-)** Tuttavia non vi impedisce affatto di utilizzare un server *X*. Scegliete **y** anche per l'opzione Support for frame buffer devices (EXPERIMENTAL), e di nuovo **y** to VESA VGA graphics console.
- Sound card support: qui scegliete **m** se nel vostro computer è presente una scheda audio, e consultate il file `/etc/conf.modules`

per determinare qual è il driver appropriato; questo presuppone che voi abbiate già configurato la vostra scheda audio, con `sndconfig`.

- Magic SysRq key: **n**.

E voilà! La fase di configurazione è finalmente terminata. Salvate le modifiche che avete apportato e uscite.

Il file di configurazione è `/usr/src/linux/.config`. È buona norma farne una copia di backup! Il luogo più appropriato dove salvare questa copia è la directory personale dell'utente `root`. Dato che la configurazione cambia molto poco tra le versioni incrementali di kernel che appartengono alla stessa serie (ovvero fra due versioni del kernel 2.2.x, oppure kernel della serie sperimentale 2.3.x), potete utilizzarla ancora per configurare versioni future del kernel.

Adesso è il momento di passare alla compilazione vera e propria.

## Compilazione del kernel e dei moduli, installazione dei moduli

Piccola precisazione prima di cominciare: se state per ricompilare un kernel con esattamente lo stesso numero di versione di quello già presente sul vostro sistema, per prima cosa è necessario cancellare i vecchi moduli. Ad esempio, se volete ricompilare la versione 2.2.10 del kernel, dovete prima cancellare la directory `/lib/modules/2.2.10`.

La compilazione del kernel e dei moduli, e poi l'installazione dei moduli, è compiuta con una sola riga di istruzioni:

```
$ make dep \&\& make bzImage \&\& make modules \&\&
  make modules_install
```

Se vi state chiedendo qual è lo scopo di questo `&&`, ecco a voi la spiegazione: con la riga `a && b` specifichiamo che per primo dev'essere eseguito il comando `a`, mentre il comando `b` viene lanciato in esecuzione **se e soltanto se** `a` è stato portato a termine con successo. Potete quindi immaginare qual è lo scopo della riga di comando vista in pre-

cedenza: se uno dei comandi fallisce, quelli che seguono non verranno eseguiti. Un'altra conseguenza di un ipotetico fallimento è la possibile presenza di un bug nel kernel! Se questo si verifica, per favore inviateci un rapporto dettagliato.

Un'ultima cosa: la compilazione di un nuovo kernel non renderà quello vecchio non funzionante! Se la compilazione non ha buon esito, non significa che il vostro sistema non sarà più in grado di riavviarsi. Per impedire al sistema di effettuare un boot regolare dovreste fare qualcosa di veramente stupido – ipotesi che, in tutta onestà, non potrà avverarsi se seguirete alla lettera le istruzioni contenute in questo capitolo **:-)**

## Installazione del nuovo kernel

Adesso che il vostro nuovo kernel è stato compilato senza intoppi, tutto quello che vi resta da fare è installarlo. Di nuovo, per amor di precisione e per identificare in maniera chiara i vostri kernel, è preferibile attenersi a una convenzione precisa per quanto riguarda i nomi di questi ultimi. Supponiamo che stiate per installare un kernel `2.2.17`. I comandi da digitare sono i seguenti:

```
$ make install
```

Dopo questo comando, dovete ancora effettuare l'aggiornamento del Boot Loader usato sul vostro sistema. Ci sono due possibilità: *GRUB* o *LILO*. Notate che al momento il Boot Loader standard utilizzato da **Linux-Mandrake** è *GRUB*.

### Aggiornamento di GRUB

Adesso dovete modificare il file `/boot/grub/menu.lst`. Questo è l'aspetto tipico di un file `menu.lst` subito dopo aver installato una distribuzione **Linux-Mandrake** e prima di qualsiasi modifica:

```
timeout 5
```

```
color black/cyan yellow/cyan
i18n (hd0,4)/boot/grub/messages
keytable (hd0,4)/boot/fr-latin1.klt
default 0

title linux
kernel (hd0,4)/boot/vmlinuz root=/dev/hda5

title failsafe
kernel (hd0,4)/boot/vmlinuz root=/dev/hda5  failsafe

title Windows
root (hd0,0)
makeactive
chainloader  +1

title floppy
root (fd0)
chainloader +1
```

Questo file si compone di due parti: l'intestazione, che contiene le opzioni più comuni (le prime cinque righe), e le diverse sezioni (o entrate), ciascuna delle quali corrisponde a un diverso kernel di *GNU/Linux* oppure a un altro OS specificato. `timeout 5` determina la quantità di tempo che avete a disposizione prima che *GRUB* lanci il sistema operativo (o il kernel) predefinito. Questa opzione predefinita è determinata da `default 0`, che significa che la prima sezione è quella di default. La riga `color` specifica i colori da usare per i menu. La riga `i18n`, invece, comunica al sistema dove si trova il messaggio di benvenuto da presentare all'utente: `(hd0,4)` significa che si trova sulla quinta partizione del primo disco rigido. `keytable (hd0,4)/boot/fr-latin1.klt` informa *GRUB* riguardo il tipo di tastiera nel caso abbiate bisogno di passare manualmente delle opzioni al momento del boot.

Quindi incontriamo la parte contenente le sezioni. In totale sono quattro: `linux`, `failsafe`, `windows`, e `floppy`.

1. La riga di opzioni per la sezione `linux` comunica a *GRUB* che vogliamo sia caricata l'immagine di boot che si chiama `vmlinuz` e si trova nella directory `/boot/` nella quarta partizione del primo

disco rigido. L'opzione `root=/dev/hda5` non è un'opzione relativa a *GRUB*, si tratta in effetti di un'informazione passata al kernel per informarlo che il filesystem di root (`/`) si trova nella partizione `/dev/hda5`.

2. La sezione `failsafe` assomiglia molto alla precedente, fatta eccezione per il fatto che verrà comunicato un argomento al kernel (`failsafe`) per ordinargli di entrare in modalità *single user* ("monoutente") o *rescue* ("salvataggio").
3. La sezione `Windows` semplicemente ordina a *GRUB* il settore di boot della prima partizione, che probabilmente contiene un settore di boot di *Windows*.
4. L'ultima sezione, `floppy`, non fa altro che eseguire il boot a partire da un floppy disk inserito nel primo lettore, qualunque sia il sistema operativo installato su di esso.

> **Nota:** A seconda del livello di sicurezza che usate sul vostro sistema, alcune delle voci qui descritte potrebbero non essere presenti nel vostro file di configurazione.

Adesso arriviamo al nocciolo della questione: dobbiamo aggiungere un'altra sezione *GNU/Linux*, in modo da poter effettuare il boot con il nuovo kernel. In questo esempio verrà inserita prima delle altre entrate, ma niente vi impedisce di collocarla in un'altra posizione[1]:

```
title linux-experimental
kernel (hd0,4)/boot/vmlinuz-2.2.17 root=/dev/hda5
```

Non dimenticate di modificare questa entrata in base alla vostra effettiva configurazione hardware! Il filesystem `root` di *GNU/Linux* qui

1. N.d.T.: Giusto per precauzione, conviene non usare il nuovo kernel come quello predefinito per il boot.

si trova in `/dev/hda5`, ma molto probabilmente sulla vostra macchina risiede altrove. Il vostro sistema è pronto, potete effettuare un riavvio del computer e godervi il vostro kernel nuovo di zecca!

Se avete compilato il kernel abilitando l'opzione relativa al framebuffer, probabilmente vorrete usarlo: in tal caso, dovete istruire il kernel riguardo la risoluzione di partenza. La lista di modi video è disponibile nel file `/usr/src/linux/Documentation/fb/vesafb.txt` (soltanto nell'eventualità che si tratti del framebuffer VESA! Altrimenti, fate riferimento al file corrispondente). Per il modo 800x600 a 32 bit[2], il numero della modalità video è `0x315`, perciò dovete aggiungere la direttiva:

```
vga=0x315
```

in maniera tale che la sezione diventi più o meno così:

```
title linux-experimental
kernel (hd0,4)/boot/vmlinuz-2.2.17 root=/dev/hda5 vga=0x315
```

Per avere ulteriori informazioni, per favore consultate le pagine info relative a *GRUB* (`info grub`).

## Aggiornamento di `LILO`

Il modo più semplice di aggiornare *LILO* è usare *DrakBoot* (si veda il capitolo *DrakBoot*: modifica della configurazione di boot nel **Manuale utente**). In alternativa, potete modificare a mano il file di configurazione secondo le istruzioni che troverete qui di seguito.

2. 8 bit significa $2^8$ colori, cioè 256; 16 bit significa $2^{16}$ colori, cioè 64k, o 65536, colori; alla risoluzione di 24 bit, come pure di 32, i colori sono codificati utilizzando 24 bit, cioè $2^{24}$ colori possibili, in altre parole 16M, o un po' più di 16 milioni di colori.

Per quanto riguarda *LILO*, dovrete aggiornare il file `/etc/lilo.conf`. Quello che segue è un tipico esempio di `lilo.conf` subito dopo aver installato il pacchetto *LILO* e prima di ogni modifica:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz-2.2.9-19mdk
        label=linux
        root=/dev/hda1
        read-only
other=/dev/hda2
        label=dos
        table=/dev/hda
```

> **Attenzione**
>
> Questo esempio presuppone che voi usiate *LILO* come boot loader principale! Se invece utilizzate *System Commander*, la direttiva `boot=` sarà diversa, e probabilmente non ci sarà nessuna sezione `other`.

Un file di configurazione `lilo.conf` consiste di una sezione principale seguita da una sezione per ogni sistema operativo (o kernel) dal quale si può effettuare il boot. Nell'esempio visto in precedenza, la sezione principale consiste delle seguenti istruzioni:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
```

La direttiva `boot=` comunica a *LILO* dove deve installare il suo settore di boot; nel nostro esempio si tratta del MBR (*Master Boot*

*Record*) del primo disco rigido IDE. Se desiderate creare un floppy disk basato su *LILO*, non dovete far altro che sostituire `/dev/hda` con `/dev/fd0` **:-)** la direttiva `prompt` impone a *LILO* di mostrare il *prompt* al momento del boot e di continuare con la procedura normale dopo 5 secondi (`timeout=50`). Se cancellate la direttiva `timeout=`, *LILO* attenderà finché non avrete digitato qualcosa.

Segue poi una sezione `linux`:

```
image=/boot/vmlinuz-2.2.9-19mdk
        label=linux
        root=/dev/hda1
        read-only
```

Una sezione `linux` comincia sempre con la direttiva `image=`, seguita dal percorso completo di un kernel *GNU/Linux*. Come ogni altra sezione, contiene un'etichetta `label=` che funge da identificatore univoco. La direttiva `root=` comunica a *LILO* qual è la partizione che ospita il filesystem `root` per questo sistema *GNU/Linux*. Ovviamente potrebbe essere diverso nel vostro caso. La direttiva `read-only` ordina a *LILO* di montare questo filesystem `root` in modalità di sola lettura al momento dell'avvio del sistema: se questa direttiva non è presente, il fatto vi verrà segnalato con un messaggio di errore.

Subito dopo viene la sezione relativa a *Windows*:

```
other=/dev/hda2
        label=dos
        table=/dev/hda
```

Una sezione che comincia con `other=`, in effetti, viene usata da *LILO* per avviare qualsiasi sistema operativo diverso da *GNU/Linux*: l'argomento passato a questa direttiva è il luogo in cui si trova il settore di boot di questo sistema operativo, e nel caso specifico si tratta di un sistema *Windows*. Per trovare il settore di boot, situato all'inizio della partizione che ospita questo secondo sistema operativo, *GNU/Linux* ha anche bisogno di sapere dove si trova la tabella delle partizioni, che gli permetterà di individuare la partizione in questione: questa informazione la ottiene grazie alla direttiva `table=`. La direttiva `label=`,

esattamente come in una sezione `linux`, serve per identificare il tipo di sistema operativo.

Prima di aggiungere la nostra sezione `linux`, vogliamo prendere due uccelli con una fava **:-)** Il prossimo passo sarà la composizione di un messaggio che venga mostrato prima che compaia il *prompt* di *LILO*, per spiegare come usare *LILO*:

```
$ cat >/boot/message <<EOF
> Benvenuti, questo è LILO (LInux LOader).
> Per elencare le scelte possibili, premi il tasto TAB.
> Sono disponibili:
> * esp : avvia Linux-Mandrake con un nuovo kernel
> * linux : il kernel originale di Linux-Mandrake
> * dos : Windows
> Premendo INVIO senza specificare un nome verrà effettuato
> il boot usando la prima immagine della lista, cioè esp.
> EOF
$
```

E voilà! Per mostrare questo messaggio ogni volta che si avvia il sistema dovete soltanto aggiungere la direttiva:

```
message=/boot/message
```

nella sezione principale del file `lilo.conf`. Adesso però è necessario aggiungere la sezione *GNU/Linux* che provvede ad avviare il sistema usando il nuovo kernel. In questo esempio la inseriremo in cima alla lista, ma nulla vi impedisce di scegliere un'altra posizione:

```
image=/boot/vmlinuz-2.2.17
        label=exp
        root=/dev/hda1
        read-only
```

Se avete compilato il vostro kernel abilitando l'opzione relativa al framebuffer, fate riferimento al paragrafo corrispondente, visto in precedenza, relativo a *GRUB*, la differenza adesso è che l'opzione si trova su una propria riga:

```
vga=0x315
```

Ecco l'aspetto che il nostro `lilo.conf` dovrebbe avere dopo tutte le modifiche apportate, compreso l'inserimento di qualche commento supplementare (tutte le righe che cominciano con un #), che verrà ignorato da *LILO*:

```
#
# Sezione principale
#
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
# Il nostro messaggio di benvenuto
message=/boot/message
# Mostra il prompt...
prompt
# ... aspetta 5 secondi
timeout=50
#
# Il nostro nuovo kernel è l'immagine predefinita
#
image=/boot/vmlinuz-2.2.17
        label=exp
        root=/dev/hda1
        read-only
# Se viene usato il framebuffer VESA:
        vga=0x315
#
# Il kernel originale
#
image=/boot/vmlinuz-2.2.15-19mdk
        label=linux
        root=/dev/hda1
        read-only
#
# Sezione Windows
#
other=/dev/hda2
        label=dos
        table=/dev/hda
```

Non dimenticate di adattare il file in modo che corrisponda alla vostra effettiva configurazione! Il filesystem `root` di *GNU/Linux* qui si trova in`/dev/hda1`, ma certo potrebbe risiedere altrove sul vostro sistema; lo stesso vale per il sistema operativo *Windows*. Adesso che il file di configurazione è stato modificato in maniera appropriata, dovete dire a *LILO* di cambiare il settore di boot:

```
$ lilo
Added exp *
Added linux
Added dos
$
```

In questo modo avete la possibilità di compilare tutti i kernel che volete, e di aggiungere le sezioni *GNU/Linux* necessarie per utilizzarli. Adesso non vi resta che riavviare il sistema per provare il nuovo kernel.

# Capitolo 15. La compilazione e l'installazione di software libero

Spesso i neofiti si domandano come si fa a installare del software libero direttamente dai sorgenti. Compilare personalmente i programmi è davvero facile, perché molti dei passi da seguire sono gli stessi a prescindere dal software specifico che desiderate installare. Lo scopo di questo capitolo è quello di guidare il principiante un passo alla volta, spiegandogli il significato di ogni mossa. Si presume che il lettore abbia una conoscenza minima di *Unix* (dei comandi `ls` o `mkdir`, ad esempio).

Questo capitolo costituisce soltanto una guida, non un manuale di riferimento; il che spiega perché al termine vengono proposti numerosi riferimenti, in maniera da poter soddisfare le domande che restano senza risposta. Questa guida molto probabilmente è suscettibile di miglioramenti, dunque ogni osservazione o correzione in merito al contenuto sarà accolta con piacere.

## Introduzione

Quello che distingue il software libero, detto anche "software liberamente distribuibile", dal software proprietario è la possibilità di accedere ai sorgenti dello stesso[1]. Questo comporta anche che il software libero viene sempre distribuito come archivi dei sorgenti. Gli utenti dei programmi liberamente distribuibili, dunque, devono compilare i sorgenti di persona prima di poter usare il software, e questo può sembrare piuttosto inusuale ai principianti.

1. Questo non è del tutto esatto, in quanto alcune case che producono software proprietario mettono a disposizione i sorgenti. Ma, a differenza di quello che succede con il software libero, l'utente finale non ha il permesso di utilizzarli come vorrebbe.

Esistono versioni compilate di molti programmi che appartengono al mondo del software libero: l'utente che va di fretta non deve far altro che installare questi binari precompilati. Una parte di questi programmi, tuttavia, non viene distribuita secondo questa modalità, oppure si tratta delle versioni iniziali che ancora non vengono rilasciate in forma binaria; se usate un sistema operativo poco diffuso, inoltre, oppure una piattaforma hardware piuttosto rara, una gran parte del software libero non è ancora stata compilata per il vostro sistema. È importante sapere che compilare di persona il software da installare vi permette di abilitare soltanto le opzioni che vi interessano, o di estendere la funzionalità del programma aggiungendo estensioni di vario tipo, in maniera da ottenere un software che sia perfettamente adeguato ai vostri bisogni.

## Prerequisiti

Per compilare software avete bisogno di:

- un computer con un sistema operativo funzionante,
- una conoscenza di base del sistema operativo che utilizzate,
- una certa quantità di spazio sul vostro disco rigido,
- un compilatore (in genere per il linguaggio `C`) e un programma di archiviazione (`tar`),
- un po' di cibo (in casi difficili la compilazione può impiegare un tempo considerevole). Un vero hacker mangia pizza - non caviale.
- qualcosa da bere (per la stessa ragione). Un vero hacker beve cola - per la caffeina.
- il numero di telefono del vostro amico smanettone che ricompila il kernel tutte le settimane,
- soprattutto pazienza, e in buona quantità!

Compilare dei sorgenti in genere non presenta grandi problemi, ma se non lo avete mai fatto prima il più piccolo intralcio può bloccarvi all'istante. Lo scopo di questo capitolo è quello di mostrarvi come potete evitare situazioni di questo tipo.

# Compilazione

## Principi di base

Per tradurre un codice sorgente in un file binario è necessario effettuare una ***compilazione***; in genere i sorgenti sono scritti in *C* o *C++*, che sono i linguaggi più diffusi nella comunità del software libero (in particolare per *Unix*). Una parte dei programmi liberamente distribuibili è scritta in linguaggi che non richiedono compilazione (il *Perl*, ad esempio, o il linguaggio della *shell*), ma necessitano comunque di una configurazione.

La compilazione di programmi scritti in *C* viene effettuata per mezzo di un compilatore *C*, ovviamente, che di solito è `gcc`, il compilatore liberamente distribuibile creato dal progetto GNU (potrete saperne di più visitando la URL http://www.gnu.org/). La compilazione di un intero pacchetto software è un'operazione complessa, che si basa sulla compilazione in sequenza dei diversi file sorgenti (per una serie di motivi, è più semplice per il programmatore distribuire parti diverse del proprio lavoro in file separati). Al fine di rendere l'intero processo più semplice, queste operazioni ripetitive vengono gestite da un programma chiamato `make`.

## Le quattro fasi della compilazione

Per capire come funziona la compilazione, così da poter risolvere eventuali problemi, dovete conoscere le quattro fasi da cui è composta. Lo scopo è quello di tradurre poco a poco un testo scritto in un linguaggio che è comprensibile per un essere umano ben allenato

(e cioè il linguaggio *C*) in un linguaggio comprensibile da una macchina (o da un essere umano **molto** ben allenato, e anche in tal caso non sarebbero molti). `gcc` esegue quattro programmi uno dopo l'altro, ciascuno dei quali gestisce una fase della compilazione:

1. `cpp`: Il primo passo consiste nel rimpiazzare le direttive (del ***preprocessore***) con delle istruzioni in *C* vere e proprie. Questo significa, di norma, inserire un header (`#include`) o definire una macro (`#define`). Al termine di questa fase viene generato un puro codice *C*.
2. *cc1*: Questa fase consiste nella conversione del codice *C* in ***linguaggio assembler***. Il codice generato dipende dalla architettura su cui deve girare il programma.
3. `as`: Questo passo consiste nella produzione di codice ***oggetto*** (o codice binario) a partire dal linguaggio assembler. Al termine di questa fase viene generato un file `.o`.
4. *ld*: Come ultimo passo (***linkage***) vengono collegati tutti i file oggetto (`.o`) e le librerie relative, e viene prodotto un file eseguibile.

## Struttura di una distribuzione

Una distribuzione di software libero strutturata in maniera corretta presenta sempre la medesima organizzazione dei file:

- Un file `INSTALL`, che descrive la procedura di installazione.
- Un file `README`, che contiene informazioni generali riguardanti il programma (breve descrizione, autore, URL del sito dove lo si può scaricare, documentazione relativa al programma, collegamenti utili, etc.). Se il file `INSTALL` non è presente, di solito una breve descrizione della procedura di installazione è contenuta nel file `README`.

- Un file `COPYING`, che contiene la licenza o descrive le condizioni secondo le quali può essere distribuito il software. A volte è rimpiazzato da un file `LICENSE`.
- Un file `CONTRIB` o `CREDITS`, che contiene una lista delle persone che hanno contribuito al progetto (partecipazione diretta, suggerimenti utili, programmi di terze parti, etc.).
- Un file `CHANGES` (o, meno frequentemente, un file `NEWS`), che contiene una lista degli ultimi miglioramenti e dei bug eliminati.
- Un file `Makefile` (si veda la sezione *`make`*, pag. 204), che permette di compilare il software (è un file di cui ha bisogno `make`). Spesso questo file all'inizio non esiste e viene generato durante il processo di configurazione.
- Molto spesso, un file `configure` o `Imakefile`, che permette di generare un nuovo file `Makefile`,
- Una directory che contiene i sorgenti, e dove viene in genere collocato il file binario alla fine della compilazione. Spesso viene chiamata `src`.
- Una directory che contiene la documentazione relativa al programma (normalmente in formato `man` o *`Texinfo`*), il cui nome spesso è `doc`.
- A volte, una directory che contiene dati specifici del software (si tratta, tipicamente, di file di configurazione, esempi di dati prodotti, o file che costituiscono risorse di vario tipo).

# Decompressione

## Archivi `tar.gz`

Il formato standard[2] per la compressione nei sistemi *`Unix`* è il formato

`gzip`, sviluppato dal progetto GNU, che viene considerato uno dei migliori strumenti per la compressione generica dei file.

`gzip` viene spesso associato con un programma che si chiama `tar`. `tar` è il superstite di tempi antidiluviani, quando gli operatori di computer archiviavano i loro dati su nastri. Oggi floppy disk e CDROM hanno rimpiazzato i nastri, ma `tar` viene ancora impiegato per creare archivi. Tutti i file presenti in una directory, ad esempio, possono essere riuniti in un unico file; quest'ultimo può essere facilmente compresso con `gzip`.

Questo è il motivo per cui gran parte del software libero è disponibile sotto forma di archivi `tar`, compressi con `gzip`. La loro estensione, perciò, è `.tar.gz` (o anche `.tgz` per brevità).

## Uso di `GNU Tar`

Per decomprimere un archivio di questo tipo, è possibile usare `gzip` e poi `tar`. Ma la versione GNU di `tar` (`gtar`) permette di usare `gzip` ***"al volo"*** (ingl. *`on-the-fly`*), e di decomprimere un archivio quasi senza accorgersene (e senza il bisogno di spazio aggiuntivo su disco).

L'uso di `tar` segue questa sintassi:

```
tar <opzioni di file> <.tar.gz file> [file]
```

L'opzione `<files>` non è necessaria. Se omessa, l'operazione richiesta verrà effettuata sull'intero archivio. Questo argomento non dev'essere specificato per estrarre il contenuto di un archivio `.tar.gz`.

2. Un nuovo programma, chiamato `bzip2`, più efficiente nei confronti dei file di testo (e più esigente in materia di potere di calcolo), viene usato sempre più di frequente. Si veda la sezione successiva *`bzip2`*, pag. 196 che si occupa specificamente di esso.

Ad esempio:

```
$ tar xvfz guile-1.3.tar.gz
-rw-r--r-- 442/1002      10555 1998-10-20 07:31 guile-1.3/Makefile.in
-rw-rw-rw- 442/1002       6668 1998-10-20 06:59 guile-1.3/README
-rw-rw-rw- 442/1002       2283 1998-02-01 22:05 guile-1.3/AUTHORS
-rw-rw-rw- 442/1002      17989 1997-05-27 00:36 guile-1.3/COPYING
-rw-rw-rw- 442/1002      28545 1998-10-20 07:05 guile-1.3/ChangeLog
-rw-rw-rw- 442/1002       9364 1997-10-25 08:34 guile-1.3/INSTALL
-rw-rw-rw- 442/1002       1223 1998-10-20 06:34 guile-1.3/Makefile.am
-rw-rw-rw- 442/1002      98432 1998-10-20 07:30 guile-1.3/NEWS
-rw-rw-rw- 442/1002       1388 1998-10-20 06:19 guile-1.3/THANKS
-rw-rw-rw- 442/1002       1151 1998-08-16 21:45 guile-1.3/TODO
...
```

Tra le opzioni di `tar`:

- `v` rende `tar` prolisso. Questo significa che mostrerà sullo schermo tutti i file che trova nell'archivio. Se questa opzione viene omessa, l'operazione verrà eseguita in modo silenzioso.
- `f` è un'opzione indispensabile: se non è indicata, `tar` cercherà di usare un nastro invece di un file archivio (e cioè il dispositivo `/dev/rmt0`).
- `z` vi permette di usare `tar` con un archivio "gzippato" (cioè compresso con `gzip`, deve presentare un'estensione `.gz`). Se vi dimenticate di questa opzione, `tar` produrrà un messaggio d'errore. Viceversa, questa opzione non dev'essere impiegata con un archivio non compresso.
- `tar` vi consente di eseguire un certo numero di azioni su un archivio (estrazione, lettura, creazione, aggiunta di file...). Un'opzione stabilisce quale azione dev'essere compiuta dal programma:
  - `x`: vi permette di estrarre file dall'archivio.
  - `t`: elenca il contenuto dell'archivio.
  - `c`: vi permette di creare un archivio. Potete usarla per fare una copia di backup dei vostri file personali, ad esempio.

- `r`: consente di aggiungere file al termine dell'archivio. Non può essere usata con un archivio compresso.

### `bzip2`

Un nuovo formato di compressione, chiamato `bzip2`, ha cominciato a rimpiazzare `gzip` nell'uso normale. `bzip2` produce archivi più piccoli rispetto a `gzip`, ma non è ancora diventato uno standard. Le estensioni `.tar.bz2` hanno iniziato a diffondersi soltanto in tempi recenti.

`bzip2` viene usato anche per mezzo del comando `tar`, proprio come `gzip`. L'unico accorgimento da adottare è la sostituzione della lettera `z` con la `y`. Ad esempio:

```
$ tar xvfy foo.tar.bz2
```

Alcune distribuzioni invece usano, o hanno usato, l'opzione `I`:

```
$ tar xvfI foo.tar.bz2
```

Un altro metodo (che sembrerebbe più compatibile, ma richiede più tempo per essere digitato!):

```
$ tar --use-compress-program=bzip2 -xvf foo.tar.bz2
```

`bzip2` deve essere installato e incluso nella vostra variabile d'ambiente `PATH` prima di lanciare `tar`.

## Fatelo e basta!

### Il modo più semplice

Adesso che siete pronti a decomprimere l'archivio, non dimenticate di eseguire questa operazione in veste di amministratori del sistema (`root`). Sarà necessario compiere azioni che non sono permesse a un

utente normale, e anche se parte di quello che farete è alla portata di quest'ultimo, è più semplice essere `root` per tutto il tempo.

Il primo passo è quello di spostarsi nella directory `/usr/local/src` e copiarvi l'archivio. In tal modo, dovreste riuscire sempre a ritrovare l'archivio di partenza se il programma installato dovesse venir rimosso. Se non avete molto spazio a disposizione sul vostro disco rigido, salvate l'archivio su un floppy disk dopo aver installato il programma. Nulla vi vieta di cancellarlo, ma accertatevi di poterlo comunque reperire sul *Web* nel caso doveste averne bisogno.

La decompressione di un archivio `tar`, in genere, dovrebbe portare alla creazione di una nuova directory (potete controllare che ciò si verifichi grazie all'opzione `t`). Una volta creata, spostatevi in tale directory, in modo da esser pronti per continuare.

## Il modo più sicuro

I sistemi *Unix* (di cui *GNU/Linux* e *FreeBSD* costituiscono validi esempi) sono sistemi sicuri. Questo significa che gli utenti normali non possono né compiere operazioni che possono mettere a rischio il sistema (la formattazione di un disco rigido, ad esempio), né modificare i file di altri utenti. Tale condizione comporta anche un'immunizzazione del sistema per quanto riguarda i virus.

`root`, d'altra parte, può fare tutto - anche lanciare un programma malevolo. Avere a disposizione il codice sorgente vi permette di esaminarlo per escludere la presenza di codice malevolo (virus o cavalli di Troia). È meglio prestare la massima attenzione a questo problema[3].

Il nostro suggerimento, pertanto, è di creare un utente dedicato a compiti amministrativi (`free` o `admin`, ad esempio) usando il coman-

3. Un proverbio del mondo BSD dice: "Non fidarti mai di un pacchetto di cui non hai i sorgenti."

do `adduser`. Questo utente deve avere il permesso di scrivere nelle seguenti directory: `/usr/local/src`, `/usr/local/bin` e `/usr/local/lib`, come pure tutta la sotto-gerarchia di `/usr/man` (è anche possibile che debba avere il permesso di copiare file altrove). Vi raccomandiamo di rendere questo utente proprietario delle directory necessarie, oppure di creare un gruppo per lui e di rendere tali directory accessibili in scrittura per il gruppo.

Dopo aver preso queste precauzioni, potete seguire le istruzioni della sezione *Il modo più semplice*, pag. 196.

## Configurazione

Una conseguenza puramente tecnica del fatto di avere a disposizione i sorgenti creati dagli autori è il ***porting*** del software. Il software libero sviluppato per un sistema *Unix* può essere utilizzato su tutti i sistemi *Unix*, che siano aperti o proprietari, una volta effettuate alcune modifiche. Questo richiede una configurazione del software prima di procedere alla compilazione.

Esistono diversi sistemi di configurazione. Siete obbligati a usare quello scelto dall'autore del programma, e, talvolta, ne serve più d'uno. In genere potete:

- Usare *Autoconf* (consultate la sezione *Autoconf*, pag. 198) se esiste un file chiamato `configure` nella directory principale della distribuzione.
- Usare *Imake* (consultate la sezione *Imake*, pag. 202) se un file di nome `Imakefile` è presente nella directory principale della distribuzione.
- Eseguire uno ***script*** dalla shell (ad esempio `install.sh`) seguendo le istruzioni contenute nel file `INSTALL` (o nel file `README`)

`Autoconf`

## La teoria

*Autoconf* viene usato per configurare il software in maniera corretta. Crea i file richiesti dal processo di compilazione (`Makefile` ad esempio), e talvolta cambia i file sorgenti direttamente (ad esempio usando un file `config.h.in`).

Il principio di funzionamento di *Autoconf* è semplice:

- Il programmatore autore del software sa quali test sono richiesti per configurarlo (ad esempio: "quale versione di questa ***libreria*** state usando?"). Perciò li raccoglie in un file chiamato `configure.in`, seguendo una sintassi ben precisa.
- Quindi esegue *Autoconf*, che genera uno script di configurazione chiamato `configure` dal file `configure.in`. Questo script esegue i test richiesti quando il programma viene configurato.
- L'utente finale esegue lo script, e *Autoconf* configura tutto ciò che è necessario per la compilazione.

## Un esempio pratico

Un esempio dell'uso di *Autoconf*:

```
$ ./configure
loading cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for main in -lX11... yes
checking for main in -lXpm... yes
checking for main in -lguile... yes
checking for main in -lm... yes
checking for main in -lncurses... yes
checking how to run the C preprocessor... gcc -E
```

```
checking for X... libraries /usr/X11R6/lib, headers /usr/X11R6/include
checking for ANSI C header files... yes
checking for unistd.h... yes
checking for working const... yes
updating cache ./config.cache
creating ./config.status
creating lib/Makefile
creating src/Makefile
creating Makefile
```

È possibile aggiungere alcune opzioni per mezzo della linea di comando, in modo da avere un controllo maggiore su quello che viene generato da `configure`. Ad esempio:

```
$ ./configure --with-gcc --prefix=/opt/GNU
```

oppure (con *Bash*):

```
$ export CC=‘which gcc‘
$ export CFLAGS=-O2
$ ./configure --with-gcc
```

o:

```
$ CC=gcc CFLAGS=-O2 ./configure
```

## Cosa fare... se non funziona?

Se `configure` si blocca, in genere comparirà un errore del tipo `configure: error: Cannot find library guile`; la maggior parte degli errori dello script `configure` hanno questo aspetto.

Questo errore significa che lo script `configure` non è riuscito a trovare una libreria (nell'esempio citato si tratta della libreria `guile`). Il metodo seguito da `configure` è quello di compilare un breve programma di prova che usa tale libreria. Se la compilazione di tale programma fallisce, ciò significa che non sarà possibile compilare il software in questione. Di conseguenza, venite avvertiti dell'errore.

- Scoprite la causa dell'errore esaminando la parte finale del file `config.log`, che contiene un registro di tutti i passi seguiti nel corso della configurazione. Il compilatore *C* è sufficientemente chiaro nei suoi messaggi d'errore, e questo vi aiuterà a risolvere il problema.
- Controllate che la sunnominata libreria sia installata correttamente: se non lo è, installatela (dai sorgenti, o usando un file binario precompilato) e lanciate di nuovo `configure`. Un metodo efficace per controllare è quello di cercare il file che contiene i simboli della libreria, che è sempre `lib<name>.so`. Ad esempio,

  ```
  $ find / -name 'libguile*'
  ```

  o anche:

  ```
  $ locate libguile
  ```

- Controllate che il compilatore possa accedervi, il che significa che deve trovarsi in una delle seguenti directory: `/usr/lib`, `/lib`, `/usr/X11R6/lib` (oppure in una di quelle specificate dalla variabile d'ambiente `LD_LIBRARY_PATH`, spiegata in *Cosa fare... se non funziona?*, pag. 207 numero 5.b. Controllate che questo file sia una libreria digitando il comando `file libguile.so`.
- Controllate che i file header che corrispondono a tale libreria siano installati nel posto giusto (in genere `/usr/include`, oppure `/usr/local/include` o anche `/usr/X11R6/include`). Se non sapete quali sono gli header di cui avete bisogno, controllate che sia stata installata la versione per sviluppatori della libreria richiesta (ad esempio, `gtk+-devel` invece di `libgtk`). La versione per sviluppatori di una libreria fornisce i file "include" necessari per compilare software che faccia uso della suddetta libreria.
- Accertatevi di avere spazio sufficiente sul disco rigido (lo script `configure` ha bisogno di una certa quantità di spazio per dei file temporanei). Usate il comando `df -k` per mostrare lo spazio disponibile sulle partizioni del vostro sistema, e prestate attenzione a quelle piene o quasi piene.

Se non capite il messaggio di errore contenuto nel file `config.log`, non esitate a chiedere aiuto alla comunità del software libero (consultate la sezione *Supporto tecnico*, pag. 217).

Controllate inoltre se `configure` risponde sempre `No`, oppure se risponde `No` anche quando siete sicuri della presenza di una data libreria. Sarebbe davvero strano, ad esempio, che la libreria `curses` non fosse presente sul vostro sistema. In tal caso, la variabile `LD_LIBRARY_PATH` probabilmente è sbagliata!

## Imake

*Imake* vi permette di configurare un programma creando un file `Makefile` sulla base di semplici regole. Queste regole determinano quali file debbano essere compilati per costruire il file binario, e *Imake* genera il `Makefile` corrispondente. Queste regole sono state specificate in un file chiamato `Imakefile`.

La caratteristica più interessante di *Imake* è il fatto che fa uso di informazioni dipendenti dalla particolare ***installazione*** (ovvero dalla particolare architettura hardware). È molto comodo per applicazioni che usano *X Window System*. Ma *Imake* viene usato anche per molte altre applicazioni.

Il modo più semplice per usare *Imake* è quello di spostarsi nella directory principale dell'archivio decompresso, e lanciare poi lo script `xmkmf`, che richiama il programma `imake`:

```
$ xmkmf -a
imake -DUseInstalled -I/usr/X11R6/lib/X11/config
make Makefiles
```

Se il la vostra installazione non funziona correttamente, ricompilate e installate *X11R6*!

## Vari script da `shell`

Leggete il file `INSTALL` o `README` per avere più informazioni. Di solito dovrete lanciare un file del tipo `install.sh` o `configure.sh`. A questo punto, lo script di installazione è non interattivo (e stabilisce autonomamente ciò di cui ha bisogno), oppure vi chiede informazioni sul sistema (percorsi, ad esempio).

Se non riuscite a determinare quale sia il file da eseguire, potete digitare `./` (in una shell *Bash*), e poi premere due volte il tasto `TAB` (tasto di tabulazione). Nella sua configurazione predefinita, *Bash* completa automaticamente la linea di comando inserendo il nome di un file eseguibile contenuto nella directory corrente (e quindi un eventuale script di configurazione). Se è presente più di un file eseguibile, vi verrà presentata una lista: non dovete far altro che scegliere il file giusto.

Un caso particolare è quello della installazione di moduli *Perl* (ma non solo). L'installazione di tali moduli viene effettuata lanciando in esecuzione uno script di configurazione, che è scritto in *Perl*. Il comando da eseguire in genere è:

```
$ perl Makefile.PL
```

## Alternative

Alcune distribuzioni di software libero sono organizzate in modo non ottimale, soprattutto durante le prime fasi di sviluppo (ma l'utente è avvertito!). A volte richiedono di cambiare "a mano" alcuni file di configurazione. Normalmente questi file sono un file `Makefile` (consultate la sezione *make*, pag. 204) e un file `config.h` (questo nome è del tutto convenzionale).

Vi sconsigliamo queste modifiche, fatta eccezione per quegli utenti che sanno quello che fanno. Queste operazioni richiedono una buona conoscenza e una certa determinazione per riuscire, ma la pratica le rende banali.

## Compilazione

Adesso che il software è stato configurato in maniera corretta, tutto quello che resta da fare è compilarlo. Questa fase in genere è facile, e non presenta seri problemi.

### `make`

Lo strumento preferito per compilare sorgenti dalla comunità del software libero è `make`. Presenta due caratteristiche particolarmente interessanti:

- Lo sviluppatore risparmia tempo, perché `make` permette di gestire la compilazione del proprio progetto in maniera assai efficiente,
- L'utente finale può compilare e installare il software con pochi comandi dalla shell, anche se non ha nessuna conoscenza preliminare di come si sviluppa un programma.

Le azioni che devono essere eseguite per ottenere una versione compilata dei sorgenti sono conservate in un file chiamato normalmente `Makefile` o `GNUmakefile`. Quando `make` viene invocato, infatti, legge questo file – se esiste – nella directory corrente. Se non esiste, è sempre possibile specificare un altro file usando l'opzione `-f` del comando `make`.

### Regole

`make` opera in accordo con un sistema di ***dipendenze***. Pertanto compilare un file binario (`target`) richiede il completamento di una serie di passi ("dipendenze"). Ad esempio, se vogliamo creare l'ipotetico file binario `glloq`, i file oggetto `main.o` e `init.o`, file intermedi del processo di compilazione, devono essere compilati e collegati (*`linked`*). Questi file oggetto sono anch'essi dei risultati della compilazione, e le loro dipendenze sono i file sorgenti.

Questo testo costituisce un'introduzione minima allo scopo di sopravvivere nel mondo spietato di `make`. Se volete saperne di più, vi consigliamo di recarvi sul sito web di **APRIL** (`http://www.april.org/groupes/doc` dove troverete una documentazione su `make` molto più dettagliata. Per una documentazione veramente esaustiva, fate riferimento a **Managing Projects with Make**, seconda edizione, **O'Reilly**, di *Andrew Oram* e *Steve Talbott*.

## Attenti, pronti, via!

In genere l'uso di `make` segue un certo numero di convenzioni. Ad esempio:

- `make` senza argomenti si limita a eseguire la compilazione del programma, senza installarlo.
- `make install` compila il programma (ma non sempre) e provvede a installare i file necessari nelle locazioni appropriate nella struttura gerarchica del filesystem. Alcuni file non sempre vengono installati correttamente (`man`, `info`), devono pertanto essere copiati a mano dall'utente stesso. Talvolta `make install` dev'essere eseguito di nuovo nelle sotto-directory; normalmente questo è necessario con moduli sviluppati da terze parti.
- `make clean` cancella tutti i file temporanei e, nella maggior parte dei casi, anche i file eseguibili creati dal processo di compilazione.

Il primo passo è quello di compilare il programma, e quindi di digitare (esempio del tutto immaginario):

```
$ make
gcc -c glloq.c -o glloq.o
gcc -c init.c -o init.o
gcc -c main.c -o main.o
gcc -lgtk -lgdk -lglib -lXext -lX11 -lm glloq.o init.o main.o -o glloq
```

Perfetto, il file binario è stato compilato in maniera corretta. Adesso siamo pronti per la fase successiva, che prevede l'installazione dei file

della distribuzione (file binari, file di dati, etc.). Consultate la sezione *Installazione*, pag. 215.

## Spiegazioni

Se la vostra curiosità vi ha spinto a dare un'occhiata al file `Makefile`, avrete scoperto che contiene dei comandi noti (`rm`, `mv`, `cp`, etc.), ma anche delle stringhe di caratteri dall'aspetto inusuale, qualcosa come `$(CFLAGS)`.

Si tratta di ***variabili***, cioè stringhe che normalmente si trovano all'inizio del file `Makefile` e vengono successivamente sostituite dal valore con il quale sono associate. Il loro uso semplifica molto le cose quando si vogliono usare le stesse opzioni di compilazione più volte nella stessa riga.

Ad esempio, per stampare su video la stringa "`foo`" usando il comando `make all`:

```
TEST = foo
all:
        echo $(TEST)
```

Nella maggior parte dei casi sono presenti le seguenti variabili:

1. `CC`: indica il compilatore. In genere è `cc`, che nella maggioranza dei sistemi aperti è sinonimo di `gcc`. Se in dubbio, inserite qui `gcc`.
2. `LD`: si tratta del programma usato per portare a termine l'ultima fase del processo di compilazione (consultate la sezione *Le quattro fasi della compilazione*, pag. 191). Come opzione predefinita, è il comando `ld`.
3. `CFLAGS`: sono argomenti supplementari che vengono forniti al compilatore durante le prime fasi di compilazione. Fra questi:

- `-I<path>`: indica al compilatore dove trovare alcuni header supplementari (`-I/usr/X11R6/include`, ad esempio, consente di includere gli header che si trovano nella directory `/usr/X11R6/include`).
- `-D<symbol>`: definisce un simbolo addizionale, utile per programmi la cui compilazione dipende dai simboli definiti (ad esempio: usa il file `string.h` se è definito `HAVE_STRING_H`).

Spesso ci sono righe di compilazione come questa:

```
$(CC) $(CFLAGS) -c foo.c -o foo.o
```

4. `LDFLAGS` (o `LFLAGS`): sono argomenti usati durante l'ultima fase del processo di compilazione. Fra di loro:
   - `-L<path>`: specifica un percorso supplementare dove cercare librerie (ad esempio: `-L/usr/X11R6/lib`).
   - `-l<library>`: specifica una libreria supplementare da usare durante l'ultima fase del processo di compilazione.

## Cosa fare... se non funziona?

Non lasciatevi prendere dal panico, può capitare a tutti. Fra le cause di errore più comuni:

1. `glloq.c:16: decl.h: No such file or directory`

   Il compilatore non è riuscito a trovare l'header corrispondente. La fase di configurazione del software, tuttavia, avrebbe dovuto prevenire questo errore. Ecco come risolvere il problema:

- controllate che l'header in questione esista realmente sul disco in una delle seguenti directory: `/usr/include`, `/usr/local/include`, `/usr/X11R6/include`, oppure una delle loro sottodirectory. Se così non fosse, cercatelo su tutto il disco rigido, con il comando `find` o `locate`), e, se ancora non riusciste a trovarlo, accertatevi di aver installato la libreria corrispondente. Troverete esempi relativi ai comandi `find` e `locate` nelle loro rispettive pagine di manuale.
- controllate che l'header in questione sia accessibile in lettura: digitate `less <percorso>/<file>.h` per esserne sicuri.
- se si trova in una directory come `/usr/local/include` o `/usr/X11R6/include`, può essere necessario aggiungere un nuovo argomento al compilatore: aprite il file `Makefile` corrispondente (fate attenzione e aprite il file giusto, nella directory dove si è interrotto il processo di compilazione [4]) con il vostro editor di testi preferito (*Emacs*, *VI*, etc.). Cercate la riga incriminata, e aggiungete la stringa `-I<percorso>` – dove `<percorso>` è il percorso che indica dove può essere trovato l'header in questione – subito dopo il punto in cui viene invocato il compilatore (`gcc`, o `$(CC)`, a volte). Se non sapete dove aggiungere questa opzione, inseritela all'inizio del file, dopo `CFLAGS=<qualcosa>` o dopo `CC=<qualcosa>`.
- lanciate `make` ancora una volta e, se si interrompe di nuovo, controllate che questa opzione (si veda il punto precedente) sia stata aggiunta durante la compilazione sulla riga in questione.

4. Analizzate il messaggio di errore restituito da `make`. Le ultime righe, in genere, contengono il nome di una directory (sono messaggi simili a questo: `make[1]: Leaving directory '/home/benj/Project/foo'`). Scegliete la riga con il numero più alto e spostatevi nella directory indicata: per essere sicuri che sia quella giusta, lanciate di nuovo `make` per ottenere lo stesso messaggio di errore.

- se ancora non funziona, chiedete aiuto al vostro guru locale, oppure rivolgetevi alla comunità del software libero per risolvere il vostro problema (consultate la sezione *Supporto tecnico*, pag. 217).

2. `glloq.c:28: ‘struct foo’ undeclared (first use this function)`

   Le strutture sono particolari tipi di dato usati da tutti i programmi. Un gran numero di strutture vengono definite dal sistema negli header: il messaggio significa che il problema è sicuramente causato da un header mancante o usato in modo inappropriato. La procedura corretta per risolvere il problema è la seguente:

   - cercate di accertare se la struttura in questione è definita nel programma o dal sistema. Un metodo possibile è l'impiego del comando `grep` per controllare se la struttura è stata definita in uno degli header.

     Ad esempio, quando siete nella directory principale della distribuzione digitate:

     ```
     $ find . -name '*.h'| xargs grep 'struct foo' | less
     ```

     È possibile che sullo schermo compaiano un gran numero di righe di testo (una per ogni volta che è definita una funzione che fa uso di questo tipo di struttura, ad esempio). Se esiste, trovate la riga dove è definita la struttura esaminando il file header che avete individuato usando il comando `grep`.

     La definizione di una struttura è:

     ```
     struct foo {
           <contenuto della struttura>
     };
     ```

     Controllate che corrisponda con quanto avete trovato: se le cose stanno così, questo significa che l'header non è incluso nel file `.c` difettoso. Ci sono due soluzioni:

  - aggiungete la riga `#include "<filename>.h"` all'inizio del file `.c` difettoso.
  - oppure fate un copia-incolla della definizione della struttura all'inizio di questo file (non è proprio elegante, ma per lo meno dovrebbe funzionare).

- Se invece non corrisponde, fate la stessa cosa con i file header di sistema (che in genere si trovano nelle directory `/usr/include`, `/usr/X11R6/include`, o `/usr/local/include`). Ma stavolta usate la riga `#include <<filename>.h>`.
- Se questa struttura risulta ancora non esistente, cercate di scoprire in quale libreria (ovvero un set di funzioni riunito in un unico pacchetto) dovrebbe essere definita: controllate nel file `INSTALL` o `README` quali sono le librerie usate dal programma, e il numero di versione richiesto. Se la versione di cui ha bisogno il programma non è quella installata sul vostro sistema, aggiornate la libreria in questione.
- Se, malgrado tutto, ancora non funziona, accertatevi che il programma funzioni correttamente con la vostra architettura (alcuni programmi non sono stati ancora portati su tutti i sistemi *Unix*). Controllate anche di aver correttamente configurato il programma (quando avete lanciato `configure`, ad esempio) per la vostra architettura.

3. `parse error`

   Si tratta di un problema piuttosto complicato da risolvere, in quanto spesso l'errore appare relativamente a una certa riga, ma dopo che il compilatore l'ha incontrato. Talvolta si tratta semplicemente di un tipo di dati non definito. Se incontrate un messaggio d'errore come:

```
main.c:1: parse error before ‘glloq_t
main.c:1: warning: data definition has no type or storage class
```

allora il problema sta nel fatto che il tipo `glloq_t` non è stato definito. La soluzione è più o meno la stessa che per il problema precedente.

> **Nota:** potrebbe esserci un `parse error` nelle vecchie librerie `curses`, se la memoria non m'inganna.

4. `no space left on device`

   Questo è un problema che può essere risolto facilmente: non è rimasto spazio sufficiente sul disco rigido per generare un file binario a partire dai sorgenti. La soluzione consiste nel liberare una quantità di spazio sufficiente nella partizione che contiene la directory di installazione (cancellate file temporanei o directory di sorgenti, disinstallate i programmi che non utilizzate). Se avete decompresso quest'ultima in `/tmp`, fatelo piuttosto in `/usr/local/src`, in modo da evitare di riempire inutilmente la partizione `/tmp`. Controllate, inoltre, se vi sono file `core`> sul vostro disco. In caso affermativo, cancellateli, o fateli cancellare se appartengono a un altro utente.

5. `/usr/bin/ld: cannot open -lglloq: No such file or directory`

   Questo significa, come avrete capito, che il programma `ld` (usato da `gcc` durante l'ultima fase del processo di compilazione) non riesce a trovare una libreria. Per includere una libreria, `ld` cerca un file il cui nome si trova negli argomenti del tipo `-l<library>`. Questo file è `lib<library>.so`, se `ld` non riesce a trovarlo, produce un messaggio d'errore. Per risolvere il problema, seguite i passi descritti qui di seguito:

a. Controllate che il file sia presente sul disco rigido usando il comando `locate`. Le librerie grafiche si trovano, in genere, in `/usr/X11R6/lib`. Ad esempio:

   ```
   $ locate libglloq
   ```

   Se questa ricerca non è fruttuosa, potete effettuarne un'altra usando il comando `find` (ad esempio: `find /usr -name libglloq.so*`). Se proprio non riuscite a trovare la libreria, vuol dire che non è presente sul vostro sistema e dunque dovrete installarla.

b. Una volta localizzata la libreria, accertatevi che `ld` possa accedervi: il file `/etc/ld.so.conf` specifica dove potrà trovare queste librerie: aggiungete il percorso della libreria incriminata al termine del file (è possibile che si debba riavviare il computer perché questa modifica venga recepita dal sistema). Potete anche aggiungere la stessa directory modificando il contenuto della variabile d'ambiente `LD_LIBRARY_PATH`: se la directory da aggiungere è `/usr/X11R6/lib`, ad esempio, digitate:

   ```
   export \ LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/X11R6/lib
   ```

   (se la vostra shell è *Bash*).

c. Se ancora non funziona, accertatevi che il formato della libreria sia quello di un file eseguibile (cioè ELF) usando il comando `file`. Se si tratta di un link (noto anche come collegamento) simbolico, controllate che il link sia valido e che non punti a un file inesistente (ad esempio con il comando `nm libglloq.so`). I permessi potrebbero essere errati, ad esempio se usate un account che non sia `root`, oppure se la libreria è protetta in lettura.

6. `glloq.c(.text+0x34): undefined reference to 'glloq_init'`

Il problema riguarda un simbolo che non è stato risolto durante l'ultima fase del processo di compilazione. Si tratta, in genere, di problemi dovuti alle librerie. Le cause possono essere molteplici:

- la prima cosa da fare è sapere se il simbolo **dovrebbe** trovarsi in una libreria. Ad esempio, se si tratta di un simbolo che comincia con `gtk`, appartiene alla libreria `gtk`. Se il nome della libreria non è immediatamente identificabile (`frobnicate_foobar`), potete elencare i simboli di una libreria usando il comando `nm`. Ad esempio,

  ```
  $ nm libglloq.so
  0000000000109df0 d glloq_message_func
  000000000010a984 b glloq_msg
  0000000000008a58 t glloq_nearest_pow
  0000000000109dd8 d glloq_free_list
  0000000000109cf8 d glloq_mem_chunk
  ```

  Aggiungendo l'opzione `-o` al comando `nm` permette di stampare il nome della libreria su ogni riga, il che facilita la ricerca. Supponiamo di dover trovare il simbolo `bulgroz_max`, una soluzione brutale ma efficace è effettuare una ricerca come:

  ```
  $ nm /usr/lib/lib*.so | grep bulgroz_max
  $ nm /usr/X11R6/lib/lib*.so | grep bulgroz_max
  $ nm /usr/local/lib/lib*.so | grep bulgroz_max
  /usr/local/lib/libfrobnicate.so:000000000004d848 T bulgroz_max
  ```

  Perfetto! Il simbolo `bulgroz_max` viene definito nella libreria `frobnicate` (la lettera maiuscola `T` si trova davanti al suo nome). A questo punto non dovete fare altro che aggiungere `-lfrobnicate` nella riga di compilazione modificando il file `Makefile`: inserite questa stringa al termine della riga in cui vengono definite le variabili `LDFLAGS` o `LFGLAGS` (oppure `CC`, nel peggiore dei casi), o nella riga che corrisponde alla creazione del file binario.

- la compilazione viene effettuata con una versione della libreria che non è quella richiesta dal software. Leggete il file `README`

o `INSTALL` della distribuzione per sapere qual è la versione che dev'essere usata.

- non tutti i file oggetto della distribuzione sono linkati correttamente. Il file dove viene definita questa funzione è assente. Digitate `nm -o *.o` per sapere di quale file si tratta e aggiunge il file `.o` corrispondente sulla riga di compilazione se è assente.
- la funzione o variabile causa del problema forse non esiste. Cercate di cancellarla: modificate il file sorgente in questione (il suo nome viene indicato all'inizio del messaggio di errore). Si tratta di una soluzione disperata, che avrà come sicura conseguenza un funzionamento "anarchico" del software (***segfault*** all'avvio, etc.).

7. `Segmentation fault (core dumped)`

   Talvolta il compilatore si pianta in modo deplorevole, e produce questo messaggio d'errore. Non possiamo darvi altro consiglio se non quello di installare una versione più recente.

8. spazio esaurito su `/tmp`

   La compilazione necessita di uno spazio di lavoro temporaneo durante le sue diverse fasi: se questo spazio non è disponibile, non viene portata a termine con successo. È quindi necessario ripulire la partizione, ma state attenti ad alcuni programmi in esecuzione (il server *X*, delle pipe, etc.) che potrebbero piantarsi se vengono cancellati alcuni dei file temporanei: dovete sapere quello che state facendo! Se `/tmp` fa parte di una partizione che contiene altro (ad esempio la directory radice), cercate e cancellate eventuali file di tipo `core`.

9. `make`/`configure` in ricorsione infinita

Molto spesso si tratta di un problema relativo al tempo sul vostro sistema. `make`, infatti, ha bisogno di conoscere la data fornita dall'orologio di sistema e la data dei file che controlla. Compara le date e usa il risultato per sapere se il bersaglio è più recente rispetto alla dipendenza.

Alcuni problemi di data possono indurre `make` a costruire se stesso in maniera ricorsiva, senza interruzioni (o a costruire e costruire di nuovo un sotto-albero in ricorsione infinita). In tal caso, l'uso del comando `touch` (il cui scopo qui è di aggiornare i file in questione all'ora corrente) in genere risolve questo problema.

Ad esempio:

```
$ touch *
```

O anche (più rozzo, ma efficace):

```
$ find . | xargs touch
```

# Installazione

## Con `make`

Adesso che la compilazione è terminata, dovete copiare i file che sono stati generati in un luogo appropriato (normalmente in una delle sotto-directory di `/usr/local`).

`make` di solito può eseguire questo compito. Il bersaglio `install` è un bersaglio particolare, riservato a questo scopo specifico. Il comando `make install`, quindi, permette di installare i file richiesti.

La procedura da seguire, in genere, è descritta nel file `INSTALL` o nel `README`. Ma a volte lo sviluppatore si è dimenticato di pensarci: in tal caso dovrete installare ogni file personalmente.

Pertanto copiate:

- i file eseguibili (programmi) nella directory `/usr/local/bin`
- le librerie (i file `lib*.so`) nella directory `/usr/local/lib`
- gli header (i file `*.h`) nella directory `/usr/local/include` (attenzione a non cancellare quelli originali)
- i file di dati in genere finiscono in `/usr/local/share`. Se non conoscete la procedura di installazione, potete cercare di lanciare in esecuzione i programmi senza aver copiato i file di dati, e metterli nel posto giusto quando vi viene richiesto (ad esempio con un messaggio d'errore come `Cannot open /usr/local/share/glloq/data.db`).
- Per la documentazione le regole da seguire sono di poco differenti:
  - I file `man` in genere vengono collocati in una delle sotto-directory di `/usr/local/man`. Di solito questi file sono in formato `troff` (o `groff`) e la loro estensione è una cifra. Il loro nome è il nome di un comando (ad esempio `echo.1`). Se la cifra è `n`, copiate il file in `/usr/local/man/man<n>`.
  - I file `info` vanno copiati nella directory `/usr/info` o `/usr/local/info`

Avete finito! Congratulazioni! Adesso siete pronti per compilare un intero sistema operativo!

## Problemi

Se avete appena installato del software libero, *`GNU tar`* ad esempio, e se, quando lo eseguite, viene lanciato un altro programma, oppure non si comporta esattamente come faceva quando lo avete provato direttamente dalla directory `src` si tratta di un problema di `PATH`, per cui i programmi vengono trovati in una directory che precede quella in cui avete installato del nuovo software. Controllate eseguendo il comando `type -a <program>`.

La soluzione consiste nel collocare la directory di installazione in una posizione più alta nella variabile `PATH`, e/o di cancellare/rinominare i file che vengono eseguiti al posto di quelli richiesti, e/o di rinominare i vostri nuovi programmi (come `gtar` in questo esempio) in maniera tale che non vi sia più nessuna confusione.

Potete anche impostare un alias, se la vostra shell lo consente (ad esempio, supponiamo che `tar` significhi `/usr/local/bin/gtar`).

# Supporto

## Documentazione

Esistono diverse fonti di documentazione:

- *`HOWTO`*, brevi documenti su punti specifici; in genere abbastanza lontani da quanto ci serve adesso, ma a volte utili. Cercateli sul vostro disco in `/usr/doc/HOWTO` (ma non sono sempre lì, a volte la locazione è diversa: controllate con il comando `locate HOWTO`),
- Le pagine di manuale. Digitate `man <command>` per ottenere documentazione in merito al comando `<command>`,
- Testi specifici sull'argomento. Molti grandi editori hanno cominciato a pubblicare libri che riguardano i sistemi aperti (in particolare su *`GNU/Linux`*). Spesso risultano molto utili se siete agli inizi e non capite tutti i termini usati in questo manuale.

## Supporto tecnico

Se avete comprato una distribuzione **Linux-Mandrake** "ufficiale" potete rivolgervi al personale di supporto tecnico per avere informazioni sul vostro sistema. Probabilmente questo personale è impegnato in

molte altre cose per poter aiutare tutti gli utenti nell'installazione di software supplementare, ma alcuni di loro potrebbero offrirvi un numero x di giorni in cui potete chiedere aiuto in merito all'installazione. Forse potranno dedicare un po' del loro tempo per aiutarvi a risolvere problemi di compilazione.

Potete anche fare affidamento sulla comunità del software libero per essere aiutati:

- nei ***gruppi di discussione*** (su *Usenet*) comp.os.linux.* (news:comp.os.linux.*) e it.comp.os.linux.* (news:it.comp.os.linux.*) viene data risposta a molte domande che riguardano *GNU/Linux*. I gruppi di discussione che appartengono alla gerarchia comp.os.bsd.* (news:comp.os.bsd.*) sono dedicati ai sistemi BSD. Potrebbero esserci altri gruppi di discussione che riguardano altri sistemi *Unix*. Ricordatevi di leggerli per un certo periodo prima di formulare la vostra richiesta.
- Nella comunità del software libero esistono numerose associazioni o gruppi di entusiasti che offrono un supporto volontario. Il modo migliore per trovare quelli più vicini al luogo in cui vivete è di consultare le liste di collegamenti sui siti web dedicati, o di leggere i gruppi di discussione che vi abbiamo suggerito prima per un certo periodo di tempo.
- Alcuni ***canali*** IRC offrono un'assistenza in tempo reale (ma alla cieca) da parte di alcuni ***guru***. Recatevi, ad esempio, sul canale `#linux`, disponibile su quasi tutta la rete IRC, oppure `#linuxhelp` su *IRCNET*.
- Come ultima risorsa, chiedete allo sviluppatore del software (se quest'ultimo ha riportato il suo nome e il suo indirizzo ***email*** in un file della distribuzione) se siete sicuri di aver trovato un bug; questo potrebbe essere dovuto alla vostra particolare architettura, ma si suppone che il software libero sia facilmente portabile, dopo tutto.

## Come reperire il software libero

Esistono molti riferimenti che vi possono aiutare a trovare del software libero:

- il grande sito FTP sunsite.unc.edu o uno dei suoi mirror
- i siti web elencati qui di seguito raccolgono un gran numero di programmi liberamente distribuibili che possono essere usati sotto *Unix* (ma su di essi è anche possibile trovare del software proprietario):
  - http://www.freshmeat.net/ è, probabilmente, il sito più completo,
  - http://www.linux-france.org/ contiene un gran numero di link relativi a programmi che girano sotto *GNU/Linux*. La maggior parte di essi, ovviamente, funziona anche con altre piattaforme *Unix* aperte,
  - http://www.gnu.org/software/ per una lista esaustiva di tutto il software GNU. Naturalmente tutti questi programmi sono software libero, la maggior parte è distribuita secondo la licenza GPL.
- potete anche effettuare un'indagine con un motore di ricerca come http://www.altavista.com/ e http://ftpsearch.lycos.com/, e fare una richiesta simile alla seguente: `+<software> +download` o `"download software"`.

# Ringraziamenti


- Revisione, correzione e commenti sgradevoli (in ordine alfabetico): *Sébastien Blondeel, Mathieu Bois, Xavier Renaut, Kamel Sehil,*
- ***Beta-testing***: *Laurent Bassaler*

- Traduzione inglese: *Fanny Drieu* Edizione inglese: *Hoyt Duff*

# Capitolo 16. Strumenti da linea di comando

Lo scopo di questo capitolo è presentare alcuni strumenti da linea di comando che possono tornare utili nell'uso quotidiano. Naturalmente potete saltare questo capitolo se intendete usare unicamente l'ambiente grafico, ma una rapida lettura potrebbe farvi cambiare opinione **:-)**.

Questo capitolo non ha una vera e propria struttura, i comandi sono elencati così come capita, dal più comunemente usato al più oscuro. Ciascun comando sarà accompagnato da un esempio, ma la ricerca di altri usi interessanti per gli stessi comandi sarà lasciata a voi, come esercizio.

## `grep`: `General Regular Expression Parse`

Va bene, il nome non è molto intuitivo, e non lo è neanche l'acronimo, ma il suo scopo è semplice: cercare in uno o più file un modello ("pattern") immesso come argomento. La sua sintassi è:

```
grep [opzioni] <schema> [uno o più file]
```

Se indicate più file insieme, prima di ciascuna riga del risultato sarà mostrato il nome del file corrispondente. Usate l'opzione `-h` per evitare che i nomi dei file siano mostrati; usate invece l'opzione `-l` per ottenere solo i nomi dei file in cui sono state trovate corrispondenze. Quest'ultima opzione può essere utile, specialmente con lunghe liste di argomenti, per analizzare i file con un ciclo di *`shell`* e usare il trucchetto `grep <modello> <nomefile> /dev/null`. Il modello è una espressione regolare, sebbene la maggior parte delle volte si tratti di una semplice parola. Le opzioni usate più di frequente sono le seguenti:

1. `-i`: Fa una ricerca che non distingue fra maiuscole e minuscole.
2. `-v`: Inverte la ricerca: mostra le righe che non contengono corrispondenze.
3. `-n`: Mostra il numero di riga per ogni riga in cui viene trovata una corrispondenza.
4. `-w`: Dice a `grep` che il modello deve corrispondere a una parola intera.

Ecco un esempio di come usarlo:

```
$ cat my_father
Hello dad
Hi daddy
So long dad

  # Cerca la stringa "hi", nessuna differenza tra maiuscole e minuscole
$ grep -i hi my_father
Hi daddy

  # Cerca la stringa "dad" come parola intera, e stampa
  # il numero di riga prima di ogni occorrenza
$ grep -nw dad my_father
1:Hello dad
3:So long dad

  # Adesso vogliamo tutte le righe che non cominciano con "H"
$ grep -v "^H" my_father
So long dad
$
```

Se volete usare `grep` in una pipe non dovete indicare il nome di un file, poiché come comportamento predefinito il comando preleva il suo ingresso dallo ***standard input***. Allo stesso modo, come comportamento predefinito, il comando stampa il risultato sullo ***standard output***, quindi potete redirigere l'output di un `grep` verso un altro programma senza alcun problema. Ad esempio:

```
$ cat /usr/doc/HOWTO/Parallel-Processing-HOWTO | \
  grep -n thread | less
```

## `find`: cerca file in base a determinati criteri

`find` è un veterano tra i comandi *Unix*. Il suo scopo è analizzare ricorsivamente una o più directory e cercare all'interno di esse i file che corrispondono a determinati criteri. Sebbene sia molto utile, la sua sintassi è veramente oscura, ed è necessario un certo impegno per utilizzarlo. La sintassi generale è:

```
find [opzioni] [directory] [criterio] [azione]
```

Se non specificate alcuna directory, `find` cercherà nella directory attuale. Se non specificate il criterio di ricerca, questo sarà considerato equivalente a "true", e quindi saranno trovati come validi tutti i file. Le opzioni, i criteri e le azioni sono così tanti che qui ne citeremo solo alcuni. Iniziamo con le opzioni:

1. `-xdev`: Esclude dalla ricerca le directory che risiedono su altri filesystem.
2. `-mindepth <n>`: Scende almeno `<n>` livelli al di sotto della directory specificata prima di iniziare la ricerca.
3. `-maxdepth <n>`: Cerca i file che si trovano al massimo `n` livelli al di sotto della directory specificata.
4. `-follow`: Segue i link simbolici corrispondenti a directory. Come comportamento predefinito, `find` non li segue.
5. `-daystart`: In caso di controlli relativi al tempo (vedi sotto), considera come orario l'inizio del giorno attuale invece del valore predefinito (24 ore prima dell'orario attuale).

Un criterio può essere costituito da uno o più controlli ***atomici*** fra i molti disponibili; alcuni controlli utili sono:

1. `-type <tipo>`: Cerca un determinato tipo di file; `<tipo>` può essere uno dei seguenti: `f` (file normale), `d` (directory), `l` (col-

legamento simbolico), `s` (socket), `b` (file in modalità a blocchi), `c` (file in modalità a caratteri) o `p` (pipe con nome).

2. `-name <modello>`: Cerca i file i cui nomi corrispondono al `<modello>` indicato. Con questa opzione, il `<modello>` è considerato come un modello di ***meta-espansione*** (si veda il capitolo *I caratteri speciali (meta-caratteri) e le **espressioni regolari** nella `shell`*, pag. 51 nel *Introduzione alla linea di comando*, pag. 43).
3. `-iname <modello>`: Come `-name`, ma non distingue tra maiuscole e minuscole.
4. `-atime <n>`, `-amin <n>`: Cerca i file che sono stati letti `<n>` giorni fa (`-atime`) o `<n>` minuti fa (`-amin`). Potete anche usare `+<n>` o `-<n>`, e in questo caso saranno cercati i file letti rispettivamente al massimo o al minimo `<n>` giorni/minuti fa.
5. `-anewer <file>`: Cerca i file che sono stati letti più recentemente del file `<file>`.
6. `-ctime <n>`, `-cmin <n>`, `-cnewer <file>`: Sono equivalenti a `-atime`, `-amin` e `-anewer`, ma vengono applicati alla data dell'ultima modifica del file.
7. `-regex <modello>`: Come `-name`, ma il `modello` viene considerato come una espressione regolare.
8. `-iregex <modello>`: Come `-regex`, ma non distingue tra maiuscole e minuscole.

Esistono molti altri tipi di controlli, fate riferimento alla pagina di manuale per ulteriori dettagli. Per combinare i controlli potete scegliere fra:

1. `<c1>` -a <c2>: È vero se sono veri sia `<c1>` che `<c2>`; `-a` è implicito, quindi potete scrivere `<c1>` <c2> <c3> ... se volete che siano verificati tutti i controlli `<c1>`, `<c2>`, ...
2. `<c1>` -o <c2>: È vero se almeno uno tra `<c1>` e `<c2>` è vero. Ricordate che `-o` ha una ***precedenza*** inferiore a `-a`, e quindi se vo-

lete, ad esempio, cercare i file che corrispondono ad almeno uno dei criteri `<c1>` e `<c2>` oltre che al criterio `<c3>`, dovrete usare le parentesi e scrivere `( <c1> -o <c2> ) -a <c3>`. Dovrete usare una ***sequenza escape*** per le parentesi (disattivandole), perché altrimenti saranno interpretate dalla shell!

3. `-not <c1>`: Inverte il controllo `<c1>`, pertanto `-not <c1>` è vero se `<c1>` è falso.

Infine, potete specificare una azione per ciascun file trovato. Le più frequentemente usate sono le seguenti:

1. `-print`: Stampa il nome di ciascun file sullo standard output. Se non specificate alcuna azione, questo è il comportamento predefinito.
2. `-ls`: Stampa sullo standard output l'equivalente del comando `ls -ilds` per ogni file trovato.
3. `-exec <comando>`: Esegue il comando `<comando>` per ogni file trovato. La linea di comando `<comando>` deve terminare con un `;`, per il quale deve essere utilizzata una sequenza escape in modo che la shell non lo interpreti; la posizione del file è rappresentata da `{}`. Per maggiori chiarimenti si vedano gli esempi di utilizzo.
4. `-ok <comando>`: Come `-exec`, ma chiede conferma per ogni comando.

Ci siete ancora? Bene, ora facciamo un po' di pratica, è sempre il modo migliore per capire questo mostro. Supponiamo che vogliate trovare tutte le directory contenute in `/usr/share`. Dovrete scrivere:

```
find /usr/share -type d
```

Supponiamo che abbiate un server HTTP, e che tutti i vostri file HTML siano in `/home/httpd/html`, che è anche la vostra directory attuale. Volete cercare tutti i file il cui contenuto non è stato modificato da un mese a questa parte. Poiché possedete pagine create da diversi autori, alcuni file hanno l'estensione `html` mentre altri hanno l'esten-

sione `htm`. Volete creare dei collegamenti a questi file nella directory `/home/httpd/obsolete`. Dovrete quindi scrivere:

```
find \( -name "*.htm" -o -name "*.html" \) -a -ctime -30 -exec ln {} /home/httpd/obsolete
```

1

Va bene, questo esempio è un po' complicato ed è necessaria una breve spiegazione. Il criterio è questo:

```
\( -name "*.htm" -o -name "*.html" \) -a -ctime -30
```

e fa quello che noi vogliamo: cerca tutti i file i cui nomi finiscono in `.htm` o `.html` "`\( -name "*.htm" -o -name "*.html" \)`", **e** (`-a`) che non sono stati modificati negli ultimi 30 giorni, che equivalgono più o meno a un mese (`-ctime -30`). Fate caso alle parentesi: in questo caso sono necessarie, perché `-a` ha una precedenza maggiore; se non le avessimo usate, sarebbero stati trovati tutti i file il cui nome finisce per `.htm`, più tutti i file il cui nome finisce per `.html` e che non sono stati modificati nell'ultimo mese, e questo non è quello che avevamo in mente. Notate anche che le parentesi sono disattivate dalla shell: se noi avessimo scritto `( .. )` invece di `\( .. \)`, la shell le avrebbe interpretate e avrebbe cercato di eseguire `-name "*.htm" -o -name "*.html"` in una sotto-shell... Un'altra soluzione poteva essere quella di racchiudere le parentesi fra virgolette doppie o singole, ma una sbarretta retroversa (\) qui è preferibile perché dobbiamo isolare un solo carattere.

E, per finire, abbiamo il comando che deve essere eseguito su ogni file:

```
-exec ln {} /home/httpd/obsolete \;
```

---

1. Notate che per questo esempio è necessario che `/home/httpd` e `/home/httpd/obsolete` siano sullo stesso filesystem!

Anche qui è necessario disattivare il ; dalla shell, altrimenti questa lo interpreterebbe come un separatore di comandi. Se non lo fate, `find` lamenterà la mancanza di un argomento `-exec`.

Un ultimo esempio: supponiamo di avere una directory `/shared/images` enorme, contenente tutti i tipi di immagini possibili. Normalmente si usa il comando `touch` per aggiornare l'orario di un file di nome `stamp` contenuto nella stessa directory, in modo da avere un riferimento temporale. Vogliamo cercare in questa directory tutte le immagini ***JPEG*** più recenti del file `stamp`, e poiché possedete immagini provenienti da diverse fonti, questi file potranno avere estensione `jpg`, `jpeg`, `JPG` o `JPEG`. Vogliamo anche evitare di cercare nella directory `old`. Vogliamo infine che questa lista di file ci venga spedita via email, e il nostro nome utente è `mario`:

```
find /shared/images -cnewer     \
     /shared/images/stamp       \
     -a -iregex ".*\.jpe?g"     \
     -a -not -regex ".*/old/.*" \
       | mail mario -s "Nuove immagini"
```

Ed ecco fatto! Naturalmente questo comando non è molto utile se dovete riscriverlo tutto ogni volta, e inoltre potreste volere che sia eseguito periodicamente... Potete fare così:

## `crontab`: analizzare o modificare il file `crontab`

`crontab` è un comando che vi permette di eseguire dei comandi a intervalli di tempo regolari, con il vantaggio che di non dovere essere necessariamente collegati al sistema, e che il risultato dei comandi vi viene spedito via email. Potete indicare l'intervallo di tempo in minuti, giorni e anche mesi. In base alle opzioni utilizzate, `crontab` si comporterà in diversi modi:

1. `-l`: Stampa il vostro file `crontab` attuale.

2. `-e`: Modifica il vostro file `crontab`.
3. `-r`: Elimina il vostro file `crontab` attuale.
4. `-u <utente>`: Applica una delle opzioni di cui sopra all'utente `<utente>`. Solo `root` può farlo.

   Iniziamo modificando un file `crontab`. Se scrivete `crontab -e`, se avete impostato le variabili d'ambiente `EDITOR` o `VISUAL` vi troverete di fronte al vostro editor di testi preferito, altrimenti sarà utilizzato *`VI`*. Una riga del file `crontab` è composta da sei campi. I primi cinque campi sono gli intervalli di tempo in minuti, ore, giorni del mese, mesi e giorni della settimana. Il sesto campo è il comando che deve essere eseguito. Le righe che iniziano con un `#` sono considerate come commenti e saranno ignorate da `crond` (il programma che si occupa dell'esecuzione dei file `crontab`). Ecco un esempio di `crontab`:

   **Nota:** per poterlo stampare con un carattere leggibile abbiamo dovuto spezzare le righe più lunghe in più parti; pertanto, alcune parti devono essere digitate su una sola riga. Quando una riga termina con il carattere ****, significa che quella riga continua sotto. Questa convenzione è valida anche nei file `Makefile`, nella *`shell`* e in altre situazioni.

```
# Se non volete ricevere posta, rimuovete il segno
#   di commento (#) che precede la riga seguente
#MAILTO=""
#
# Ogni due 2 giorni, alle 14.00 in punto, fammi un rapporto in merito
#   a nuove immagini partendo dall'esempio qui sopra - dopo di che,
#   "ritocca" il file "stamp". Il carattere "%" viene trattato come un
#   comando 'a capo', questo vi permette di inserire più
#   comandi sulla stessa linea.
0 14 */2 * *  find /shared/images              \
  -cnewer /shared/images/stamp                \
  -a -iregex ".*\.jpe?g"                      \
  -a -not -regex                              \
    ".*/old/.*"%touch /shared/images/stamp
```

```
#
# Suona questa musica ogni volta che è Natale :)
0 0 25 12 * mpg123 $HOME/sounds/merryxmas.mp3
#
# Ogni martedì alle 17.00 stampa la lista della spesa...
0 17 * * 2 lpr $HOME/shopping-list.txt
```

Ci sono parecchi modi per specificare gli intervalli, oltre a quelli mostrati in questo caso. Ad esempio, potete indicare un insieme di valori ***discreti*** separati da virgole (`1,14,23`) o una gamma di valori (`1-15`), o anche combinare le due cose insieme (`1-10,12-20`), specificando eventualmente un passo (`1-12,20-27/2`). Ora tocca a voi trovare dei comandi utili da inserire in questo file **:-)**.

## `at`: programmare un comando, ma per una sola volta

Potreste anche voler avviare un comando in un giorno particolare, e non periodicamente. Ad esempio, supponiamo che vogliate vi sia ricordato che avete un appuntamento oggi pomeriggio alle 6; voi avviate *X*, e volete essere avvertiti alle 5:30 del pomeriggio che è ora di andare. In questo caso, `at` è quello che vi serve:

```
$ at 5:30pm
  # Ora siete in presenza del promp "at"
at> xmessage "E' il momento di andare! Appuntamento alle 18.00"
  # Premete C-d per uscire
at> <EOT>
$
```

Potete indicare il tempo in diversi modi:

1. `now +<intervallo>`: Significa “adesso” più un intervallo di tempo (opzionale, se non lo indicate significa semplicemente “adesso”). La sintassi per l’intervallo è `<n>` (minutes|hours|days|weeks|months).

Ad esempio, potete scrivere `now + 1 hour`, `now + 3 days` e così via.

2. `<orario> <giorno>`: Specifica esattamente la data. Il parametro `<orario>` è obbligatorio. `at` è molto flessibile riguardo al formato: ad esempio, potete scrivere `0100`, `04:20`, `2am`, `0530pm`, `1800`, o uno fra i tre seguenti valori speciali: `noon` (mezzogiorno), `teatime` (4pm) o `midnight` (mezzanotte). Il parametro `<giorno>` è opzionale; anche questo può essere scritto in diversi modi: `12/20/2001`, ad esempio, che indica il 20 dicembre del 2001, oppure nel formato europeo, `20.12.2001`. Potete omettere l'anno, ma in questo caso viene accettato solo il formato europeo: `20.12`. Potete anche indicare il mese per esteso: `Dec 20` e `20 Dec` sono entrambe espressioni consentite.

Anche `at` accetta diverse opzioni:

1. `-l`: Stampa la lista dei job attualmente accodati; il primo campo è il numero associato al job. È equivalente al comando `atq`.
2. `-d <n>`: Rimuove il job numero `<n>` dalla coda. Potete vedere i numeri dei job usando `atq`. È equivalente a `atrm <n>`.

Come sempre, guardate la pagina di manuale `man 1 at` per le altre opzioni.

## `tar`: Tape ARchiver

Pur avendo già visto un utilizzo per `tar` nel capitolo *La compilazione e l'installazione di software libero*, pag. 189, non vi abbiamo spiegato come funziona: lo scopo di questa sezione è proprio questo. Come `find`, anche `tar` è un antico strumento di *Unix*, e come tale ha una sintassi un po' particolare. La sua sintassi è:

```
tar [opzioni] [file...]
```

Eccovi ora un elenco di opzioni. Ricordate che esse possiedono tutte un'opzione estesa equivalente, ma per queste dovrete fare riferimento alla relativa pagina di manuale, perché qui non saranno indicate. E, naturalmente, qui non saranno indicate neanche tutte le opzioni disponibili **:-)**.

**Nota:** l'uso del trattino iniziale (-) per le opzioni brevi di `tar` è da evitare, fatta eccezione per quelle che seguono un'opzione estesa.

1. `c`: Questa opzione è usata per creare nuovi archivi. ~
2. `x`: Questa opzione è usata per estrarre file da un archivio esistente.
3. `t`: Elenca i file contenuti in un archivio esistente.
4. `v`: Elenca semplicemente i file mentre vengono aggiunti o estratti da un archivio, oppure, insieme all'opzione `t` (vedi sopra), mostra una lista estesa dei file invece di una breve.
5. `f <file>`: Crea un archivio di nome `<file>`, estrae dall'archivio `<file>` oppure elenca i file dell'archivio `<file>`. Se questo parametro non viene indicato, il file predefinito sarà `/dev/rmt0`, che generalmente è un file speciale associato a uno ***streamer*** (unità a nastri). Se al posto del nome del file viene usato `-` (un trattino), l'uscita o l'ingresso (a seconda se state estraendo da un archivio o creandone uno) saranno associati a standard output o standard input.
6. `z`: Dice a `tar` che l'archivio da creare deve essere compresso con `gzip`, oppure che l'archivio da cui estrarre è compresso con `gzip`.
7. `y`: Come `z`, ma il programma usato per la compressione sarà `bzip2`.
8. `p`: Quando vengono estratti file da un archivio conserva tutti i loro attributi, compresi proprietario, ultima data d'accesso e così via.

Molto utile per copie di interi filesystem.

9. `r`: Accoda a un archivio esistente i file indicati nella linea di comando. Ricordate che l'archivio al quale accodate i file **non** deve essere compresso!
10. `A`: Aggiunge gli archivi indicati nella linea di comando a quello associato all'opzione `f`. Come per `r`, affinché questo funzioni gli archivi non devono essere compressi.

Ci sono tante, tante, tante altre opzioni, potete fare riferimento alla pagina `man 1 tar` per una lista completa; guardate ad esempio l'opzione `d`. Ora facciamo un po' di pratica; supponiamo che vogliate creare un archivio di tutte le immagini contenute in `/shared/images`, compresso con `bzip2`, di nome `images.tar.bz2` e posizionato nella vostra home directory. Dovrete quindi scrivere:

```
 #
 # Attenzione: dovete trovarvi nella directory che contiene
 #   i file che volete archiviare!
 #
$ cd /shared
$ tar cyf ~/images.tar.bz2 images/
```

Come potete vedere, abbiamo usato tre opzioni: `c` per dire a `tar` che volevamo creare un archivio, `y` per dirgli che lo volevamo compresso con `bzip2`, e `f ~/images.tar.bz2` per dirgli che l'archivio doveva essere creato nella nostra directory base, con il nome `images.tar.bz2`. Ora potremmo voler controllare che l'archivio sia corretto; possiamo farlo semplicemente vedendo l'elenco dei suoi file:

```
 #
 # Torniamo alla nostra directory home
 #
$ cd
$ tar tyvf images.tar.bz2
```

Qui abbiamo detto a `tar` di elencare (`t`) i file dell'archivio `images.tar.bz2` (`f images.tar.bz2`), avvertendolo che questo archivio era compresso con `bzip2` (`y`), e richiedendo una lista estesa (`v`). Ora supponiamo che voi abbiate cancellato la directory delle immagini; fortu-

natamente il vostro archivio è intatto, e ora volete estrarre di nuovo la directory nella sua posizione originale, in `/shared`. Ma, poiché volete che il vostro comando `find` per cercare nuove immagini continui a funzionare correttamente, dovete preservare tutti gli attributi dei file:

```
 #
 # spostatevi nella directory dove volete estrarre
 # i file contenuti nell'archivio
 #
$ cd /shared
$ tar yxpf ~/images.tar.bz2
```

Ed ecco fatto!

Ora, supponiamo che vogliate estrarre dall'archivio la directory `images/cars`, e nient'altro. Potete scrivere così:

```
$ tar yxf ~/images.tar.bz2 images/cars
```

Nel caso ve ne stiate preoccupando, state tranquilli: se provate ad archiviare file speciali, `tar` li considererà per quello che sono, file speciali, e non ne copierà il contenuto. Perciò, sì, potete tranquillamente includere `/dev/mem` in un archivio **:-)**. Ah, si comporta correttamente anche con i collegamenti, quindi non preoccupatevi neanche di questo. Per quel che riguarda i collegamenti simbolici, date uno sguardo anche all'opzione `h` nella pagina di manuale.

## `bzip2` e `gzip`: programmi per la compressione di dati

Avrete notato che abbiamo già parlato di questi due programmi quando ci siamo occupati di `tar`. A differenza di *WinZip* su *Windows*, qui l'archiviazione e la compressione vengono fatte usando due strumenti separati – `tar` per l'archiviazione, e i due programmi di cui stiamo per parlare per comprimere i dati: `bzip2` e `gzip`.

Inizialmente `bzip2` era stato scritto per sostituire `gzip`: i suoi rapporti di compressione sono generalmente migliori, ma d'altra parte richiede

più memoria. Il motivo per cui `gzip` esiste ancora è che è ancora più diffuso di `bzip2`. Forse un giorno `bzip2` sostituirà definitivamente `gzip`, o forse no.

I due comandi hanno una sintassi simile:

```
gzip [opzioni] [uno o più file]
```

Se non viene indicato alcun file, sia `gzip` che `bzip2` si mettono in attesa di dati dallo standard input e inviano il risultato sullo standard output. Quindi potete usare entrambi i programmi nelle pipe. Possiedono anche alcune opzioni in comune:

1. `-1`, ..., `-9`: Imposta il rapporto di compressione. Più alto il numero, migliore la compressione; ma migliore significa anche più lenta: "Niente è gratis".
2. `-d`: Decomprime i file. Equivale ad usare `gunzip` o `bunzip2`.
3. `-c`: Invia sullo standard output il risultato della compressione/decompressione dei file indicati nella linea di comando.

Attenzione! Come comportamento predefinito, sia `gzip` che `bzip2` cancellano i file che hanno compresso (o decompresso) se non usate l'opzione `-c`. Con `bzip2` potete evitarlo usando l'opzione `-k`, ma `gzip` non la possiede!

Ora vediamo alcuni esempi. Supponiamo che vogliate comprimere con `bzip2` tutti i file della directory attuale il cui nome finisce per `.txt`; userete quindi:

```
$ bzip2 -9 *.txt
```

Supponiamo che vogliate condividere il vostro archivio di immagini con qualcuno che non possiede `bzip2`, ma solo `gzip`. Non è necessario decomprimere l'archivio e poi ricomprimerlo, potete semplice-

mente decomprimerlo sullo standard output, usare una pipe, comprimere dallo standard input e redirigere l'uscita verso il nuovo archivio:

```
bzip2 -dc images.tar.bz2 | gzip -9 >images.tar.gz
```

Ecco fatto. Avreste anche potuto scrivere `bzcat` invece di `bzip2 -dc`. Esiste anche un equivalente per `gzip`, ma il suo nome è `zcat`, e non `gzcat`. Avete anche a disposizione `bzless` (e rispettivamente `zless`) se volete visualizzare direttamente un file compresso, senza doverlo prima decomprimere. Come esercizio, provate a trovare il comando da usare per visualizzare file compressi senza prima decomprimerli, e senza usare `bzless` o `zless` **:-)**

## Tanti, tanti altri...

Ci sono così tanti comandi che un libro completo su di essi sarebbe grande come un'enciclopedia. Questo capitolo non ha coperto neanche un decimo dell'argomento, ma tuttavia potete fare molte cose con quello che avete imparato qui. Se lo desiderate, potete leggere alcune pagine di manuale: `man 1 sort`, `man 1 sed`, `man 1 zip` (sì, è proprio quello che pensate: potete estrarre o creare archivi ZIP con *GNU/Linux*), `man 1 convert`, e così via. Il modo migliore per prendere confidenza con questi strumenti è fare pratica con essi e sperimentare, e probabilmente troverete per loro moltissime applicazioni, anche le più inaspettate. Buon divertimento! **:-)**

# Appendice A. La Licenza Pubblica Generica GNU

Questa è una traduzione italiana non ufficiale della Licenza Pubblica Generale GNU (), applicabile alla maggior parte dei programmi che sono contenuti nelle distribuzioni **Linux-Mandrake**. Non è pubblicata dalla Free Software Foundation e non ha valore legale nell'esprimere i termini di distribuzione del software che usa la licenza GPL. Solo la versione originale in inglese della licenza ha valore legale. Ad ogni modo, speriamo che questa traduzione aiuti le persone di lingua italiana a capire meglio il significato della licenza GPL.



## Preambolo

Le licenze della maggior parte dei programmi hanno lo scopo di togliere all'utente la libertà di condividere e modificare il programma stesso. Viceversa, la Licenza Pubblica Generica GNU è intesa a garantire la libertà di condividere e modificare il software libero, al fine di assicurare che i programmi siano liberi per tutti i loro utenti. Questa Licenza si applica alla maggioranza dei programmi della Free Software Foundation e ad ogni altro programma i cui autori hanno deciso di usare questa Licenza. Alcuni altri programmi della Free Soft-

ware Foundation sono invece coperti dalla Licenza Pubblica Generica Minore. Chiunque può usare questa Licenza per i propri programmi.

Quando si parla di software libero (free software), ci si riferisce alla libertà, non al prezzo. Le nostre Licenze (la GPL e la LGPL) sono progettate per assicurarsi che ciascuno abbia la libertà di distribuire copie del software libero (e farsi pagare per questo, se vuole), che ciascuno riceva il codice sorgente o che lo possa ottenere se lo desidera, che ciascuno possa modificare il programma o usarne delle parti in nuovi programmi liberi e che ciascuno sappia di potere fare queste cose.

Per proteggere i diritti dell'utente, abbiamo bisogno di creare delle restrizioni che vietino a chiunque di negare questi diritti o di chiedere di rinunciarvi. Queste restrizioni si traducono in certe responsabilità per chi distribuisce copie del software e per chi lo modifica.

Per esempio, chi distribuisce copie di un programma coperto da GPL, sia gratis sia in cambio di un compenso, deve concedere ai destinatari tutti i diritti che ha ricevuto. Deve anche assicurarsi che i destinatari ricevano o possano ottenere il codice sorgente. E deve mostrar loro queste condizioni di licenza, in modo che essi conoscano i propri diritti.

Proteggiamo i diritti dell'utente in due modi: (1) proteggendo il software con un copyright, e (2) offrendo una licenza che dia il permesso legale di copiare, distribuire e modificare il Programma.

Inoltre, per proteggere ogni autore e noi stessi, vogliamo assicurarci che ognuno capisca che non ci sono garanzie per i programmi coperti da GPL. Se il programma viene modificato da qualcun altro e ridistribuito, vogliamo che gli acquirenti sappiano che ciò che hanno non è l'originale, in modo che ogni problema introdotto da altri non si rifletta sulla reputazione degli autori originari.

Infine, ogni programma libero è costantemente minacciato dai brevetti sui programmi. Vogliamo evitare il pericolo che chi ridistribuisce un programma libero ottenga la proprietà di brevetti, rendendo in pratica il programma cosa di sua proprietà. Per prevenire questa evenienza,

abbiamo chiarito che ogni brevetto debba essere concesso in licenza d'uso a chiunque, o non avere alcuna restrizione di licenza d'uso.

Seguono i termini e le condizioni precisi per la copia, la distribuzione e la modifica.

## Termini e condizioni per la copia, la distribuzione e la modifica

1. Questa Licenza si applica a ogni programma o altra opera che contenga una nota da parte del detentore del copyright che dica che tale opera può distribuita sotto i termini di questa Licenza Pubblica Generica. Il termine "Programma" nel seguito si riferisce ad ogni programma o opera così definita, e l'espressione "opera basata sul Programma" indica sia il Programma sia ogni opera considerata "derivata" in base alla legge sul copyright; in altre parole, un'opera contenente il Programma o una porzione di esso, sia letteralmente sia modificato o tradotto in un'altra lingua. Da qui in avanti, la traduzione è in ogni caso considerata una "modifica". Vengono ora elencati i diritti dei beneficiari della licenza.

   Attività diverse dalla copia, distribuzione e modifica non sono coperte da questa Licenza e sono al di fuori della sua influenza. L'atto di eseguire il Programma non viene limitato, e l'output del programma è coperto da questa Licenza solo se il suo contenuto costituisce un'opera basata sul Programma (indipendentemente dal fatto che sia stato creato eseguendo il Programma). In base alla natura del Programma il suo output può essere o meno coperto da questa Licenza.

2. È lecito copiare e distribuire copie letterali del codice sorgente del Programma così come viene ricevuto, con qualsiasi mezzo, a condizione che venga riprodotta chiaramente su ogni copia una

appropriata nota di copyright e di assenza di garanzia; che si mantengano intatti tutti i riferimenti a questa Licenza e all'assenza di ogni garanzia; che si dia a ogni altro destinatario del Programma una copia di questa Licenza insieme al Programma.

È possibile richiedere un pagamento per il trasferimento fisico di una copia del Programma, è anche possibile a propria discrezione richiedere un pagamento in cambio di una copertura assicurativa.

3. È lecito modificare la propria copia o copie del Programma, o parte di esso, creando perciò un'opera basata sul Programma, e copiare o distribuire tali modifiche o tale opera secondo i termini del precedente comma 1, a patto che siano soddisfatte tutte le condizioni che seguono:

   a. Bisogna indicare chiaramente nei file che si tratta di copie modificate e la data di ogni modifica.

   b. Bisogna fare in modo che ogni opera distribuita o pubblicata, che in parte o nella sua totalità derivi dal Programma o da parti di esso, sia concessa in licenza gratuita nella sua interezza ad ogni terza parte, secondo i termini di questa Licenza.

   c. Se normalmente il programma modificato legge comandi interattivamente quando viene eseguito, bisogna fare in modo che all'inizio dell'esecuzione interattiva usuale esso stampi un messaggio contenente una appropriata nota di copyright e di assenza di garanzia (oppure che specifichi il tipo di garanzia che si offre). Il messaggio deve inoltre specificare che chiunque può ridistribuire il programma alle condizioni qui descritte e deve indicare come reperire questa Licenza. Se però il programma di partenza è interattivo ma normalmente non stampa tale messaggio, non occorre che un'opera basata sul Programma lo stampi.

   Questi requisiti si applicano all'opera modificata nel suo comp-

lesso. Se sussistono parti identificabili dell'opera modificata che non siano derivate dal Programma e che possono essere ragionevolmente considerate lavori indipendenti, allora questa Licenza e i suoi termini non si applicano a queste parti quando queste vengono distribuite separatamente. Se però queste parti vengono distribuite all'interno di un prodotto che è un'opera basata sul Programma, la distribuzione di quest'opera nella sua interezza deve avvenire nei termini di questa Licenza, le cui norme nei confronti di altri utenti si estendono all'opera nella sua interezza, e quindi ad ogni sua parte, chiunque ne sia l'autore.

Quindi, non è nelle intenzioni di questa sezione accampare diritti, né contestare diritti su opere scritte interamente da altri; l'intento è piuttosto quello di esercitare il diritto di controllare la distribuzione di opere derivati dal Programma o che lo contengano.

Inoltre, la semplice aggregazione di un'opera non derivata dal Programma col Programma o con un'opera da esso derivata su di un mezzo di memorizzazione o di distribuzione, non è sufficiente a includere l'opera non derivata nell'ambito di questa Licenza.

4. È lecito copiare e distribuire il Programma (o un'opera basata su di esso, come espresso al comma 2) sotto forma di codice oggetto o eseguibile secondo i termini dei precedenti commi 1 e 2, a patto che si applichi una delle seguenti condizioni:

    a. Il Programma sia corredato dal codice sorgente completo, in una forma leggibile da calcolatore, e tale sorgente sia fornito secondo le regole dei precedenti commi 1 e 2 su di un mezzo comunemente usato per lo scambio di programmi.

    b. Il Programma sia accompagnato da un'offerta scritta, valida per almeno tre anni, di fornire a chiunque ne faccia richiesta una copia completa del codice sorgente, in una forma leggibile da calcolatore, in cambio di un compenso non superiore al costo del trasferimento fisico di tale copia, che deve essere

fornita secondo le regole dei precedenti commi 1 e 2 su di un mezzo comunemente usato per lo scambio di programmi.

c. Il Programma sia accompagnato dalle informazioni che sono state ricevute riguardo alla possibilità di ottenere il codice sorgente. Questa alternativa è permessa solo in caso di distribuzioni non commerciali e solo se il programma è stato ottenuto sotto forma di codice oggetto o eseguibile in accordo al precedente comma B.

Per "codice sorgente completo" di un'opera si intende la forma preferenziale usata per modificare un'opera. Per un programma eseguibile, "codice sorgente completo" significa tutto il codice sorgente di tutti i moduli in esso contenuti, più ogni file associato che definisca le interfacce esterne del programma, più gli script usati per controllare la compilazione e l'installazione dell'eseguibile. In ogni caso non è necessario che il codice sorgente fornito includa nulla che sia normalmente distribuito (in forma sorgente o in formato binario) con i principali componenti del sistema operativo sotto cui viene eseguito il Programma (compilatore, kernel, e così via), a meno che tali componenti accompagnino l'eseguibile.

Se la distribuzione dell'eseguibile o del codice oggetto è effettuata indicando un luogo dal quale sia possibile copiarlo, permettere la copia del codice sorgente dallo stesso luogo è considerata una valida forma di distribuzione del codice sorgente, anche se copiare il sorgente è facoltativo per l'acquirente.

5. Non è lecito copiare, modificare, sublicenziare, o distribuire il Programma in modi diversi da quelli espressamente previsti da questa Licenza. Ogni tentativo di copiare, modificare, sublicenziare o distribuire il Programma non è autorizzato, e farà terminare automaticamente i diritti garantiti da questa Licenza. D'altra parte ogni acquirente che abbia ricevuto copie, o diritti, coperti da questa Licenza da parte di persone che violano la Licenza come qui indicato non vedranno invalidata la loro Licenza, purché si

comportino conformemente ad essa.

6. L'acquirente non è obbligato ad accettare questa Licenza, poiché non l'ha firmata. D'altra parte nessun altro documento garantisce il permesso di modificare o distribuire il Programma o i lavori derivati da esso. Queste azioni sono proibite dalla legge per chi non accetta questa Licenza; perciò, modificando o distribuendo il Programma o un'opera basata sul programma, si indica nel fare ciò l'accettazione di questa Licenza e quindi di tutti i suoi termini e le condizioni poste sulla copia, la distribuzione e la modifica del Programma o di lavori basati su di esso.

7. Ogni volta che il Programma o un'opera basata su di esso vengono distribuiti, l'acquirente riceve automaticamente una licenza d'uso da parte del licenziatario originale. Tale licenza regola la copia, la distribuzione e la modifica del Programma secondo questi termini e queste condizioni. Non è lecito imporre restrizioni ulteriori all'acquirente nel suo esercizio dei diritti qui garantiti. Chi distribuisce programmi coperti da questa Licenza non e' comunque tenuto a imporre il rispetto di questa Licenza a terzi.

8. Se, come conseguenza del giudizio di un tribunale, o di una imputazione per la violazione di un brevetto o per ogni altra ragione (non limitatamente a questioni di brevetti), vengono imposte condizioni che contraddicono le condizioni di questa licenza, che queste condizioni siano dettate dalla corte, da accordi tra le parti o altro, queste condizioni non esimono nessuno dall'osservazione di questa Licenza. Se non è possibile distribuire un prodotto in un modo che soddisfi simultaneamente gli obblighi dettati da questa Licenza e altri obblighi pertinenti, il prodotto non può essere affatto distribuito. Per esempio, se un brevetto non permettesse a tutti quelli che lo ricevono di ridistribuire il Programma senza obbligare al pagamento di diritti, allora l'unico modo per soddisfare contemporaneamente il brevetto e questa Licenza e' di non distribuire affatto il Programma.

   Se una qualunque parte di questo comma è ritenuta non valida o non applicabile in una qualunque circostanza, deve comunque

essere applicata l'idea espressa da questo comma; in ogni altra circostanza invece deve essere applicato questo comma nel suo complesso.

Non è nelle finalità di questo comma indurre gli utenti ad infrangere alcun brevetto né ogni altra rivendicazione di diritti di proprietà, né di contestare la validità di alcuna di queste rivendicazioni; lo scopo di questo comma è unicamente quello di proteggere l'integrità del sistema di distribuzione dei programmi liberi, che viene realizzato tramite l'uso di licenze pubbliche. Molte persone hanno contribuito generosamente alla vasta gamma di programmi distribuiti attraverso questo sistema, basandosi sull'applicazione fedele di tale sistema. L'autore/donatore può decidere di sua volontà se preferisce distribuire il software avvalendosi di altri sistemi, e l'acquirente non può imporre la scelta del sistema di distribuzione.

Questo comma serve a rendere il più chiaro possibile ciò che crediamo sia una conseguenza del resto di questa Licenza.

9. Se in alcuni paesi la distribuzione o l'uso del Programma sono limitati da brevetto o dall'uso di interfacce coperte da copyright, il detentore del copyright originale che pone il Programma sotto questa Licenza può aggiungere limiti geografici espliciti alla distribuzione, per escludere questi paesi dalla distribuzione stessa, in modo che il programma possa essere distribuito solo nei paesi non esclusi da questa regola. In questo caso i limiti geografici sono inclusi in questa Licenza e ne fanno parte a tutti gli effetti.

10. All'occorrenza la Free Software Foundation può pubblicare revisioni o nuove versioni di questa Licenza Pubblica Generica. Tali nuove versioni saranno simili a questa nello spirito, ma potranno differire nei dettagli al fine di coprire nuovi problemi e nuove situazioni.

    Ad ogni versione viene dato un numero identificativo. Se il Programma asserisce di essere coperto da una particolare versione

di questa Licenza e "da ogni versione successiva", l'acquirente può scegliere se seguire le condizioni della versione specificata o di una successiva. Se il Programma non specifica quale versione di questa Licenza deve applicarsi, l'acquirente può scegliere una qualsiasi versione tra quelle pubblicate dalla Free Software Foundation.

11. Se si desidera incorporare parti del Programma in altri programmi liberi le cui condizioni di distribuzione differiscano da queste, è possibile scrivere all'autore del Programma per chiederne l'autorizzazione. Per il software il cui copyright è detenuto dalla Free Software Foundation, si scriva alla Free Software Foundation; talvolta facciamo eccezioni alle regole di questa Licenza. La nostra decisione sarà guidata da due finalità: preservare la libertà di tutti i prodotti derivati dal nostro software libero e promuovere la condivisione e il riutilizzo del software in generale.

## NON C'È GARANZIA

12. POICHÉ IL PROGRAMMA È CONCESSO IN USO GRATUITAMENTE, NON C'È GARANZIA PER IL PROGRAMMA, NEI LIMITI PERMESSI DALLE VIGENTI LEGGI. SE NON INDICATO DIVERSAMENTE PER ISCRITTO, IL DETENTORE DEL COPYRIGHT E LE ALTRE PARTI FORNISCONO IL PROGRAMMA "COSÌ COM'È", SENZA ALCUN TIPO DI GARANZIA, NÉ ESPLICITA NÉ IMPLICITA; CIÒ COMPRENDE, SENZA LIMITARSI A QUESTO, LA GARANZIA IMPLICITA DI COMMERCIABILITÀ E UTILIZZABILITÀ PER UN PARTICOLARE SCOPO. L'INTERO RISCHIO CONCERNENTE LA QUALITÀ E LE PRESTAZIONI DEL PROGRAMMA È DELL'ACQUIRENTE. SE IL PROGRAMMA DOVESSE RIVELARSI DIFETTOSO, L'ACQUIRENTE SI ASSUME IL COSTO DI OGNI MANUTENZIONE, RIPARAZIONE O CORREZIONE NECESSARIA.

13. NÉ IL DETENTORE DEL COPYRIGHT NÉ ALTRE PARTI CHE POSSONO MODIFICARE O RIDISTRIBUIRE IL PROGRAMMA COME PERMESSO IN QUESTA LICENZA SONO RESPONSABILI PER DANNI NEI CONFRONTI DELL' ACQUIRENTE, A MENO CHE QUESTO NON SIA RICHIESTO DALLE LEGGI VIGENTI O APPAIA IN UN ACCORDO SCRITTO. SONO INCLUSI DANNI GENERICI, SPECIALI O INCIDENTALI, COME PURE I DANNI CHE CONSEGUONO DALL'USO O DALL' IMPOSSIBILITÀ DI USARE IL PROGRAMMA; CIÒ COMPRENDE, SENZA LIMITARSI A QUESTO, LA PERDITA DI DATI, LA CORRUZIONE DEI DATI, LE PERDITE SOSTENUTE DALL' ACQUIRENTE O DA TERZI E L'INCAPACITÀ DEL PROGRAMMA A INTERAGIRE CON ALTRI PROGRAMMI, ANCHE SE IL DETENTORE O ALTRE PARTI SONO STATE AVVISATE DELLA POSSIBILITÀ DI QUESTI DANNI.

FINE DEI TERMINI E DELLE CONDIZIONI

# Glossario

***account***

su un sistema *Unix*, un account (o *login*) è la combinazione di un nome, una directory personale, una password e una *shell* che consentono a un utente di connettersi al sistema.

***alias***

il meccanismo usato in una *shell* per sostituire una stringa con un'altra prima di eseguire il comando. Potete vedere tutti gli alias definiti nella sessione corrente digitando `alias` al prompt.

***al volo***

Si dice che qualcosa viene eseguito "al volo" (ingl. *on the fly*) quando questo accade contemporaneamente a qualcos'altro e senza che l'operazione in questione sia stata esplicitamente richiesta, o anche notata dall'utente.

***ambiente***

è il contesto di esecuzione di un processo. Include tutte le informazioni di cui ha bisogno il sistema operativo per gestire il processo, e quanto serve al processore per eseguire il processo in maniera corretta.

*Si veda anche:* processo.

***APM***

*Advanced Power Management*. Una caratteristica propria di alcuni *BIOS* che permette alla macchina di entrare in standby dopo un certo periodo di inattività. Sui portatili, APM ha anche il compito di verificare la carica residua della batteria e, se questa

caratteristica è supportata, di stimare il periodo di funzionamento che questa permette.

***arp***

*Address Resolution Protocol*. Il protocollo Internet usato per rimappare dinamicamente un indirizzo Internet su indirizzi fisici (hardware) che appartengono a una rete locale. Il suo uso è limitato a reti che supportano il broadcasting hardware.

***ASCII***

*American Standard Code for Information Interchange*. La codifica standard impiegata per archiviare caratteri, inclusi caratteri di controllo, su un computer. Molti codici a 8-bit (come l'ISO 8859-1, il set di caratteri predefinito di Linux) includono l'ASCII come prima metà (si veda *ISO 8859*).

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | |
| 2 | | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | |

**Figura 1. Tabella ASCII**

***assembler***

è il linguaggio di programmazione più vicino al metodo di funzionamento interno di un computer, pertanto è definito come lin-

guaggio di programmazione di “basso livello”. Il principale vantaggio dell’assembler è la velocità, in quanto i programmi in assembler sono scritti sotto forma di istruzioni del processore, di conseguenza sono poche o comunque semplici le traduzioni richieste al momento di generare i file eseguibili. Il suo grande svantaggio è dato dal fatto che è un linguaggio strettamente dipendente da un processore (o da un’architettura). La scrittura di programmi complessi in assembler, inoltre, è un processo lungo e complesso. In conclusione, si tratta del linguaggio di programmazione che garantisce la maggior velocità in esecuzione, ma certo non durante la scrittura dei programmi, e non è portabile fra architetture hardware diverse.

***ATAPI***

(“AT Attachment Packet Interface”) Un’estensione delle specifiche ATA (“Advanced Technology Attachment”, più comunemente note come IDE, *Integrated Drive Electronics*) che mette a disposizione comandi supplementari per controllare lettori CD-ROM e unità a nastro magnetico. I controller IDE che sono equipaggiati con tale estensione sono noti anche come controller EIDE (*Enhanced IDE*).

***atomico***

un insieme di operazioni si dice atomico quando queste vengono eseguite tutte contemporaneamente, e non possono essere interrotte.

***attraversamento***

per una directory di un sistema *Unix*, significa che l’utente ha il permesso di accedere a tale directory e forse anche alle directory sotto di essa. Perché questo sia possibile, l’utente deve avere anche il permesso di esecuzione sulla directory.

***background***

nell'ambito di una *shell*, un processo viene eseguito in background ("in secondo piano") se potete digitare comandi mentre il processo in questione è in esecuzione.

*Si veda anche:* job, foreground.

***backup***

il salvataggio di dati importanti su di un supporto sicuro, archiviato in un luogo sicuro. I backup dovrebbero essere effettuati con regolarità, soprattutto nel caso di informazioni e file di configurazione d'importanza critica (le prime directory di cui bisogna fare un backup sono `/etc`, `/home` e `/usr/local`). Molte persone usano strumenti tradizionali come il programma `tar`, usato con `gzip` o `bzip2`, per effettuare il backup di directory e file. Potete usare questi strumenti, o altri come `dump` e `restore`, come pure molte applicazioni di backup disponibili per Linux, liberamente distribuibili o commerciali.

***batch***

è un metodo di elaborazione per cui il processore riceve una serie di compiti, e li esegue uno dopo l'altro finché non ha portato a termine anche l'ultimo; da quel momento è pronto per un'altra serie di processi.

***beep***

il piccolo rumore emesso dall'altoparlante interno del computer per avvertirvi di qualche ambiguità quando usate il completamento automatico della linea di comando (nel caso, ad esempio, che ci sia più di una soluzione possibile per il completamento automatico). Altri programmi potrebbero emettere un beep per informarvi di qualche particolare situazione.

***beta testing***

è il nome dato al processo di prova della versione beta di un programma. In genere i programmi vengono rilasciati in versioni alfa

e beta proprio per essere sottoposti a varie prove prima di arrivare alla versione finale.

***bit***

sta per *BInary uniT*. Una singola cifra che può avere valore 0 o 1, in quanto il calcolo viene effettuato in base due.

***boot***

la procedura d'avvio che si verifica subito dopo l'accensione di un computer, quando le periferiche vengono riconosciute una dopo l'altra e il sistema operativo viene caricato in memoria.

***bootloader***

è un programma che provvede ad avviare un sistema operativo. Molti bootloader offrono la possibilità di caricare più di un sistema operativo grazie a un menu di boot che vi permette di scegliere il sistema desiderato. Bootloader come *GRUB* sono molto diffusi proprio grazie a questa caratteristica, e sono molto utili nel caso di sistemi in cui risiedano due o più sistemi operativi.

***BSD***

*Berkeley Software Distribution*. Una variante di *Unix* sviluppata presso il dipartimento di informatica dell'Università di Berkeley. Questa versione è sempre stata considerata più avanzata sul piano tecnico rispetto alle altre, e ha introdotto molte innovazioni nel mondo informatico in generale e, in particolare, per quanto riguarda *Unix*.

***buffer***

una piccola parte della memoria di dimensioni fisse, che può essere associata a un file in modalità a blocchi, a una tabella di sistema, a un processo, e così via. La consistenza di tutti i buffer viene mantenuta per mezzo della buffer cache. Si veda **buffer cache**.

***buffer cache***

un parte essenziale del kernel di un sistema operativo, ha il compito di tenere aggiornati tutti i buffer, ridimensionando la cache quando necessario, svuotando i buffer non più necessari, e altro ancora. Si veda anche **buffer**.

***bug***

comportamento illogico o incoerente di un programma in una situazione particolare, oppure un comportamento che non rientra nei casi previsti dalla documentazione del programma o dagli standard cui quest'ultimo dovrebbe essere conforme. Nuove funzionalità spesso introducono nuovi bug in un programma. Il termine, dal punto di vista storico, deriva dai tempi (antichi!) delle schede perforate: un insetto (ingl. *bug*) era scivolato in un buco di una scheda perforata e, di conseguenza, il programma non si era comportato correttamente. Dopo aver scoperto la causa, Ada Lovelace esclamò "It's a bug!", e da allora la definizione si è diffusa ed è diventata universale.

***byte***

otto bit consecutivi, interpretati come un numero tra 0 e 255 in base due. Si veda **bit**.

***canali IRC***

sono i "luoghi" all'interno dei server IRC dove potete chiacchierare con altre persone. I canali sono creati nei server IRC, e gli utenti entrano in questi canali per comunicare con altre persone. I messaggi scritti su un canale sono visibili soltanto alle persone connesse a quel canale. Due o più utenti possono creare un canale "private" per non essere disturbati da altri utenti. I nomi dei canali cominciano con un `#`.

***carattere nullo***

il carattere o numero di byte 0, viene utilizzato per indicare la fine di una stringa. Il suo nome tecnico è `NULL`.

`case`

senza equivalente in italiano, in relazione alle stringhe di caratteri il *case* è la differenza fra lettere minuscole e lettere maiuscole.

*Si veda anche:* PAP.

***CHAP***

*Challenge-Handshake Authentication Protocol*: protocollo usato dagli ISP nella fase di autenticazione dei loro client. Secondo tale schema, viene inviato un valore al client (la macchina che si collega), quest'ultimo calcola un *hash* sulla base di tale valore e lo invia al server, infine il server compara l'*hash* con quello che ha calcolato. Si veda anche **PAP**.

***CIFS***

*Common Internet FileSystem* Il predecessore del filesystem SMB, usato sui sistemi *DOS*.

***client***

programma o computer che in maniera intermittente e temporanea si connette a un altro programma o computer per comunicargli comandi o chiedere informazioni. È uno dei componenti di un **sistema client/server**.

***codice oggetto***

è il codice generato dal processo di compilazione, che deve essere collegato ad altri codici oggetto e librerie per formare un file eseguibile. Il codice oggetto è comprensibile dalla macchina.

*Si veda anche:* compilazione, linkage.

***compilazione***

si tratta del processo di traduzione del codice sorgente, comprensibile per un essere umano (beh, con un po' di allenamento) e scritto in un qualsiasi linguaggio di programmazione (il *C*, ad esempio), in un file binario comprensibile dall'elaboratore.

***completamento automatico***

la capacità di una *shell* di espandere automaticamente una sottostringa in un nome di file, nel nome di un utente o in altro ancora, a condizione che vi sia una corrispondenza.

***compressione***

è il metodo di ridurre le dimensioni dei file o di diminuire il numero di caratteri inviati nel corso di una connessione di rete. Tra i programmi di compressione dei file citiamo *compress*, `zip`, `gzip`, e `bzip2`.

***configurabile via temi***

un'applicazione grafica è configurabile via temi se è in grado di cambiare il suo aspetto in tempo reale. Anche molti window manager sono configurabili via temi.

***console***

è il nome dato a quello che una volta era chiamato terminale. Si trattava di una macchina utente (uno schermo più una tastiera) connessa a un computer centrale molto potente (mainframe). Sui *PC* un terminale fisico è costituito da schermo e tastiera.

*Si veda anche:* console virtuali.

***console virtuali***

conservano il nome dato a quelli che una volta erano chiamati terminali. Nei sistemi *GNU/Linux* avete a vostra disposizione delle "console virtuali", così chiamate perché vi permettono di usare un solo schermo e una sola tastiera per molte sessioni indipendenti. Come opzione predefinita, sono disponibili in tutto

sei console virtuali, cui si può accedere premendo le combinazioni di tasti da **ALT-F1** a **ALT-F6**. Esiste una settima console, anche questa caratteristica predefinita del sistema, **ALT-F7**, che vi consente di accedere a una sessione X in esecuzione. Se vi trovate già in X, invece, potete accedere alle console di testo premendo le combinazioni di tasti che vanno da **CTRL-ALT-F1** a **CTRL-ALT-F6**.

*Si veda anche:* console.

***cookie***

file temporanei scritti sul disco rigido locale da un *web* server remoto. Permettono al server di ricordare le preferenze dell'utente quando quest'ultimo si connette nuovamente.

***denominazione***

una parola usata comunemente nel campo dell'informatica per indicare un metodo di identificazione degli oggetti. Avrete sentito spesso parlare di "convenzioni di denominazione" per file, funzioni nei programmi, e così via.

***desktop***

Se state usando il sistema X Window, il desktop (o "scrivania") è la rappresentazione sullo schermo dell'ambiente grafico in cui lavorate, all'interno del quale sono visualizzate le finestre e le icone. È chiamato anche background (o "sfondo"), e in genere è visualizzato come un colore pieno, una sfumatura di due colori o anche un'immagine.

*Si veda anche:* .

***DHCP***

*Dynamic Host Configuration Protocol*. Un protocollo progettato per far sì che le macchine presenti in una rete locale ottengano dinamicamente un indirizzo IP da un server DHCP.

***dipendenze***

sono le fasi di compilazione che è necessario siano soddisfatte prima di poter passare a quelle successive e portare a termine la compilazione di un programma.

***directory***

Parte della struttura del filesystem. All'interno di una directory possono trovarsi file o altre directory. A volte, all'interno di una directory è presente una serie di sotto-directory: spesso ci si riferisce a questa struttura gerarchica con il termine "albero delle directory", all'interno del quale le sotto-directory costituiscono i rami e i file le foglie. Per vedere il contenuto di un'altra directory, dovrete spostarvi in essa o elencarlo da dove vi trovate. Le directory sono soggette alle stesse limitazioni dei file, per quanto i permessi abbiano un significato differente. Le directory speciali '.' e '..' si riferiscono, rispettivamente, alla directory stessa e a quella immediatamente superiore.

***directory home***

spesso abbreviata con "home", è il nome della directory personale di un dato utente. Si veda anche **account**.

***directory radice***

Si riferisce alla directory di livello più alto in un filesystem. Questa non ha una directory superiore, pertanto il simbolo '..' fa riferimento a se stessa. La directory root viene indicata con il simbolo '/'.

***disco di boot***

un floppy disk in grado di effettuare il boot del computer, contenente i programmi necessari per caricare il sistema operativo dal disco rigido (ma talvolta all'interno di un disco di boot si trova tutto quanto serve per avviare un sistema operativo).

***distribuzione***

è il termine usato per indicare il sistema *GNU/Linux* progettato e assemblato da uno specifico venditore. Una distribuzione consiste del kernel centrale di Linux e dei programmi indispensabili al sistema, oltre a questi sono presenti programmi di installazione, programmi di terze parti, un buon numero di applicazioni liberamente distribuibili e, a volte, anche del software proprietario.

***DMA***

*Direct Memory Access*. Una caratteristica dell'architettura *PC* che permette a una periferica di leggere o scrivere dalla memoria centrale senza alcun aiuto da parte della CPU. Le periferiche PCI usano la tecnica del bus mastering e non hanno bisogno del DMA.

***DNS***

*Domain Name System*. Il meccanismo distribuito nome/indirizzo utilizzato su Internet. Questo meccanismo vi permette di rimappare un nome di dominio con un indirizzo IP, ed è grazie ad esso che potete cercare e caricare un sito senza che sia necessario conoscere il suo indirizzo IP. Il DNS permette anche l'operazione contraria, cioè ricavare il nome di una macchina dal suo indirizzo IP.

***DPMS***

*Display Power Management System*. Protocollo usato da tutti i monitor moderni al fine di gestire le funzionalità di risparmio energetico. I monitor che supportano questa caratteristica sono comunemente noti come "monitor verdi".

***echo***

è il fenomeno per cui i caratteri digitati nel campo di testo del nome utente, ad esempio, vengono mostrati "tali e quali", invece di visualizzare "*" per ciascuno di loro.

***editor***

è il termine più comune per programmi utilizzati per modificare file di testo (da qui anche l'espressione "editor di testi"). Gli editor più famosi sotto *GNU/Linux* sono l'editor GNU Emacs (*Emacs*) editor e l'editor *Unix*, *VI*.

***ELF***

*Executable and Linking Format*. È il formato per i file eseguibili usato oggi da quasi tutte le distribuzioni *GNU/Linux*.

***email***

sta per *Electronic Mail*, ovvero "posta elettronica". Si tratta dello scambio di messaggi in formato elettronico fra persone che si trovano sulla stessa rete. In modo simile alla posta tradizionale (detta anche *snail mail*, "posta lumaca"), per funzionare la posta elettronica necessita di una destinazione e di un mittente. Il mittente deve avere un indirizzo come "mittente@dominio.del.mittente" e il destinatario deve avere un indirizzo simile ("destinatario@dominio.del.destinatario"). La posta elettronica è un metodo di comunicazione estremamente rapido, in genere occorrono solo pochi minuti per raggiungere chiunque, non importa dove si trovi nel mondo. Per scrivere messaggi di posta elettronica è necessario usare un client email come *Pine* o `mutt`, programmi con interfaccia testuale, oppure client dotati di una GUI, come *KMail*.

***escape***

nel contesto della shell, è l'azione di circondare qualche stringa di testo fra doppi apici in maniera da impedire alla shell di interpretare tale stringa. Ad esempio, quando avete la necessità di usare

degli spazi in qualche linea di comando e inviare con una pipe i risultati a qualche altro comando, dovete racchiudere il primo comando fra doppi apici, altrimenti la shell non lo interpreterà in maniera corretta e non funzionerà come desiderato.

***espressione regolare***

un potente strumento concettuale utilizzato per cercare e confrontare stringhe di testo. Permette di specificare modelli ai quali devono conformarsi le stringhe. Molti programmi di utilità sotto *Unix* fanno uso di espressioni regolari: `sed`, `awk`, `grep`, `perl` e altri ancora.

***ext2***

abbreviazione per "Extended 2 filesystem". Questo è il filesystem nativo di *GNU/Linux*, e possiede tutte le caratteristiche di un qualsiasi filesystem *Unix*: supporto per file speciali (dispositivi a caratteri, link simbolici, etc.), permessi e proprietà relativi a file e directory, e così via.

***FAQ***

*Frequently Asked Questions*: documento che contiene una serie di domande e risposte riguardo un argomento specifico. Storicamente le FAQ sono comparse nei gruppi di discussione su Usenet, ma questo tipo di documento adesso è comune su molti siti *web*, e persino dei prodotti commerciali hanno le loro FAQ. In genere costituiscono ottime fonti d'informazione.

***FAT***

*File Allocation Table*. Il filesystem usato da *DOS* e *Windows*.

***FDDI***

*Fiber Distributed Digital Interface*. Uno strato fisico di rete ad alta velocità, che usa le fibre ottiche per la comunica-

zione. Usato unicamente su reti molto estese, soprattutto a causa del suo costo.

***FHS***

*Filesystem Hierarchy Standard*. Un documento che contiene le linee guida per una coerente organizzazione della struttura gerarchica del filesystem su sistemi *Unix*. La distribuzione **Linux-Mandrake** è quasi del tutto conforme a questo standard.

***FIFO***

*First In, First Out*. Una struttura dati o un buffer hardware dai quali gli oggetti vengono estratti nello stesso ordine in cui erano stati inseriti. Le pipe di *Unix* costituiscono uno dei più complessi esempi di FIFO.

***file in modalità a blocchi***

file il cui contenuto è bufferizzato. Tutte le operazioni di lettura/scrittura che riguardano tali file passano attraverso dei buffer, fatto che permette scritture in modo asincrono sull'hardware sottostante e, per quanto riguarda le operazioni di lettura, di non leggere di nuovo ciò che è già presente in un buffer.

*Si veda anche:* buffer, buffer cache.

***file in modalità a caratteri***

file il cui contenuto non è bufferizzato. Tutte le operazioni di input/output vengono eseguite immediatamente in maniera diretta. Corrispondono ai flussi di dati.

*Si veda anche:* file in modalità a blocchi.

***file nascosto***

è un file che non può esser "visto" quando si digita il comando `ls` senza opzioni. I nomi dei file nascosti iniziano con un `.` (per antica convenzione Unix). Tali file vengono usati per archiviare

le preferenze personali e le configurazioni dei diversi programmi usati da un utente. Ad esempio, l'elenco degli ultimi comandi eseguiti da *Bash* viene salvato nel file `.bash_history`, che è un file nascosto.

***filesystem***

il metodo usato per registrare i file su supporti fisici (dischi rigidi, floppy, etc.) in maniera coerente. Esempi di filesystem sono la FAT, l'*ext2fs* di *GNU/Linux*, l'`iso9660` (usato sui CDROM), e così via.

***filesystem radice***

è il filesystem di livello più alto. Questo è il filesystem sul quale *GNU/Linux* monta l'intero albero delle directory. È necessario che il filesystem radice si trovi su una partizione dedicata, in quanto è la base dell'intero sistema. Contiene la directory radice.

***firewall***

una macchina o comunque un dispositivo hardware dedicato che, nella topologia di una rete locale, è l'unico punto di connessione alla rete esterna, e filtra tale connessione, o controlla l'attività di alcune porte, o si accerta che soltanto alcune specifiche interfacce IP possano accedere a queste ultime.

***flag***

è un indicatore (in genere un solo bit) usato per segnalare (*flag* significa "bandiera") una particolare condizione a un programma. Un filesystem, ad esempio, possiede, fra gli altri, un flag che indica se è giunto il momento di effettuare un backup con `dump`, così quando il flag è attivo viene effettuata una copia di sicurezza del filesystem, mentre se è inattivo questo non succede.

***focus***

la possibilità, per una finestra, di ricevere segnali dalla tastiera (come la pressione e il rilascio dei tasti) e clic del mouse, a meno che tali eventi siano intercettati dal window manager.

***foreground***

nell'ambito della *shell*, il processo in foreground ("primo piano") è quello attualmente in corso di esecuzione. Siete obbligati ad aspettare che tale processo sia terminato prima di poter digitare di nuovo dei comandi. Si veda **job**, **background**.

*Si veda anche:* job, background.

***framebuffer***

la proiezione della RAM presente su una scheda grafica nella memoria centrale. Questo permette alle applicazioni di accedere alla memoria video senza dover prima dialogare con la scheda grafica. Tutte le workstation grafiche di fascia alta, ad esempio, usano dei framebuffer.

***FTP***

*File Transfer Protocol*. È il protocollo standard usato su *Internet* per trasferire file da una macchina all'altra.

***gateway***

anello di collegamento fra due reti IP.

***GIF***

*Graphics Interchange Format*. Un formato di file per immagini, ampiamente usato sul *web*. Le immagini GIF possono essere compresse, e anche animate. A causa di problemi relativi al copyright, non è una buona idea usare questo formato, è consigliabile rimpiazzarle il più possibile con il più avanzato formato PNG.

***GNU***

*GNU's Not Unix*. Il progetto GNU è nato per iniziativa di *Richard Stallman* all'inizio degli anni '80, il suo scopo è lo sviluppo di un sistema operativo libero ("libero" nel senso di "libertà di parola"). Al momento attuale, tutte le varie componenti sono disponibili, con l'eccezione... del kernel. Il kernel del progetto GNU, *Hurd*, non è ancora affidabile al 100%. *GNU/Linux* utilizza due componenti, in particolare, del progetto GNU: il suo compilatore *C*, `gcc`, e la licenza GPL. Si veda **GPL**.

***GPL***

*General Public License*. La licenza del kernel *GNU/Linux*, è l'esatto contrario di tutte le licenze di software proprietario in quanto non impone nessuna limitazione per quanto riguarda la copia, la modifica e la redistribuzione del software, a condizione che il codice sorgente sia sempre disponibile. L'unica limitazione, se possiamo chiamarla così, consiste nel fatto che le persone alle quali redistribuite il software devono poter beneficiare degli stessi diritti.

***gruppo proprietario***

nel contesto degli utenti e dei loro file, il gruppo proprietario di un file è il gruppo cui appartiene l'utente che lo ha creato.

***gruppi di discussione***

(ingl. *newsgroups*) si riferisce ad aree dedicate alla discussione, a informazioni e a notizie varie cui si può accedere con un programma client appropriato per leggere e scrivere messaggi in relazione all'argomento del gruppo. Il gruppo di discussione `alt.os.linux.mandrake`, ad esempio, è un gruppo alternativo (alt) che tratta del sistema operativo (os) *GNU/Linux*, in particolare di **Linux-Mandrake** (mandrake). I nomi dei gruppi di discussione sono strutturati in questo modo per rendere più facile la ricerca di un argomento specifico.

***GUI***

*Graphical User Interface*. L'interfaccia verso un computer, consiste di menu, pulsanti, icone e così via. La quasi totalità degli utenti preferisce una GUI alla CLI (*Command Line Interface*, "Interfaccia a Linea di Comando") per la maggiore facilità d'uso, malgrado quest'ultima sia più flessibile.

***host***

si riferisce a un computer, è il termine usato normalmente per indicare computer connessi a una rete.

***HTML***

*HyperText Markup Language*. Il linguaggio usato per creare documenti *web*.

***HTTP***

*HyperText Transfer Protocol*. Il protocollo usato per connettere i siti *web* e trasferire documenti HTML.

***icona***

termine che indica una piccola immagine o disegno (normalmente di 16x16, 32x32, 48x48, e a volte 64x64 pixel) che rappresenta, in un ambiente grafico, un documento, un file o un programma.

***IDE***

*Integrated Drive Electronics*. Il bus di gran lunga più diffuso sui *PC* attuali per la connessione di dischi rigidi. Un bus IDE può ospitare fino a due dispositivi, e la velocità del bus è limitata dal dispositivo che ha la coda di comando più lenta (e non la velocità di trasferimento più lenta!).

*Si veda anche:* ATAPI.

***indirizzo IP***

è un indirizzo numerico, composto da quattro elementi, che identifica il vostro computer su Internet. Gli indirizzi IP sono strutturati secondo un metodo gerarchico, con livelli superiori e domini nazionali, domini, sotto-domini e l'indirizzo individuale di ciascuna macchina. Un indirizzo IP ha l'aspetto di qualcosa tipo 192.168.0.1. L'indirizzo individuale di una macchina può appartenere a due tipi diversi: statico o dinamico. Gli indirizzi IP di tipo statico sono indirizzi permanenti, che non cambiano mai. Gli indirizzi IP di tipo dinamico cambiano con ogni nuova connessione alla rete. Gli utenti di connessioni via modem e via cavo in genere hanno indirizzi IP di tipo dinamico, mentre gli utenti di connessioni DSL e altre connessioni ad alta velocità in genere dispongono di indirizzi IP statici.

***inode***

punto di entrata che porta al contenuto di un file su un filesystem della famiglia *Unix*. Un inode è identificato in maniera univoca da un numero, e contiene meta-informazioni riguardo il file cui fa riferimento, quali le sue date di accesso, il suo tipo, le sue dimensioni, ma non il suo nome!

***Internet***

la rete di dimensioni smisurate che connette computer di tutto il mondo.

***IP masquerading***

si riferisce all'uso di un firewall per nascondere il vero indirizzo IP del vostro computer. Grazie al firewall, ogni connessione alla rete esterna erediterà l'indirizzo IP di quest'ultimo. Questo è utile nelle situazioni in cui è disponibile una connessione a Internet veloce con un unico indirizzo IP, ma desiderate usare più di un computer con indirizzi IP interni già assegnati per connettervi a Internet.

***IRC***

*Internet Relay Chat*. Uno dei pochi standard di *Internet* per conversazioni in tempo reale. Permette la creazione di canali, discussioni private, e anche lo scambio di file. Inoltre è progettato in maniera tale da permettere la connessione dei server l'uno con l'altro, il che spiega perché oggi esistono numerose reti IRC: **Undernet**, **DALnet**, **EFnet** per nominarne soltanto alcune.

***ISA***

*Industry Standard Architecture*. Il primissimo bus usato sui *PC*, viene pian piano abbandonato a favore del bus PCI. Alcuni produttori di hardware, tuttavia, continuano a usarlo: è un evento ancora molto comune trovare delle schede SCSI di tipo ISA fornite a corredo di periferiche come scanner, masterizzatori, etc. Un vero peccato.

***ISDN***

*Integrated Services Digital Network*. Un insieme di standard di comunicazione aventi come scopo la trasmissione di voce, servizi di rete e video per mezzo di un singolo cavo o fibra ottica. È stato progettato per rimpiazzare, nel lungo periodo, gli attuali sistemi di comunicazione telefonica.

***ISO***

*International Standards Organisation*. Un gruppo di società, consulenti, università e altre componenti che ha come scopo l'elaborazione di standard applicabili in vari ambiti, incluso quello dell'informatica. I documenti che descrivono tali standard sono caratterizzati da un numero: lo standard numero 9660, ad esempio, descrive il filesystem utilizzato sui CDROM.

***ISO 8859***

Lo standard ISO 8859 comprende un certo numero di estensioni a 8-bit al set di caratteri ASCII (si veda *ASCII*). Riveste particolare importanza l'ISO 8859-1, l'"Alfabeto Latino N. 1", che ha

conosciuto un gran numero di implementazione ed è spesso visto come lo standard che rimpiazzerà l'ASCII.

L'ISO 8859-1 (Figura 2) offre il supporto per le lingue che seguono: Afrikaans, Basco, Catalano, Danese, Olandese, Inglese, Faroese, Finnico, Francese, Galizio, Tedesco, Islandese, Irlandese, Italiano, Norvegese, Portoghese, Scozzese, Spagnolo, e Svedese.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| A | | ¡ | ¢ | £ | ¤ | ¥ | ¦ | § | ¨ | © | ª | « | ¬ | – | ® | ¯ |
| B | ° | ± | ² | ³ | ´ | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| C | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| D | Ð | Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| E | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ī |
| F | ð | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

**Figura 2. La tabella ISO-8859-1**

Notate che i caratteri dell'ISO 8859-1 costituiscono anche i primi 256 caratteri dell'ISO 10646 (Unicode). Questo standard, tuttavia, non ha il simbolo dell'EURO e non copre in maniera completa Finnico e Francese. L'ISO 8859-15 (Figura 3) è una modifica dell'ISO 8859-1 soddisfa tali necessità.

Il set completo degli alfabeti dell'ISO 8859 comprende:

| Nome | Lingua (o Lingue) |
|---|---|
| ISO 8859-1 | lingue dell'Europa occidentale (Latin-1) |
| ISO 8859-2 | lingue dell'Europa orientale (Latin-2) |

| Nome | Lingua (o Lingue) |
|---|---|
| ISO 8859-3 | lingue dell'Europa sud-orientale e altre (Latin-3) |
| ISO 8859-4 | lingue Scandinave/Baltiche (Latin-4) |
| ISO 8859-5 | Latin/Cirillico |
| ISO 8859-6 | Latin/Arabico |
| ISO 8859-7 | Latin/Greco |
| ISO 8859-8 | Latin/Ebraico |
| ISO 8859-9 | modifica Latin-1 per il Turco (Latin-5) |
| ISO 8859-10 | lingue Lapponi/Nordiche/Eschimesi (Latin-6) |
| ISO 8859-11 | Tailandese |
| ISO 8859-13 | lingue dell'area Baltica (Latin-7) |
| ISO 8859-14 | Celtico (Latin-8) |
| ISO 8859-15 | lingue dell'Europa occidentale con il simbolo dell'Euro (Latin-9) |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | |
| A | | ¡ | ¢ | £ | € | ¥ | Š | § | š | © | ª | « | ¬ | – | ® | ¯ |
| B | ° | ± | ² | ³ | Ž | µ | ¶ | · | ž | ¹ | º | » | Œ | œ | Ÿ | ¿ |
| C | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| D | Ð | Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| E | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| F | ð | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

**Figura 3. Tabella dell'ISO-8859-15**

***ISP***

*Internet Service Provider*. Una società che vende l'accesso a *Internet* ai suoi clienti, accesso che può avvenire per mezzo delle normali linee telefoniche o su linee dedicate.

***job***

nell'ambito di una *shell*, un job è un processo che viene eseguito in background. Potete avere più di un job nella stessa shell, e controllarne l'esecuzione. Si veda anche **background**, **foreground**.

***jolly***

I caratteri '*' e '?' vengono usati come caratteri jolly e possono rappresentare qualsiasi cosa. L'asterisco ('*') rappresenta qualsiasi numero di caratteri, incluso nessun carattere. Il punto interrogativo ('?') rappresenta esattamente un unico carattere. I caratteri jolly sono usati spesso nelle espressioni regolari.

***JPEG***

*Joint Photographic Experts Group*. Un altro formato di file per immagini molto comune. Il formato JPEG è adatto soprattutto alla compressione di immagini tratte dal mondo reale, e non funziona molto bene con immagini non realistiche.

***kernel***

si riferisce alla parte più interna e importante del sistema operativo. Il kernel è responsabile di assegnare risorse e di separare i processi l'uno dall'altro. Gestisce tutte le operazioni a basso livello che permettono ai programmi di interagire direttamente con l'hardware del vostro computer, gestisce la buffer cache, e così via.

***kill ring***

sotto *Emacs*, è l'insieme di aree di testo tagliate o copiate dal momento in cui si è utilizzato l'editor (buffer di copia/cancellazione); queste possono essere richiamate per essere inserite nuovamente, e sono organizzate sotto forma di anello.

***LAN***

*Local Area Network*. Nome generico per indicare una rete di macchine connesse dallo stesso cavo fisico.

***lanciare***

l'azione di richiamare, o avviare, un programma.

***LDP***

*Linux Documentation Project*. Un'organizzazione senza scopo di lucro che cura della documentazione relativa a *GNU/Linux*. I suoi documenti più conosciuti sono gli *HOWTO*, ma si occupa anche dell'organizzazione e dell'aggiornamento di numerose FAQ, e persino di un certo numero di libri.

***libreria***

si riferisce a una collezione di procedure e funzioni in forma binaria che i programmatori possono usare nei loro programmi (purché la licenza della libreria glielo consenta). Il programma cui spetta il compito di caricare le librerie condivise al momento dell'esecuzione è chiamato *dynamic linker*.

***linea di comando***

quella fornita da una shell, che permette all'utente di digitare comandi direttamente. È anche l'oggetto di un'eterna "flame war" tra i suoi sostenitori e i suoi detrattori **:-)**

***link***

il riferimento a un inode in una directory, che pertanto assegna un nome (di file) all'inode. Fra gli inodi che non hanno un link,

e quindi nemmeno un nome, possiamo citare, ad esempio: pipe anonime (quelle usate dalla shell), socket (ovvero connessioni di rete), dispositivi di rete, e così via.

***linkage***

l'ultima fase di un processo di compilazione, consiste nel collegare insieme tutti i file oggetto per produrre un file eseguibile; in questa stessa fase, tutti i simboli non risolti vengono identificati con librerie dinamiche (a meno che non sia stato richiesto un linkage di tipo statico, nel qual caso il codice di questi simboli verrà incluso nel file eseguibile).

***link simbolico***

file speciali, che non contengono nulla a parte una stringa che fa riferimento a un altro file. Ogni accesso a essi è equivalente a un accesso al file il cui nome è rappresentato da detta stringa; quest'ultimo può esistere o no, e il suo percorso può essere dato in termini relativi o assoluti.

***Linux***

si riferisce a un sistema operativo simile a *Unix* che può girare su computer di vario tipo, il cui uso e adattamento è libero e gratuito per chiunque. Linux (il kernel) è stato scritto da Linus Torvalds.

***livelli di sicurezza***

una caratteristica esclusiva di **Linux-Mandrake** che vi permette di impostare diversi livelli di restrizioni a seconda di quanto vogliate rendere il vostro sistema. Esistono 6 livelli predefiniti, che vanno da 0 a 5, dove 5 indica il livello più alto di sicurezza. Potete anche definire un vostro personale livello di sicurezza.

***login***

il nome di accesso di un utente su un sistema *Unix*, e l'atto del connettersi al sistema.

***lookup table***

è una tabella che stabilisce una corrispondenza fra codici (o etichette) e il loro reale significato. Spesso si tratta di un file di dati usato da un programma per ottenere ulteriori informazioni riguardo un oggetto particolare.

*HardDrake*, ad esempio, usa una simile tabella per sapere cosa significa il codice assegnato da un produttore a un dispositivo hardware. Questa è una riga appartenente a tale tabella, che ci fornisce informazioni in merito all'oggetto `CTL0001`

```
CTL0001 sound   sb      Creative Labs   SB16 \
                         HAS_OPL3|HAS_MPU401|HAS_DMA16|HAS_JOYSTICK
```

`loopback`

interfaccia di rete virtuale di una macchina con se stessa, che permette ai programmi in esecuzione di fare a meno di prendere in considerazione il caso speciale per cui due entità di rete sono, di fatto, la stessa macchina.

***major***

numero specifico alla classe del dispositivo (numero primario).

***MBR***

*Master Boot Record*. nome dato al primo settore di un disco rigido in grado di effettuare il boot. L'MBR contiene il codice utilizzato per caricare il sistema operativo in memoria o un bootloader (come *LILO*), e la tabella delle partizioni del disco rigido cui appartiene il settore che lo ospita.

***menu a discesa***

è un menu che si presenta come “arrotolato”, con un bottone in uno dei suoi angoli: quando cliccate su quel bottone, il menu “si srotola” mostrandovi l’intero contenuto.

***meta-espansione***

nell’ambito della *shell*, la possibilità di raggruppare un certo numero di nomi di file grazie a un modello di meta-espansione. Si veda **modello di meta-espansione**.

***MIME***

*Multipurpose Internet Mail Extensions*. Una stringa con il formato `tipo/sotto-tipo` che descrive il contenuto di un file allegato a un messaggio di posta elettronica. Questo permette ai client di posta elettronica in grado di sfruttare lo standard MIME di definire azioni in corrispondenza del tipo del file.

***minor***

numero che identifica un particolare dispositivo fisico all’interno di una classe (numero secondario).

***modalità comandi***

sotto *VI* o uno dei suoi cloni, è lo stato del programma in cui la pressione di un tasto (questo riguarda soprattutto le lettere) non avrà come conseguenza il suo inserimento nel file che viene modificato, bensì l’esecuzione di un azione specifica per quel tasto (a meno che il clone non offra la possibilità di rimappare i comandi e voi abbiate personalizzato la vostra configurazione). Se ne può uscire digitando uno dei comandi “torna alla modalità inserimento”: **i**, **I**, **a**, **A**, **s**, **S**, **o**, **O**, **c**, **C**, ...

***modalità inserimento***

sotto *VI* o uno dei suoi cloni, è lo stato del programma in cui premendo un tasto inserirà il carattere corrispondente nel file che viene modificato (con l’eccezione di alcuni casi patologici co-

me il completamento automatico di un'abbreviazione, la giustificazione a destra alla fine della riga, etc.). È possibile uscirne premendo il tasto `Esc` (o `Ctrl-[`).

***modalità lettura-scrittura***

per un file, significa che può essere scritto. Potete leggere il suo contenuto e modificarlo. Se disponete di questo tipo di permesso, potete anche cancellare il file.

***modalità sola lettura***

per un file, significa che non può essere scritto. Potete leggere il suo contenuto, ma non potete modificarlo, e non potete cancellare il file.

***modalità prolissa***

Per quanto riguarda i comandi, la modalità prolissa significa che il comando comunica allo standard output (o allo standard error) tutte le azioni che compie e il risultato di queste azioni. A volte i comandi possono definire un "livello di prolissità", in altre parole può essere controllata la quantità di informazioni che il comando trasmette al momento del suo utilizzo.

***modello di meta-espansione***

una stringa composta di caratteri normali e caratteri speciali. I caratteri speciali (meta-caratteri) sono interpretati ed espansi dalla *`shell`*.

***monoutente***

termine usato per descrivere la condizione di un sistema operativo, o persino il sistema operativo stesso, che permette l'accesso al sistema e il suo uso soltanto a un utente per volta.

***montare, montato***

Un dispositivo è detto montato quando è connesso e riconosciuto dal filesystem *GNU/Linux*. Quando montate un dispositivo potete esplorarne i contenuti. Questo termine è almeno in parte obsoleto, in quanto grazie al programma "supermount" incluso in **Linux-Mandrake**, gli utenti non sono più costretti a montare manualmente supporti rimuovibili.

*Si veda anche:* punto di mount.

***MPEG***

*Moving Pictures Experts Group*. Un comitato dell'ISO che definisce gli standard per la compressione di audio e video. MPEG è anche il nome dei loro algoritmi. Sfortunatamente la licenza per questo formato è molto restrittiva, e di conseguenza non esistono ancora riproduttori MPEG che siano *Open Source*...

***multitasking***

la capacità di un sistema operativo di condividere il tempo di elaborazione della CPU fra più processi contemporaneamente in esecuzione. A basso livello, questa caratteristica è conosciuta anche come multi-programmazione. Il passaggio da un processo a un altro richiede che tutto il contesto attuale del processo venga salvato e ricaricato quando è di nuovo il turno di quel processo. Questa operazione è nota come *context switch* ("passaggio da un contesto all'altro") e, su macchine Intel, si verifica 100 volte al secondo: dunque è sufficientemente veloce da dare a un utente l'impressione che il sistema operativo stia eseguendo più applicazioni nello stesso momento. Esistono due tipi di multitasking: il *preemptive multitasking* ("multitasking basato su diritti di precedenza"), in cui il sistema operativo ha la responsabilità di sottrarre la CPU a un processo per passarla a un altro; e il *cooperative multitasking* ("multitasking basato sulla cooperazione"), in cui è il processo stesso che decide quand'è il momento di restituire la CPU. La prima variante, ovviamente, è la scelta migliore, in quanto nessun programma può assorbire l'in-

tero tempo di elaborazione della CPU e bloccare così altri processi; *GNU/Linux* è caratterizzato da un multitasking di questo tipo. Il metodo in base al quale viene scelto il processo da eseguire dipende da diversi parametri, ed è chiamato *scheduling* ("programma di lavoro").

***multiutente***

è usato per descrivere un sistema operativo che consente l'accesso e l'uso del sistema a più utenti nello stesso momento, ciascuno dei quali è messo in grado di svolgere la propria attività in maniera indipendente dagli altri utenti. Per offrire un supporto multiutente è indispensabile un sistema operativo caratterizzato dal multitasking. *GNU/Linux* è un sistema operativo multitasking e multiutente, come qualsiasi altro sistema *Unix*.

***NCP***

*NetWare Core Protocol*: protocollo definito dalla **Novell** per accedere a file e servizi di stampa tramite Novell NetWare.

***NFS***

*Network FileSystem*. Un filesystem di rete creato dalla **Sun Microsystems**, avente come obiettivo la condivisione di file all'interno di una rete in modo del tutto trasparente.

***NIC***

*Network Interface Controller*: adattatore installato in un computer per ottenere una connessione fisica a una rete, tipicamente una scheda *Ethernet*.

***NIS***

*Network Information System*. Il NIS era conosciuto anche come "*Yellow Pages*", ma la **British Telecom** possiede un copyright su questo nome. Il NIS è un protocollo definito dalla **Sun Microsystems** avente come scopo la condivisione di informazio-

ni all'interno di un **dominio** NIS, che può comprendere un'intera LAN, parte di una LAN o anche più LAN. Può esportare database di password, database di servizi, informazione su gruppi e altro ancora.

***nome utente***

è un nome (o, più in generale, una parola) che identifica un utente in un sistema. Ad ogni nome utente corrisponde un'unica e univoca UID (*user ID*)

*Si veda anche:* login.

***open source***

si riferisce al nome dato al codice liberamente distribuibile di un programma reso disponibile per la comunità di programmatori e pubblico in un senso più generale. Secondo la teoria che sta alla base di questo concetto, se si permette al codice sorgente di essere usato e modificato da un ampio gruppo di programmatori il risultato finale sarà, in ultima analisi, un prodotto migliore per tutti. Tra i più diffusi programmi open source citiamo *Apache*, *sendmail* e *GNU/Linux*.

***pagina di manuale***

un breve documento contenente la definizione di un comando e il suo uso, da consultarsi per mezzo del comando `man` nome_del_comando. La prima cosa che si dovrebbe (imparare a) leggere quando si sente pronunciare il nome di un comando sconosciuto **:-)**

***PAP***

*Password Authentication Protocol*: protocollo usato da molti ISP nella fase di autenticazione del client. Secondo questo metodo, il client (e cioè voi) invia una coppia identificatore/password non criptata al server. Si veda anche **CHAP**.

***password***

è una parola segreta, o la combinazione di parole o lettere, usata per garantire la sicurezza di particolari operazioni. Le password sono usate insieme ai nomi di login degli utenti su sistemi operativi multiutente, siti web, siti FTP, e così via. Dovrebbero essere frasi o combinazioni alfa-numeriche difficili da indovinare, e non dovrebbero mai essere basate su semplici parole reperibili in un dizionario. Le password assicurano che nessun altro possa accedere a un computer o a un sito utilizzando il vostro account.

***password ombra***

un metodo di gestione delle password sui sistemi *Unix* (ingl. "shadow password") in cui il file che contiene le password criptate non è più accessibile in lettura, mentre lo è quando si usa il sistema di gestione delle password normale.

***patch, applicare una patch***

file che contiene una serie di correzioni da applicare a un codice sorgente al fine di aggiungere nuove caratteristiche, di rimuovere bug, o di modificarlo secondo i desideri e i bisogni dell'autore. L'azione, ovvero l'applicazione di una patch (ingl. *to patch*), consiste nell'applicare tali correzioni direttamente all'archivio del codice sorgente.

***PCI***

*Peripheral Components Interconnect*. Un bus creato dalla **Intel** che ha raggiunto lo status di standard per le architetture *PC* odierne, e che viene usato anche su altre architetture. È il successore del bus ISA, e offre numerosi servizi: identificazione del dispositivo, informazioni sulla configurazione, condivisione di IRQ, bus mastering e altri ancora.

***PCMCIA***

*Personal Computer Memory Card International Association*: chiamato sempre più spesso "PC Card" per motivi di sem-

plicità, è lo standard relativo alle schede esterne inserite in un computer portatile: modem, dischi rigidi, espansioni di memoria, schede *Ethernet* e altro ancora. L'acronimo originario viene talvolta letto, a fini umoristici, come *People Cannot Memorize Computer Industry Acronyms*...

***percorso***

si riferisce alla collocazione di file e directory nel filesystem. I diversi componenti di un percorso sono separati dal carattere '/'. Esistono due tipi di percorso sui sistemi *GNU/Linux*: il percorso **relativo** corrisponde alla posizione di un file o una directory in relazione alla directory corrente; il percorso **assoluto** è la posizione di un file o una directory in relazione alla directory radice.

***pipe***

uno speciale tipo di file *Unix*. Un programma scrive dati in una pipe, e un altro programma legge i dati dall'altra estremità. Le pipe *Unix* sono di tipo FIFO, in modo che i dati vengono letti esattamente nello stesso ordine in cui erano stati immessi. Usate molto frequentemente con la shell. Si veda anche **pipe con nome**.

***pipe con nome***

una pipe *Unix* a cui è assegnato un nome nel filesystem, a differenza delle pipe usate nelle shell. Si veda anche **pipe**, **link**.

***pixmap***

acronimo per "pixel map" ("mappa basata su pixel"). È un altro modo per riferirsi alle immagini di tipo bitmap.

***plugin***

programma aggiuntivo usato per mostrare o suonare i contenuti multimediali di un documento *web*. Nel caso in cui il vostro navigatore non sia ancora capace di mostrare o suonare quel tipo

di informazione, non dovrebbe essere troppo difficile reperire il plugin relativo sul *web*.

***PNG***

*Portable Network Graphics*. Formato di file per immagini creato principalmente per essere usato sul *web*, è stato progettato per essere un sostituto non vincolato da brevetti del formato GIF, rispetto al quale offre alcune caratteristiche supplementari.

***PNP***

*Plug'N'Play*. Nato come un espansione del bus ISA mirata ad aggiungere informazioni di configurazione per i dispositivi, è diventato un termine dall'uso più generico che riunisce tutti i dispositivi in grado di comunicare i loro parametri di configurazione. In quanto tali, tutti i dispositivi PCI sono Plug'n'Play.

***POP***

*Post Office Protocol*. Il protocollo usato comunemente per scaricare la posta elettronica da un ISP.

***porting***

si riferisce al metodo di traduzione di un programma in maniera tale che possa essere usato in un sistema operativo diverso da quello per il quale era stato scritto, o che possa essere usato in sistemi "simili". Per poter eseguire un programma nativo di *Windows* sotto *GNU/Linux* (in modo nativo), ad esempio, è necessario che prima venga portato su *GNU/Linux*.

***PPP***

*Point to Point Protocol*. Questo è il protocollo usato per trasmettere dati usando linee seriali. È usato comunemente per inviare pacchettiIP su Internet, ma può essere usato anche con altri protocolli, quale ad esempio il protocollo IPX di Novell.

***precedenza***

determina l'ordine di valutazione degli operandi in una espressione. Ad esempio: nel caso di `4 + 3 * 2` il risultato è 10, in quanto la moltiplicazione ha la precedenza rispetto all'addizione. Se volete che prima venga calcolata l'addizione, dovete aggiungere delle parentesi così: `(4 + 3) * 2`, e il risultato finale ' 14, poiché le parentesi hanno la precedenza rispetto all'addizione e alla moltiplicazione, pertanto le operazioni che si svolgono fra le parentesi vengono calcolate prima di ogni altra.

***preprocessori***

sono direttive di compilazione che ordinano al compilatore di rimpiazzare tali direttive con codice usato nel linguaggio di programmazione del file sorgente. Esempi di preprocessori in *C*'s sono `#include`, `#define`, etc.

***processo***

in ambito *Unix*, un processo è costituito dall'instanza di un programma in esecuzione, insieme con il suo ambiente.

***prompt***

è la stringa di caratteri che precede il cursore di una *shell*. Quando è visibile è possibile digitare dei comandi.

***proprietario***

nel contesto degli utenti e dei loro file, il proprietario di un file è l'utente che lo ha creato.

***protocollo***

I protocolli organizzano la comunicazione fra macchine differenti all'interno di una rete, usando mezzi hardware o software. Definiscono il formato dei dati trasferiti, se una macchina ne controlla un'altra, etc. Tra i protocolli più diffusi citiamo HTTP, FTP, TCP, e UDP.

***proxy***

un macchina che si interpone fra una rete e *Internet*, il cui ruolo è quello di accelerare i trasferimenti di dati per i protocolli più diffusi (HTTP e FTP, ad esempio). Conserva una cache dei dati ottenuti in seguito alle ultime richieste, il che evita la necessità di dover richiedere il trasferimento di un file presente nella cache se un'altra macchina chiede lo stesso file. I proxy sono molto utili su reti che dispongono di una scarsa ampiezza di banda (ovvero: le connessioni via modem). A volte il proxy è l'unica macchina in grado di accedere all'esterno.

***punto di mount***

è la directory alla quale una partizione o un dispositivo di altro tipo è connesso al filesystem *GNU/Linux*. Il vostro CDROM, ad esempio, è montato sulla directory `/mnt/cdrom`: per esplorare il contenuto di ogni CD che verrà montato dovrete partire da tale directory.

***quota***

è un metodo per limitare l'uso del disco da parte degli utenti. Gli amministratori possono imporre limiti alle dimensioni delle directory home degli utenti impostando dei valori di quota su specifici filesystem.

***RAID***

*Redundant Array of Independent Disks*. Un progetto nato presso il dipartimento di scienze informatiche dell'Università di Berkeley, il cui scopo è distribuire l'archiviazione dei dati su di un insieme di dischi rigidi usando diversi schemi. In un primo momento, questa caratteristica è stata implementata usando dei drive floppy, che è il motivo per cui l'acronimo originario significava *Redundant Array of Inexpensive Disks*.

***RAM***

*Random Access Memory*. Termine utilizzato per indicare la memoria principale di un computer.

***RFC***

*Request For Comments*. Gli RFC sono i documenti ufficiali che descrivono gli standard di *Internet*. Documentano tutti i protocolli, il loro uso, i loro requisiti, e così via. Quando volete sapere come funziona un certo protocollo, procuratevi l'RFC corrispondente.

***root***

è il super-utente di un qualsiasi sistema *Unix*. Tipicamente root (ovvero l'amministratore del sistema) è la persona responsabile per la manutenzione e la supervisione del sistema, inoltre ha anche l'accesso completo a qualunque cosa si trovi sul sistema.

***route***

si riferisce al percorso che esiste tra una macchine e un'altra in una rete.

***RPM***

*Redhat Package Manager*. Un formato sviluppato dalla **Red Hat** per creare pacchetti software, viene usato in molte distribuzioni *GNU/Linux*, inclusa la **Linux-Mandrake**.

***run level***

è una configurazione del software di sistema che permette solo ad alcuni specifici processi di esistere. I processi consentiti sono definiti, per ciascun runlevel, nel file `/etc/inittab`. I runlevel definiti sono otto: 0, 1, 2, 3, 4, 5, 6, S; il passaggio dall'uno all'altro può essere effettuato soltanto da un utente dotato di privilegi eseguendo i comandi`init` e `telinit`.

***schermo pieno***

Questo termine si riferisce ad applicazioni che occupano l'intera area visibile dello schermo con la loro finestra.

***SCSI***

*Small Computers System Interface*. Un bus caratterizzato da un'alta velocità di trasferimento dati, progettato per consentire la connessione di periferiche di vario tipo. A differenza del bus IDE, il bus SCSI non è limitato dalla velocità massima alla quale le periferiche accettano i comandi. Solo le macchine di fascia alta integrano un bus SCSI direttamente sulla scheda madre, i *PC* in genere necessitano di schede aggiuntive.

***script***

gli script della *shell* sono sequenze di comandi che devono essere eseguiti come se fossero stati digitati uno dopo l'altro. Gli script della *shell* sono l'approssimativo equivalente *Unix* dei file batch del *DOS*.

***server***

programma o computer che offre una caratteristica o un servizio e attende la connessione dei **client** per eseguire i loro ordini o dar loro le informazioni che chiedono. Uno dei componenti di un **sistema client/server**.

***shell***

La *shell* è l'interfaccia di base al kernel del sistema operativo, e offre la linea di comando nella quale gli utenti possono digitare comandi per eseguire programmi e software di sistema. Tutte le shell offrono un linguaggio script che può essere usato per rendere automatiche certe operazioni o semplificare operazioni complesse eseguite frequentemente. Gli script della *shell* sono simili ai file batch del sistema operativo *DOS*, ma sono molto più potenti. Tra le shell più comuni citiamo *Bash*, *sh*, e *tcsh*.

***sistema client/server***

sistema o protocollo basato su di un **server** e uno o più **client**.

***sistema operativo***

è l'interfaccia tra le applicazioni e l'hardware. Il compito principale di un qualsiasi sistema operativo è quello di gestire tutte le risorse specifiche della macchina. In un sistema *GNU/Linux* questo compito è svolto dal kernel e dai moduli caricabili. Tra gli altri sistemi operativi più conosciuti citiamo *AmigaOS*, *MacOS*, *FreeBSD*, *OS/2*, *Unix*, *Windows NT*, e *Windows 9x*.

***site dependent***

significa che le informazioni usate da programmi come *Imake* e `make` per compilare dei file sorgenti dipendono dal sito, l'architettura del computer, le librerie installate nel sistema, e così via.

***SMB***

*Server Message Block*. Protocollo usato dalle macchine *Windows* (*9x* o *NT*) per condividere file e stampanti all'interno di una rete. Si veda anche **CIFS**.

***SMTP***

*Simple Mail Transfer Protocol*. Si tratta del più diffuso protocollo per trasmettere messaggi di posta elettronica. Tra i programmi che svolgono questa funzione *sendmail* e *postfix* usano entrambi il protocollo SMTP; a volte sono chiamati "server SMTP".

***socket***

tipo di file corrispondente a una qualsiasi connessione di rete.

***soft link***

si veda "link simbolico".

***standard error***

il descrittore di file numero 2, aperto da ogni processo, convenzionalmente usato per stampare messaggi di errore, e associato normalmente allo schermo del terminale. Si vedano anche **standard input**, **standard output**.

***standard input***

il descrittore di file numero 0, aperto da ogni processo, convenzionalmente usato come il descrittore di file dal quale il processo riceve i dati. Si vedano anche **standard error**, **standard output**.

***standard output***

il descrittore di file numero 1, aperto da ogni processo, convenzionalmente usato come il descrittore di file nel quale il processo stampa il suo output. Si vedano anche **standard error**, **standard input**.

***streamer***

si riferisce a un dispositivo che riceve "flussi" (ingl. *streams*, un invio di dati non interrotti o divisi in frammenti più piccoli) di caratteri come input. Un tipico esempio di streamer è un'unità a nastro magnetico.

***SVGA***

*Super Video Graphics Array*: standard di visualizzazione grafica progettato dalla VESA per l'architettura *PC*. La risoluzione è di 800 x 600 x 16 colori.

***switch***

Gli switch ("interruttori") sono usati per cambiare il comportamento di alcuni programmi, vengono chiamati anche opzioni o argomenti da linea di comando. Per stabilire se un programma ha degli switch opzionali che possono essere usati, leggete le pagine

`man` relative o provate a inviare l'argomento `-help` al programma (cioè `nome_programma -help`).

***target***

lett. ,"bersaglio", è l'oggetto della compilazione, ovvero il file binario che verrà generato dal compilatore.

***TCP***

*Transmission Control Protocol*. Questo è il più diffuso e affidabile protocollo che usa l'IP per trasmettere pacchetti di rete. Il TCP aggiunge i controlli necessari all'IP in modo da avere la certezza che i pacchetti sono stati consegnati. A differenza dell'UDP, il TCP lavora in modalità di connessione, che significa che due macchine devono stabilire una connessione prima che sia possibile uno scambio di dati.

***telnet***

crea una connessione a un host remoto e consente l'accesso a quella macchina, a condizione che si disponga di un account. Telnet è il metodo più usato per gli accessi a distanza, tuttavia ci sono alternative migliori e più sicure, come `ssh`.

***URL***

*Uniform Resource Locator*. Una stringa in un formato speciale usata per identificare una risorsa su *Internet* in maniera univoca. Questa risorsa può essere un file, un server o altro ancora. La sintassi per una URL è `protocol://server.name[:port]/percorso/della/risorsa`. Quando viene inserito soltanto il nome di una macchina e il protocollo è http://, come opzione predefinita viene trasmesso il file `index.html` eventualmente presente sul server.

***valori discreti***

valori che non sono continui. In altre parole, esiste qualche forma di "spaziatura" fra due valori consecutivi.

***variabili***

sono stringhe usate nei file `Makefile` che verranno rimpiazzate dal valore loro assegnato ogni volta che appaiono. In genere si trovano all'inizio del `Makefile`. Vengono usate per semplificare la gestione di `Makefile` e dell'albero dei file sorgenti.

In termini più generali, nella programmazione le variabili sono parole che fanno riferimento ad altre entità (numeri, stringhe, tabelle, etc.) suscettibili di variazione durante l'esecuzione del programma.

***variabili d'ambiente***

una parte dell'ambiente di un processo. Le variabili d'ambiente sono direttamente visibili dalla *`shell`*.

*Si veda anche:* processo.

***VESA***

*`Video Electronics Standards Association`*. Un'associazione che si occupa di standard in relazione all'architetture *`PC`*. Dobbiamo a essa lo standard SVGA, ad esempio.

***visualizzatore a pagine***

programma che visualizza un file di testo una schermata alla volta (ingl. *`pager`*), in modo da facilitare gli spostamenti avanti e indietro nel file, e la ricerca di stringhe. Vi consigliamo di usare `less`.

***WAN***

*`Wide Area Network`*. Questa rete, per quanto simile a una LAN, connette computer che non sono fisicamente collegati agli stessi cavi e sono separati da un distanza più grande.

***window manager***

il programma responsabile per il "*look and feel*" di un ambiente grafico, che si occupa delle barre delle finestre, delle cornici, dei pulsanti, dei menu predefiniti e di alcune scorciatoie da tastiera. Senza di esso, sarebbe molto difficile, se non impossibile, avere dei desktop virtuali, ridimensionare le finestre in tempo reale, spostarle sullo schermo, ...

# Indice


account, 25
ambiente
  di un processo, 33, 148
amministrazione
  delle stampanti, 75
attributi
  dei file, 47, 145
carattere
  di meta-espansione, 51
collegamento, 137
comando
  at, 229
  bzip2, 196, 233
  cat, 38
  cd, 35
  chgrp, 48
  chmod, 49
  chown, 48
  configure, 199
  cp, 46
  crontab, 227
  find, 223
  grep, 221
  gzip, 233
  init, 155
  less, 38
  ls, 39
  make, 204
  mkdir, 43
  mount, 130
  msec, 108
  mv, 45
  patch, 167
  pwd, 35
  rm, 44
  tar, 194, 230
  touch, 43
  umount, 130
completamento automatico, 56
console, 27
directory
  radice, 122, 149
editor
  Emacs, 61
  vi, 67
FHS, 121
file
  in modalità a blocchi, 136, 141
  in modalità a caratteri, 135
framebuffer, 182
GID, 28
GPL, 237
grub, 179
gruppo, 25, 48
inode, 137
lilo, 183
link, 137
  hard, 144
  simbolico, 136, 143
livello
  di sicurezza, 108
Makefile, 193, 206
modalità promiscua, 117
moduli, 152
NFS, 113
password, 26
permessi, 28, 49

dei file, 116
PID, 33
pipe, 55
anonima, 139
con nome, 136, 139
processo, 32, 58, 147
prompt, 27, 34
proprietario, 28, 47
redirezione, 53
runlevel, 157
shell, 34, 43
socket, 136
standard
error, 53
input, 53, 222
output, 53, 222
SUID, 116
UID, 28
umask, 114
Unix, 25
GNU/Linux, 25
utente, 25
root, 28
servizi di stampa, 85
variabile
di ambiente, 36
virus, 33