

Informatica e GNU/Linux

- 2003.12.15 -

DANIELE MASINI

15 dicembre 2003

Titolo originale dell'opera: **Informatica e GNU/Linux**

Versione 2003.12.15 del 15 dicembre 2003.

Numero totale di pagine: 551

Autore: **Daniele Masini**

Copyright © 2003 Daniele Masini.

È permesso copiare, distribuire e/o modificare quest'opera seguendo i termini della Licenza per Documentazione Libera GNU, Versione 1.2 o ogni versione successiva pubblicata dalla Free Software Foundation; con le sezioni non modificabili intitolate "Prefazione", "The GNU General Public License" and "GNU Free Documentation License", con nessun testo di fronte copertina, e con nessun testo di retro copertina. Una copia della licenza è acclusa nella sezione C.1 intitolata "GNU Free Documentation License".

In caso di modifica e/o di riutilizzo parziale della presente opera, secondo i termini della licenza, le annotazioni riferite a queste modifiche ed i riferimenti all'opera originale, devono risultare evidenti e apportati secondo modalità appropriate alle caratteristiche dell'opera stessa. IN NESSUN CASO È COMUNQUE CONSENTITA LA MODIFICA DI QUANTO, IN MODO EVIDENTE, ESPRIME IL PENSIERO O L'OPINIONE DELL'AUTORE.

Quest'opera è stata realizzata nella speranza che possa risultare utile, ma SENZA ALCUNA GARANZIA ESPRESSA O IMPLICITA. Sebbene le informazioni siano state riportate in maniera tale da cercare di assicurarne quanto più possibile la correttezza e l'accuratezza, l'autore non si assume nessuna responsabilità per qualsiasi errore o danneggiamento del sistema che possa derivare in maniera diretta o meno dall'utilizzo delle informazioni in essa contenute. Tutti i nomi di aziende, prodotti ed i relativi marchi, nonché eventuali loghi riportati nella presente opera, appartengono ai legittimi proprietari.

Queste condizioni e questo copyright si applicano all'opera nel suo complesso, salvo ove espressamente indicato in modo diverso.

Copyright © 2003 Daniele Masini.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with the Invariant Sections being entitled "Prefazione", "The GNU General Public License" and "GNU Free Documentation License" and with no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the section C.1 entitled "GNU Free Documentation License".

If you modify this work and/or reuse it partially, under the terms of the license, the notices about these changes and the references about the original work, must be evidenced conforming to the work characteristics. IN NO EVENT IS ALLOWED TO MODIFY WHAT ARE CLEARLY THE THOUGHTS, THE OPINIONS AND/OR THE FEELINGS OF THE AUTHOR.

This work is distributed hoping that it will be useful but WITHOUT ANY EXPLICIT NOR IMPLICIT WARRANTY. Even if information were reported keeping attention to their correctness and accuracy, the author does not assume any responsibility about any error or system damage that could be caused by using the information contained in this work.

All Company names, Product names and related Trademarks and Logos contained in this work, belong to their legitimate owners.

These conditions and this copyright apply to the whole work, except where clearly stated in a different way.

Indice

Prefazione	xiii
Organizzazione del testo	xvi
Convenzioni tipografiche	xviii
L'autore	xx
I Il sistema	1
1 Introduzione	3
1.1 Il computer	3
1.2 La rappresentazione delle informazioni	4
1.2.1 I sistemi di numerazione	4
1.2.2 Bit, byte, nibble, word	12
1.2.3 La logica binaria	12
1.2.4 L'ASCII	13
1.2.5 ISO 8859-X, Unicode e UTF-8	15
1.3 Gli utenti	16
1.4 L'interfaccia utente	16
1.5 L'hardware	17
1.5.1 La motherboard	18
1.5.2 La CPU	18
1.5.3 La memoria	19
1.5.4 Le periferiche	21
1.5.5 I bus	21
1.5.6 I controller	27
1.6 I dischi	28
1.6.1 CSH	28
1.6.2 LBA	29
1.6.3 I dischi ottici	29
1.6.4 Le partizioni	29
1.6.5 Il caching	31
1.6.6 Impostazioni dei dischi	31
1.7 Il filesystem	34
1.8 Il layout della tastiera	35
1.9 Il software	36
1.9.1 Compilatore ed interprete	36
1.9.2 L'input, l'output e l'exit status	37
1.9.3 Le versioni	37
1.10 Il kernel ed i processi	38
1.11 La notifica degli eventi	38
1.12 Le reti di computer	39
1.13 Le distribuzioni di GNU/Linux	39
1.14 L'installazione	40
1.15 I pacchetti	42
1.16 Reperibilità della documentazione	42

1.17	Riferimenti	47
2	Avvio ed arresto del sistema	49
2.1	Il boot	49
2.1.1	Il MBR - Master Boot Record	51
2.1.2	Il boot loader	52
2.1.3	Il boot da dischetto	52
2.1.4	LOADLIN	53
2.1.5	SYSLINUX	54
2.1.6	GRUB	55
2.1.7	LILLO	60
2.2	Parametri di avvio	63
2.3	Avvio del sistema	63
2.4	I runlevel	68
2.5	I daemon	70
2.5.1	inetd e xinetd	75
2.6	Accesso al sistema	83
2.6.1	Accesso da interfaccia carattere	84
2.6.2	Accesso da interfaccia grafica	84
2.7	Arresto del sistema	85
2.8	Riferimenti	86
3	Il filesystem	87
3.1	I dispositivi di memoria di massa	87
3.2	Partizionamento del disco	87
3.3	I file di dispositivo	88
3.4	Inizializzazione del filesystem	89
3.5	La struttura del filesystem	91
3.5.1	La struttura logica	92
3.5.2	La struttura fisica	95
3.6	Il filesystem ext2	96
3.6.1	Struttura di base	97
3.6.2	I file	99
3.6.3	I tipi di file	103
3.6.4	I formati dei file	110
3.6.5	I permessi	113
3.7	Journalized filesystem	119
3.8	Il VFS	122
3.9	mount e umount	125
3.10	“Navigare” nel filesystem	130
3.11	La directory /proc	141
3.12	La directory /dev	143
3.13	La directory /etc/sysconfig	145
3.14	I permessi e l’umask	149
3.15	Limitazioni di accesso al filesystem	150
3.16	Lo swap	150
3.17	Il quota	153
3.18	RAID	158
3.19	LVM	170
3.19.1	Concetti di base	170
3.19.2	Utilizzo	172
3.20	RAM disk	176
3.21	Copia e ripristino di partizioni	176
3.21.1	Partition Image	177
3.22	Comandi utili	180
3.23	Riferimenti	183

4	Operazioni su file e directory	185
4.1	Creazione di file	185
4.2	Creazione di directory	186
4.3	Copia di file o directory	186
4.4	Spostamento o rinominazione di file o directory	188
4.5	Cancellazione di file o directory	189
4.6	Visualizzazione del contenuto di un file	190
4.7	Modifica del contenuto di un file	199
4.8	Ricerca di file o nel contenuto di file	202
4.9	Differenze tra file	203
4.10	Ordinamento	203
4.11	Archiviazione e compressione	203
4.12	Le espressioni regolari	203
4.13	Riferimenti	204
5	Utenti ed accesso al sistema	205
5.1	User account	205
5.2	Group account	210
5.3	La cifratura della password	212
5.4	Il meccanismo delle shadow password	213
5.4.1	Attivare/disattivare il meccanismo delle shadow password	214
5.5	Impersonare un altro utente	215
5.6	Casi particolari	216
5.6.1	System user e system group	216
5.6.2	Gestione particolare delle utenze	217
5.7	La registrazione degli eventi	218
5.7.1	Il system log	218
5.7.2	Altri file di log	221
5.8	La libreria PAM	221
5.8.1	Configurazione	223
5.9	La procedura di login	228
5.9.1	Accesso al terminale	229
5.9.2	login	231
5.10	Comandi utili	233
5.11	Riferimenti	234
6	Il kernel ed i processi	235
6.1	Il kernel	235
6.1.1	Le versioni	236
6.1.2	La compilazione	236
6.1.3	L'avvio	241
6.1.4	Gli argomenti	244
6.2	I moduli	255
6.2.1	Caricamento di moduli in memoria	256
6.2.2	Scarico di moduli dalla memoria	260
6.2.3	Carico/scarico automatico dei moduli	260
6.2.4	Elenco dei moduli caricati in memoria	261
6.2.5	Il file di configurazione	261
6.3	I processi	261
6.3.1	Creazione	266
6.3.2	Gestione della memoria	268
6.3.3	Terminazione	269
6.4	Lo scheduler	270
6.5	La comunicazione tra processi	272
6.5.1	I segnali	272
6.5.2	Le pipe	275
6.5.3	Le FIFO	276

6.5.4	I socket	276
6.5.5	I meccanismi di IPC di System V	276
6.6	I file di lock	277
6.7	Riferimenti	277
7	La shell ed i job	279
7.1	La shell	279
7.2	Le varie shell	282
7.3	<i>Bash</i> - Bourne Again Shell	283
7.3.1	Il prompt	284
7.3.2	Command line editing	284
7.3.3	L'ambiente	285
7.3.4	Name completion	290
7.3.5	Alias	290
7.3.6	Command history	291
7.3.7	Comandi interni ed esterni	291
7.3.8	Gestione dei metacaratteri	293
7.3.9	Scripting language	294
7.3.10	Impostazioni	300
7.3.11	La redirectione dei comandi	302
7.3.12	Job management	303
7.4	Riferimenti	304
8	La stampa	305
8.1	Riferimenti	305
9	Il tempo	307
9.1	L'orologio di sistema	307
9.1.1	Impostazione della data/ora del sistema	308
9.1.2	Il calendario	310
9.2	La schedulazione dei job	311
9.2.1	<i>cron</i>	311
9.2.2	<i>anacron</i>	313
9.2.3	<i>at</i>	314
9.2.4	<i>batch</i>	314
9.2.5	<i>watch</i>	314
9.3	Riferimenti	314
10	Il suono	315
10.1	Concetti di base	315
10.2	La riproduzione del suono	317
10.2.1	La sintesi FM	317
10.2.2	La sintesi con wavetable	317
10.2.3	Il mixing	317
10.2.4	MIDI	317
10.3	Configurazione	318
10.4	Riferimenti	318
11	L'interfaccia grafica	319
11.1	Concetti di base	319
11.1.1	Le immagini	321
11.1.2	La risoluzione dello schermo	322
11.1.3	Le schede video	322
11.2	X Window System	323
11.2.1	XFree86	324
11.3	X Display Manager	340
11.4	Window manager	342

11.4.1	FVWM	342
11.4.2	IceWM	342
11.4.3	Window Maker	342
11.4.4	After Step	342
11.4.5	Enlightment	342
11.4.6	Metacity	342
11.4.7	Sawfish	342
11.5	Desktop environment	342
11.5.1	GNOME	342
11.5.2	KDE	342
11.6	Gestione dei font	343
11.6.1	Fontconfig	343
11.6.2	X Font server	344
11.7	Riferimenti	345
12	Cenni sui database	347
12.1	Introduzione	347
12.2	Tipi di database	347
12.3	Entità e relazioni	348
12.4	I RDBMS	351
12.4.1	Le tabelle	352
12.4.2	Le chiavi e gli indici	353
12.4.3	Le relazioni e l'integrità referenziale	354
12.4.4	La normalizzazione dei dati	355
12.4.5	Il linguaggio SQL	355
12.4.6	Le stored procedure	355
12.4.7	I trigger	355
12.4.8	Le transazioni	355
12.5	La gestione degli utenti	356
12.6	Implementazioni di database	356
12.6.1	PostgreSQL	356
12.6.2	MySQL	356
12.7	Riferimenti	356
13	Applicazioni utili	357
13.1	Gestione del filesystem	357
13.1.1	Midnight Commander	357
13.1.2	Nautilus e Konqueror	357
13.2	Terminale grafico	357
13.3	Calcoli	357
13.4	Scrittura di testi	357
13.5	Grafica	357
13.6	Audio	357
13.7	Office automation	358
13.8	Visualizzazione file PDF	358
13.9	Web browser	358
13.10	Tool per lo sviluppo	358
13.11	Riferimenti	358
14	Installazione del software	359
14.1	I pacchetti	359
14.1.1	Pacchetti tarball	359
14.1.2	Pacchetti deb	363
14.1.3	Pacchetti rpm	364
14.2	Riferimenti	370

15 Sviluppo di applicazioni	371
15.1 Introduzione	371
15.2 I programmi	371
15.2.1 L'interprete	371
15.2.2 Il compilatore	371
15.3 I linguaggi di programmazione	371
15.3.1 C	371
15.3.2 C++	371
15.3.3 Java	371
15.3.4 Perl	371
15.3.5 Python	372
15.3.6 Ruby	372
15.3.7 Tcl/Tk	372
15.4 Gli strumenti	372
15.4.1 Anjuta	372
15.4.2 Quanta	372
15.4.3 Glade	372
15.4.4 KDevelop	372
15.5 Riferimenti	372
 II La rete	 373
16 Concetti di base	375
16.1 Le interfacce	375
16.2 I server ed i client	375
16.3 I protocolli	376
16.4 LAN e WAN	377
16.5 Lo stack OSI	377
16.6 Lo stack TCP/IP	378
16.7 Altri stack di protocolli	379
16.8 Il livello fisico e di collegamento	379
16.8.1 Ethernet	381
16.8.2 Token ring	383
16.8.3 Protocolli Punto-punto	383
16.8.4 MTU	384
16.9 I dispositivi di rete	384
16.9.1 Hub	384
16.9.2 Bridge	385
16.9.3 Switch	385
16.9.4 Router	385
16.9.5 Gateway	385
16.10 Riferimenti	385
 17 TCP/IP - Network	 387
17.1 Network byte order	387
17.2 IP - Internet Protocol (IPv4)	389
17.2.1 Gli indirizzi	389
17.2.2 CIDR	391
17.2.3 Il routing	392
17.2.4 Unicast e multicast	393
17.2.5 L'IP datagram	394
17.3 IPv6	395
17.4 Frammentazione	395
17.5 ARP - Address Resolution Protocol	396
17.6 ICMP - Internet Control Message Protocol	397
17.7 Diagnosi di problemi di connettività	398

17.7.1 ping	398
17.7.2 traceroute	400
17.8 Riferimenti	402
18 TCP/IP - Transport	403
18.1 Porte e socket	403
18.2 UDP - User Datagram Protocol	403
18.2.1 L'UDP datagram	403
18.3 TCP - Transmission Control Protocol	404
18.3.1 Il TCP segment	404
18.3.2 La connessione	406
18.4 Comunicazione tra client e server	408
18.5 netstat e tcpdump	409
18.6 Riferimenti	410
19 TCP/IP - Application	411
19.1 SMTP - Simple Mail Transfer Protocol	411
19.2 POP - Post Office Protocol	411
19.3 HTTP - HyperText Transfer Protocol	411
19.3.1 Apache	411
19.3.2 HTML - HyperText Markup Language	411
19.3.3 Javascript	411
19.3.4 CGI - Common Gateway Interface	411
19.3.5 PHP	411
19.3.6 Tomcat	411
19.3.7 JSP - Java Scripting Page	412
19.4 FTP - File Transfer Protocol	412
19.5 NTP - Network Time Protocol	412
19.6 DHCP - Dynamic Host Control Protocol	412
19.7 SNMP - Simple Network Message Protocol	412
19.8 DNS - Domain Name Server	412
19.9 NFS - Network FileSystem	412
19.10 SAMBA	412
19.11 Telnet	412
19.12 rcp	412
19.13 rlogin	412
19.14 rsh	412
19.15 Riferimenti	412
20 Impostazioni di rete	413
20.1 Le interfacce di rete	413
20.2 Impostazioni di sistema	413
20.3 Impostazioni delle interfacce	415
20.3.1 Interfacce ethernet	416
20.3.2 Interfacce dialup	417
20.4 La routing table	418
20.5 La risoluzione dei nomi	418
20.6 Riferimenti	419
21 Gestione centralizzata degli utenti	421
21.1 NIS	421
21.2 LDAP - Lightweight Directory Access Protocol	421
21.3 Riferimenti	421

22 Cluster	423
22.1 Load balancing	423
22.2 High Availability (HA)	423
22.3 High Performance Computing (HPC)	423
22.4 Riferimenti	423
 III Sicurezza	 425
23 Protezione delle informazioni	427
23.1 Concetti generali	427
23.2 Steganografia	428
23.3 Crittografia	428
23.3.1 Un po' di matematica	430
23.3.2 Cifratura a chiave simmetrica	433
23.3.3 Cifratura a chiave asimmetrica	436
23.3.4 Protocollo Diffie-Hellman	439
23.3.5 Funzioni hash	440
23.3.6 PKI - Public Key Infrastructure	441
23.4 La firma digitale	441
23.4.1 Il MAC - Message Authentication Code	442
23.5 I certificati digitali	443
23.6 Riferimenti	443
 24 Comunicazioni sicure	 445
24.1 SSL/TLS - Secure Sockets Layer / Transport Layer Security	445
24.2 SSH - Secure SHell	445
24.2.1 Autenticazione	451
24.2.2 Login	452
24.2.3 Generazione di chiavi asimmetriche	453
24.2.4 <code>ssh</code>	453
24.2.5 <code>scp</code>	459
24.2.6 <code>sftp</code>	459
24.3 GNU Privacy Guard	460
24.4 Cifratura della posta elettronica	462
24.5 VPN - Virtual Private Network	462
24.6 Kerberos	462
24.7 Riferimenti	463
 25 Protezione del sistema	 465
25.1 Gli attacchi	465
25.2 L'installazione del software	467
25.3 I firewall	467
25.3.1 Packet filtering	469
25.4 Proxy server	485
25.5 IDS - Intrusion Detection System	485
25.6 Riferimenti	485
 IV Appendici	 487
A Il software e le licenze	489
A.1 Le licenze	489
A.2 Il software libero	490
A.3 Garanzie del software	490
A.4 The GNU General Public License	491
A.5 Licenza Pubblica Generica GNU	495

A.6	Riferimenti	500
B	Utilizzo pratico	501
B.1	Dispositivi	501
B.1.1	Configurazione flash pen	501
B.2	Interfaccia grafica	502
B.2.1	Personalizzazione dei cursori del mouse	502
B.2.2	Personalizzazione dei menù di Gnome	505
B.3	Interfaccia carattere	505
B.3.1	Disabilitare lo shutdown con Ctrl-Alt-Del	505
C	Licenza	507
C.1	GNU Free Documentation License	507
C.2	GNU Free Documentation License (italiano)	512

Prefazione

“Prima ti ignorano, poi ti deridono, poi ti combattono. Poi vinci.”

– M. K. Gandhi

Nel 1991, dal progetto di uno studente universitario, nacque il kernel di un sistema operativo Unix-like¹ che il progetto GNU (Gnu’s Not Unix)² attendeva per la realizzazione di un vero e proprio sistema operativo *open source*³. Tale kernel, denominato **Linux** in onore del suo autore Linus Torvalds, unito alle librerie di sistema già sviluppate da GNU, dette origine a **GNU/Linux**, ovvero il sistema operativo Unix-like open source che negli ultimi anni, grazie anche ad *Internet*, si è diffuso in tutto il mondo con una velocità senza eguali.

Linux

GNU/Linux

Si tenga presente che Linux è soltanto uno dei possibili kernel del sistema operativo GNU, ideato dalla *Free Software Foundation*⁴, un organismo fondato nel 1985 da R. Stallman, con lo scopo di creare del software libero, a disposizione di tutti, sorgenti compresi. Esistono anche altri kernel, come Hurd, che permettono di far funzionare tale sistema, anche se per il momento non sono arrivati al livello di sviluppo di Linux.



Figura 1: Il simbolo di progetto GNU ed il pinguino Tux (di L. Ewing), simbolo di Linux.

Linux, grazie al fatto di essere governato dalla licenza GNU GPL (General Public License)⁵, quindi libero da qualunque restrizione legale al suo utilizzo, modifica e diffusione, non è rimasto appannaggio di una sola persona o azienda, ma ha coinvolto nello sviluppo un gran numero di persone in tutto il mondo.

La velocità dello sviluppo di GNU/Linux è impressionante tanto che in pochi anni tale sistema operativo è divenuto adatto anche per utenti inesperti. Infatti, oltre all'interfaccia a caratteri o *shell* (v. fig. 2), GNU/Linux è dotato anche di un'interfaccia grafica⁶ (un esempio della quale è riportato in fig. 3) e di un software di corredo che nulla hanno da invidiare a quelli già presenti in commercio per altri sistemi.

¹un sistema che ha caratteristiche simili al sistema operativo Unix creato dai *Bell Labs*, una divisione della *AT&T*.

²v. <http://www.gnu.org>.

³questo significa che il codice sorgente dei programmi che compongono il sistema operativo è disponibile per chiunque (v. app. A).

⁴v. <http://www.fsf.org>.

⁵v. app. A

⁶in realtà per GNU/Linux sono disponibili più interfacce grafiche (v. cap. 11).

```
[daniele@rhserver /]$ ls -l
total 213
drwxr-xr-x  2 root    root      4096 Nov 14 23:57 bin
drwxr-xr-x  4 root    root      1024 Nov 14 23:28 boot
drwxr-xr-x 20 root    root     118784 Dec 11 22:32 dev
drwxr-xr-x 29 root    root      4096 Dec  9 20:07 documents
drwxr-xr-x 64 root    root      8192 Dec 11 23:04 etc
drwxr-xr-x  3 root    root      4096 Dec 11 21:31 home
drwxr-xr-x  2 root    root      4096 Jun 21  2001 initrd
drwxr-xr-x  7 root    root      4096 Nov 14 23:57 lib
drwx----- 2 root    root     16384 Nov 14 23:18 lost+found
drwxr-xr-x  2 root    root      4096 Aug 27 06:49 misc
drwxr-xr-x  5 root    root      4096 Dec 11 22:32 mnt
drwxr-xr-x  2 root    root      4096 Aug 23  1999 opt
dr-xr-xr-x 88 root    root         0 Dec 11 23:32 proc
drwxr-xr-x  5 root    root      4096 Oct  1 23:23 public
drwxr-xr-x 24 root    root      4096 Dec 11 22:33 root
drwxr-xr-x  2 root    root      8192 Nov 15 00:03 sbin
drwxr-xr-x  3 root    root      4096 Nov 14 23:53 tftpboot
drwxrwxrwt 13 root    root      4096 Dec 11 23:57 tmp
drwxr-xr-x  4 root    root      4096 Oct  1 23:20 TotemPrj
drwxr-xr-x 16 root    root      4096 Nov 14 23:24 usr
drwxr-xr-x 24 root    root      4096 Nov 15 00:00 var
[daniele@rhserver /]$ _
```

Figura 2: Esempio di interfaccia a carattere di GNU/Linux.

Ma ciò che più di ogni altra cosa contraddistingue GNU/Linux è, come per altri sistemi operativi Unix-like, la sua affidabilità. Nessun processo infatti può compromettere il funzionamento del sistema operativo, neanche la stessa interfaccia utente (a meno che non si tratti di un processo che gira in *kernel space*). Inoltre GNU/Linux ha un'altra importante caratteristica: può gestire più terminali per l'accesso alla stessa macchina, ovvero si presenta come un vero sistema operativo multiutente. Questo va a vantaggio anche della manutenibilità del sistema, poiché anche se un processo potesse rendere impossibile l'utilizzo di un terminale, ci sarebbe la possibilità di accedere al sistema attraverso un altro terminale e poter così terminare il processo che tiene bloccato il primo terminale.

GNU/Linux, inoltre, come molti programmi sviluppati per questo sistema, fa parte del *free software*⁷ (software libero) e questo è il suo vero punto di forza. Chiunque può leggere, modificare e ridistribuire il codice sorgente che lo compone. Dunque GNU/Linux è distribuito gratuitamente, lo si può reperire senza dover pagare nessuna azienda poiché è rilasciato sotto una licenza particolare che ne autorizza la libertà di copia e addirittura di modifica.

Oggi un'azienda non può permettersi di creare un prodotto proprietario come il software (chiuso agli utilizzatori), e fare affidamento sul fatto che questo funzioni in qualunque situazione. Un software totalmente bug free (senza falle) è molto difficile da realizzare. Inoltre un bug trovato da un utilizzatore del software diviene di pubblico dominio in tempi brevissimi grazie ad *Internet*, cosicché chiunque può conoscere la falla presente nel software e sfruttarla per comprometterne l'uso o addirittura danneggiare l'intero sistema (questo dipende dal ruolo che riveste il software che contiene il bug all'interno del sistema). E questa possibilità c'è per tutto il tempo che intercorre tra la scoperta del bug ed il rilascio di una *patch* (toppa) o di una nuova versione del software da parte dell'azienda fornitrice del prodotto, che risolve il problema. Con il software libero invece, qualora venga rilevato un bug, chiunque può proporre una soluzione al problema (dato che il codice sorgente è modificabile) e chi lo desidera può reperirla ed utilizzarla nel proprio sistema.

⁷v. app. A.



Figura 3: Esempio di interfaccia grafica di GNU/Linux.

Un'altra caratteristica di GNU/Linux è la sua elevatissima configurabilità, tanto che è possibile utilizzarlo su PC (Personal Computer), palmari, supercomputer e macchine specifiche, utilizzando lo stesso kernel, modificando semplicemente la configurazione del sistema. Nel testo quindi verrà fatto molto uso delle espressioni “in generale”, “generalmente”, ... che indicano appunto il comportamento di base (*default*) del sistema, che, se diversamente configurato, potrebbe essere diverso da quello riportato nel testo. Anche i programmi sviluppati per il sistema GNU/Linux, in modo particolare quelli che appartengono alla categoria del software libero, sono caratterizzati da un'elevata configurabilità, tendendo a portare avanti l'idea che è il software che si deve adattare all'utente e non viceversa (gli utenti possono avere esigenze diverse).

Infine, GNU/Linux, come i sistemi Unix-like, riesce a gestire molto bene reti di computer, tanto che molti dei computer utilizzati per la gestione dei servizi su *Internet* sono basati su questo sistema operativo. Esso utilizza i protocolli definiti da standard pubblici, che danno quindi una maggiore garanzia di sicurezza ed affidabilità rispetto a quelli non pubblici (proprietary). Non solo, il sistema è anche in grado di gestire dei meccanismi di sicurezza molto efficaci, tanto che le macchine sulle quali esso gira, possono anche essere utilizzate come veri e propri firewall.

Per tutti gli aspetti sopra elencati, molte aziende hanno già adottato il sistema GNU/Linux per il proprio parco macchine. Esso infatti è una valida alternativa gratuita ai costosi sistemi operativi proprietari, spesso anche troppo avidi di risorse.

Il presente documento, facendo riferimento al kernel Linux 2.4⁸ per piattaforma *Intel X386*, è una trattazione dei vari aspetti di GNU/Linux che riguardano il sistema nel suo complesso. Per questo, si rivolge ad un pubblico che si avvicina per la prima volta al “pinguino” (soprannome di Linux)⁹ poiché i concetti saranno spiegati cercando di non dare niente per scontato. Pertanto questo testo non ha la pretesa di essere un manuale da cui l'utente inesperto possa imparare tutto su GNU/Linux (“tutto” è una parola grossa...), ma si pone lo scopo di concentrare l'attenzione sui suoi aspetti principali. Il testo si pone come guida essenziale all'utilizzo di GNU/Linux cercando di fornire una

⁸la stesura del testo è iniziata nell'Ottobre 2002.

⁹deriva dal fatto che il simbolo di Linux è Tux, un pinguino, v. fig. 1.

trattazione degli argomenti che possa interessare la maggior parte dei lettori. I vari argomenti, quindi, non saranno trattati in maniera superficiale, ma nemmeno troppo approfondita, anche perché si correrebbe il rischio che le informazioni riportate nel testo potrebbero risultare obsolete, vista la rapidità di sviluppo del sistema; nel testo comunque saranno presenti riferimenti per eventuali approfondimenti, oltre ad incoraggiare il lettore a consultare i manuali forniti (in forma elettronica) col sistema¹⁰. Ottime fonti di spunto per gli argomenti illustrati nel presente testo sono state [1] e [2].

Per sfruttare al massimo il contenuto del presente testo, è consigliabile che il lettore abbia disponibile un PC sul quale sia installato GNU/Linux (in caso di problemi con l'installazione si può far riferimento alla guida relativa alla *distribuzione*¹¹ considerata), in modo tale da sperimentare i concetti qui esposti e prendere pratica fin da subito con questo sistema operativo che poi non è così difficile da utilizzare come si può sentir dire in giro, ma diventa semplice man mano che lo si conosce.

Per la stesura del testo, al momento in lingua italiana, è stato scelto L^AT_EX per l'eccezionale qualità tipografica ottenibile.

Organizzazione del testo

La trattazione dei vari argomenti è distribuita secondo il seguente schema:

parte I

Tratta gli aspetti dei singoli sistemi (sistemi *stand-alone*), senza considerare l'interconnessione con altri sistemi;

capitolo 1

In questo capitolo sono illustrate tutte le nozioni di base necessarie alla comprensione del testo. Viene fornita una panoramica sui sistemi di elaborazione con particolare riferimento ai PC ed a GNU/Linux;

capitolo 2

In questo capitolo viene analizzata la procedura di avvio del sistema, i *runlevel*, il meccanismo di avvio e terminazione dei servizi;

capitolo 3

In questo capitolo viene trattata la gerarchia del filesystem e la sua struttura sia fisica che logica. Vengono illustrati i file, le directory, il *mount* dei dischi, i dispositivi, ...;

capitolo 4

In questo capitolo vengono illustrati rapidamente i comandi che permettono l'utilizzo di base degli oggetti del filesystem come la creazione, visualizzazione, modifica, ricerca e cancellazione dei file e directory;

capitolo 5

In questo capitolo è trattata la gestione degli *account* e delle *password*, la procedura di *login* ed il sistema di autenticazione;

capitolo 6

In questo capitolo viene trattato il *kernel* del sistema, la gestione dei processi nonché i meccanismi di comunicazione tra gli stessi;

capitolo 7

In questo capitolo viene trattata la *shell*, i comandi essenziali ed il linguaggio di scripting di *Bash*, la shell di default di GNU/Linux;

capitolo 8

In questo capitolo viene illustrato il sistema di stampa e di gestione delle stampanti utilizzato da GNU/Linux;

capitolo 9

In questo capitolo viene illustrato come il sistema tiene conto del tempo e come può essere pianificata l'esecuzione dei comandi;

¹⁰per una spiegazione più dettagliata v. cap. 1.

¹¹v. sez. 1.13.

capitolo 10

In questo capitolo sono illustrati gli aspetti del suono e la gestione dell'audio da parte del sistema;

capitolo 11

In questo capitolo è trattato l'avvio dell'interfaccia grafica *X Window*, nonché degli *window manager* più conosciuti e le relative procedure di *login*;

capitolo 12

In questo capitolo vengono introdotti i concetti di base relativi ai database ed alla loro gestione;

capitolo 13

In questo capitolo sono passate in rassegna le applicazioni principali presenti nelle distribuzioni di GNU/Linux (dotate di interfaccia grafica), che permettono la navigazione del filesystem, la navigazione internet, l'automazione per l'ufficio, ...;

capitolo 14

In questo capitolo vengono illustrate le operazioni necessarie all'installazione di applicazioni con particolare riferimento alla gestione dei pacchetti;

capitolo 15

In questo capitolo vengono illustrati i concetti di base ed i programmi per lo sviluppo di applicazioni;

parte II

Tratta gli aspetti relativi all'interconnessione di più sistemi ed alla loro comunicazione;

capitolo 16

In questo capitolo vengono introdotti i concetti di base delle reti di computer e della comunicazione tra i sistemi;

capitolo 17

In questo capitolo vengono trattati i meccanismi di comunicazione a livello di rete, come i protocolli IP, ARP e ICMP;

capitolo 18

In questo capitolo vengono trattati i meccanismi di comunicazione a livello di trasporto, come i protocolli TCP e UDP;

capitolo 19

In questo capitolo vengono trattati i meccanismi di comunicazione a livello di applicazione, come i protocolli HTTP, FTP, SMTP, ... ed i servizi come DNS, NFS, ...;

capitolo 20

In questo capitolo vengono illustrate le impostazioni per l'uso della rete con un sistema GNU/Linux;

capitolo 21

In questo capitolo vengono trattati i meccanismi di gestione centralizzata degli utenti in una rete di computer, come NIS e LDAP;

capitolo 22

In questo capitolo vengono trattati i meccanismi di gestione dei *cluster*, ovvero particolari insiemi di sistemi interconnessi tra loro in rete;

parte III

Tratta gli aspetti legati alla sicurezza del sistema e delle informazioni;

capitolo 23

In questo capitolo vengono presi in esame i meccanismi che riguardano la protezione delle informazioni: i sistemi di cifratura e la firma digitale;

capitolo 24

In questo capitolo vengono esaminati i sistemi per la trasmissione sicura delle informazioni su canali insicuri;

capitolo 25

In questo capitolo vengono illustrati i meccanismi di protezione del sistema da eventuali attacchi: tecniche di intrusion detection, firewalling, masquerading e proxy server;

parte IV

contiene appendici relative ad argomenti di contorno o che non hanno trovato una posizione specifica all'interno del testo.

Convenzioni tipografiche

grassetto

in questo modo saranno evidenziati i nuovi termini man mano che verranno introdotti nel testo. Un esempio è il seguente:

... dette origine a **GNU/Linux**, ovvero il sistema operativo ...

inclinato

in questo modo saranno evidenziate le espressioni su cui si vuol momentaneamente porre l'attenzione ed i termini particolari. Un esempio è il seguente:

... utilizzata per il *mounting* delle periferiche ...

italic

in questo modo saranno evidenziate le espressioni su cui si vuol momentaneamente porre l'attenzione ed i termini particolari. Un esempio è il seguente:

... motore di ricerca come ad esempio *Google*, ...

monospaced

in questo modo sarà evidenziato tutto l'input/output del sistema (ciò che è digitato con la tastiera e che è visualizzato sullo schermo dal sistema), i nomi ed il contenuto dei file, gli indirizzi di pagine web e quelli di posta elettronica. Un esempio è il seguente:

```
$ man 8 mount
```

monospaced inclinato

in questo modo saranno evidenziati i nomi dei processi. Un esempio è il seguente:

... che viene avviato dal kernel è *init* tramite l'esecuzione ...

meta

in questo modo saranno evidenziate le parti del metalinguaggio per l'illustrazione della sintassi dei comandi e dei dati. Le parti opzionali saranno racchiuse tra parentesi quadre '[' e ']'. Un esempio è il seguente:

```
# useradd [option] username
```

In tal caso *option* sarà un argomento opzionale, ovvero l'utente può impartire il comando `useradd` senza specificare *option*, ma *username* deve essere sempre specificato.

Gli elementi che fanno parte di elenchi di scelta vengono racchiusi tra parentesi graffe '{' e '}', separati tra loro dal carattere *pipe* '|'. Ad esempio

```
# httpd {start|restart|stop|...}
```

In tal caso gli elementi **start**, **restart**, **stop** (ecc.) fanno parte di un elenco di parametri tra cui l'utente può scegliere per impartire il comando **httpd**. In genere tali parametri sono da ritenersi mutuamente esclusivi, ovvero soltanto uno di essi può essere specificato.

key

in questo modo saranno evidenziati i tasti da premere. Un esempio è il seguente:

... della macchina in genere premendo il tasto Canc (o Del).

In questo modo saranno rappresentati i tasti che non hanno un corrispondente carattere visualizzato sullo schermo o tali che la funzione che essi richiamano all'interno di un programma viene effettuata subito dopo la pressione del tasto, cioè non è necessario premere nessun ulteriore tasto di conferma (tipicamente il tasto Return).

Nel testo saranno anche evidenziati dei paragrafi nel modo seguente:

In questo modo saranno evidenziati i paragrafi che si riferiscono a dettagli implementativi, note storiche o commenti vari.

Un'altra convenzione utilizzata è quella relativa alla rappresentazione del simbolo della *prompt* della shell di sistema: essa denota immediatamente i privilegi che si devono avere per effettuare determinate operazioni. Un cancelletto (**#**), tipico prompt dell'amministratore del sistema nei sistemi Unix-like, indica che l'utente che digita i comandi deve avere privilegi amministrativi, ovvero quelli dell'utente detto *superuser*; un simbolo di dollaro (**\$**), tipico prompt di tutti gli utenti non amministratori, indica che non è necessario che l'utente goda dei diritti amministrativi sul sistema.

Si dà per scontato, salvo specifico avviso, che all'interno dei file di configurazione e degli *script* le linee vuote e quelle che iniziano con un cancelletto (**#**) sono ignorate.

I simboli matematici utilizzati nel testo sono riportati in tab. 1.

Simbolo	Significato
\mathbb{N}	insieme dei numeri naturali
\mathbb{Z}	insieme dei numeri interi
\mathbb{Q}	insieme dei numeri razionali
\mathbb{R}	insieme dei numeri reali
∞	infinito
$\{\dots\}$	insieme (elenco di valori)
$[\dots]$	insieme chiuso (intervallo di valori, estremi inclusi)
(\dots)	insieme aperto (intervallo di valori, estremi esclusi)
\in	elemento appartenente ad un insieme
\subset	insieme contenuto in un insieme
\forall	per ogni
\exists	esiste
$:$	tale che
\cong	congruenza
$ a $	valore assoluto di a . Esso vale $\begin{cases} a & \text{se } a > 0 \\ -a & \text{se } a < 0 \end{cases}$
$\sum_{i=0}^n a_i$	sommatoria (equivale a $a_0 + a_1 + a_2 + \dots + a_n$)
$\prod_{i=0}^n a_i$	produttoria (equivale a $a_0 \times a_1 \times a_2 \times \dots \times a_n$)

Tabella 1: Simboli matematici utilizzati nel testo.

Infine è opportuno notare che, sebbene nella pratica vengano spesso utilizzati impropriamente i prefissi legati ai multipli del sistema metrico decimale (riportati in tab. 2), nel testo viene fatto uso dei prefissi relativi ai multipli del sistema binario come indicato dal **SI** (Sistema Internazionale)¹² e riportato in tab. 3. Si avranno pertanto i kibibyte (1 KiB = 1.024 byte), i mibibyte (1 MiB = 1.048.576 byte), i gibibyte (1 GiB =

¹²v. <http://www.themeter.net>.

Prefisso	Simbolo	Coeff. moltiplicativo
kilo-	k	$1.000 = 10^3$
mega-	M	$1.000.000 = 10^6$
giga-	G	$1.000.000.000 = 10^9$
tera-	T	$1.000.000.000.000 = 10^{12}$
peta-	P	$1.000.000.000.000.000 = 10^{15}$
exa-	E	$1.000.000.000.000.000.000 = 10^{18}$
zetta-	Z	$1.000.000.000.000.000.000.000 = 10^{21}$
yotta-	Y	$1.000.000.000.000.000.000.000.000 = 10^{24}$

Tabella 2: Multipli del sistema metrico decimale secondo il SI.

Prefisso	Simbolo	Coeff. moltiplicativo
kibi-	Ki	$1.024 = 2^{10}$
mebi-	Mi	$1.048.576 = 2^{20}$
gibi-	Gi	$1.073.741.824 = 2^{30}$
tebi-	Ti	$1.099.511.627.776 = 2^{40}$
pebi-	Pi	$1.125.899.906.842.624 = 2^{50}$
exbi-	Ei	$1.152.921.504.606.846.976 = 2^{60}$

Tabella 3: Multipli di quantità binarie secondo il SI.

1.073.741.824 byte) e così via, al posto dei kilobyte (1 kB = 1.000 byte), megabyte (1 MB = 1.000.000 byte), gigabyte (1 GB = 1.000.000.000 byte), ...

L'autore

Nato a Firenze nel 1970 e laureato con lode in Ingegneria elettronica nel 1996 presso la facoltà di Ingegneria dell'Università degli Studi di Firenze, ha manifestato fin dall'adolescenza un vivido interesse per l'elettronica ed in particolare per i computer. Ha conosciuto GNU/Linux nel 1995 e ci si è dedicato più approfonditamente a partire dal 1998. Attualmente lavora come sistemista e team leader per lo sviluppo software presso una nota azienda in provincia di Firenze.

Firenze, Italia
Dicembre 2003

Daniele Masini
d.masini@tiscali.it

Parte I

Il sistema

Capitolo 1

Introduzione

“So di non sapere.”
– *Socrate*

In questo capitolo saranno fornite le nozioni necessarie alla comprensione degli argomenti trattati nei capitoli successivi. Dopo una rapida panoramica su come vengono rappresentate le informazioni all'interno di un elaboratore elettronico, si danno le definizioni di base per il software e l'hardware. Quindi si forniscono i termini relativi al filesystem, al kernel, ai processi ed agli utenti. Infine vengono riportate le distribuzioni di GNU/Linux più conosciute con le loro rispettive caratteristiche e le modalità di reperimento della documentazione sul sistema.

1.1 Il computer

Un **computer** o *elaboratore elettronico* è una macchina che esegue rapidamente istruzioni opportunamente codificate: i *programmi*. *computer*



Figura 1.1: Un computer.

L'utilizzo del computer è diventato ormai di uso quotidiano in tutti gli ambiti delle varie attività, dalla scuola, all'ufficio, all'uso domestico e questo per il semplice fatto che l'elaboratore permette di svolgere compiti, per i quali è stato opportunamente “istruito” per mezzo dei *programmi* o *applicazioni* (v. sez. 1.9), in tempi piuttosto brevi. Il termine “piuttosto” dipende dalla velocità dei circuiti elettronici che costituiscono il computer e dalla qualità dei programmi.

È quindi sempre più sentita l'esigenza di avere programmi che vadano il più possibile incontro a chi li usa (gli utenti), ovvero un buon programma non è soltanto quello che svolge correttamente il proprio compito, ma che lo fa anche più velocemente possibile ed il suo funzionamento avviene attraverso un'interfaccia quanto più possibile *user friendly* (amichevole per l'utente) o intuitiva.

1.2 La rappresentazione delle informazioni

Un elaboratore, come tutti i dispositivi elettronici digitali¹, è in grado di lavorare con una logica *binaria*, cioè basata su due soli stati logici: “0” e “1” (“falso” e “vero”, “spento” e “acceso”). In questo modo si riduce al minimo la possibilità di errore dovuto al rumore di tipo elettromagnetico. Infatti, si supponga di avere un intervallo si devono definire i livelli da associare ai vari stati logici. È evidente che per avere la massima distanza tra i livelli, è opportuno definirne il minimo numero possibile (e, di conseguenza, avere il minimo numero di stati logici), in modo tale che i livelli siano alle due estremità dell’intervallo. In pratica si ha a disposizione un intervallo di tensione ad esempio da 0 e 5 V (Volt); quindi i livelli di tensione associati ai due stati logici vengono fissati l’uno a 0 V, e l’altro a 5 V (v. fig. 1.2). Inoltre, si definiscono dei sottointervalli, intorno ai livelli associati agli stati logici, all’interno dei quali il valore di tensione è sempre riconosciuto come quello associato al livello in questione: ad esempio, il sottointervallo che si riferisce allo stato “0” è compreso tra 0 e 2 V, mentre il sottointervallo che si riferisce allo stato “1” è compreso tra 3 e 5 V; in questo modo, il valore che si riferisce allo stato “1”, trasmesso come 5 V dal circuito che lo emette, viene riconosciuto come tale dal circuito destinatario, può subire un eventuale degrado, dovuto al rumore, pari a 2 V. Un discorso analogo vale per lo stato logico “0”. La zona tra 2 e 3 V non è attribuita a nessuno stato logico, come muro di confine tra i due stati riconoscibili: un segnale il cui valore cade al suo interno non è attribuibile a nessuno stato, potrebbe derivare sia da un segnale di partenza di 0 V che da uno di 5 V. In questo modo si ha un’elevata insensibilità al rumore all’interno di un intervallo di tensione di 5 V.

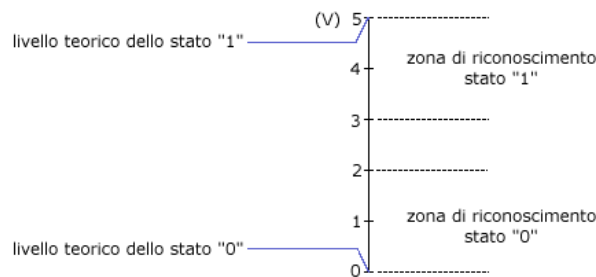


Figura 1.2: Esempio di suddivisione di un intervallo in livelli.

Qualunque informazione gestita da un computer, di qualunque tipo essa sia (numerico, alfabetico, logico, ...) viene rappresentata utilizzando due sole cifre numeriche, “0” e “1”, ovvero in forma binaria.

1.2.1 I sistemi di numerazione

sistema di numerazione
cifre

Un **sistema di numerazione** è un modo per rappresentare dei valori numerici. Un numero si compone di **cifre**, cioè di simboli ad ognuno dei quali viene associato univocamente un valore diverso dagli altri.

I sistemi di numerazione moderni sono sistemi di numerazione *posizionali*, ovvero una stessa cifra ha un “peso” diverso dipendentemente dalla posizione che essa occupa all’interno del numero in questione. Per esempio nel numero 44, la cifra 4 che si trova a sinistra vale 10 volte di più di quella che le sta immediatamente a destra.

base

Da ciò nasce il concetto di **base**: la base indica il numero di cifre a disposizione del sistema di numerazione e, di conseguenza, quante volte di più vale la stessa cifra man mano che questa occupa una posizione sempre più a sinistra all’interno di un numero. Il sistema di numerazione decimale, o a base 10, con il quale si ha a che fare quotidianamente, ha dieci cifre diverse (da “0” a “9”) ed ogni cifra assume un significato 10 volte maggiore per ogni posizione che essa occupa spostandosi verso sinistra

¹dall’inglese *digit* (cifra).

all'interno di un numero. Pertanto si può pensare che ad ogni posizione all'interno di un numero sia associato un peso che non è altro che una *potenza* della base del sistema di numerazione considerato.

Richiami di matematica

Nel corso della presente sezione si farà riferimento all'operazione di *elevamento a potenza*. Tale operazione identifica due entità, l'una detta **base** della potenza e l'altra detta **esponente**. La base è il valore che viene elevato alla potenza indicata dall'esponente. Indicando con b la base e con n l'esponente, la sintassi utilizzata per rappresentare l'operazione di elevamento alla potenza n -esima della base b è la seguente

$$b^n$$

e si legge “ b elevato alla n -esima potenza” o semplicemente “ b alla n ”. A tale scrittura viene associato il seguente significato

$$b^n = \prod_{i=1}^n b = \underbrace{b \times b \times \cdots \times b}_n$$

ovvero b moltiplicato per sé stesso n volte. In particolare si definisce

$$b^0 = 1$$

ovvero, qualunque base elevata alla potenza 0 assume il valore 1.

Inoltre si definiscono le potenze negative come le potenze del reciproco della base, ovvero

$$b^{-n} = \frac{1}{b^n}$$

Alcuni esempi di elevamento a potenza sono i seguenti:

$$\begin{aligned} 3^2 &= 3 \times 3 = 9 \\ 5^3 &= 5 \times 5 \times 5 = 125 \\ 2^3 &= 2 \times 2 \times 2 = 8 \\ 4^2 &= 4 \times 4 = 16 \\ 1^5 &= 1 \times 1 \times 1 \times 1 \times 1 = 1 \\ 14^1 &= 14 \\ 27^1 &= 27 \\ 7^0 &= 1 \\ 32^0 &= 1 \\ 2^{-3} &= \frac{1}{2^3} = \frac{1}{8} = 0,125 \\ 10^{-2} &= \frac{1}{10^2} = \frac{1}{100} = 0,01 \end{aligned}$$

Per facilitare i calcoli illustrati nelle pagine seguenti, in tab. 1.1 sono riportate alcune potenze delle basi 2 e 10.

I numeri naturali

I numeri naturali sono i numeri interi senza segno, come 0, 1, 2, 3, ... la cui rappresentazione, nel sistema di numerazione decimale, può essere espressa come

$$\sum_{i=0}^{n-1} d_i \times 10^i = d_{n-1} \times 10^{n-1} + \dots + d_2 \times 10^2 + d_1 \times 10^1 + d_0 \times 10^0$$

dove n è il numero di cifre di cui si compone il valore decimale, d_i sono le cifre che lo costituiscono e 10^i i pesi ad esse associati. Poiché i pesi hanno **base** 10, si parla di sistema di numerazione **decimale**. Dunque, ad esempio, nel sistema di numerazione decimale, il numero 7345, costituito da 4 cifre, è “pesato” nel modo seguente:

Potenza	Valore	Potenza	Valore
2^{-5}	0,03125	10^{-5}	0,00001
2^{-4}	0,0625	10^{-4}	0,0001
2^{-3}	0,125	10^{-3}	0,001
2^{-2}	0,25	10^{-2}	0,01
2^{-1}	0,5	10^{-1}	0,1
2^0	1	10^0	1
2^1	2	10^1	10
2^2	4	10^2	100
2^3	8	10^3	1.000
2^4	16	10^4	10.000
2^5	32	10^5	100.000
2^6	64	10^6	1.000.000
2^7	128	10^7	10.000.000
2^8	256	10^8	100.000.000
2^9	512	10^9	1.000.000.000
2^{10}	1.024	10^{10}	10.000.000.000
2^{11}	2.048	10^{11}	100.000.000.000
2^{12}	4.096	10^{12}	1.000.000.000.000

Tabella 1.1: Alcune potenze del 2 e del 10.

pesi	10^3	10^2	10^1	10^0
cifre	7	3	4	5

Questo significa che la cifra 7 ha un peso 10 volte superiore a quello della cifra 3, che a sua volta ha un peso dieci volte superiore rispetto a quello della cifra 4, la quale ha un peso dieci volte superiore rispetto a quello della cifra 5. Quindi si può scrivere:

$$7 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 = 7345$$

In questo modo, si possono definire altri sistemi di numerazione, semplicemente cambiando la *base* di riferimento. Scegliendo come base il valore più piccolo possibile, cioè 2, si ottiene il sistema di numerazione **binario**, con base 8 si ha il sistema di numerazione *ottale*, con base 16 si ottiene il sistema di numerazione *esadecimale*, ... Poiché, come accennato in precedenza, in un elaboratore le informazioni si basano soltanto su due valori possibili (due cifre: “0” e “1”), la rappresentazione di questi avviene per mezzo del sistema di numerazione binario. La posizione occupata da una cifra all’interno di un numero binario è dunque pesata, anziché da una potenza di 10 (come avviene nel sistema di numerazione decimale), da una potenza di 2, cioè un numero binario può essere espresso come

binario

$$\sum_{i=0}^{n-1} b_i \times 2^i = b_{n-1} \times 2^{n-1} + \dots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 \times 2^0$$

dove n è il numero di cifre di cui si compone il valore binario, b_i sono le cifre che lo costituiscono e 2^i i pesi ad esse associati. Pertanto il valore binario 1101, costituito da 4 cifre, corrisponde al valore decimale 13, infatti

pesi	2^3	2^2	2^1	2^0
cifre	1	1	0	1

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13$$

Quindi si può scrivere $1101_2 = 13_{10}$.²

Dunque, un numero binario di n cifre, può rappresentare valori interi senza segno (numeri naturali) da 0 a $2^n - 1$.

Il sistema di numerazione binario è piuttosto scomodo in quanto, disponendo soltanto dei simboli “0” e “1”, necessita di molte cifre per rappresentare un valore. Per questo vengono utilizzate altre rappresentazioni numeriche che raggruppano più cifre binarie

esadecimale

in un'unica cifra. Ne sono un esempio i numeri **esadecimale** (sistema di numerazione a base 16) che raggruppano 4 cifre binarie in ogni cifra esadecimale. Tale sistema di numerazione si basa sull'utilizzo di 16 simboli: "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F". I simboli alfabetici da "A" ad "F" rappresentano i valori decimali che vanno da 10 a 15.³

Dunque il valore binario 10011101_2 può essere rappresentato come valore esadecimale $9D_{16}$, infatti le prime quattro cifre binarie corrispondono al valore esadecimale "9", $1001_2 = 9_{10} = 9_{16}$, e il secondo gruppo composto dalle seconde ed ultime 4 cifre binarie corrisponde al valore esadecimale "D", $1101_2 = 13_{10} = D_{16}$. Ragionando come in precedenza, si può avere riprova di quanto affermato:

pesi	16^1	16^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
cifre	9	D	1	0	0	1	1	1	0	1

$$9 \times 16^1 + 13 \times 16^0 = 157$$

$$1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 157$$

In genere i numeri esadecimali sono rappresentati anche posponendo il pedice "H" al numero considerato. Pertanto le scritture $9D_{16}$ e $9D_H$ sono equivalenti.

Per convertire un numero decimale nel corrispondente valore binario si può utilizzare il metodo delle divisioni successive. Un valore infatti, se diviso per la base del sistema di numerazione dà come resto il valore della cifra meno significativa del numero. Quindi per trovare il valore binario corrispondente al valore decimale 25_{10} si può operare come segue

Operazione	Risultato	Resto
$25/2$	12	1
$12/2$	6	0
$6/2$	3	0
$3/2$	1	1
$1/2$	0	1

Il meccanismo delle divisioni, può essere arrestato quando il quoziente ottenuto è 0. Quindi, considerando le cifre ottenute dai resti, lette nell'ordine opposto a quello nel quale sono state ottenute, si può scrivere

$$25_{10} = 11001_2$$

I numeri interi negativi

Per quanto riguarda i numeri interi negativi, si introduce l'operazione di *complemento*. Il complemento è l'operazione che fornisce il valore complementare del numero considerato, rispetto ad un numero con le stesse cifre di quello in oggetto poste ad un valore prefissato. Ad esempio, nel sistema di numerazione decimale, il complemento a 9 del numero 324 su 4 cifre (0324) è 9675, infatti il complementare di 0324 rispetto a 9999 è 9675 ($9999 - 0324 = 9675$).

Valore decimale	Complemento a 9	Valore di riferimento
0324	9675	9999

Si tratta praticamente di un'operazione di sottrazione. Nel sistema di numerazione binario si parla in maniera equivalente di complemento ad 1. Ad esempio, il complemento ad 1 del numero 00010010_2 è 11101101_2 , che si può ottenere sostituendo tutte le cifre "0" con la cifra "1" e viceversa (questo lo si può ottenere per mezzo dell'operazione di negazione logica, v. sez. 1.2.3).

²per chiarezza i numeri verranno rappresentati con un pedice che indica la base del sistema di numerazione considerato.

³le lettere utilizzate nelle cifre esadecimali, possono essere indifferentemente sia maiuscole che minuscole.

Valore binario	Complemento a 1	Valore di riferimento
00010010	11101101	11111111

Si può parlare anche di complemento a b , dove b è la base del sistema di numerazione considerato, ovvero per il sistema decimale è il complemento a 10, mentre per il sistema di numerazione binario è il complemento a 2. In tal senso si ricerca il complementare di un numero rispetto ad un valore composto dallo stesso numero di cifre poste a $b - 1$, a cui si aggiunge 1. Ad esempio, il complemento a 10 del numero decimale 0072 (su 4 cifre) è 9938 poiché $(9999 + 1) - 0072 = 10000 - 0072 = 9938$. Il complemento a b si può ottenere calcolando il complemento a $b - 1$ e quindi aggiungendo 1 al risultato. Ad esempio, il complemento a 2 del numero 00010010_2 è 11101110_2 , ovvero $11101101_2 + 1$.

Valore decimale	Complemento a 10	Valore di riferimento
0072	9938	10000

Valore binario	Complemento a 2	Valore di riferimento
00010010	11101110	100000000

Dal punto di vista matematico, la sottrazione tra due numeri A e B si può ottenere con l'operazione di somma tra A ed il complemento a b di B , scartando l'eventuale cifra in più rispetto a quelle tra le quali viene effettuato il calcolo. Ad esempio, si supponga di voler calcolare $43 - 16$, senza utilizzare l'operazione di sottrazione. Il complemento a 10 di 16 (ad esempio su 3 cifre) è 974, quindi, applicando quanto affermato precedentemente si ha $043 + 974 = 1017$. Il risultato occupa più di 3 cifre, quindi le cifre in più (quelle più significative) vengono scartate, ottenendo 017, cioè 17, proprio il risultato della sottrazione. Analogamente in binario (con 8 cifre) $00101011_2 - 00010010_2$ si riduce a $00100011_2 + 11101110_2 = 100010001_2$, quindi, scartando le cifre più significative in eccesso, si ha 00010001_2 , cioè 10001_2 . Questo procedimento, che può sembrare complicato, permette di dimenticarsi completamente dell'operazione di sottrazione nel sistema di numerazione binario, infatti il complemento a 1 si ottiene negando tutte le cifre e il complemento a due aggiungendo 1 al complemento a 1.

Dunque, un numero negativo $-n$ può essere praticamente rappresentato dal complemento a b del numero n . Quindi, nel sistema di numerazione binario, i numeri negativi sono rappresentati con il complemento a 2 del corrispondente numero senza segno. Ad esempio, il valore -15_{10} viene rappresentato come il complemento a 2 di 15_{10} , ovvero, considerando 8 cifre, $15_{10} = 00001111_2$, quindi $-15_{10} = 11110001_2$. In particolare, il numero -1 è rappresentato su 8 cifre come 11111111_2 ovvero è il complemento a 2 di 00000001_2 .

Da quanto illustrato, segue che un numero binario di n cifre può rappresentare, in complemento a 2, valori interi con segno, cioè sia positivi che negativi, che vanno da -2^{n-1} a $2^{n-1} - 1$.

I valori negativi, ovvero rappresentati in complemento a 2, sono riconoscibili dal fatto che la prima cifra binaria è posta a 1.

Il tentativo di memorizzare su n cifre, su un computer, un valore numerico maggiore di $2^{n-1} - 1$ o minore di -2^{n-1} genera un errore di *overflow* (il numero richiederebbe un numero di cifre binarie maggiore per poter essere rappresentato).

I numeri con parte non intera

I numeri che contengono una parte non intera sono rappresentati da due gruppi di cifre, separati da una virgola: il gruppo di cifre alla sinistra della virgola rappresenta la parte intera del numero, mentre quella a destra è la parte non intera. Ad esempio, il numero 352,74 è composto dalla parte intera 352 e dalla parte non intera 0,74 ed i pesi associati alle varie cifre sono i seguenti

pesi	10^2	10^1	10^0		10^{-1}	10^{-2}
cifre	3	5	2	,	7	4

La rappresentazione di un valore secondo il sistema di numerazione decimale è la seguente

$$\sum_{i=-m}^{n-1} d_i \times 10^i = d_{n-1} \times 10^{n-1} + \dots + d_0 \times 10^0 + d_{-1} \times 10^{-1} + d_{-2} \times 10^{-2} + \dots + d_{-m} \times 10^{-m}$$

mentre nel sistema di numerazione binario è

$$\sum_{i=-m}^{n-1} b_i \times 2^i = b_{n-1} \times 2^{n-1} + \dots + b_0 \times 2^0 + b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \dots + b_{-m} \times 2^{-m}$$

dove n è il numero di cifre di cui si compone la parte intera, m è il numero di cifre di cui si compone la parte non intera, d_i sono le cifre che compongono il numero decimale e b_i quelle che compongono il numero binario, con i relativi pesi (10^i e 2^i).

La parte non intera del numero è costituita dalle cifre che seguono la virgola, ovvero dalle cifre con pedice negativo, d_{-1} , d_{-2} , ... per i valori decimali e b_{-1} , b_{-2} , ... per quelli binari.

Per convertire la parte non intera di un numero decimale nel corrispondente valore binario, si può procedere con moltiplicazioni successive. Infatti, moltiplicando la parte non intera per la base del sistema di numerazione, si ottiene, come parte intera, la cifra più significativa della parte non intera, cioè, con riferimento al simbolismo utilizzato precedentemente, d_{-1} o b_{-1} . Ad esempio, il valore decimale 0,853 moltiplicato per la base (10) dà come risultato 8,53 ovvero 8 è la cifra più significativa della parte non intera. Moltiplicando la nuova parte non intera (0,53) nuovamente per la base si ottiene 5,3 che fornisce come parte intera la seconda cifra non intera 5. Procedendo ulteriormente con le moltiplicazioni si ottengono, una dopo l'altra, tutte le cifre della parte non intera. Applicando lo stesso ragionamento con un'altra base, ovvero un altro sistema di numerazione, si ottengono via via tutte le cifre relative alla parte non intera del valore. Quindi per trovare il valore binario corrispondente al valore decimale 18,3125 si può operare come segue: si considera innanzi tutto la parte intera, 18, a cui si applica il procedimento descritto in precedenza ottenendo $18_{10} = 10010_2$. Poi si considera la sola parte non intera ($18,3125 - 18 = 0,3125$) e si applicano le moltiplicazioni successive appena descritte

Operazione	Risultato	Parte intera	Parte non intera
$0,3125 \times 2$	0,625	0	0,625
$0,625 \times 2$	1,25	1	0,25
$0,25 \times 2$	0,5	0	0,5
$0,5 \times 2$	1	1	0

Il meccanismo delle divisioni, può essere arrestato quando la parte non intera si è ridotta a 0. Quindi, considerando le cifre ottenute come parti intere dei risultati parziali, prese nell'ordine nel quale sono state ottenute con le operazioni, si può scrivere

$$0,3125_{10} = 0,0101_2$$

per cui

$$18,3125_{10} = 10010,0101_2$$

Infatti

pesi	2^4	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}
cifre	1	0	0	1	0	,	0	1	0	1

$$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 18,3125$$

Si noti che non tutti i numeri reali \mathbb{R} sono rappresentabili all'interno di un computer. Ad esempio i numeri irrazionali, come $\pi \simeq 3,1415\dots$, oppure i periodici (composti da un numero infinito di cifre dopo la virgola), come $1/3 \simeq 0,33333\dots$, non sono ovviamente memorizzabili avendo a disposizione un numero finito di celle di memoria (tali numeri possono essere approssimati, troncandoli a partire da una determinata cifra dopo la virgola). Inoltre, poiché il numero di cifre dopo la virgola varia in funzione del sistema di numerazione utilizzato (ad esempio $0,2_{10} = 0,00110011\dots_2$), anche numeri decimali con un numero finito di cifre dopo la virgola possono non essere rappresentabili esattamente nel sistema di numerazione binario.

Un numero con la virgola (tranne il valore 0), può essere sempre rappresentato da un numero con parte intera non nulla ma inferiore alla base, moltiplicato per una potenza della base (ad esempio $346,57 = 3,4657 \times 10^2$), che si ottiene effettuando sullo stesso un'operazione di *normalizzazione* (moltiplicazione del valore per un'opportuna potenza della base). La parte prima del simbolo di moltiplicazione, è detta **mantissa** o *fraction* (nell'esempio 3,4657) ed il valore che appare in alto a destra della base è detto **esponente** (nell'esempio 2). I numeri con la virgola vengono rappresentati all'interno del computer mediante la coppia mantissa ed esponente. Poiché la virgola, durante l'operazione di normalizzazione e denormalizzazione⁴ subisce degli spostamenti verso sinistra o verso destra rispetto alle varie cifre che compongono il numero, tale tipo di rappresentazione viene detta anche *rappresentazione in virgola mobile* (*floating point*).

Lo standard IEEE 754 è quello utilizzato per la rappresentazione dei valori numerici in virgola mobile nei computer. Tale standard prevede due tipi di rappresentazioni: **single precision** (precisione singola) e **double precision** (precisione doppia), che si differenziano dal numero di bit⁵ necessari per la rappresentazione dei valori stessi: entrambi utilizzano la rappresentazione dei valori della mantissa e dell'esponente come se fossero dei numeri naturali, ed un bit per la rappresentazione del segno. Vengono inoltre utilizzate le seguenti regole: per quanto riguarda la mantissa viene memorizzata soltanto la sua parte non intera (poiché la parte intera della stessa sarà sicuramente 1); per quanto riguarda l'esponente, esso è memorizzato come un numero naturale, ma per far ciò è necessario sommarlo preventivamente ad un valore, detto *bias*, che rappresenta il valore 0. Il valore decimale corrispondente alla sua rappresentazione secondo lo standard IEEE 754 è dato da

$$(-1)^S \times 2^{E-bias} \times (1 + M \times 2^{-b_M})$$

dove S è il valore del bit di segno, E è il valore dell'esponente, M è il valore della mantissa e b_M è il numero di bit utilizzato per la rappresentazione della mantissa.

Single precision

il valore viene memorizzato su 32 bit, di cui il primo è riservato al segno (S), gli 8 bit successivi sono utilizzati per la memorizzazione dell'esponente e gli altri 23 bit sono utilizzati per memorizzarvi la mantissa (v. fig. 1.3).

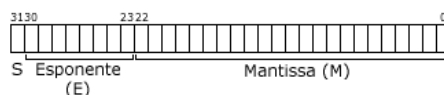


Figura 1.3: Rappresentazione dei numeri secondo lo standard IEEE 754 (single precision).

Un valore binario memorizzato secondo questo standard, corrisponde al valore decimale espresso da

$$(-1)^S \times 2^{E-127} \times (1 + M \times 2^{-23})$$

dove S è il valore del bit di segno, E è il valore dell'esponente (il bias è 127) e M è il valore della mantissa. Alcuni esempi sono riportati di seguito.

⁴l'operazione complementare a quella di normalizzazione.

⁵bit sta per *binary digit*, cioè *cifra binaria* (v. sez. 1.2.2).

per memorizzare la mantissa) genera un errore di *overflow*. L'errore di *underflow* sta a significare il fatto che sarebbero necessari più bit per rappresentare un esponente molto piccolo (ma in valore assoluto molto grande). Infatti, si considerino i valori sempre più piccoli, che si approssimano sempre di più allo 0: $0,1 = 1 \times 10^{-1}$, $0,01 = 1 \times 10^{-2}$, $0,001 = 1 \times 10^{-3}$, ... per la rappresentazione dei quali sono necessari esponenti sempre più piccoli ($-1, -2, -3, \dots$), ma in valore assoluto sempre più grandi ($1, 2, 3, \dots$). Oltre una certa precisione il valore dell'esponente non può essere rappresentato su 8 o 11 bit.

1.2.2 Bit, byte, nibble, word

bit

byte

nibble

word

double-word

LSB

MSB

Una cifra binaria è convenzionalmente detta **bit** (binary digit) e rappresenta la quantità minima di informazione utilizzabile da un computer. Un gruppo contiguo di 8 bit è detto **byte**. Un gruppo contiguo di 4 bit, cioè la metà di un byte, è detto *half-byte* o **nibble**. L'insieme di due byte adiacenti è detto **word** (parola). Esiste anche il termine **double-word** per indicare l'insieme di due word adiacenti.

Il bit meno significativo di un gruppo di bit (sia esso un byte, nibble o word) è detto **LSB** (Less Significant Bit), mentre quello più significativo è detto **MSB** (Most Significant Bit).

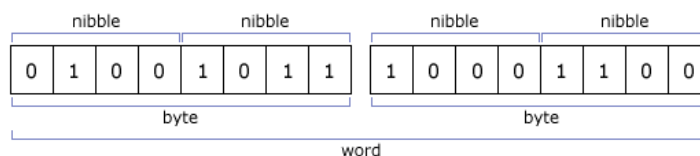


Figura 1.5: Raggruppamenti di bit.

Analogamente si parla di **MSB** (Most Significant Byte) in riferimento al byte più significativo di un gruppo di byte adiacenti e di **LSB** (Less Significant Byte) in riferimento a quello meno significativo.

1.2.3 La logica binaria

predicato

Un computer è in grado di eseguire operazioni logiche utilizzando l'*algebra di Boole*. Tale algebra consiste di un insieme di regole per la valutazione della veridicità dei *predicati*. Un **predicato** è una condizione che può assumere soltanto due valori: “vero” e “falso”, che nell'algebra di Boole sono associati rispettivamente ai simboli “1” e “0”.

Un predicato può essere assimilato ad una frase. Ad esempio, se si considera il predicato “Mario mangia una mela”, esso può risultare vero (1) o falso (0) dipendentemente dal fatto che Mario stia effettivamente mangiando una mela o meno. Un'altro esempio di predicato può essere “Luigi cammina per la strada”. In genere i predicati con cui il sistema ed i programmi avranno a che fare, saranno del tipo “Il valore contenuto all'indirizzo di memoria 1034293 è 35” oppure “La somma dei valori contenuti agli indirizzi di memoria 2470234 e 2470238 è maggiore di 20”.

Le operazioni logiche di base sono tre: l'*intersezione*, l'*unione* e la *negazione*.

Intersezione

intersezione logica

and

L'operazione di **intersezione logica**, indicata in genere con i simboli \wedge , \otimes o \odot , è un'operazione binaria (ovvero ha due operandi) ed è equivalente alla congiunzione “e” nella valutazione della veridicità di una frase, tanto che viene denominata anche con il nome **and**⁶. Quindi il predicato risultante dall'intersezione di due predicati A e B

$$A \wedge B$$

risulterà vero soltanto quando lo saranno entrambi i predicati A e B .

⁶il termine anglosassone *and* ha il significato di e.

L'operazione di intersezione logica (*and*) è definita dalla seguente *tavola di verità*

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Unione

L'operazione di **unione logica**, indicata in genere con i simboli \vee o \oplus , è un'operazione binaria (ovvero ha due operandi) ed è equivalente alla congiunzione “o” nella valutazione della veridicità di una frase, tanto che l'operazione viene denominata anche con il nome **or**⁷. Quindi il predicato risultante dall'unione di due predicati A e B

unione logica

or

$$A \vee B$$

risulterà vero quando lo sarà uno dei due predicati A o B .

L'operazione di unione logica (*or*) è definita dalla seguente *tavola di verità*

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Negazione

L'operazione di **negazione logica**, indicata in genere con il simbolo \neg o \sim , è un'operazione unaria (ovvero ha un solo operando) ed è equivalente alla negazione “non” nella valutazione della veridicità di una frase, tanto che l'operazione viene denominata anche con il nome **not**⁸. Quindi il predicato risultante dalla negazione di un predicato A

negazione logica

not

$$\bar{A}$$

risulterà vero quando sarà falso il predicato A e falso quando il predicato A sarà vero.

L'operazione di negazione logica (*not*) è definita dalla seguente *tavola di verità*

A	\bar{A}
0	1
1	0

1.2.4 L'ASCII

Poiché tutti i circuiti elettronici presenti in un computer lavorano su informazioni in forma “numerica”, anche le informazioni alfabetiche devono essere trasformate in forma numerica. E questo avviene per mezzo di una codifica che associa ad ogni carattere alfanumerico un corrispondente valore numerico di 1 byte che l'elaboratore potrà quindi gestire. Il sistema di codifica adottato dalla maggior parte dei sistemi è l'**ASCII** (American Standard Code for Information Interchange) definito dall'ANSI (American National Standards Institute)⁹ riportato in tab. 1.3.¹⁰

ASCII

Alcuni elementi della tabella ASCII non corrispondono a caratteri alfanumerici (v. i primi 32 elementi di tab. 1.3), ma sono dei codici di controllo a cui è stato attribuito un significato particolare riportato in tab. 1.4.

Dunque, anche le sequenze di caratteri alfanumerici, dette **stringhe**, vengono rap-

stringhe

⁷il termine anglosassone *or* ha il significato di o.

⁸il termine anglosassone *not* ha il significato di non.

⁹l'ANSI è un'organismo americano che redige standard, v. <http://www.ansi.org>.

¹⁰esistono anche altri tipi di codifiche come l'EBCDIC (Extended Binary Coded Decimal Interchange Code), utilizzata su sistemi IBM.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NUL	32	20		64	40	@	96	60	`
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	\$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	'	71	47	G	103	67	g
8	08	BS	40	28	(72	48	H	104	68	h
9	09	TAB	41	29)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	
29	1D	GS	61	3D	=	93	5D]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Tabella 1.3: La tabella ASCII.

Codice	Significato	Codice	Significato
NUL	null (carattere nullo)	DC1	device control 1
SOH	start of heading	DC2	device control 2
STX	start of text	DC3	device control 3
ETX	end of text	DC4	device control 4
EOT	end of transmission	NAK	negative acknowledge
ENQ	enquiry	SYN	synchronous idle
ACK	acknowledge	ETB	end of transmitted block
BEL	bell	CAN	cancel
BS	backspace	EM	end of medium
TAB	carattere di tabulazione	SUB	substitute
LF	line feed	ESC	escape
VT	vertical tab (tabulazione verticale)	FS	file separator
FF	form feed (salto pagina)	GS	group separator
CR	carriage return	RS	record separator
SO	shift out	US	unit separator
SI	shift in	DEL	delete
DLE	data link escape		

Tabella 1.4: I codici di controllo della tabella ASCII.

presentate e gestite dall'elaboratore come sequenze numeriche. In particolare, secondo la conversione ASCII, la stringa costituita da "prova" è convertita nella seguente sequenza numerica

stringa	p	r	o	v	a
esadecimale	70	72	6F	76	61
binario	01110000	01110010	01101111	01110110	01100001

È evidente che 128 codici (di cui alcuni soltanto di controllo) non possono essere sufficienti per codificare qualunque carattere utilizzato dalle varie lingue esistenti. So-

no così stati aggiunti altri 128 codici per la rappresentazione di caratteri particolari, definendo l'ASCII esteso (formato da un totale di 256 codici). Esistono varie tabelle ASCII estese, dipendentemente dal set di caratteri aggiunti ai 128 di base (ASCII standard). Questo crea però delle difficoltà di comunicazione tra sistemi che utilizzano codifiche ASCII estese diverse, poiché uno stesso codice ASCII esteso può corrispondere ad un carattere diverso dipendentemente dalla tabella ASCII estesa considerata per la codifica/decodifica.

1.2.5 ISO 8859-X, Unicode e UTF-8

Dall'insieme delle tabelle ASCII estese è nato lo standard ISO¹¹ 8859-1 (o Latin-1) che raccoglie in una tabella ASCII estesa i principali caratteri delle varie lingue, in modo tale che gli svedesi potessero scrivere “knäckebröd” ed i danesi “rødgrød med fløde”. Tale standard non era sufficiente a soddisfare tutte le esigenze delle varie lingue, quindi sono nati altri standard, come ISO 8859-2 (per le lingue dell'europa centrale), KOI-8 (per la lingua russa), JIS (per la lingua giapponese), GB5 (per la lingua cinese), ...

L'utilizzo di varie tabelle di codifica delle informazioni alfanumeriche, porta a correre il rischio di perdere delle informazioni quando le stesse passano da un sistema all'altro. A tale scopo è nato un sistema di codifica detto **Unicode**¹² che associa un valore univoco (espresso su più di un byte) ad ogni carattere, indipendentemente dal sistema e dalla lingua utilizzata. Parallelamente, con lo standard ISO/IEC 10646-1 è stata definita una codifica numerica per ogni simbolo di ogni lingua, ovvero l'UCS (Universal Character Set), che ha essenzialmente la stessa codifica di Unicode. Unicode è supportato da molti sistemi operativi, da tutti i più moderni web browser e da molti altri prodotti.

Unicode

Poiché la codifica Unicode non è molto agevole da utilizzare così com'è definita, a causa del fatto che molte delle routine già scritte si troverebbero ad avere a che fare con un elevato numero di byte nulli¹³ (ad esempio per i caratteri la cui codifica è rappresentabile per mezzo di un solo byte), è stata definita la codifica **UTF-8** (Unicode Transformation Format-8), come descritto nella norma ISO 10646-1:2000 Annex D e nella RFC 2279. Si tratta di una codifica che associa ad ogni carattere un numero di byte variabile da 1 fino a 6, dipendentemente dalla codifica dello stesso secondo l'Unicode di base (v. tab. 1.5)¹⁴: essa codifica ulteriormente le coppie di byte che rappresentano i caratteri alfanumerici, in maniera tale da non far risultare byte nulli, tranne nel caso in cui il carattere da codificare sia il carattere NUL. UTF-8 è stata definita in modo che i caratteri ASCII standard (00_H - 7F_H) vengano codificati su un solo byte ed assumono i valori previsti dalla codifica ASCII (v. tab. 1.3).

UTF-8

Unicode	UTF-8
00000000 _H - 0000007F _H	0XXXXXX ₂
00000080 _H - 000007FF _H	110XXXX ₂ 10XXXXXX ₂
00000800 _H - 0000FFFF _H	1110XXX ₂ 10XXXXXX ₂ 10XXXXXX ₂
00010000 _H - 001FFFFF _H	1110XXX ₂ 10XXXXXX ₂ 10XXXXXX ₂ 10XXXXXX ₂
00200000 _H - 003FFFFF _H	11110XXX ₂ 10XXXXXX ₂ 10XXXXXX ₂ 10XXXXXX ₂ 10XXXXXX ₂
04000000 _H - 7FFFFFFF _H	111110XX ₂ 10XXXXXX ₂ 10XXXXXX ₂ 10XXXXXX ₂ 10XXXXXX ₂ 10XXXXXX ₂

Tabella 1.5: Codifica dei caratteri secondo UTF-8.

Ad esempio, il simbolo del copyright (©), che secondo la codifica Unicode è rappresentato dal valore U-00A9_H¹⁵ (= 00000000₂ 10101001₂), secondo la codifica UTF-8 è rappresentato come C2A9_H (= 11000010₂ 10101001₂). Il carattere di disuguaglianza (≠), che secondo la codifica Unicode è rappresentato dal valore U-2260_H (= 00100010₂

¹¹l'ISO (International Organization for Standardization) è un organismo internazionale che redige standard (v. <http://www.iso.ch>).

¹²v. <http://www.unicode.org>.

¹³il byte nullo corrisponde al carattere NUL secondo la codifica ASCII, che assume un significato particolare: la fine di una stringa.

¹⁴si noti che il numero di 1 nella parte più significativa del primo byte della codifica UTF-8 indica il numero di byte utilizzati nella codifica del carattere stesso.

¹⁵i codici Unicode sono generalmente rappresentati con il prefisso “U-”.

01100000₂), secondo la codifica UTF-8 è rappresentato come E289A0_H (= 11100010₂ 10001001₂ 10100000₂).

1.3 Gli utenti

Le persone che interagiscono con il sistema si dicono **utenti** ed in GNU/Linux, come in qualunque sistema operativo multiutente, è possibile definirne più di uno per ogni sistema. La definizione degli utenti è necessaria poiché il sistema deve riconoscere l'utente al momento del suo accesso.

L'insieme degli utenti che possono accedere al sistema è generalmente memorizzato all'interno del sistema stesso. Ogni utente è identificato univocamente da un valore numerico, lo **UID** (User IDentifier) ed a questo sono associati uno **username** (il nome dell'utente) ed una **password** (parola d'ordine). Al momento dell'accesso, il sistema richiede all'utente di inserire il proprio username e per verificare che si tratti proprio di lui, richiede anche la relativa password, che solo l'utente in questione dovrebbe conoscere. Se l'utente viene riconosciuto, in base allo username ed alla password inseriti, questi viene fatto accedere all'*interfaccia utente* del sistema (v. sez. 1.4).

Tra tutti gli utenti ne esiste uno speciale detto **superuser**, il cui UID è 0, che generalmente ha uno username uguale a "root". Questo rappresenta l'*amministratore del sistema* (intendendo per amministratore colui che lo gestisce, lo configura, ...) ed il sistema non attua nessun meccanismo di sicurezza per l'accesso alle risorse nei confronti di tale utente. È pertanto opportuno ridurre al minimo gli accessi al sistema in qualità di *superuser*, per evitare di modificare inavvertitamente parti del sistema che potrebbero comprometterne il funzionamento.



I concetti qui introdotti saranno trattati più in dettaglio nel cap. 5.

1.4 L'interfaccia utente

interfaccia utente

Una volta acceduto al sistema tramite un'opportuna procedura di riconoscimento, un utente si trova davanti quella che viene detta **interfaccia utente**, dalla quale si possono impartire comandi al sistema stesso. GNU/Linux ha la possibilità di presentare due tipi di interfaccia utente

shell

(detta anche *interfaccia testuale* o *interfaccia a riga di comando*) è l'interfaccia di base, la più vecchia ma dalle enormi potenzialità. Questa si presenta essenzialmente con uno schermo nero suddiviso in 24 righe per 80 colonne, sul quale appare una sequenza di simboli, il *prompt*, che indica che il sistema è pronto per ricevere un comando dall'utente (v. fig. 2). L'utente può quindi inserire un comando tramite la tastiera e premere il pulsante  (o ) (v. sez. 1.8) per confermarlo. I comandi vengono impartiti con la seguente sintassi

\$ *command* [*args*]

dove

\$ rappresenta il prompt della shell;

command è il comando da eseguire (generalmente il nome di un file eseguibile);

args è un eventuale elenco di argomenti (parametri) che possono essere specificati al comando che li interpreta e si comporta di conseguenza (gli argomenti sono specifici per ogni comando);

Il comando per esteso (comprensivo degli eventuali argomenti), cioè la parte **command** [*args*] costituisce quella che viene chiamata **riga di comando**, che è poi l'ordine impartito al sistema.

X Window System

(spesso denominata anche *X Window*, o soltanto *X*, *interfaccia grafica* o **GUI** – Graphic User Interface) è l'interfaccia più innovativa, ma anche la più avida di risorse (memoria). Questa si presenta con una schermata analoga a quella di fig. 3 (pag. xv), cioè quello che viene definito *desktop* (scrivania), sul quale vi sono delle immagini, dette *icone* (*icon*), con le quali l'utente può interagire per mezzo del dispositivo di puntamento (mouse). Le applicazioni sviluppate per tale interfaccia si basano su finestre (*window*), ovvero si presentano all'utente come delle “schede” con una veste grafica, nelle quali si possono inserire dei dati con la tastiera, cliccare pulsanti grafici, ... (v. fig. 1.6). Le *window* costituiscono l'interfaccia grafica delle applicazioni.

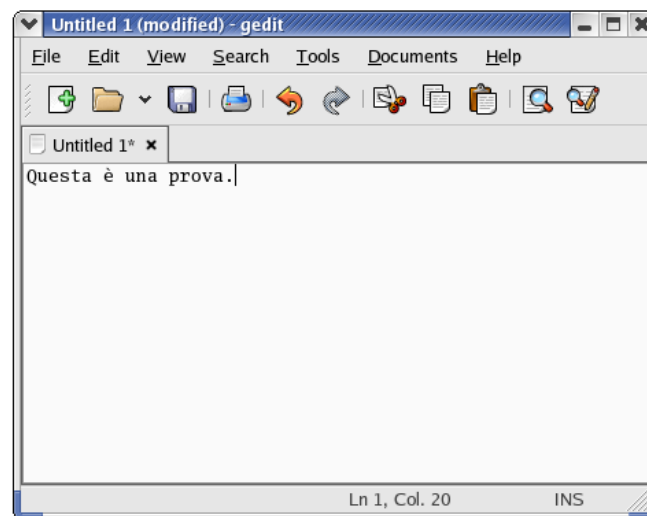


Figura 1.6: Esempio di *window*.

La shell è l'interfaccia di base del sistema poiché le applicazioni vengono sempre lanciate in esecuzione attraverso una riga di comando (anche se il processo utilizza l'interfaccia grafica). Quindi tutti i comandi, come anche quelli riportati nel presente testo, sono impartibili tramite la shell.

Nel seguito del testo, salvo specifico avviso, la shell di riferimento sarà *Bash* (v. cap. 7).

Le interfacce utente saranno descritte più in dettaglio nei cap. 7 e 11.

1.5 L'hardware

Un computer è costituito da un'unità centrale, cioè l'insieme dei circuiti elettronici contenuti all'interno di una “scatola” metallica, detta **case** o *chassis*, alla quale sono collegati una **tastiera**, un *dispositivo di puntamento*, costituito generalmente da un **mouse** (piccolo dispositivo con due o più pulsanti)¹⁶, un **monitor** ed eventualmente una **stampante** (*printer*) ed altri dispositivi. La tastiera ed il mouse sono dei *dispositivi di input* (*input device*) o dispositivi di ingresso, attraverso i quali possono essere impartiti i comandi al sistema, mentre il monitor (come la stampante) è un *dispositivo di output* (*output device*) o dispositivo di uscita, ovvero un dispositivo attraverso il quale il sistema comunica le sue risposte ai vari comandi.

case
tastiera
mouse
monitor
stampante

¹⁶esistono anche altri dispositivi di puntamento come le *tavolette grafiche*, i *touch pad*, le *trackball*, ...

All'interno del case è presente una *motherboard* (il cuore del sistema) alla quale sono connesse una (o più) *CPU*, la *memoria*, i *bus*, i *dispositivi (device)* o *periferiche* (mouse, tastiera, scheda video, scheda audio, stampante, ...), ovvero quello che viene definito **hardware** (la parte dura, fisica).

hardware

1.5.1 La motherboard

motherboard

La **motherboard** (o *scheda madre*) è un circuito stampato che raccoglie gran parte della circuiteria necessaria per il funzionamento del computer. Su questa sono integrati i controller della memoria e delle porte di comunicazione con l'esterno (PS/2, USB, seriale, parallela, ...). Essa ha gli alloggiamenti per la CPU, per la memoria centrale, per il collegamento ai dischi ATA e per la connessione alle periferiche. Un esempio di motherboard è riportata in fig. 1.7.

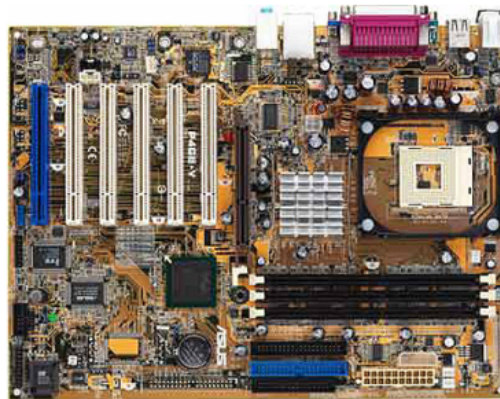


Figura 1.7: Una motherboard.

1.5.2 La CPU

CPU

La **CPU** (Central Processing Unit) o *microprocessore* (o semplicemente *processore*) è il cervello del computer, un circuito integrato che funge da centrale di gestione dell'hardware, esegue calcoli ed operazioni logiche, nonché sequenze di istruzioni, ovvero i programmi (il *software*¹⁷).

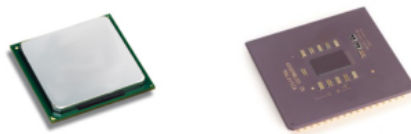


Figura 1.8: Due CPU di due diversi produttori.

Le operazioni svolte dalla CPU sono temporizzate per mezzo di un dispositivo esterno alla CPU stessa, il **system clock** (orologio di sistema), un circuito che genera segnali elettrici ad intervalli regolari, detti in genere *impulsi di clock* (tipicamente produce un segnale che ha la forma di un onda quadra), in corrispondenza dei quali la CPU esegue delle operazioni. La velocità di una CPU è descritta in termini di *frequenza* o *cicli di clock* (*clock tick*), cioè una CPU che lavora a 500 MHz riceverà 500.000.000 impulsi di clock al secondo. L'indicazione della *frequenza* di clock non è effettivamente indicativa della potenza di calcolo di una CPU, poiché CPU tecnologicamente diverse possono eseguire più o meno istruzioni per ogni impulso di clock. Un parametro più utile per confrontare CPU differenti è il numero di **MIPS** (Mega Instructions Per Second), cioè

system clock

MIPS

¹⁷v. sez. 1.9.

il numero dei milioni di istruzioni che una CPU può effettuare in un secondo. Più tale valore è elevato, più operazioni può effettuare la relativa CPU nell'unità di tempo.

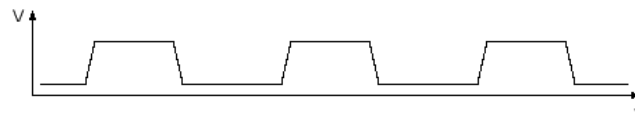


Figura 1.9: Esempio del segnale di temporizzazione emesso dal system clock.

Spesso, con il termine CPU, si fa riferimento addirittura all'intera parte del sistema che è contenuto all'interno del case.

1.5.3 La memoria

La **memoria** è la parte del sistema adibita appunto a memorizzare le informazioni. Si divide in *memoria centrale* e *memoria di massa*.

La **memoria centrale** è costituita da circuiti integrati suddivisi in gruppi che vengono denominati **banchi** di memoria, ed è situata sulla motherboard. Si tratta di una memoria molto veloce nella quale ogni **cella** ha la capacità di memorizzare 1 byte. Ogni cella della memoria centrale è identificata da un valore numerico, detto **indirizzo di memoria**. Quindi l'accesso ad una cella di memoria viene sempre effettuato specificando il relativo *indirizzo*.

La memoria centrale si suddivide in *ROM* e *RAM*. La **ROM** (Read Only Memory) è una memoria *non volatile*, contenente le informazioni necessarie per il funzionamento della macchina ed è accessibile soltanto in lettura, ovvero i dati in essa contenuti non possono essere modificati (in questo modo è impossibile per gli utenti modificare il suo contenuto e quindi rendere inavviabile la macchina)¹⁸. La **RAM** (Random Access Memory) è la memoria utilizzata dal sistema per eseguire i programmi. Si tratta di una memoria *volatile*, ovvero allo spegnimento della macchina tutti i dati in essa contenuti vengono perduti. La RAM è in genere organizzata in maniera gerarchica, dipendentemente dalla velocità di risposta dei dispositivi che la compongono: la memoria più veloce è detta **cache** ed è utilizzata per memorizzare temporaneamente delle informazioni a cui si accede di frequente¹⁹. Questo tipo di memoria è la più costosa ed è anche quella che consuma più energia. Le informazioni memorizzate nella cache e nell'altra parte della RAM devono essere mantenute allineate (le informazioni presenti nella cache devono essere coerenti con le stesse presenti in RAM): ovvero quando il contenuto della cache viene aggiornato, il sistema deve aggiornare anche le stesse informazioni presenti nella RAM. La coerenza delle informazioni contenute in tale porzione di memoria è garantita in parte dall'hardware ed in parte dal sistema operativo.



Figura 1.10: Un "banco" di RAM.

La **memoria di massa** è utilizzata dal sistema per archiviare informazioni (programmi e dati). Si tratta di una memoria non volatile ma più lenta della memoria

¹⁸con le versioni più recenti dell'hardware, alcune informazioni sono memorizzate in **EEPROM** (Electrically Erasable and Programmable Read Only Memory), o **E²PROM**, ovvero un chip che può essere eventualmente riprogrammato dall'utente per eventuali futuri aggiornamenti distribuiti dall'azienda fornitrice dell'hardware.

¹⁹possono esistere anche più livelli di memoria cache: quella di primo livello, la più veloce, è in genere contenuta all'interno della CPU stessa.

memoria

memoria centrale

banchi

cella

indirizzo di memoria

ROM

RAM

cache

memoria di massa

centrale ed in genere si compone di dischi/nastri magnetici e/o di dispositivi a tecnologia ottica (o magneto-ottica). Il supporto fisico utilizzato per la memorizzazione delle informazioni è generalmente detto *media*²⁰, mentre il dispositivo in grado di leggere e/o scrivere le informazioni sul media è detto unità a disco o *drive*. I media più comuni sono di seguito descritti.



Figura 1.11: Un hard disk.

Generalmente un PC (Personal Computer) utilizza dei dischi magnetici per l'archiviazione delle informazioni. Questi possono essere fissi, come gli **hard disk** (*dischi rigidi* o *dischi fissi*), o rimovibili, come i **floppy disk**²¹ (*dischetti*). I floppy disk hanno le dimensioni standard di $3\frac{1}{2}$ " e possono contenere da 400 KiB a 1,4 MiB (dipendentemente dal filesystem utilizzato).

hard disk
floppy disk



Figura 1.12: Nell'ordine: un floppy disk, uno zip disk ed una flash pen.

I lettori di floppy disk (e quindi i relativi supporti, cioè i floppy disk stessi) stanno ormai scomparendo dal mercato poiché il livello tecnologico attuale li ha resi obsoleti. Al loro posto esistono dei supporti di dimensioni analoghe in grado di memorizzare molte più informazioni, come gli *zip disk*. L'unico svantaggio, rispetto ai floppy disk, sono i costi sia dei supporti, che degli appositi lettori ed il fatto che il sistema non riconosce automaticamente tali dispositivi come invece avviene per i lettori di floppy disk.

Recentemente stanno prendendo campo anche dei dispositivi di memorizzazione detti **flash pen**²², basati su memorie di tipo *flash*²³ che, tramite opportuni driver, riescono a farsi "vedere" dal sistema operativo come se fossero dei dischi. Tali periferiche vengono collegate al sistema tramite la porta USB e la loro capacità di memorizzazione va da 32 MiB ad 1 GiB.

flash pen



Figura 1.13: Un CD.

L'archiviazione delle informazioni può essere effettuata anche su media che utilizzano la tecnologia ottica (**CD** – Compact Disc – ed i **DVD** – Digital Versatile Disc²⁴) e magneto-ottica (MO). I supporti ottici hanno forma e dimensioni standard da $5\frac{1}{4}$ " (esistono anche da $3\frac{1}{2}$ "), ma, mentre i CD possono contenere fino a 640 o 700 MiB, i

²⁰inglesismo che deriva dal latino medium (mezzo, tramite).

²¹il termine *floppy* (molliccio) deriva dalla consistenza dei primi dischetti da $5\frac{1}{4}$ " (ormai obsoleti) in contrapposizione a quella dei dischi non rimovibili presenti sulla macchina.

²²il nome deriva dalle dimensioni ridotte del dispositivo, analoghe a quelle di una penna biro.

²³particolari memorie RAM non volatili (i dati rimangono memorizzati anche se il dispositivo viene scollegato dal computer).

²⁴quando è stato coniato, l'acronimo aveva il significato di Digital Video Disc.

DVD possono arrivare a contenere fino a 4,37 GiB su ogni strato (esistono supporti DVD con 2 strati per faccia, che possono contenere fino a 15,9 GiB).

Mentre per tutti gli altri media le informazioni possono essere scritte, cancellate e sovrascritte più volte, per i CD esistono due tipi diversi di supporti: quelli scrivibili solo una volta (CD-R) e quelli riscrivibili più volte (CD-RW), con costi notevolmente diversi tra loro.

1.5.4 Le periferiche

Le **periferiche**, non sono altro che dispositivi collegati alla motherboard, secondo diverse modalità. *periferiche*



Figura 1.14: Alcune periferiche.

Le periferiche si possono suddividere in *interne* ed *esterne* dipendentemente dal fatto se possono essere alloggiate fisicamente all'interno dello chassis del computer o meno. Ad esempio la scheda video, la scheda audio, la scheda di rete, gli hard disk, ... sono tipicamente periferiche interne, mentre dispositivi come la tastiera, il mouse, la stampante, ... sono periferiche esterne. Possono esistere periferiche interne ed esterne che realizzano lo stessa funzione: esistono modem, lettori CD, ed altre periferiche sia interne che esterne.

Le periferiche sono collegate alla motherboard per mezzo di linee di collegamento denominate *bus*. Alcune periferiche interne che si presentano sottoforma di schede elettroniche, sono in genere fisicamente inserite negli appositi alloggiamenti (*slot*) presenti sulla motherboard stessa.

Un sistema operativo ha la necessità di conoscere la data e l'ora corrente e per questo i PC includono sulla motherboard una periferica speciale: il **RTC** (Real Time Clock). Questa periferica costituisce l'orologio del sistema ed è alimentata da una propria batteria che gli permette di funzionare anche quando il PC è spento.

1.5.5 I bus

I **bus** sono i canali attraverso i quali avvengono le comunicazioni tra la CPU e gli altri dispositivi. Essi sono costituiti dall'insieme dei collegamenti tra i vari circuiti elettronici e possono essere interni o esterni dipendentemente dal fatto se essi sono tipicamente utilizzati all'interno o all'esterno dello chassis. *bus*

Si possono distinguere due tipologie di bus dipendentemente dal numero di linee che lo compongono: si parla di

bus seriale

è un bus costituito da una sola linea, sul quale le informazioni vengono trasmesse serialmente, una dopo l'altra;

bus parallelo

è un bus costituito da più linee, sul quale le informazioni vengono trasmesse in modo parallelo (più informazioni contemporaneamente, in parallelo una all'altra, una su ogni linea);

Il *bus di sistema*, ovvero quello che collega la CPU al resto del sistema, si può suddividere logicamente in 3 tipi di bus distinti, dipendentemente dalle funzioni delle connessioni: l'*address bus*, il *data bus* ed il *control bus*. L'**address bus** è costituito dalle linee che trasportano gli indirizzi di memoria e tipicamente tali linee sono unidirezionali, ovvero le informazioni viaggiano su tale bus soltanto in una direzione, dalla CPU verso la memoria. Il **data bus** è costituito dalle linee che trasportano le informazioni vere e proprie; ogni linea di questo bus è bidirezionale, ovvero può trasportare le informazioni in entrambi i sensi: da e verso la CPU. Il **control bus** è costituito da tutte le linee che trasportano segnali di controllo in tutto il sistema. Ogni linea di questo bus è unidirezionale.

In genere il numero di linee che compongono l'address bus è utilizzato come riferimento dei tipi di architettura con i quali sono costruiti i computer. Ad esempio, si parla di *architettura a 32 bit* se le linee dell'address bus sono 32, cioè la CPU può indicare gli indirizzi di memoria centrale con un massimo di 32 bit (ovvero da 0 a $2^{32} - 1 = 4.294.967.295$).

I bus si possono suddividere anche in base allo standard tecnologico con il quale sono realizzati. Di seguito è riportata una breve descrizione per ogni tipo di bus.

ISA (Industry Standard Architecture) si tratta di un bus interno che inizialmente utilizzava 8 linee per la trasmissione dei dati (8 bit) e successivamente è stato esteso a 16. È un bus piuttosto lento;

MCA (Micro Channel Architecture) si tratta di un bus interno a 32 bit introdotto da *IBM* nel 1987;

EISA (Extended ISA) si tratta di un bus interno a 32 bit introdotto da *Compaq* nel 1987, compatibile con il bus ISA, cioè gli alloggiamenti per le schede inseribili su tale bus sono composti da due parti: una ISA e l'altra per i segnali aggiunti dall'EISA;

VLB (VESA Local Bus) si tratta di un bus interno a 32 bit con specifiche VESA (Video Electronic Standards Association) introdotto nel 1992. È compatibile con il bus ISA e permette al massimo due alloggiamenti per altrettante eventuali schede;

PCI (Peripheral Component Interconnect) si tratta di un bus interno a 32 e 64 bit (dipendentemente dalla piattaforma) introdotto nel 1993 e presente ormai in tutti i PC. GNU/Linux permette di gestire tale bus con i comandi `lspci` (man page²⁵ `lspci(8)`) e `setpci` (man page `setpci(8)`).

Comando: `lspci`
 Path: `/sbin/lspci`
 SINTASSI
 # `lspci [option]`

DESCRIZIONE

option indica la modalità di funzionamento di `lspci`. Può assumere i seguenti valori:

- v visualizza informazioni dettagliate per ogni dispositivo;
- vv visualizza il massimo livello di dettagli delle informazioni relative ad ogni dispositivo;
- n visualizza i codici numerici dei dispositivi PCI e dei relativi produttori al posto della corrispondente denominazione;

²⁵le *man page* sono pagine di manuale in forma elettronica presenti sui sistemi Unix-like (v. sez. 1.16).

- x visualizza i primi 64 byte (standard header) della configurazione di ogni dispositivo PCI;
- xxx visualizza tutti i byte della configurazione di ogni dispositivo PCI;
- b visualizza tutti gli IRQ e gli indirizzi “visti” da ogni dispositivo, invece che dal kernel;
- t visualizza un diagramma ad albero che riporta le dipendenze dei vari dispositivi PCI;
- s `[[bus]:][slot][.function]`
seleziona soltanto i dispositivi relativi al bus *bus*, all'alloggiamento *slot* ed alla funzione *function*. Ognuno dei parametri può essere omesso o specificato con il simbolo '*': in tal caso qualsiasi valore viene considerato valido;
- d `[vendor]:[device]`
seleziona soltanto i dispositivi relativi al produttore *vendor* e identificativo *device*. Ognuno dei parametri può essere omesso o specificato con il simbolo '*': in tal caso qualsiasi valore viene considerato valido;
- i *file* utilizza il file specificato da *file* come database per gli identificatori dei dispositivi PCI al posto di `/usr/share/hwdata/pci.ids`;
- p *dir* | -P *dir*
utilizza la directory specificata da *dir*, come contenitore delle informazioni relative al bus PCI al posto di `/proc/bus/pci`;
- m visualizza le informazioni in “machine readable form” per facilitarne il parsing;
- M esamina il bus PCI in modo da riportare informazioni dei dispositivi mal configurati (può bloccare il sistema);
- H1 utilizza l'accesso diretto all'hardware per mezzo del meccanismo di configurazione Intel 1;
- H2 utilizza l'accesso diretto all'hardware per mezzo del meccanismo di configurazione Intel 2 (funziona solo con le prime 16 periferiche e non sembra essere molto affidabile);
- F *file* visualizza le informazioni contenute nel file *file* secondo il formato di output del comando `lspci -x`;
- G visualizza ulteriori informazioni durante il funzionamento di `lspci`;
- version
visualizza la versione di `lspci`;

Comando: `setpci`

Path: `/sbin/setpci`

SINTASSI

`setpci [option] [device] [operation] [...]`

DESCRIZIONE

option indica la modalità di funzionamento di `setpci`. Può assumere i seguenti valori:

- v visualizza informazioni dettagliate per ogni dispositivo;
- f indica di non visualizzare nessun messaggio di errore se nessun dispositivo è stato indicato;
- D “Demo mode” - simula l'operazione sul bus PCI ma non la effettua realmente;
- s `[[bus]:][slot][.function]`
seleziona soltanto i dispositivi relativi al bus *bus*, all'alloggiamento *slot* ed alla funzione *function*. Ognuno dei parametri può essere omesso o specificato con il simbolo '*': in tal caso qualsiasi valore viene considerato valido;

-d [vendor]:[device]
 seleziona soltanto i dispositivi relativi al produttore *vendor* e identificativo *device*. Ognuno dei parametri può essere omissso o specificato con il simbolo '*': in tal caso qualsiasi valore viene considerato valido;

-P dir utilizza la directory specificata da *dir*, come contenitore delle informazioni relative al bus PCI al posto di */proc/bus/pci*;

-H1 utilizza l'accesso diretto all'hardware per mezzo del meccanismo di configurazione Intel 1;

-H2 utilizza l'accesso diretto all'hardware per mezzo del meccanismo di configurazione Intel 2 (funziona solo con le prime 16 periferiche e non sembra essere molto affidabile);

-F file visualizza le informazioni contenute nel file *file* secondo il formato di output del comando **lspci -x**;

-G visualizza ulteriori informazioni durante il funzionamento di **lspci**;

--version
 visualizza la versione di **setpci**;

device specifica l'identificativo numerico del dispositivo PCI da considerare;

operation specifica l'operazione da effettuare secondo la sintassi

reg[=value]

dove

reg è il nome del registro o il suo indirizzo seguito dal suffisso **.b**, **.w** o **.l** che indica la dimensione del registro (rispettivamente byte, word o longword). Il nome dei registri possono essere ottenuti dalla man page di **setpci**;

value è il valore da assegnare al registro specificato;

È inoltre possibile visualizzare l'elenco dei dispositivi presenti sul bus PCI riconosciuti dal sistema, visualizzando il contenuto del file */proc/pci*.

AGP (Accelerated Graphic Port) si tratta di un bus interno introdotto recentemente, appositamente progettato per un elevato trasferimento di dati per adattatori video molto veloci. Permette l'inserimento di una sola scheda video poiché presenta un solo adattatore.

ATA (AT – Advanced Technology – Attachment) spesso chiamato impropriamente *IDE* (Integrated Drive Electronics) o *EIDE* (Enhanced IDE), è un bus di tipo parallelo, sviluppato inizialmente da *Compaq*, *Western Digital* ed *Impris* nel 1986, utilizzato per collegare gli hard disk e le unità a disco o CD/DVD (*ATAPI* – ATA Peripheral Interface) interne, che permette l'impiego di hardware a più basso costo (il controller viene in genere integrato sul dispositivo). Si basa sul bus ISA e permette velocità di trasferimento delle informazioni da 8,3 MiB/s a 133 MiB/s su 16 linee di dati (16 bit).²⁶

Sulla motherboard sono integrati generalmente i connettori relativi a due **canali** ATA e per ognuno di essi possono essere collegate fino ad un massimo di due periferiche. Tali canali sono detti l'uno **primary** (primario) e l'altro **secondary** (secondario).

Ogni periferica connessa a tale bus ha a bordo il relativo controller ed è caratterizzata da una sorta di priorità: su ogni canale è possibile definire una periferica **master** (prioritaria) ed una **slave** (meno prioritaria). La periferica identificata come *master* gode del privilegio che durante la fase di boot del sistema²⁷ viene interrogata per prima rispetto all'altra (eventualmente presente sullo stesso canale) per l'avvio del sistema stesso. Per il resto, le due periferiche sono trattate

²⁶inizialmente tale bus era a 8 bit.

²⁷v. cap. 2.

nello stesso identico modo. I dispositivi presenti sullo stesso canale devono sempre avere priorità diverse: devono essere impostati l'uno come master e l'altro come slave. Esiste anche la possibilità di impostare una configurazione automatica di ogni dispositivo, dipendentemente dal connettore a cui il dispositivo stesso è connesso (*cable select*). Infatti, le periferiche vengono collegate al canale ATA tramite un cavo piatto (*flat cable*, v. fig. 1.20) e questo cavo ha tre connettori, uno dei quali viene connesso alla motherboard ed ognuno degli altri due viene connesso ad un dispositivo (unità a disco): con particolari tipi di cavo, il dispositivo è in grado di riconoscere a quale dei due connettori (apparentemente identici) è connesso ed in questo modo viene impostato automaticamente il suo livello di priorità.

Di recente è stata introdotta sul mercato anche una nuova versione di tale bus denominata **SATA** (Serial ATA), che si basa su di una connessione seriale ad alta velocità (fino a 150 MiB/s nella prima versione e 300 e 600 MiB/s per le versioni successive), che soppianderà l'ormai vetusto (Parallel) ATA. Tale tecnologia è stata sviluppata dal *Serial ATA Working Group*²⁸, un consorzio di cui fanno parte *Intel*, *IBM*, *Dell*, *Maxtor*, *Quantum* e *Seagate*. Con questo tipo di tecnologia non c'è più bisogno dell'identificazione dei dispositivi master e slave ed è, per il momento, pienamente compatibile con ATA. L'utilizzo di Serial ATA è pertanto trasparente per i sistemi operativi.

SCSI (Small Computer System Interface) è un bus parallelo utilizzato per collegare sia le unità a disco (interne ed esterne) che altre periferiche come stampanti o scanner con le migliori performance (tempi di accesso più bassi e *transfer rate*²⁹ più elevati).³⁰ Queste richiedono la presenza di un controller apposito che in genere va acquistato separatamente. Questo innalza ulteriormente i costi di installazione delle periferiche SCSI.

Ad uno stesso bus SCSI possono essere collegate fino ad un massimo di 16 periferiche. Ogni periferica è individuata da un numero identificativo univoco (*ID*), oltre che da un valore numerico detto *LUN* (Logical Unit Number). La periferica che ha il valore ID più elevato è quella che ha la priorità più alta rispetto alle altre presenti sul bus SCSI. Ogni periferica è generalmente dotata di due interfacce SCSI, un connettore di ingresso ed uno di uscita, in modo tale che il bus possa "attreversare" la periferica stessa, per poter raggiungere tutte le periferiche connesse al bus SCSI in cascata, una dopo l'altra. L'ultima periferica collegata fisicamente sul bus SCSI deve essere "terminata", cioè il connettore di uscita dell'ultima periferica SCSI deve essere chiuso su sé stesso in modo da informare l'ultima periferica SCSI del fatto che dopo di essa non c'è nessun'altra periferica presente su tale bus.

Come per il bus ATA, anche per quello SCSI è allo studio, da parte del *Serial Attached SCSI Working Group*³¹, uno standard seriale basato su tecnologie di trasmissione simili a Serial ATA, il bus **SAS** (Serial Attached SCSI) che rimpiazzerà lo SCSI attuale.

PS/2 è un canale di comunicazione esterno generalmente utilizzato da tastiera e mouse. Tale standard è stato sviluppato da *IBM* e si presenta all'esterno del case con un connettore 6-pin³² mini-DIN³³ o 5-pin DIN. Generalmente i dispositivi che si connettono tramite un connettore a 5-pin DIN (più vecchio) sono detti di tipo *AT*, mentre quelli che utilizzano il connettore 6-pin mini-DIN (più recente) sono detti di tipo *PS/2*;

²⁸v. <http://www.serialata.org>.

²⁹indica la quantità di informazioni lette/scritte dalla/sulla periferica nell'unità di tempo.

³⁰lo standard SCSI si è evoluto in *Wide SCSI*, *Fast Wide SCSI*, *Ultra SCSI*, ... offrendo prestazioni sempre più elevate.

³¹v. <http://www.serialattachedscsi.com>.

³²*pin* è il conduttore fisico: 6-pin significa che il connettore è formato da 6 conduttori, cioè 6 linee.

³³*DIN* (Deutsches Institut für Normung) è un'organo tedesco che redige standard tecnici.



Figura 1.15: Connettore PS/2 (6-pin mini-DIN).

Parallela

spesso chiamata anche *Centronics* (dal nome dell'azienda che ne definì il primo standard) è il canale di comunicazione esterno in grado di effettuare lo scambio di più segnali contemporaneamente con il sistema (da cui il nome parallela). Si presenta all'esterno del case con un connettore a 25 pin (DB-25) ed è utilizzato per connettere dispositivi che hanno bisogno di velocità di trasferimento delle informazioni abbastanza elevate. Questo bus viene utilizzato prevalentemente da dispositivi come stampanti e scanner;



Figura 1.16: Connettore DB-25.

Seriale

spesso chiamata anche RS-232 (dal nome del primo standard utilizzato "Recommended Standard-232C" della EIA – *Electronic Industries Alliance* – sebbene recentemente la EIA abbia definito un nuovo standard RS-422 totalmente compatibile con esso), si presenta all'esterno del case con un connettore DB-25 o DB-9 (i pin effettivamente utilizzati sono soltanto 9). È il canale di comunicazione esterno in grado di scambiare un'informazione alla volta tra il sistema e le periferiche ad esso collegate (da cui il nome seriale). Questo bus viene generalmente utilizzato da dispositivi come mouse e modem;



Figura 1.17: Connettore DB-9.

USB (Universal Serial Bus) è lo standard più recente che soppianderà tutti gli standard relativi alla porta seriale e quella parallela. Sempre più periferiche vengono realizzate per essere collegate con questo tipo di bus esterno. Si tratta essenzialmente di un bus seriale con una velocità massima di trasmissione dei dati di 60 MiB/s. La connessione di dispositivi al sistema per mezzo di tale bus è semplicissima, dal fatto che esso permette il riconoscimento automatico dei dispositivi;

PCMCIA

(Personal Computer Memory Card International Association) è il canale di comunicazione utilizzato per connettere periferiche esterne ai computer portatili (*laptop*);

NIC (Network Interface Card) L'interfaccia di rete (scheda di rete) è quella che permette il collegamento tra più PC e/o periferiche con elevata velocità di scambio dei dati (da 10 Mbit/s a 1 Gbit/s). Il collegamento può essere effettuato mediante appositi cavi di rete o tramite onde radio (*wireless*);

Firewire

(IEEE³⁴ 1394) è un canale di comunicazione esterno che permette l'interfacciamento con periferiche che necessitano di elevate velocità di trasmissione dati

³⁴l'IEEE (Institute of Electrical and Electronics Engineers) è un organismo che pubblica articoli



Figura 1.18: Connettore USB.



Figura 1.19: Cavo di rete.

come le telecamere digitali. Tale bus può supportare comunicazioni fino a 50 MiB/s;

IrDA (Infrared Data Association) è un canale di comunicazione esterno che non necessita di cavo di collegamento. La trasmissione dei dati avviene per mezzo di onde elettromagnetiche nella banda di frequenze degli infrarossi. È utilizzata per dispositivi mobili che non necessitano di velocità di scambio dati particolarmente elevate (cellulare, mouse, tastiera, ...);

Attualmente esistono molte periferiche di tipo **PNP** (*Plug & Play* o *Plug'n'Play*) che hanno cioè la capacità di autoconfigurarsi sul sistema.

Esistono anche altri tipi di bus, ormai obsoleti, come **XT** (eXtended Technology) ed **ESDI** (Enhanced Small Device Interface).



Figura 1.20: Un *flat cable*.

Le linee che permettono il collegamento ai bus di tipo parallelo, all'interno dello chassis del computer, sono costituiti generalmente da cavi piatti (*flat cable*) che raggruppano più linee, ognuna protetta da un rivestimento plastico, una parallela all'altra (v. fig. 1.20).

L'impostazione della "priorità" di una periferica ATA o la terminazione di un dispositivo SCSI, viene effettuata generalmente per mezzo di *jumper* (ponticelli), ovvero dei piccoli pezzi metallici che chiudono il circuito elettrico tra due connettori terminali, o di piccoli interruttori (*microswitch*).

1.5.6 I controller

I **controller** sono parte dell'hardware dedicata alla gestione di circuiti particolari (sono dei processori dedicati). Ne sono un esempio, il controller della memoria centrale che si preoccupa di gestire le richieste di lettura/scrittura delle informazioni dalla/nella memoria centrale, il controller dei dischi ATA che si preoccupa di gestire l'accesso ai vari dischi ATA collegati agli opportuni alloggiamenti (presenti sulla motherboard), ...

controller

scientifici e standard tecnici internazionali e viene spesso riferito con la dizione anglosassone "Eye-triple-E", v. <http://www.ieee.org>.

1.6 I dischi

I dischi magnetici sono la periferica più utilizzata nei PC come memoria di massa. Un disco magnetico può essere composto da più *piatti* (v. fig. 1.21) di materiale trasparente ai campi magnetici, ricoperto su entrambe le facce da un deposito di materiale ferromagnetico. I piatti costituiscono il supporto fisico sul quale sono memorizzate le informazioni. L'accesso alle informazioni contenute sul disco avviene per mezzo di trasduttori magnetici che sono le **testine**. Queste sono in numero pari al doppio dei piatti che costituiscono il disco: una per ogni faccia. Quando il disco è in funzione, i piatti ruotano con una velocità angolare costante. Le testine possono muoversi soltanto trasversalmente al senso di rotazione dei piatti. Quando le testine sono in una determinata posizione, individuano una **traccia** su ogni faccia dei piatti del disco, ovvero la porzione del supporto magnetico che transita sotto di esse alla velocità di rotazione del disco. Ogni traccia viene suddivisa in sezioni tutte della stessa grandezza: i **settori**³⁵. Quest'ultimi hanno generalmente la capacità di contenere 512 byte.

testine

traccia

settori

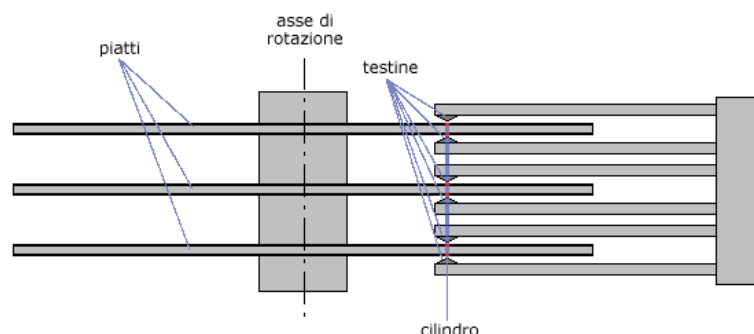


Figura 1.21: Vista di un disco magnetico in sezione longitudinale all'asse di rotazione.



Figura 1.22: Interno di un hard disk.

La meccanica del disco (l'insieme dei bracci che sostengono le testine ed il motore che li muove) è comandata dall'apposita circuiteria presente sull'unità a disco, che comprende dei circuiti integrati contenenti quello che viene detto *firmware* (microcodice o logica microprogrammata) del disco. Il firmware è l'interfaccia tra il bus ed il disco stesso.

1.6.1 CSH

Ogni settore può essere individuato col metodo **CHS** (Cylinder, Head, Sector), ovvero da tre coordinate: il *cilindro*, la *testina* ed il *settore*. Il *cilindro* rappresenta l'insieme delle tracce contenute sulle facce dei vari piatti ad una determinata distanza dall'asse di rotazione (una traccia per ogni faccia). La *testina* identifica la traccia da considerare,

³⁵ poiché il disco ruota a velocità angolare costante, i settori delle tracce più esterne saranno fisicamente più grandi (più lunghi) di quelli delle tracce più interne, ma conterranno la stessa quantità di informazioni.

ovvero quella dello stesso cilindro che passa sotto la testina di lettura/scrittura specificata. Il *settore* è appunto il segmento della traccia individuata per mezzo del cilindro e della testina a cui si vuole accedere.

1.6.2 LBA

Esiste anche un'altra modalità per accedere ai settori di un disco: il **LBA** (Logical Block Addressing). Ogni settore viene identificato da un numero sequenziale a partire dal primo settore che corrisponde al numero 0. LBA

Spesso viene confusa l'indicizzazione dei settori tramite il LBA con il supporto dei dischi di grandi capacità. Molti dei dischi che supportano il LBA non richiedono necessariamente che questo venga utilizzato. Quello che è necessario per gestire dischi con grandi capacità è un algoritmo di traduzione degli indirizzi dei settori. Esistono essenzialmente due algoritmi principali per la traduzione dell'indirizzamento dei settori dei dischi: l'**EC**HS (Extended CHS) e l'**Assisted LBA**. L'EC³⁶HS, noto anche come *CHS to CHS* o *bit-shift*, che in alcuni BIOS va sotto la voce "Large", non fa altro che dimezzare il numero di cilindri, raddoppiando il numero di testine, finché il numero di cilindri non è inferiore a 1024. L'altro algoritmo di traduzione degli indirizzi è l'*Assisted LBA*, chiamato anche semplicemente *LBA*. EC³⁶HS
Assisted LBA

L'algoritmo *LBA* è stato da sempre utilizzato per i dischi SCSI, mentre i dischi ATA hanno iniziato ad utilizzarlo da quando la loro capacità ha superato i 500 MiB. Per poter utilizzare il LBA è necessario che sia il disco che il BIOS lo supportino.

1.6.3 I dischi ottici

Mentre la gestione e l'organizzazione dei dischi magneto-ottici (MO) sono sostanzialmente analoghe a quelle dei dischi magnetici (la scrittura delle informazioni avviene con tecnologia magnetica, mentre la lettura delle stesse viene effettuata per mezzo di un raggio laser – tecnologia ottica), i dischi ottici (CD, DVD), nei quali la lettura e scrittura delle informazioni avviene per mezzo di un raggio laser, richiedono una memorizzazione dei dati in maniera sequenziale, consentendo eventualmente un'aggiunta in coda. Per questo, i dati vengono registrati sui dischi ottici utilizzando un'unica traccia a spirale dal centro verso la periferia del disco.

1.6.4 Le partizioni

I dischi fissi e quelli rimovibili di grandi dimensioni, possono essere suddivisi in **partizioni** partizioni che sono aree del disco in ognuna delle quali è possibile memorizzare le informazioni secondo uno specifico filesystem (v. sez. 1.7). Il sistema di suddivisione dei dischi in partizioni riconosciuto e gestito da GNU/Linux è quello utilizzato dal DOS³⁶ e dai sistemi operativi da esso derivati.

In pratica, i dischi fissi vengono sempre suddivisi in partizioni (al limite soltanto una). Nel caso di dischi rimovibili di grandi dimensioni, non suddivisi in partizioni, si parla di *superfloppy*.

La **partition table**, ovvero l'elenco delle partizioni in cui è suddiviso il disco e la relativa posizione fisica sul supporto magnetico, è memorizzata nella parte finale del primo settore del disco che è escluso dal partizionamento, cioè non è contenuto in nessuna partizione. Tale settore è detto **MBR** (Master Boot Record) poiché contiene anche il codice per l'avvio del *boot loader* (v. cap. 2). Lo spazio riservato nel MBR per annotare i dati delle partizioni è limitato e consente la suddivisione del disco in un massimo di quattro partizioni (*partizioni primarie*). La possibilità di suddividere lo spazio di un disco in un massimo di quattro partizioni, può essere troppo limitante, per partition table
MBR

³⁶il DOS (Disk Operating System) è un sistema operativo analogo a MS-DOS (*Microsoft* DOS) e DR-DOS (*Digital Research* DOS).

questo sono state introdotte le *partizioni estese*, partizioni che possono contenere al loro interno delle *partizioni logiche*.

La fig. 1.23 rappresenta un esempio di suddivisione di un disco in partizioni.

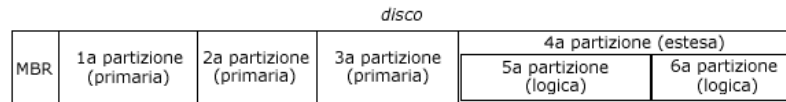


Figura 1.23: Schematizzazione della suddivisione di un disco in partizioni.

La suddivisione del disco in partizioni può essere effettuata con il comando **fdisk** (man page **fdisk(8)**).

Comando: **fdisk**
Path: **/sbin/fdisk**

SINTASSI
fdisk [*option*] [*device*]

DESCRIZIONE

option indica la modalità di funzionamento di **fdisk**. Può assumere i seguenti valori:

- b *sectorsize*
specifica la dimensione dei settori (512, 1024 o 2048). I kernel più recenti riconoscono da soli la dimensione dei settori;
- l
elenca la tabella delle partizioni (*partition table*) del disco specificato da *device*. Se non è stato specificato nessun *device*, vengono considerati quelli contenuti nel file **/proc/partitions** (se esiste);
- s *partition*
visualizza la dimensione della partizione specificata da *partition*;
- u
visualizza le dimensioni delle partizioni in settori anziché in cilindri;
- v
visualizza la versione di **fdisk**;

device è il disco sul quale si desidera operare;

In un sistema GNU/Linux, un disco viene indicato con un'espressione analoga alle seguenti:

/dev/hdx

indica un disco ATA identificato dalla lettera x: **/dev/hda** rappresenta il primo disco ATA, **/dev/hdb** il secondo, e così via (con x che va da a ad h);

/dev/sdx

indica un disco SCSI identificato dalla lettera x: **/dev/sda** rappresenta il primo disco SCSI, **/dev/sdb** il secondo, e così via (con x che va da a a p);

/dev/edx

indica un disco ESDI (obsoleto) identificato dalla lettera x: **/dev/eda** rappresenta il primo disco ESDI, **/dev/edb** il secondo, e così via (con x che va da a a d);

/dev/xdx

indica un disco XT (obsoleto) identificato dalla lettera x: **/dev/xda** rappresenta il primo disco XT e **/dev/xdb** il secondo;

Una partizione è specificata dal nome del disco seguito da un numero intero da 1 a 15. Ad esempio **/dev/hda1** rappresenta la prima partizione del primo disco ATA e **/dev/sdb3** rappresenta la terza partizione del secondo disco SCSI.

1.6.5 Il caching

L'accesso ai dati contenuti nella memoria di massa è un'operazione relativamente lenta rispetto ai tempi di CPU (il ciclo di clock di sistema è dell'ordine dei nanosecondi ($1 \text{ ns} = 10^{-9} \text{ s}$), mentre l'accesso ai dischi è dell'ordine dei millisecondi ($1 \text{ ms} = 10^{-3} \text{ s}$)). Inoltre spesso vengono effettuati accessi successivi a zone contigue del disco.

Per ridurre il numero di accessi al disco, i sistemi operativi utilizzano generalmente un *buffer*³⁷, o *memoria cache*³⁸, dedicando parte della memoria RAM alla “bufferizzazione” delle operazioni di lettura e scrittura su disco. Tutte le operazioni di scrittura relative al disco vengono effettuate in una zona della memoria centrale (la memoria *cache* sopra citata), quindi più veloce rispetto al disco, e solo in un secondo momento vengono replicate sul disco (l'operazione di scrittura sul disco viene rinviata ad un momento successivo in modo da avere accumulato un blocco di informazioni da scrivere abbastanza consistente). Analogamente, le informazioni sono lette dal disco a blocchi composti da un numero consistente di byte (anche nel caso in cui si abbia bisogno di leggere un solo byte) e sono copiate nella memoria cache. In questo modo, una successiva richiesta di un'informazione contenuta nel blocco appena letto viene velocemente soddisfatta senza dover accedere nuovamente al disco. A causa di questo meccanismo di caching, uno spegnimento accidentale dell'elaboratore può comportare una perdita parziale delle informazioni, se le operazioni di scrittura accodate nella memoria cache non sono state trasferite in tempo sul disco.

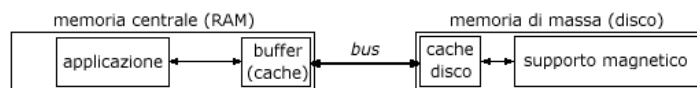


Figura 1.24: Schematizzazione del meccanismo di caching.

Oltre al meccanismo di caching da parte del sistema operativo, si aggiunge spesso una memoria cache nell'hardware della stessa unità a dischi (v. fig. 1.24). Questa non può essere controllata dal sistema, il quale, in genere, si limita a tentare l'accesso in scrittura più volte (3 volte) e/o ad attendere qualche secondo prima di considerare avvenuta l'effettiva scrittura delle informazioni sul dispositivo stesso.

1.6.6 Impostazioni dei dischi

Le impostazioni delle unità a disco possono essere modificate con il comando **hdparm** (man page **hdparm(8)**).

ATTENZIONE: l'utilizzo di questo comando può compromettere la struttura del filesystem (v. sez. 1.7) relativa all'unità a disco considerata.

Comando: **hdparm**
 Path: **/sbin/hdparm**
 SINTASSI
hdparm [option] [device]

DESCRIZIONE

option indica la modalità di funzionamento di **hdparm**. Può assumere i seguenti valori:

- a legge o imposta il numero di settori che il filesystem deve prelevare con un'operazione di lettura (il valore di default è 8 settori);

³⁷significa *serbatoio* ed indica un'area temporanea di scambio di dati.

³⁸non si confonda questa con la memoria cache di primo o secondo livello della CPU, che serve per velocizzare lo scambio di informazioni tra la CPU e la memoria centrale con un meccanismo analogo a quello qui descritto.

La logica microprogrammata delle unità a disco in genere prevede già un *read-ahead* (numero di settori prelevati con una sola operazione di lettura) di più di un settore.

- A[*value*]
imposta (*value* = 1) o disabilita (*value* = 0) la caratteristica di *look-ahead* dell'unità a disco (per default è attiva);
- b[*value*]
legge o imposta lo stato del bus (0 = off, 1 = on, 2 = tristate);
- B[*value*]
imposta l'APM (Advanced Power Management) se l'unità lo supporta (*value* = 1-255). Un valore *value* basso indica una gestione del consumo di tipo aggressivo, mentre un valore alto indica migliori prestazioni, ma consumi più elevati (indicando per *value* un valore di 255, si disabilita l'APM);
- c[*value*]
abilita (1) o disabilita (0) l'I/O a 32 bit oppure ne verifica la relativa impostazione. Specificando *value* = 3 si abilita l'I/O a 32 bit con una particolare sequenza dei segnali di sincronismo;
- C
Legge lo stato dell'alimentazione dell'unità, visualizzando i messaggi riportati in tab. 1.6;

Messaggio	Significato
unknown	il dispositivo non supporta il comando
active/idle	il dispositivo sta funzionando normalmente
standby	il dispositivo ha attivato la modalità a basso consumo
sleeping	il dispositivo ha attivato la modalità al più basso consumo

Tabella 1.6: Possibili stati dell'unità visualizzati da **hdparm**.

- d[*value*]
abilita o disabilita l'utilizzo della gestione DMA per l'unità;
- D[*value*]
abilita o disabilita la gestione dei malfunzionamenti del disco della logica microprogrammata presente sull'unità (*on-drive defect management*);
- E[*value*]
imposta la velocità del lettore CD-ROM;
- f
scarica la cache sul dispositivo;
- g
visualizza la geometria del disco (cilindri, testine, settori), la sua dimensione (in settori) ed il settore a cui esso inizia (offset);
- h
visualizza un aiuto sommario di **hdparm**;
- i
visualizza le informazioni identificative dell'unità (se disponibili);
- I
visualizza le informazioni identificative, in maniera più dettagliata, leggendole direttamente dall'unità;
- k[*value*]
imposta (1 = attiva, 0 = disattiva) o legge la caratteristica di memorizzazione (per default disattiva) delle impostazioni (relative alle opzioni **-dmu**) anche dopo un riavvio della macchina;
- K[*value*]
imposta (1 = attiva, 0 = disattiva) o legge la caratteristica di memorizzazione (per default disattiva) delle caratteristiche (impostate con le opzioni **-APSWXZ**) anche dopo un riavvio della macchina;
- m[*value*]
imposta o legge l'impostazione relativa al numero di settori trasferiti per ogni richiesta di I/O all'unità (IDE Block Mode).
- M[*value*]
imposta o legge l'impostazione relativa all'AAM (Automatic Acoustic Management). Il valore *value* può variare tra 0 e 254 (128 indica al dispositivo di essere più silenzioso possibile, ma anche il più lento, mentre 254 il più rumoroso ma più il veloce).

- n[*value*] imposta o legge l'impostazione relativa all'ignorare eventuali errori di scrittura sul disco.
- p[*value*] tenta di impostare la modalità PIO (Parallel Input Output) indicata da *value* (se non indicata, tenta di riconoscere automaticamente quale sia la migliore modalità PIO);
- P[*value*] imposta il numero massimo di settori relativo al meccanismo di pre-fetch interno dell'unità;
- q[*value*] tratta in maniera silente l'opzione che la segue, non visualizzandone i relativi messaggi di output;
- r[*value*] imposta o legge l'impostazione di accesso in sola lettura per l'unità considerata;
- R[*value*] registra un'interfaccia ATA (v. l'opzione -U);
- S[*value*] imposta il timeout relativo allo spin-down (standby) dell'unità, cioè il valore che indica al disco quanto tempo attendere, se non sta effettuando nessuna attività, prima di arrestare la rotazione, per risparmiare energia (0 indica nessuna gestione del risparmio energetico, i valori che vanno da 1 a 240 indicano timeout di multipli di 5 secondi, i valori da 241 a 251 specificano multipli di 30 minuti, il valore 252 indica un timeout di 21 minuti, 253 indica di utilizzare il timeout preimpostato dal costruttore e 255 indica 21 minuti e 15 secondi);
- T[*value*] esegue un test della velocità di lettura dalla cache che il sistema associa all'unità, senza effettuare alcun accesso al dispositivo fisico;
- t[*value*] esegue un test della velocità di lettura direttamente dal dispositivo;
- u[*value*] imposta o legge l'impostazione relativa all'*unmaskirq* (IRQ – Interrupt ReQuest) dell'unità. Consente (1) o meno (0) al driver di considerare altri interrupt mentre ne sta già servendo un altro);
- U[*value*] deregistra un'interfaccia ATA. Da utilizzare soltanto per dispositivi realizzati per supportare l'*hot-swap*³⁹;
- v[*value*] visualizza tutte le impostazioni (come -acdgmnr per le unità ATA, -gr per quelle SCSI o -adgr per quelle XT). È l'opzione di default quando non ne viene specificata nessuna;
- w[*value*] reimposta un'unità (pericoloso);
- W[*value*] abilita o meno la caratteristica di *write-caching* dell'unità;
- x[*value*] attiva (1) o meno (0) lo stato di alta impedenza (*tristate*) dell'unità (pericoloso);
- X[*value*] imposta la modalità di trasferimento delle informazioni (*transfer mode*) per il bus ATA, secondo la PIO⁴⁰ o il DMA⁴¹ (v. tab. 1.7);
- y[*value*] forza l'unità ATA ad attivare la modalità di basso consumo energetico (*standby*), facendolo fermare;

³⁹tale caratteristica, secondo la quale i dischi fissi possono essere rimossi ed inseriti nel sistema senza dover necessariamente arrestare quest'ultimo, è in genere supportata dai dispositivi SCSI, ma non da quelli ATA.

⁴⁰La PIO (Parallel Input/Output) è un dispositivo (circuitto integrato) che gestisce gli accessi al bus di sistema.

⁴¹Il DMA (Direct Memory Access) è un meccanismo che permette ai dispositivi di accedere direttamente alla memoria senza impegnare le CPU.

Value	Significato
mdma2	modalità DMA multiword
sdma1	modalità DMA semplice
udma2	modalità Ultra DMA
0	modalità PIO di default
1	disabilita lo IORDY (Input/Output ReaDY)
9,...,11	modalità PIO <i>value</i> -8
33,34	modalità DMA <i>value</i> -32
65,...	modalità Ultra DMA <i>value</i> -8

Tabella 1.7: Modalità di trasferimento delle informazioni impostabili con `hdparm`.

`-Y[value]`
forza l'unità ATA ad attivare la modalità di più basso consumo energetico (*sleep*), disattivandolo completamente;

`-z[value]`
forza il kernel a rileggere la *partition table*⁴² dell'unità;

`-Z[value]`
disabilita la funzione di risparmio energetico di alcuni dischi Seagate;

device è il disco sul quale si desidera operare;

Se nessun'opzione viene specificata, vengono assunte le opzioni `-acdghknru`.

1.7 Il filesystem

Il *filesystem* è il meccanismo che permette la memorizzazione e la gestione delle informazioni sulla memoria di massa. In GNU/Linux, come in tutti i sistemi operativi Unix-like, il filesystem è strutturato in maniera gerarchica ad *albero* (v. fig. 1.25), in cui ogni *nodo* può essere una *directory* o un *file*. Una **directory** è un contenitore che può contenere *file* e *directory*. Un **file** è l'unità (logica) per la memorizzazione di un insieme di informazioni.

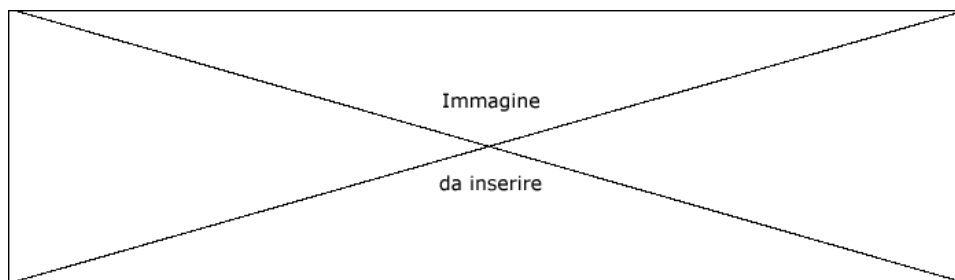


Figura 1.25: Esempio di *albero*.

La *root directory*, o *directory radice*, è rappresentata dal simbolo `'/'` (*slash*). Da questa discendono tutte le altre *directory*.

Nei sistemi Unix-like è definita anche una *working directory* (*directory di lavoro* o *directory corrente*) che indica la *directory* di default da considerare per determinate operazioni sul filesystem (una sorta di *directory sott'intesa*).

Il *path* (percorso) è l'elenco delle *directory* da attraversare per raggiungere un determinato nodo dell'albero (*file* o *directory*). Questo può essere *assoluto* o *relativo*. Un *path assoluto* è il percorso per raggiungere un determinato nodo a partire dalla *root directory* (compresa). Quello *relativo* è il percorso per raggiungere un determinato nodo a partire da una determinata *directory* (che in genere è la *working directory*).

Il simbolo utilizzato come separatore dei nomi dei nodi del filesystem è `'/'`, lo stesso utilizzato per rappresentare la *root directory*. Non ci sarà ambiguità nello specificare il

⁴²v. sez. 1.6.4.

path di un nodo del filesystem, poiché il simbolo ‘/’ sarà interpretato come root directory soltanto nel caso in cui sia specificato come primo carattere del path, altrimenti viene considerato come carattere di separazione tra i nodi dell’albero del filesystem.

Dunque, si supponga di voler raggiungere il file **prova** contenuto nella directory **seconda** contenuta nella directory **prima** che a sua volta è contenuta in / (root directory). Il path assoluto del file in questione è quindi **/prima/seconda/prova**, mentre il suo path relativo alla working directory, supponendo che questa sia la directory **prima** contenuta nella root directory, è **seconda/prova**.

Nel testo si cercherà sempre di fare riferimento a file e directory con path assoluti a meno che il contesto non sia già sufficientemente chiaro.

I concetti qui accennati saranno trattati nel cap. 3.

1.8 Il layout della tastiera

La tastiera è l’elemento principe che permette all’utente di inviare informazioni al sistema, ovvero permette di digitare⁴³ comandi o dati. Questa ha differenti **layout** (disposizione dei tasti) dipendentemente dalla lingua per cui è stata realizzata. Si parla infatti di tastiere italiane, statunitensi (US english), inglesi (UK english), francesi, ... In tali tastiere infatti, sebbene la maggior parte dei tasti siano disposti nello stesso modo (secondo l’ormai diffuso standard “QWERTY” – le prime 5 lettere alfabetiche della tastiera da sinistra verso destra), alcuni simboli sono assegnati a tasti diversi (ad esempio i simboli “, £, \$, %, ...) e addirittura in alcuni layout sono presenti dei simboli che in altri layout non esistono (è, é, ç, ò, à, ...). Anche la posizione di alcuni tasti particolari può cambiare da un tipo di tastiera all’altro.

layout

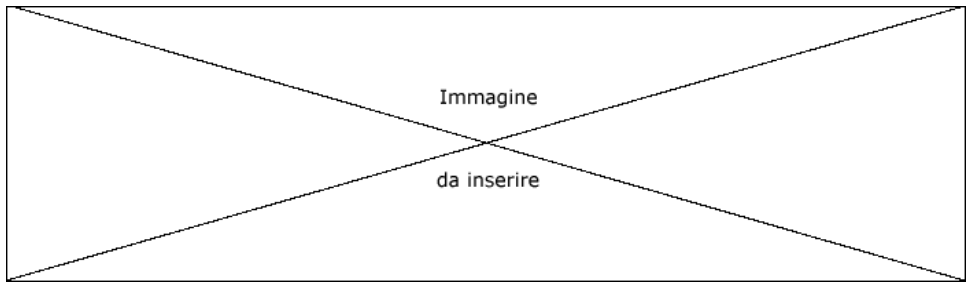


Figura 1.26: Esempio di layout di tastiera.

In tab. 1.8 è riportato un elenco dei tasti particolari con i loro equivalenti nei layout più utilizzati (statunitense e italiano). Nel testo sarà fatto sempre riferimento alla versione riportata nelle colonna “Tasto” di tale tabella.

Tasto	Equivalenze	Tasto	Equivalenze
Return	Invio Enter	Del	Canc
Spacebar	barra spaziatrice	Home	Inizio ↶
PrtScn	Print Stamp	End	Fine
SysReq	RSist	PgUp	PgSù Pag↑
Scroll Lock	Bloc Scorr	PgDn	PgGiù Pag↓
Num Lock	Bloc Num	Tab	→ ⇄
Caps Lock	Bloc Maiu	Shift	↑
Pause	Pausa	BackSpace	←
Break	Interr		

Tabella 1.8: Equivalenze di alcuni tasti sulla tastiera.

⁴³inserire carattere dopo carattere l’informazione da inserire.

1.9 Il software

macchina a stati

Un computer ha una caratteristica fondamentale: è una *macchina a stati* che può essere programmata. Per **macchina a stati** si intende che la generazione dei risultati (*output*) dipende, oltre che dai dati di ingresso (*input*), anche dallo stato in cui si trova il sistema al momento dell'elaborazione dell'input stesso, cioè l'elaborazione degli stessi dati di input può portare alla generazione di output diversi poiché il sistema si trova in uno stato diverso.

*software
programma*

L'elaborazione dei dati avviene attraverso l'esecuzione del **software**⁴⁴, ovvero dei **programmi**. Un **programma** (o algoritmo⁴⁵) è una sequenza ordinata di istruzioni scritte in un linguaggio comprensibile dall'esecutore. L'unico "linguaggio" che la CPU è in grado di comprendere è il *linguaggio macchina*⁴⁶ (o *codice macchina*), ovvero un insieme di istruzioni molto semplici formate soltanto da gruppi di cifre binarie. Poiché risulta molto difficile scrivere i programmi direttamente in linguaggio macchina, questi vengono scritti in linguaggi che si avvicinano a quello naturale: i **linguaggi di programmazione**. I linguaggi di programmazione sono costituiti da insiemi di regole sintattiche diverse da linguaggio a linguaggio ed ognuno di essi è caratterizzato da un determinato livello di astrazione che questo fornisce al programmatore: più il livello di astrazione è elevato (linguaggio di alto livello) e più i dettagli del sistema sono nascosti al programmatore, più è basso (linguaggio di basso livello) e più dettagliate sono le potenzialità che il linguaggio mette a disposizione del programmatore. Esistono così il linguaggio C, il C++, l'Assembly, il Java, il Pascal, il Fortran, l'Eiffel, il Perl, il Python, il Ruby, ... Poiché il C è il linguaggio con il quale è stata scritta la quasi totalità dei sistemi operativi Unix-like, esso è il linguaggio di riferimento per tali sistemi, ovvero le funzioni di libreria che si interfacciano con il sistema operativo sono a disposizione dei programmatori direttamente in tale linguaggio.

*linguaggi di program-
mazione*

applicazioni

driver

Il software si può suddividere in due grossi filoni: le *applicazioni* ed i *driver*. Le **applicazioni** sono programmi che si interfacciano con l'utente e gli permettono di effettuare operazioni di qualunque tipo come scrivere documenti, creare immagini, riprodurre musica, navigare sul web, ... I **driver** sono programmi di più basso livello (generalmente non interagiscono direttamente con l'utente) che gestiscono la comunicazione del sistema con i vari dispositivi, ovvero con i relativi controller.

1.9.1 Compilatore ed interprete

file sorgenti

interprete

compilatore

I programmi, scritti in uno specifico linguaggio di programmazione, sono memorizzati all'interno di file, detti **file sorgenti** (*source file*), che non sono direttamente comprensibili dall'elaboratore, ma devono essere "tradotti" in linguaggio macchina. Esistono essenzialmente due tipologie di traduzione dei file sorgenti: l'*interpretazione* e la *compilazione*. Un file *sorgente* può essere interpretato, ovvero deve esistere un programma scritto in linguaggio macchina, l'**interprete** (*interpreter*), che traduce in linguaggio macchina, una dopo l'altra, tutte le istruzioni presenti nel sorgente (programma da tradurre) passandole via via all'elaboratore che le esegue. Un file *sorgente* può essere compilato, ovvero deve esistere un programma scritto in linguaggio macchina, il **compilatore** (*compiler*), che traduce in linguaggio macchina tutte le istruzioni presenti nel sorgente, memorizzandole all'interno di un file *eseguibile*. La differenza è sostanziale: ogni volta che si desidera far eseguire un programma interpretato è necessario far eseguire anche l'interprete a cui si fa quindi tradurre il sorgente sul momento, al volo (*on-the-fly*); se si desidera invece far eseguire un programma compilato, già tradotto quindi in linguaggio macchina dal compilatore, non è necessario far eseguire insieme ad esso anche il compilatore, poiché il programma compilato è comprensibile direttamente dalla CPU (è in linguaggio macchina): il compilatore è necessario soltanto nella fase

⁴⁴il termine *software* (la parte soffice, morbida) deriva come contrapposizione all'*hardware* (la parte dura), cioè alla circuiteria elettronica che compone il computer (v. sez. 1.5).

⁴⁵dal nome del matematico arabo Al-Khwarizmi (750-850 d.C.).

⁴⁶il linguaggio macchina comprensibile dall'elaboratore dipende dalla CPU dello stesso: ogni tipo di CPU è in grado di comprendere un proprio linguaggio macchina.

di traduzione (compilazione) dei file sorgenti del programma, che avviene una tantum. Per questo l'esecuzione di un programma compilato è notevolmente più veloce rispetto a quella dello stesso programma interpretato: nel tempo di esecuzione di un programma interpretato viene ovviamente incluso anche il tempo di traduzione di ogni istruzione, mentre con il programma compilato non c'è traduzione durante la fase di esecuzione.

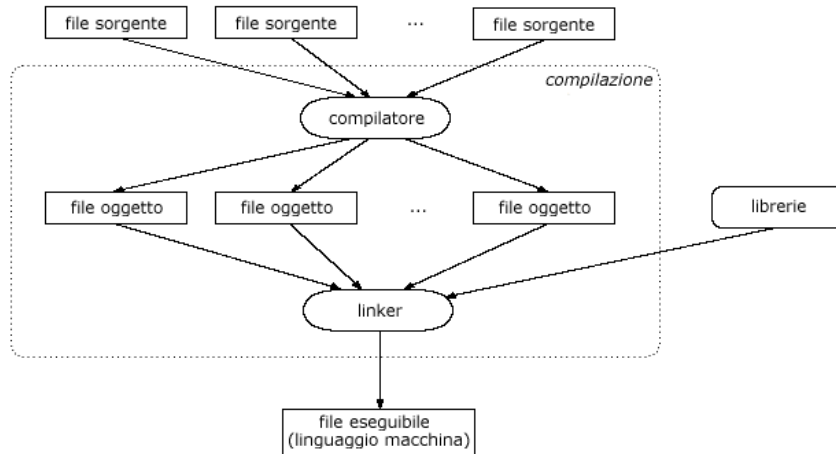


Figura 1.27: Schematizzazione della realizzazione di un file eseguibile tramite compilazione.

Come mostrato in fig. 1.27, più file sorgenti possono realizzare un unico file compilato (in linguaggio macchina). La fase di compilazione si compone della compilazione vera e propria che porta alla creazione di file oggetto e della fase di linking che, tramite un apposito programma eseguibile, il *linker*, collega i file oggetto alle routine presenti nelle librerie, creando il file compilato (per i dettagli sul meccanismo di compilazione dei file, si vedano i testi relativi ai compilatori degli specifici linguaggi di programmazione).

La stessa interfaccia a caratteri (la *shell*⁴⁷) è un *interprete* di comandi. È possibile scrivere quindi dei programmi con la sintassi compresa da tale interprete. Tali programmi sono denominati **script**.

script

1.9.2 L'input, l'output e l'exit status

Un'applicazione in genere si aspetta dei dati di ingresso, ovvero ciò che è chiamato **input**, che possono essere quelli sui quali eseguirà le operazioni oppure possono servire per fornirgli delle indicazioni sulla modalità di esecuzione delle operazioni stesse. La stessa applicazione fornirà poi dei risultati, ovvero l'**output**, che potrà essere visualizzato sullo schermo, scritto all'interno di un file, inviato in stampa alla stampante, ...

input

output

Quando un programma termina la sua esecuzione, esso ritorna un valore di uscita, detto **exit status**, cioè un valore numerico di un byte, che generalmente indica se il programma ha svolto correttamente o meno il suo compito secondo la convenzione di considerare terminata con successo (*successful*) l'esecuzione di un programma che ritorna un *exit status* uguale a 0 e terminata con un errore altrimenti.

exit status

1.9.3 Le versioni

In genere il software viene realizzato, distribuito ed utilizzato. Successivamente può essere realizzato un aggiornamento del software stesso che ne migliora degli aspetti o ne aggiunge delle funzionalità. Ogni realizzazione del software è identificata dalla **release** (o versione): un identificatore costituito da un insieme di numeri che vengono

release

⁴⁷la shell è trattata nel cap. 7.

incrementati man mano che lo stesso software viene aggiornato. In genere il *versioning* (la gestione delle *release*) del software è basato sull'uso della seguente notazione

MajorNumber.MinorNumber.RevisionNumber[-*BuildNumber*]

dove

MajorNumber

è il numero che identifica la release e viene incrementato quando il software subisce una grossa modifica strutturale.

MinorNumber

è il numero che identifica l'aggiornamento della release e viene incrementato quando il software subisce una qualunque modifica degna di nota.

RevisionNumber

è il numero che identifica la revisione della release e viene incrementato quando il software subisce correzioni minime o piccole patch.

BuildNumber

(talvolta indicato col nome *PatchNumber*) è l'eventuale valore che identifica la ricompilazione della release e viene incrementato quando il software viene ricompilato con modifiche alle opzioni/parametri di compilazione.

È naturale che l'incremento di un valore implica l'azzeramento di quelli meno significativi rispetto ad esso (quelli che gli stanno sulla destra).

1.10 Il kernel ed i processi

kernel

Il **kernel** è il cuore del sistema operativo ovvero un software che permette il funzionamento del sistema mettendo a disposizione degli altri programmi un insieme minimo di *system call* (routine, funzioni) per colloquiare con l'hardware.

moduli

Il kernel di GNU/Linux è costituito soltanto da un file, ma ha la possibilità di appoggiarsi a file di contorno detti **moduli** per la gestione di componenti specifici che devono poter essere attivati e disattivati durante il normale funzionamento della macchina.

processo

Un **processo** è l'istanza di esecuzione di un determinato *programma*. Per poter fare iniziare l'esecuzione di un programma, si ordina al sistema di eseguire il file (eseguibile) che lo contiene, specificandolo sulla riga di comando. È bene sottolineare il fatto che per lanciare l'esecuzione del file, o più brevemente lanciare (o avviare) un file, è necessario che tale file sia identificato come file eseguibile⁴⁸. Dunque, l'avvio di tale file comporta la creazione di un processo che esegue le istruzioni indicate dal programma contenuto nel file stesso.

Poiché, come sarà illustrato nel cap. 6, più processi possono girare sullo stesso sistema, il kernel gestisce l'utilizzo della CPU da parte dei processi in modo tale che questa possa essere utilizzata da ogni processo, a rotazione, per brevi intervalli di tempo (time sharing).

I concetti qui illustrati saranno trattati più in dettaglio nel cap. 6.

1.11 La notifica degli eventi

polling

interrupt

Esistono due meccanismi per mezzo dei quali un sistema può accorgersi del verificarsi di eventi particolari: il *polling* e l'*interrupt*. Il meccanismo di **polling** consiste nel controllare ad intervalli regolari se un certo evento si è verificato o meno. Il meccanismo di **interrupt** invece si basa sul fatto che è l'evento stesso a segnalare la sua occorrenza. È evidente che il meccanismo di interrupt è migliore per quanto riguarda il destinatario: questo viene avvertito soltanto in corrispondenza del verificarsi dell'evento stesso senza dover sprecare parte del proprio tempo nel controllare periodicamente se l'evento si è

⁴⁸Le proprietà dei file sono trattate nel cap. 3.

verificato o meno. Ciò però comporta una gestione più complessa del meccanismo di supporto del sistema di notifica stesso.

L'interrupt viene utilizzato sia per l'hardware che per il software: un **interrupt hardware** è un segnale elettrico che arriva ad un dispositivo per notificare l'avvenimento di un evento, mentre un **interrupt software** è un meccanismo di notifica dell'occorrenza di un evento particolare ad un processo tramite meccanismi software.

interrupt hardware

interrupt software

1.12 Le reti di computer

Due o più computer possono essere collegati tra loro per mezzo di opportune interfacce, per formare quella che viene denominata una **rete** (*network*) di computer. Le **interfacce di rete**, dette anche *schede di rete* o **NIC** (Network Interface Card), sono delle schede che si inseriscono in appositi connettori situati sulla motherboard di un computer ed in genere vengono collegate tra loro per mezzo di appositi cavi, detti appunto **cavi di rete** (con apposite NIC è possibile collegare i computer senza nessun bisogno di cavi, utilizzando segnali radio).

rete

interfacce di rete

NIC

I segnali inviati da un computer ad un altro, per mezzo del collegamento di rete, devono rispettare un determinato insieme di regole, ovvero uno specifico **protocollo** di comunicazione. Ogni tipo di comunicazione ha il proprio protocollo, ad esempio IP (Internet Protocol), TCP (Transfer Control Protocol), HTTP (Hyper Text Transfer Protocol), ...

Le reti si distinguono in *reti locali* o **LAN** (Local Area Network), cioè reti relativamente circoscritte, la cui estensione totale è contenuta all'interno di un unico edificio, e *reti estese* o **WAN** (Wide Area Network) che possono avere estensione anche a livello mondiale, come *Internet*. Queste si differenziano sia per il tipo di interconnessione che per il protocollo utilizzato nella comunicazione.

LAN

WAN

Una rete di computer permette agli utenti di poter comunicare tra loro e di utilizzare servizi **remoti** (per "remoto" si intende che il servizio è presente su un computer diverso da quello sul quale sta fisicamente lavorando l'utente).

remoti

Le reti saranno trattate in dettaglio nella parte II.

1.13 Le distribuzioni di GNU/Linux

GNU/Linux, essendo un sistema non proprietario, non è prodotto da un'azienda, ma esistono varie aziende che compilano i sorgenti del sistema operativo e personalizzano la configurazione del sistema (aspetto dell'interfaccia grafica, applicazioni particolari, ...) producendo le cosiddette **distribuzioni** (o più semplicemente *distro*) di GNU/Linux, ovvero un insieme di CD per l'installazione di GNU/Linux (oltre alla relativa documentazione). In genere le distribuzioni di GNU/Linux contengono, oltre al sistema operativo e del software libero⁴⁹, anche del software non libero (proprietario dell'azienda che ha realizzato la distribuzione).

distribuzioni

Le distribuzioni sono acquistabili presso le aziende che le realizzano e le vendono. La parte fondamentale delle distribuzioni (escluso il software proprietario), che comunque permette di installare il sistema operativo di base con un corredo abbastanza consistente di software, è in genere reperibile gratuitamente su *Internet* o assieme all'acquisto di riviste del settore.

Al momento esistono circa un centinaio di distribuzioni di GNU/Linux e di seguito sono elencate, in ordine alfabetico, alcune tra quelle più note.

Debian Il progetto *Debian* è un'associazione di persone che ha come scopo comune la creazione di un sistema operativo libero basato su GNU/Linux. Debian si presenta con più di 8700 pacchetti (v. sez. 1.15) tutti completamente gratuiti, mantenuti unicamente dal lavoro di numerosi volontari (v. <http://www.debian.org>).

⁴⁹il software libero è trattato in app. A.

Madeinlinux Si tratta di una distribuzione interamente italiana che tiene conto delle esigenze degli utenti italiani. I suoi principali obiettivi sono la facilità d'uso e la sicurezza. È prodotta da *MLX s.r.l.* che ha sede a Milano (Italia) (v. <http://www.madeinlinux.com>).

Mandrake *MandrakeSoft* è un'azienda nata verso la fine del 1998 con sedi principali in Altadena (California), Parigi (Francia) e Montreal (Quebec). Il suo scopo è quello di costituire un'interfaccia affidabile tra gli utenti delle tecnologie dell'informazione e gli operatori dell'open source. Il progetto *Mandrake Linux*, creato nel 1998 come derivazione di una distribuzione Red Hat, ha lo scopo di rendere Linux più facile da usare per tutti. È stata una delle prime aziende a fornire delle distribuzioni GNU/Linux user friendly sia nell'utilizzo delle applicazioni che nella procedura di installazione (v. <http://www.mandrakesoft.com>).

Red Hat *Red Hat Inc.* è un'azienda fondata nel 1994 con sede in Raleigh (North Carolina) che fornisce soluzioni open source. Lo staff di sviluppatori include 6 dei più accreditati sviluppatori del kernel Linux. Lo scopo che regola il suo business è quello di portare Linux e la tecnologia open source nelle aziende. È stata una delle prime aziende a fornire delle distribuzioni di GNU/Linux che è, ad oggi, la più diffusa. Anch'essa si presenta con applicazioni user friendly ed un'installazione molto semplice (v. <http://www.redhat.com>).

Slackware Fin dalla sua prima apparizione (Aprile 1993), tale distribuzione è caratterizzata da un'elevata semplicità e stabilità e per questo è divenuta una tra le più popolari distribuzioni di GNU/Linux (v. <http://www.slackware.com>).

SuSE *SuSE Linux AG* è un'azienda la cui sede principale è a Norimberga (Germania). È uno dei maggiori fornitori di soluzioni basate sul sistema operativo open source GNU/Linux (v. <http://www.suse.com>).

Esistono inoltre delle *minidistribuzioni* che non sono altro che delle versioni ridottissime del sistema operativo che possono essere contenute in un dischetto. Ne è un esempio **muLinux** (v. <http://www.sunsite.dk/mulinux>).

Oltre a quelle indicate, ci sono le cosiddette distribuzioni *live-CD* che permettono di utilizzare il sistema operativo GNU/Linux senza installare niente sull'hard disk. Il CD stesso sul quale sono distribuite è bootabile e, se inserito nel PC, all'accensione della macchina avvia automaticamente un sistema GNU/Linux che può essere tranquillamente utilizzato. Ne sono un esempio **Knoppix** (v. <http://www.knoppix.org>) e **DemoLinux** (v. http://www.xplinux.biz/demo_linux.htm).

Un elenco delle distribuzioni di GNU/Linux disponibili lo si può trovare su <http://www.linuxhq.com/dist.html> e per un confronto approfondito tra le varie distribuzioni si può vedere <http://www.distrowatch.com/>.

1.14 L'installazione

Il sistema operativo GNU/Linux può funzionare, ad oggi, su piattaforme hardware basate su di una grande varietà di CPU: *Intel* (X836, X486, PentiumX, Itanium, StrongARM), *AMD* (K5, K6, Duron, Athlon), Power PC, *Alpha*, *Sun* (SPARC e UltraSPARC), *Motorola* 68000, ...; ed inoltre, vista la sua flessibilità, è utilizzato per gestire di tutto, da computer palmari a cluster scientifici e supercomputer, a router o firewall, passando da tutto ciò che sta in mezzo (basti pensare che esistono versioni di GNU/Linux anche per console da gioco come *Sony Playstation* e *Microsoft XBox*).

La scelta della distribuzione da installare è dettata dai gusti personali e dal tipo di utilizzo del sistema, sebbene al momento le distribuzioni più utilizzate siano *Mandrake* e *Red Hat*.

Prima di installare GNU/Linux è opportuno creare un'apposita partizione sul disco sul quale il sistema deve essere installato: ogni sistema operativo viene generalmente

installato in una partizione diversa del disco. Come descritto in sez. 1.6.4, il partizionamento di un disco in genere distrugge le eventuali informazioni già presenti sullo stesso, ma esistono anche programmi che riescono ad effettuare il ridimensionamento delle partizioni su un disco che contiene già delle informazioni, senza perderle.

L'installazione di un sistema GNU/Linux avviene in genere mediante l'inserimento del primo CD della distribuzione scelta nel lettore CD e avviando il PC. In questo modo verrà avviato il programma di installazione presente sul CD, che varia da una distribuzione all'altra, ma le caratteristiche sono essenzialmente le stesse per tutti.⁵⁰ Per i dettagli relativi all'installazione del sistema operativo è opportuno riferirsi ai manuali relativi alla distribuzione considerata.

Prima di installare un sistema GNU/Linux è necessario stabilire quale sarà l'utilizzo del sistema che si va ad installare: quali servizi deve fornire il nuovo sistema al suo avvio (DHCP, HTTP, DNS, NFS, LDAP, NIS, ...) ⁵¹, se la macchina sulla quale si installa GNU/Linux è connessa ad una rete di computer e se dovrà effettuare l'autenticazione degli utenti appoggiandosi ad un altro server, il livello di sicurezza del sistema (se si tratta di una macchina esposta su di un canale insicuro – come *Internet* – è opportuno impostare delle politiche di sicurezza più restrittive rispetto a quelle nel caso in cui la macchina si trovi in una rete locale sicura) e quali sono le applicazioni che saranno utilizzate su tale macchina. Il programma di installazione di GNU/Linux in genere effettua

- l'impostazione del tipo di tastiera, del suo layout (inglese, italiana, ...) e del tipo di mouse;⁵²
- il partizionamento del disco e la formattazione⁵³ dei filesystem;
- l'impostazione della data/ora del sistema;
- la configurazione dell'interfaccia di rete (indirizzo IP, subnet, default gateway, DNS),⁵⁴
- la password da assegnare al *superuser* (utente con username *root*);⁵⁵
- l'eventuale creazione di altri utenti (non *superuser*);
- la configurazione per l'autenticazione degli utenti che richiedono l'accesso al sistema;
- la configurazione dell'avvio del sistema (boot loader);⁵⁶
- la configurazione di sicurezza del firewall per la sicurezza del sistema;⁵⁷
- la creazione di un dischetto di avvio (di emergenza);
- l'installazione del software sulla macchina (i servizi e le applicazioni da installare);

Una volta terminata l'installazione, al successivo riavvio del PC si avrà la possibilità di accedere al sistema GNU/Linux.

Nel caso in cui non si riesca ad avviare la macchina da un lettore CD, è possibile creare un dischetto di avvio attraverso una console DOS. Il primo CD delle distribuzioni di GNU/Linux in genere contiene la directory **images** (o **disks**) che contiene il file per l'avvio del programma di installazione di GNU/Linux da copiare su un dischetto. Per copiare tale file su un dischetto si deve utilizzare il comando **rawrite.exe** da una

⁵⁰l'avvio automatico del CD avviene se l'unità di lettura dei CD è configurata per l'avvio (boot) del sistema.

⁵¹i servizi elencati fanno parte dei servizi di rete che sono trattati nel cap. 16.

⁵²le periferiche saranno trattate nel cap. 3.

⁵³v. cap. 3.

⁵⁴per i dettagli v. cap. 16.

⁵⁵per i dettagli relativi agli utenti v. cap. 5.

⁵⁶per i dettagli v. cap. 2.

⁵⁷per i dettagli v. cap. 25.

console DOS (o la sua versione con interfaccia grafica **rawritewin.exe**). Con il comando (supponendo di aver copiato il file **rawrite.exe** in **C:**)

```
C:\>RAWRITE
```

e seguendo le istruzioni visualizzate di seguito, si prepara un dischetto da inserire all'avvio del PC nell'apposito lettore, che lancerà il programma di installazione di GNU/Linux.⁵⁸

Allo stesso risultato si può arrivare disponendo di una macchina sulla quale è in esecuzione un sistema operativo Unix-like, con il comando

```
# dd if=path_file_immagine of=/dev/fd0
```

dove *path_file_immagine* è il file (completo di path) immagine da copiare nel dischetto.

L'installazione può essere effettuata anche utilizzando il collegamento di rete, tramite HTTP, FTP o NFS⁵⁹, o tramite connessione attraverso la porta seriale o parallela del PC.

Comunque, nei CD di installazione delle varie distribuzioni è sempre contenuto un file **README** (o qualcosa di simile) che contiene le indicazioni necessarie per avviare la procedura di installazione del sistema operativo nei vari casi.

1.15 I pacchetti

Per installare del software su un sistema GNU/Linux è sufficiente copiare i file necessari all'applicazione desiderata, mantenendone la relativa gerarchia all'interno dell'albero delle directory. Poiché è piuttosto scomodo reperire e trasportare tutti i file necessari ad un'applicazione, l'insieme di tali file è generalmente raccolto all'interno di un altro file che costituisce un **pacchetto** di installazione. Quindi, per ogni applicazione esisterà un pacchetto di installazione, ovvero un file, generalmente in formato compresso (per ridurre al massimo le dimensioni del pacchetto stesso), che contiene l'insieme dei file che costituiscono l'applicazione stessa.

I pacchetti possono essere di vari tipi. Il più datato e più semplice è costituito da un file compresso con estensione⁶⁰ **tar.gz** o **tgz**. Tale file contiene semplicemente tutto l'albero delle directory che compone l'applicazione da esso fornita. Esistono anche pacchetti specifici per le varie distribuzioni (per citare i più famosi, file con estensione **deb** o **rpm**) che, oltre a contenere l'albero dei file e delle directory che costituiscono l'applicazione, contengono informazioni con le quali viene aggiornato un *database*⁶¹ che tiene conto dei pacchetti (e delle relative applicazioni) installati sul sistema. In questo modo è più facile per l'amministratore del sistema gestire l'installazione, l'aggiornamento e l'eventuale rimozione dei vari pacchetti.

La gestione dei pacchetti è trattata in dettaglio nel cap. 14.

1.16 Reperibilità della documentazione

La prima fonte di informazioni è costituita dalla documentazione fornita assieme alla distribuzione di GNU/Linux considerata. Le applicazioni sono generalmente accompagnate dalla relativa documentazione contenuta in file spesso in formato testo o HTML (HyperText Markup Language), che generalmente vengono raccolti all'interno della directory **/usr/share/doc** (o simili).⁶²

Per quanto riguarda i comandi del sistema, essi accettano sempre un argomento sulla riga di comando che indica di visualizzare un aiuto sommario del comando stesso, ovvero

⁵⁸questo avviene se il lettore dei dischetti è configurato per l'avvio della macchina.

⁵⁹i protocolli HTTP, FTP e NFS saranno trattati nel cap. 19.

⁶⁰l'estensione di un file è costituita dalla parte finale del suo nome dopo un punto '.'.

⁶¹un *database* è un insieme di dati organizzati secondo una particolare struttura che, generalmente, fornisce dei meccanismi per la ricerca delle informazioni al suo interno.

⁶²il contenuto delle pagine HTML è visualizzabile con un *browser* come Mozilla (v. cap. 13).

un elenco di tutti i possibili parametri comprensibili dal comando, ognuno accompagnato da una breve descrizione. L'aiuto in genere viene visualizzato specificando di seguito al comando l'argomento (opzione) `-h` o `--help`.

Ad esempio, il comando

```
$ date --help
```

visualizza un aiuto sommario del comando `date`.

Inoltre, nei sistemi Unix-like è presente un meccanismo per la gestione della documentazione relativa ai programmi, raggiungibile con l'uso dei comandi `man` e `info`.

Il comando `man` visualizza delle "pagine di manuale" o **man page**, relative ad un argomento. La documentazione relativa alle *man page* è organizzata in sezioni, ognuna delle quali si riferisce ad una classe di argomenti. Le sezioni sono le seguenti

- 1 Programmi eseguibili o comandi di shell;
- 2 System call (chiamate a routine del kernel);
- 3 Library call (chiamate a routine di libreria di sistema - `glibc`);⁶³
- 4 File speciali;
- 5 Formati di file e convenzioni;
- 6 Giochi;
- 7 Pacchetti;
- 8 Amministrazione del sistema;
- 9 Routine del kernel;
- n Interprete Tcl/Tk.

In genere si fa riferimento agli argomenti trattati nelle *man page* con la notazione seguente:

argument (section)

dove

argument

è l'argomento di cui si desidera ottenere la documentazione;

section è il numero della sezione che contiene la documentazione relativa all'argomento *argument* in questione.

Ad esempio, la *man page* del comando `ls` è riferita come `ls(1)`, poiché `ls` è un programma eseguibile, ovvero appartiene alla sezione 1. Le *man page* relative ad un argomento sono visualizzate per mezzo del comando `man`

Comando: `man`

Path: `/usr/bin/man`

SINTASSI

`$ man [option] [section] argument`

DESCRIZIONE

option è l'insieme (opzionale) delle opzioni con cui lanciare il comando `man`. Alcune tra le più comuni sono le seguenti:

⁶³sono le librerie realizzate da GNU, che costituiscono le routine richiamabili da un programma per effettuare operazioni sul sistema: lettura/scrittura sui file, operazioni matematico-logiche, ...

- C *config_file*
specifica il file di configurazione da utilizzare. Se non è specificato viene utilizzato il file `/etc/man.config` (man page `man.config(5)`);
- M *path*
specifica l'elenco delle directory, separate dal carattere ':', in cui cercare le man page;
- P *pager*
specifica il programma da utilizzare come visualizzatore della man page (il comando di visualizzazione di default è `/usr/bin/less -isr`);
- a
indica a `man` di visualizzare tutte le man page relative alla ricerca effettuata (*section* e *argument*), anziché soltanto la prima trovata;
- h
visualizza un aiuto in forma compatta del comando `man`;
- K
ricerca *argument* all'interno del contenuto di tutte le man page (o quelle relative alla sezione *section*);

Per l'elenco di tutte le possibili opzioni si veda la man page relativa (`man(1)`);
section è il valore (opzionale) che identifica la sezione a cui limitare la ricerca di *argument*;
argument è l'argomento da ricercare;

Alcuni argomenti sono trattati in più sezioni, dipendentemente dal contesto: per esempio esistono due man page che si riferiscono a *mount*, una in corrispondenza della chiamata di libreria C e l'altra per il comando di gestione del sistema. Quindi per visualizzare la man page relativa a *mount*, riguardante la gestione del sistema, si può digitare il comando

```
$ man 8 mount
```

che visualizzerà una prima schermata di informazioni analoga a quella riportata di seguito

```

MOUNT(8)                                Linux Programmer's Manual                MOUNT(8)

NAME
    mount - mount a file system

SYNOPSIS
    mount [-lhV]

    mount -a [-fFnrsvw] [-t vfstype] [-O optlist]
    mount [-fnrsvw] [-o options [...]] device | dir
    mount [-fnrsvw] [-t vfstype] [-o options] device dir

DESCRIPTION
    All files accessible in a Unix system are arranged in one big tree, the
    file hierarchy, rooted at /. These files can be spread out over sev-
    eral devices. The mount command serves to attach the file system found
    on some device to the big file tree. Conversely, the umount(8) command
    will detach it again.

    The standard form of the mount command, is
        mount -t type device dir
    This tells the kernel to attach the file system found on device (which
    is of type type) at the directory dir. The previous contents (if any)
    :
```

che può essere scorsa con i tasti `↑`, `↓`, `PgUp`, `PgDn`, ... dipendentemente dal program-
 ma utilizzato per la visualizzazione (*pager*) (quello di default è `less`⁶⁴).

⁶⁴una descrizione dei comandi di `less` è riportata in sez. 5.10.

La versione di GNU/Linux di **man** utilizza diverse variabili di ambiente⁶⁵ tra cui **MANPAGER** e **MANPATH** che specificano rispettivamente il comando da utilizzare per la visualizzazione dei documenti (*pager*) ed il percorso (*path*) da utilizzare per la ricerca nel filesystem dei file relativi ai vari argomenti.

Il comando **info** (`man page info(1)`) permette la visualizzazione di altra documentazione costituita da **info page** organizzate ad albero: l'argomento principale è "l'indice" delle info page ed elenca gli argomenti in esso contenuti; tramite l'interfaccia messa a disposizione dal programma è possibile selezionare l'argomento di interesse, il quale mostrerà delle informazioni e gli argomenti in esso contenuti e così via. *info page*

Comando: **info**

Path: `/usr/bin/info`

SINTASSI

\$ info [option] [arg [node [...]]]

DESCRIZIONE

option specifica il comportamento del comando **info**. Può assumere i seguenti valori:

- apropos=string**
ricerca la stringa *string* in tutte le info page;
- d dir | --directory=dir**
aggiunge la directory *dir* al percorso di ricerca delle info page (variabile di ambiente **INFOPATH**);
- dribble=filename**
memorizza le combinazioni di tasti dell'utente nel file *filename*;
- f filename | --file=filename**
specifica di visualizzare il file *filename*;
- h | --help**
visualizza un aiuto sommario del comando **info**;
- index-search=string**
visualizza la info page relativa alla voce dell'indice *string*;
- n nodename | --node=nodename**
specifica i nodi della prima info page visualizzata;
- o filename | --output=filename**
salva il contenuto delle info page selezionate nel file *filename*;
- R | --raw-escapes**
indica di non rimuovere le sequenze di escape ANSI dalle info page;
- restore=filename**
indica di leggere le combinazioni di tasti dal file *filename*;
- O | --show-options | --usage**
indica di attivare la modalità di opzioni da riga di comando;
- subnodes**
visualizza tutte le voci dell'indice;
- vi-keys**
indica di utilizzare le combinazioni di tasti di **vi** e **less**;
- version**
visualizza la versione di **info**;

arg è l'argomento di cui visualizzare la info page.

node è il nodo relativo all'argomento di cui visualizzare la info page.

Ad esempio, il comando

```
$ info bash
```

visualizzerà una prima schermata di informazioni analoga a quella riportata di seguito

⁶⁵le variabili di ambiente sono trattate nel cap. 7

File: bashref.info, Node: Top, Next: Introduction, Prev: (dir), Up: (dir)

Bash Features

This text is a brief description of the features that are present in the Bash shell.

This is Edition 2.5b, last updated 15 July 2002, of 'The GNU Bash Reference Manual', for 'Bash', Version 2.05b.

Copyright (C) 1991-2002 Free Software Foundation, Inc.

Bash contains features that appear in other popular shells, and some features that only appear in Bash. Some of the shells that Bash has borrowed concepts from are the Bourne Shell ('sh'), the Korn Shell ('ksh'), and the C-shell ('csh' and its successor, 'tcsh'). The following menu breaks the features up into categories based upon which one of these other shells inspired the feature.

This manual is meant as a brief introduction to features found in Bash. The Bash manual page should be used as the definitive reference
 --zz-Info: (bash.info.gz)Top, 68 lines --Top-----
 Welcome to Info version 4.3. Type C-h for help, m for menu item.

Per muoversi tra le info page relative ad un determinato argomento si possono utilizzare i tasti **N** (pagina successiva), **P** (pagina precedente), **Return** (salta al nodo collegato alla parola che si trova correntemente sotto il cursore) e **Backspace** (ritorna al nodo dal quale si era saltati precedentemente). L'elenco completo dei comandi è visualizzato premendo il tasto **H**.

Esistono anche altri comandi che visualizzano delle brevi descrizioni dell'utilizzo dei comandi di sistema: **whatis** (man page **whatis(1)**) e **apropos** (man page **apropos(1)**).

Comando: **whatis**
 Path: **/usr/bin/what**
 SINTASSI
\$ **whatis** *keyword*

DESCRIZIONE

Effettua una ricerca della parola specificata come *keyword* all'interno di un database. Esso visualizza soltanto la descrizione dei comandi che contengono *keyword* come parola intera.

Comando: **apropos**
 Path: **/usr/bin/apropos**
 SINTASSI
\$ **apropos** *keyword*

DESCRIZIONE

Effettua una ricerca della parola specificata come *keyword* all'interno di un database. Esso visualizza la descrizione dei comandi che contengono *keyword* anche come parte di altre parole.

Infine, oltre alla documentazione relativa alla distribuzione utilizzata (cartacea o direttamente on-line sul sito della distribuzione), un'enorme quantità di informazioni è comunque presente su *Internet* (per reperirla si può utilizzare un qualunque motore di ricerca come, ad esempio, *Google*, <http://www.google.com>) a partire dalla *Italian Linux Society* o ILS (<http://www.linux.it>) e dai vari *Linux User Group* (LUG), da *The Linux Documentation Project* (<http://www.tldp.org>) a *The Linux Information Headquarters*

(<http://www.linuxhq.com>) e numerosissimi altri siti che contengono documenti inerenti i vari aspetti del sistema ed *how-to*⁶⁶ che sono dei manuali mirati per argomenti specifici.

È vivamente consigliato di consultare le informazioni fornite con la propria installazione di GNU/Linux per conoscere la sintassi ed i dettagli relativi ai comandi riportati nel testo, poiché le indicazioni fornite potrebbero essere obsolete vista la velocità di sviluppo del sistema. Nel testo comunque, via via che saranno trattati i vari comandi, saranno indicate anche le man page relative.

1.17 Riferimenti

- Unicode e UTF-8
<http://www.unicode.org>
<http://www.cl.cam.ac.uk/~mgk25/unicode.html>
- Linguaggi di programmazione
B. W. Kernighan, D. M. Ritchie, *The C Programming Language*, Prentice Hall 1988
B. Stroustrup, *The C++ Programming Language*, Addison-Wesley
B. Eckel, *Thinking in C++*
B. Eckel, *Thinking in Java*

⁶⁶dalle due parole anglosassoni *how* (come) e *to*, il suo significato può essere espresso con le parole italiane “come fare a ...” o “come fare per ...”.

Capitolo 2

Avvio ed arresto del sistema

“Un viaggio di mille miglia comincia con un solo passo.”

– Lao Tze

In questo capitolo sarà trattato l’avvio dei sistemi basati su architettura *Intel* X386 o derivati, dall’accensione della macchina fino all’accesso al sistema da parte dell’utente (login). Sarà presentata una panoramica sui possibili modi di avviare un sistema GNU/Linux e quindi verrà descritta la procedura di avvio del sistema con un breve accenno alla procedura di accesso all’interfaccia, che verrà trattata nel cap. 5.

2.1 Il boot

All’avvio di un PC si ha quella che in gergo viene chiamata *procedura di boot* (*bootstrap* o semplicemente **boot**) del sistema.

boot

Un bootstrap è un processo che viene effettuato per passi successivi: prima si fanno dei semplici passi, poi, sfruttando questi, se ne fanno altri sempre più complessi. Il termine *bootstrap* deriva dal detto anglosassone “Pulling yourself up by the bootstraps”, cioè “Tirarsi su a partire dai lacci delle scarpe”.

Questa si basa su fattori sia hardware che software secondo lo schema riportato in fig. 2.1 e descritto di seguito.



Figura 2.1: Schema del boot del sistema.

1. L’hardware viene alimentato dalla corrente ed un apposito circuito presente sulla motherboard controlla la funzionalità sommaria degli altri circuiti principali presenti sulla motherboard stessa.

BIOS

La CPU inizia ad eseguire una procedura di inizializzazione la cui prima istruzione si trova all'indirizzo di memoria FFFF0_H , ovvero gli ultimi 16 byte della memoria convenzionale, che contengono un'istruzione di *jump*, che indica alla CPU di eseguire le istruzioni contenute a partire da un'altro indirizzo di memoria. Tale insieme di routine (programmi) è quello che in gergo viene chiamato **BIOS** (Basic Input/Output System) e sono memorizzate in un apposito circuito integrato di sola lettura (ROM o EPROM) della motherboard.

firmware

Generalmente ci si riferisce ai programmi memorizzati nei circuiti integrati (ad accesso in sola lettura) col termine **firmware**, per distinguerli dal *software* che viene generalmente memorizzato in file, sulla memoria di massa.

POST

2. Il BIOS effettua quindi il **POST** (Power-On Self Test), ovvero la verifica del funzionamento di base dell'hardware del sistema. Se durante tale operazione viene trovato un malfunzionamento, la procedura si arresta e vengono emessi una serie di suoni dallo speaker del PC, che indicano, secondo un determinato codice che dipende dal fornitore del BIOS, il tipo di problema che è stato incontrato.

Viene eseguita una procedura di inizializzazione della scheda video (in genere contenuta in memoria a partire dall'indirizzo C000_H), durante la quale è probabile che vengano visualizzate sullo schermo delle informazioni ad essa relative.

Il BIOS ricerca eventuali programmi di inizializzazione (driver) specifici di altri dispositivi e li esegue. Ad esempio, generalmente viene eseguito il driver del controller ATA che si trova in memoria a partire dall'indirizzo C8000_H .

Viene quindi visualizzata la schermata di avvio del BIOS.

Viene controllata la funzionalità della memoria centrale (RAM) e durante il test viene visualizzato sullo schermo la quantità di memoria esaminata.

3. Il BIOS controlla il circuito di gestione degli interrupt (PIC), riconosce gli hard disk ed i dispositivi Plug & Play (se il BIOS supporta questa funzionalità) e visualizza quindi un prospetto della configurazione del sistema.

boot loader

4. Viene avviato il **boot loader** secondo la sequenza sotto riportata
 - (a) Viene considerata la sequenza di avvio dei dischi impostata e ogni disco viene testato per assicurarsi della sua presenza e del fatto che sia effettivamente avviabile (*bootable*) (questa caratteristica viene riconosciuta dai primi byte del suo MBR);

La sequenza con cui vengono scandite le unità a dischi nella fase di boot, può essere impostata dall'utente per mezzo dell'interfaccia messa a disposizione dal fornitore del BIOS, alla quale si accede nei momenti iniziali dell'avvio della macchina in genere premendo il tasto Canc (o Del).

MBP

SBP

- (b) Viene avviato il codice eseguibile contenuto nel MBR del primo disco avviabile, cioè il **MBP** (Master Boot Program);
 - (c) Il MBP controlla la *partition table* ed avvia lo **SBP** (Slave Boot Program), cioè il codice eseguibile contenuto nel primo settore (*boot sector*) della partizione del disco (memoria di massa) contenente il sistema operativo che l'utente desidera avviare (v.fig. 2.2);
 - (d) Lo SBP avvia quindi il processo di caricamento del sistema: nel caso di GNU/Linux procede al caricamento e all'avvio del *kernel* (il nucleo del sistema operativo) che a sua volta lancerà in esecuzione il *processo* iniziale necessario per il funzionamento del sistema.

cold boot

warm boot

Quella precedentemente descritta è la procedura di avvio del sistema, che viene eseguita in seguito ad un **cold boot**, cioè quando il sistema viene avviato da spento, a freddo (*cold*). Nel caso di riavvio del sistema, da software, si parla invece di **warm boot**, ovvero un riavvio a caldo (*warm*), ed anche in tal caso viene eseguita la procedura di avvio del sistema, ma il POST non viene effettuato.

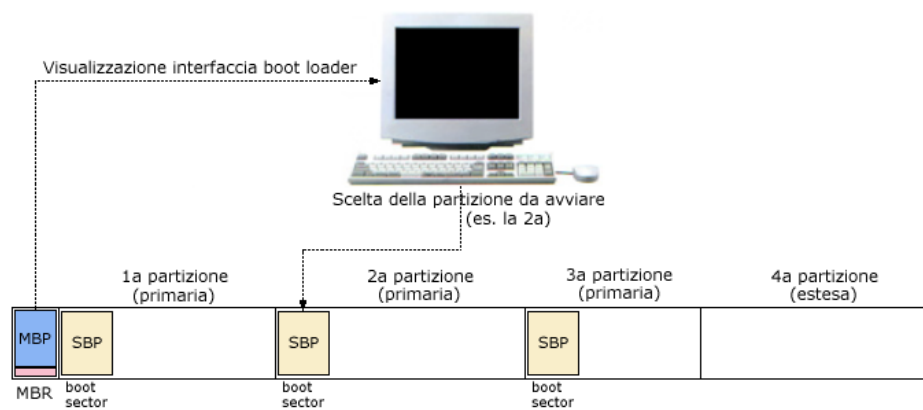


Figura 2.2: Schema dell'avvio del SBP.

2.1.1 Il MBR - Master Boot Record

Il master boot record o **MBR** il primo settore di un disco (cilindro 0, testina 0, settore 1), destinato a contenere le informazioni essenziali per l'avvio del sistema (v. fig. 2.3).

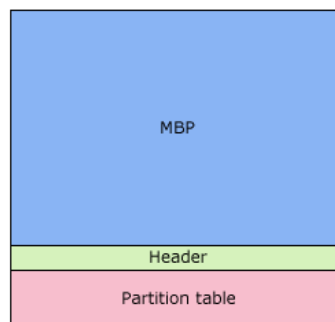


Figura 2.3: Schematizzazione del MBR.

Come ogni settore del disco, il MBR ha una dimensione di 512 byte, di cui i primi 446 sono destinati a contenere il MBP (Master Boot Program), cioè le istruzioni da eseguire quando viene avviato il disco, mentre i rimanenti 66 byte sono suddivisi in un *header* di 2 byte ed in 4 elementi, di 16 byte l'uno, che identificano le partizioni primarie (al massimo 4) in cui può essere suddiviso il disco: la **partition table**. Ogni elemento della partition table contiene le informazioni riportate in tab. 2.1

Offset	Contenuto
0	Indicatore di boot (contiene il valore 80 _H se la partizione è attiva)
1	Prima testina della partizione
2	Primo settore della partizione
3	Primo cilindro della partizione
4	Identificatore del filesystem contenuto nella partizione
5	Ultima testina della partizione
6	Ultimo settore della partizione
7	Ultimo cilindro della partizione
8	Numero di settori prima della partizione
12	Numero di settori che costituiscono la partizione

Tabella 2.1: Partition table entry.

Per com'è stato realizzato il MBR, lo SBP deve essere contenuto nei primi 1024 cilindri, 256 testine e 63 settori del disco (cioè nei primi 8 GiB) e non contiene nessun

riferimento ad eventuali partizioni logiche contenute in una partizione estesa. Quindi lo SBP (Slave Boot Program) che deve essere eseguito per caricare il sistema operativo desiderato, deve essere contenuto in una partizione primaria al di sotto del 1024° cilindro, altrimenti il BIOS non sarà in grado di raggiungerlo. Per ovviare questo problema, nelle versioni recenti dei BIOS è possibile utilizzare la traduzione dell'indirizzamento dei settori (come accennato nel sez. 1.6). Questo permette di oltrepassare il limite sull'indirizzamento dei cilindri suddetto.

2.1.2 Il boot loader

boot loader

Il **boot loader** o *boot manager* è il meccanismo di avvio del sistema operativo e si compone del MBP e del SBP relativo alla partizione considerata. Il cuore del boot loader è il MBP che permette di scegliere la partizione da avviare e quindi la avvia, ovvero avvia lo SBP relativo, mentre lo SBP è caratteristico del meccanismo di avvio del sistema operativo considerato.

Tra i boot loader in grado di far avviare un sistema GNU/Linux, i più diffusi sono GRUB, LILO, LOADLIN e SYSLINUX. In genere, tali boot loader fanno riferimento alla directory `/boot` che, per quanto illustrato precedentemente, è opportuno che risieda in una partizione del disco rigido contenuta al di sotto del 1024° cilindro.

2.1.3 Il boot da dischetto

Il modo più semplice di avviare GNU/Linux senza dover modificare il MBR od i *boot sector* (i primi settori delle partizioni) del disco fisso, è quello di avviare il sistema per mezzo di un dischetto (o floppy disk). A tale scopo, il dischetto deve contenere una copia del kernel di GNU/Linux, che può essere copiato sul dischetto con i comandi seguenti

```
# cd /boot
# cp vmlinuz /dev/fd0
```

In questo modo non si effettua la copia di un file sul dischetto secondo un filesystem, ma si effettua una copia di un unico file sulla periferica specificata (`/dev/fd0`, cioè il dischetto stesso), ovvero il dischetto viene a coincidere con il file in esso contenuto (il kernel di GNU/Linux) e non può contenere nient'altro che questo.¹

Dopo aver copiato il kernel è necessario informarlo su quale sia la partizione dell'hard disk da *montare*² come root directory (`/`).³ Questo lo si può fare alterando il file *immagine* del kernel stesso, con il comando `rdev` (man page `rdev(8)`).

Comando: `rdev`

Path: `/usr/sbin/rdev`

SINTASSI

```
# rdev [option] [image [{value [offset] | [root_device [offset]]}]]
```

DESCRIZIONE

option indica la modalità di funzionamento di `rdev`. Può assumere i seguenti valori:

`-o offset`

specifica il numero di byte dall'inizio del file *image* (*offset*) interessati dal comando. In particolare `rdev` agisce sulla coppia di byte presenti agli *offset* (in decimale) del file *image* riportati in tab. 2.2;

`-h` mostra un aiuto sommario del comando `rdev`;

`-r` imposta la dimensione della memoria centrale (RAM) col valore *value*;

`-R` imposta la modalità di montaggio della root directory col valore *value*;

¹v. cap. 3.

²v. cap. 3.

³il kernel di GNU/Linux accetta un discreto numero di parametri (v. cap. 6).

Offset	Significato
498	Root flags
500	Reserved
502	Reserved
504	RAM Disk Size
506	VGA Mode
508	Root Device
510	Boot Signature

Tabella 2.2: Offset del kernel GNU/Linux.

`-v` imposta la modalità video di default col valore *value*;
image è il file che individua il kernel;
root_device è l'indicazione del dispositivo da montare come *root directory*;

Ad esempio, con il comando

```
# rdev /dev/fd0 /dev/hda2
```

si indica al kernel contenuto nel dischetto (`/dev/fd0`) di montare come root directory la seconda partizione (2) del primo hard disk (`/dev/hda`).

È opportuno che il kernel monti la root directory in sola lettura in modo da poter così effettuare dei controlli sul filesystem. Sarà poi compito del processo di inizializzazione del sistema, successivo al caricamento del kernel, di montare tale partizione per accessi in lettura e scrittura.

Per far montare dal kernel presente sul dischetto (`/dev/fd0`) la root directory in sola lettura, è sufficiente digitare il comando

```
# rdev -R /dev/fd0 1
```

2.1.4 LOADLIN

LOADLIN

LOADLIN⁴ (LOAD LINux) non è un vero e proprio boot loader, ma permette di avviare GNU/Linux dal boot loader del DOS. Esso si compone del file **LOADLIN.EXE** che è un file eseguibile per DOS e deve trovarsi in una partizione con filesystem **FAT** (File Allocation Table), che è riconosciuto dal DOS. Il kernel di GNU/Linux deve essere raggiungibile dal comando **LOADLIN.EXE** e quindi anch'esso deve risiedere in una partizione con filesystem FAT. Per i dettagli relativi al funzionamento di **LOADLIN** è opportuno consultare la documentazione ad esso allegata.

FAT

Per avviare GNU/Linux da DOS è necessario digitare il comando

```
C:\>LOADLIN kernel_image_file_pathname [args]
```

dove

kernel_image_file_pathname

è il nome del file dell'immagine del kernel di GNU/Linux da lanciare, comprensivo di path;

args sono eventuali argomenti (parametri) da passare al kernel di GNU/Linux;

Ad esempio, il seguente comando

```
C:\>LOADLIN C:\VMLINUZ root=/dev/hda2 ro
```

⁴licenza GPL.

indica di avviare il kernel di GNU/Linux, rappresentato dal file `C:\VMLINUZ`, con i parametri `root=/dev/hda2 ro`, che indicano il fatto che si desidera “montare” come root directory la seconda partizione del primo hard disk ATA (`/dev/hda2`) in sola lettura (`ro`).

Eventualmente, prima di avviare `LOADLIN.EXE` si può scaricare la *cache* su disco fisso (per assicurarsi che l’operazione di scrittura sul hard disk sia stata effettivamente effettuata), con il comando DOS

```
C:\>SMARTDRV /C
```

Dunque, non rimane altro che inserire opportunamente i suddetti comandi nei file di avvio del sistema DOS (`C:\AUTOEXEC.BAT` e `C:\CONFIG.SYS`) in modo da permettere l’avvio di più sistemi operativi, direttamente dal menù di avvio del boot loader del DOS.

2.1.5 SYSLINUX

SYSLINUX

SYSLINUX⁵ è un boot loader basato su filesystem FAT (il filesystem nativo del DOS), piuttosto semplice da gestire, tanto che molte delle distribuzioni di GNU/Linux lo utilizzano per creare i dischetti di avvio di GNU/Linux in caso di emergenza. Si compone di un file eseguibile DOS, `SYSLINUX.EXE`, e di un file eseguibile per GNU/Linux, `syslinux`. Entrambe le versioni di questo programma, copiano il file `LDLINUX.SYS` su un dischetto con filesystem FAT e lo rendono avviabile, creando su quest’ultimo un settore di avvio opportuno. Per far ciò è sufficiente digitare (da GNU/Linux)

```
# syslinux /dev/fd0
```

e, analogamente, da DOS

```
C:\>SYSLINUX A:
```

A questo punto non rimane altro che copiare il kernel di GNU/Linux in un file all’interno della root directory del dischetto, oltre ad un file di configurazione `SYSLINUX.CFG`.

Il settore di boot del dischetto avvia il file `LDLINUX.SYS` che legge l’eventuale file `SYSLINUX.CFG` ed interpreta le direttive in esso presenti. Il file di configurazione `SYSLINUX.CFG` è un file di testo (formato DOS o Unix). Un esempio di tale file è riportato di seguito.

```
DEFAULT linux
TIMEOUT 70
DISPLAY INTRO.TXT
PROMPT 1
```

```
F1 VARIE.TXT
```

```
LABEL linux
    KERNEL LINUX
```

```
LABEL hda1
    KERNEL LINUX
    APPEND "root=/dev/hda1 ro"
```

Segue una breve spiegazione delle direttive contenute nell’esempio.

```
DEFAULT linux
```

indica a SYSLINUX la scelta di default identificato dall’etichetta (`LABEL`) “linux”. Se tale etichetta non esistesse, SYSLINUX tenterebbe di avviare il kernel contenuto nel file col nome uguale a quello dell’etichetta, ovvero `LINUX` (le maiuscole e minuscole sono indifferenti, in questo caso, nell’indicazione del nome di un file poiché il filesystem del dischetto è FAT);

⁵licenza GPL (v. <http://syslinux.zytor.com>).

TIMEOUT 70

indica a SYSLINUX di avviare automaticamente la selezione di default se entro 70 decimi di secondo non viene premuto nessun tasto. Se il valore è 0, non si ha alcun avvio automatico: SYSLINUX attende sempre la scelta del sistema da avviare in maniera interattiva;

DISPLAY INTRO.TXT

indica a SYSLINUX di visualizzare il contenuto del file **INTRO.TXT** che si deve trovare nella root directory del dischetto;

PROMPT 1

indica a SYSLINUX di visualizzare un prompt, cioè un invito ad inserire qualcosa, come `'boot:.'`. Per non visualizzare tale prompt è sufficiente sostituire `'1'` con `'0'`;

F1 VARIE.TXT

indica a SYSLINUX di associare la visualizzazione del contenuto del file **VARIE.TXT** alla pressione del tasto F1;

LABEL linux

definisce l'etichetta "linux";

KERNEL LINUX

indica a SYSLINUX di caricare il kernel contenuto nel file **LINUX** che si trova nella root directory del dischetto, quando si seleziona l'etichetta "linux";

LABEL hda1

definisce l'etichetta "hda1";

APPEND "root=/dev/hda1 ro"

indica a SYSLINUX i parametri da passare al kernel; in questo caso si tratta di montare come root directory la prima partizione del primo disco rigido ATA in sola lettura.

Durante l'avvio del sistema si può specificare, di seguito alla visualizzazione dell'eventuale prompt, il nome dell'etichetta relativa all'avvio del sistema desiderato. In riferimento all'esempio precedente, digitando **hda1** si ha l'interpretazione delle direttive che seguono, nel file di configurazione, la riga **LABEL hda1**.

2.1.6 GRUB*GRUB*

GRUB⁶ (GRand Unified Boot loader) è il boot loader utilizzato nelle recenti distribuzioni di GNU/Linux. Le sue caratteristiche principali sono le seguenti

- supporto nativo per il boot di sistemi operativi aderenti alle specifiche Multiboot (GNU/Linux, FreeBSD, NetBSD, OpenBSD, ...);
- supporto per il boot in chain-loading di sistemi operativi non-Multiboot (DOS, Windows, OS/2, ...);
- supporto di vari tipi di filesystem;
- gestione dell'avvio del sistema operativo desiderato attraverso un'interfaccia utente a menù;
- supporto per la modalità LBA: accesso a dischi con capacità superiore a 8 GiB;
- elevata configurabilità;
- possibilità di avvio via rete;
- protezione dei file di configurazione con password;

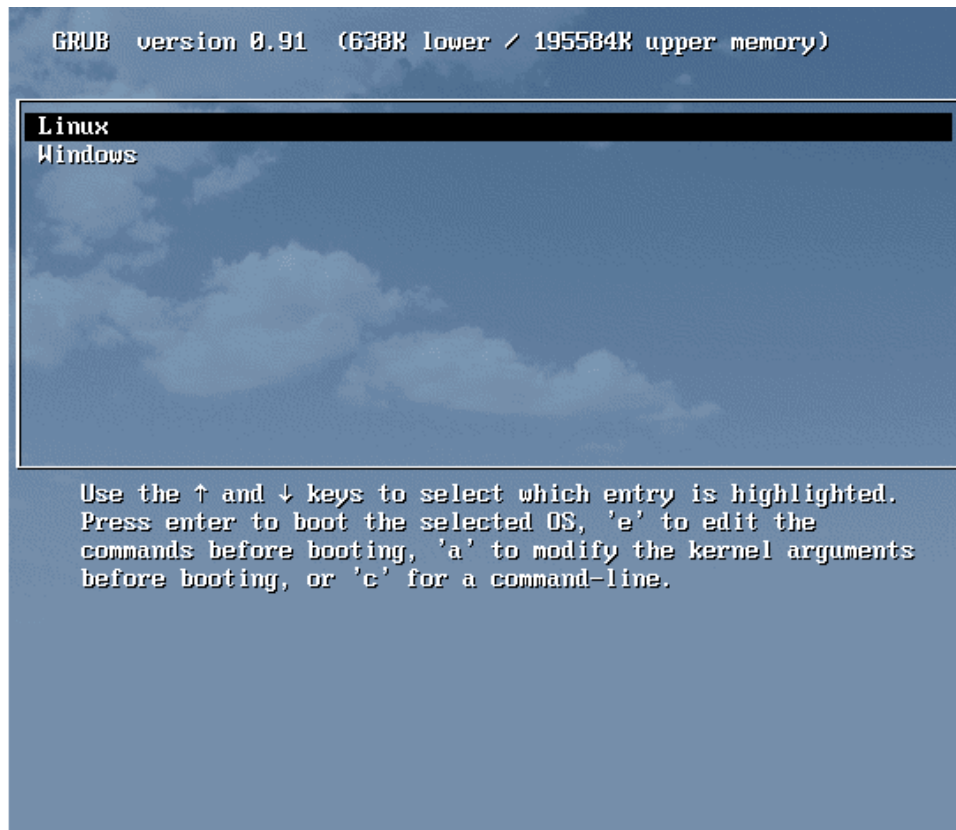


Figura 2.4: Esempio del menù di avvio di GRUB.

All'avvio, GRUB visualizza un elenco dei possibili sistemi operativi presenti sulla macchina tra i quali è possibile scegliere quale avviare (v. fig. 2.4).

Le possibilità di avvio di un sistema operativo da parte di GRUB sono due: in maniera diretta o per mezzo del *chain-load*. I sistemi operativi che aderiscono alle specifiche Multiboot (come Linux, FreeBSD, NetBSD e OpenBSD) possono essere caricati direttamente da GRUB, mentre gli altri sistemi operativi possono essere caricati soltanto tramite il meccanismo di chain-loading, ovvero GRUB chiama il boot loader del sistema specifico per poterlo avviare.

Stadi di avvio

GRUB è costituito da vari stadi (*stage*) di avvio, ognuno dei quali è rappresentato da un file: due stage essenziali (file `stage1` e `stage2`) e degli stage opzionali (file `*stage1_5`⁷).

stage1 Questo è un file essenziale per poter effettuare il boot con GRUB. Tipicamente si trova nel MBR o nel boot sector di una partizione. Poiché un boot sector è grande 512 byte, la dimensione di tale file è esattamente 512 byte. Tutto ciò che fa `stage1` è di caricare il file `stage2` o lo specifico file `*stage1_5` dal disco (`stage1` non gestisce nessun tipo di filesystem).

stage2 Questo è il nucleo di GRUB. Generalmente risiede su un filesystem, ma ciò non è necessario.

***stage1_5**

Questa famiglia di file ha lo scopo di collegare il file `stage1` al file `stage2`. Cioè `*stage1_5` è caricato da `stage1` e, a sua volta, carica `stage2`. Lo

⁶licenza GPL (v. <http://www.gnu.org/software/grub>).

⁷il simbolo asterisco '*' utilizzato all'interno dei nomi dei file sta ad indicare una qualunque sequenza di caratteri (v. cap. 3).

specifico ***stage1_5** rende GRUB capace di gestire un determinato filesystem. In questo modo, si può spostare il file **stage2** in una posizione diversa sul filesystem senza dover necessariamente reinstallare GRUB.

In un sistema GNU/Linux in cui è installato GRUB, una copia dei file relativi ai vari stadi di avvio del sistema si trova in `/usr/lib/grub/i386-pc`.

Identificatori dei dispositivi

Il modo in cui un dispositivo viene identificato da GRUB (nel suo linguaggio di scripting, ovvero nelle direttive contenute nell'apposito file di configurazione - v. più avanti) è illustrato di seguito. Per prima cosa ogni identificatore di dispositivo deve essere racchiuso tra parentesi tonde ed ha il seguente formato:

`(device[,partition])`

dove

device è il dispositivo e può assumere i seguenti valori:

hd0 il primo hard disk;
hd1 il secondo hard disk;
fd0 il primo floppy disk;
 ...

GRUB non fa differenza tra dispositivi ATA (IDE) e SCSI. Generalmente i valori numerici che identificano i dischi ATA sono minori di quelli dei dischi SCSI presenti sulla stessa macchina.

partition

è un numero che identifica la partizione di un determinato disco. Tale valore inizia da 0 ed assume i seguenti significati:

0 la prima partizione del disco;
1 la seconda partizione del disco;
2 la terza partizione del disco;
 ...

Quindi, ad esempio, il file **vmlinuz** che si trova nella directory `/boot` presente nella seconda partizione del primo hard disk potrà essere specificato come `(hd0,1)/boot/vmlinuz`.

Installazione

Se GRUB non è già utilizzato sul sistema e si desidera utilizzarlo come boot loader, è necessario installarlo seguendo i passi sotto riportati.

Innanzitutto si copia il contenuto della directory `/usr/lib/grub/i386-pc` contenente una copia dei file di GRUB nella directory `/boot/grub`. In tale directory si crea inoltre un file di configurazione denominato **grub.conf** ed un symbolic link che si riferisce ad esso denominato **menu.lst**. Questo sarà il file di configurazione di GRUB per l'avvio del sistema (come sarà mostrato in seguito). Adesso non rimane altro che inserire una copia del file `/boot/grub/stage1` nel MBR o nel settore di boot di una partizione dell'hard disk.

Generalmente è una buona idea effettuare una copia (*backup*) del primo settore della partizione sulla quale si intende copiare il **stage1** di GRUB. Ciò non è fondamentale se si installa GRUB sul MBR poiché lo si può ripristinare facilmente lanciando ad esempio il comando DOS

```
C:\>FDISK /MBR
```

(sempre che si abbia la possibilità di avviare il DOS).

Quindi, si può lanciare l'interfaccia a riga di comando di GRUB utilizzando il comando `grub` (man page `grub(8)`).

```
Comando: grub
Path: /sbin/grub
SINTASSI
# grub [option]
```

DESCRIZIONE

option indica la modalità di funzionamento di `grub`. Può assumere i seguenti valori:

```
--batch
    attiva la modalità non interattiva (batch). Analogo a --no-config-file
--no-curses;
--boot-drive=drive
    specifica il disco di boot per stage2 come indicato da drive (per
    default è 0);
--config-file=file
    specifica il file di configurazione per stage2 come indicato da file (per
    default è /boot/grub/grub.conf);
--device-map=file
    indica di utilizzare il file che contiene l'elenco dei dispositivi indicato
    da file;
--help visualizza un aiuto sommario di grub;
--hold attende finché un debugger non si è agganciato;
--install-partition=par
    specifica la partizione di installazione di stage2 come indicato da par
    (per default è 0x20000);
--no-config-file
    indica di non utilizzare nessun file di configurazione;
--no-curses
    indica di non utilizzare interfacce di tipo curse;
--no-floppy
    indica di non eseguire il test per i lettori di dischetti;
--no-pager
    indica di non utilizzare un pager (visualizzatore di testo) predefinito;
--preset-menu
    indica di utilizzare il menù preimpostato;
--probe-second-floppy
    indica di eseguire il test per il secondo lettore di dischetti;
--read-only
    indica di non scrivere niente sui dispositivi;
--verbose
    visualizza messaggi informativi;
--version
    visualizza la versione di grub;
```

Questa è una sorta di shell, con un insieme di comandi e direttive che rendono agevole le gestione dell'avvio. La lista dei comandi/direttive è ottenibile attraverso il comando `help`.

La prima cosa da fare è quella di impostare come root directory la directory `/boot` (quella che contiene il file `grub/stage1`⁸ ed il file contenente il kernel di GNU/Linux) con il comando

```
grub> root (hd0,1)
```

⁸per trovare tale directory si può anche utilizzare il comando `find` interno a GRUB.

A questo punto si può installare GRUB nel MBR del primo hard disk con il comando

```
grub> setup (hd0)
```

Se si vuole, invece, installare GRUB nel settore di boot della prima partizione del primo hard disk, si può utilizzare il comando

```
grub> setup (hd0,0)
```

Se si desidera installare GRUB in una partizione o un disco diverso dal primo, per poterlo far funzionare, si deve far lanciare GRUB tramite chain-load da un altro boot loader.

Si può anche installare GRUB su un dischetto di avvio. Per far ciò si devono copiare i file **stage1** e **stage2** rispettivamente sul primo e secondo settore del floppy.

Attenzione: questa procedura cancellerà qualsiasi dato memorizzato precedentemente sul dischetto.

I comandi da lanciare sono i seguenti:

```
# cd /usr/share/grub/i386-pc
# dd if=stage1 of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
# dd if=stage2 of=/dev/fd0 bs=512 seek=1
153+1 records in
153+1 records out
```

Configurazione

Il file di configurazione di GRUB è `/boot/grub/grub.conf` a cui si può generalmente accedere anche attraverso il link simbolico `/boot/grub/menu.lst`. Tale file contiene le direttive per la presentazione a video dell'elenco dei possibili sistemi operativi da avviare in fase di boot e quelle per il loro effettivo avvio. Un tipico file di configurazione è quello riportato di seguito:

```
#
# Sample boot menu configuration file
#

# By default, boot the first entry.
default 0

# Boot automatically after 10 secs.
timeout 10

# For booting GNU/Linux
title GNU/Linux
kernel (hd0,1)/vmlinuz root=/dev/hda1

# For booting Windows
title Windows boot menu
root (hd0,0)
makeactive
chainloader +1
```

Segue una breve spiegazione delle direttive contenute nell'esempio.

```
default 0
```

indica a GRUB la scelta di default, in questo caso la prima voce del menù (le voci del menù di avvio sono quelle contraddistinte dalla parola chiave **title**), poiché il conteggio inizia da zero;

```

timeout 10
    indica a GRUB di avviare automaticamente la macchina con la scelta di default
    se non è stato premuto nessun tasto entro 10 secondi;

title GNU/Linux
    definisce una voce di menù contenente il testo "GNU/Linux";

kernel (hd0,1)/vmlinuz root=/dev/hda1
    indica a GRUB di lanciare il file /vmlinuz, il kernel di GNU/Linux, che (in
    questo caso) si trova nella seconda partizione del primo hard disk, passandogli
    il parametro root=/dev/hda1 che indica di "montare" come root directory la
    prima partizione del primo disco ATA;

title Windows boot menu
    definisce una voce di menù contenente il testo "Windows boot menu";

root (hd0,0)
    indica a GRUB di impostare come partizione di avvio la prima partizione del
    primo hard disk;

makeactive
    indica a GRUB di rendere effettiva l'impostazione della partizione di avvio;

chainloader +1
    indica a GRUB di caricare il boot loader presente sul primo settore della prima
    partizione del primo hard disk (+1).9

```

Se è installato il DOS o Windows in un hard disk diverso dal primo, è necessario utilizzare la tecnica del *disk swapping* perché tali sistemi operativi possono essere caricati soltanto dal primo hard disk. GRUB mette a disposizione il comando **map** che può essere usato per scambiare virtualmente gli hard disk con le istruzioni seguenti

```

map (hd0) (hd1)
map (hd1) (hd0)

```

potendo così ovviare al problema.

2.1.7 LILO

LILO¹⁰ (Linux LOader) è il boot loader utilizzato da molte delle distribuzioni per il caricamento del sistema GNU/Linux.

LILO si compone dei file presenti nella directory **/boot**, dal file eseguibile **lilo** e dal file di configurazione **/etc/lilo.conf**. La directory **/boot** contiene i file utilizzati per l'avvio del sistema. Qui vi possono essere file sia per GNU/Linux che per altri sistemi operativi. Contiene almeno i seguenti file:

```

boot.b    file che contiene il boot sector di LILO;

map       map file (creato da LILO) contenente il nome del file immagine del kernel e
          del suo path (può fare riferimento anche a più di un kernel);

vmlinuz   (o simile) il kernel di GNU/Linux.

```

Oltre ai file precedentemente indicati, LILO utilizza le direttive presenti nel file di configurazione **/etc/lilo.conf**, relative sia all'installazione di LILO che all'avvio dei vari sistemi operativi presenti sulla macchina.

L'indicazione dei dischi e delle partizioni è effettuata in LILO secondo la sintassi utilizzata da GNU/Linux: quindi **/dev/hda** è il primo hard disk ATA, **/dev/hdb** il secondo, ... **/dev/hda1** la prima partizione del primo hard disk ATA, **/dev/hda2** la seconda partizione del primo hard disk ATA, ... (v. sez. 3.3).

Per installare LILO come boot loader, si utilizza il comando **lilo** (man page **lilo(8)**).

⁹nel caso di prima partizione del primo hard disk, l'indicazione della partizione (**hd0,0**) può essere sott'inteso: la direttiva poteva anche essere espressa come **chainloader (hd0,0)+1**.

¹⁰licenza GPL.

Comando: `lilo`
 Path: `/sbin/lilo`

SINTASSI

`lilo` [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di `lilo`. Può assumere i seguenti valori:

- v incrementa il livello di verbosità (può essere specificata più volte);
- q elenca i file dei kernel ed il loro path;
- m *map_file*
indica di utilizzare il map file specificato da *map_file* al posto di quello di default `/boot/map`;
- C *config_file*
indica di utilizzare il file di configurazione specificato da *config_file* al posto di quello di default `/etc/lilo.conf`;
- d *delay*
indica il tempo da attendere (in decimi di secondo) prima di avviare il kernel di default, secondo quanto specificato da *delay*;
- D *label*
indica l'etichetta (*label*) del kernel da considerare come default;
- r *root_directory*
indica di impostare come root directory quella specificata da *root_directory*;
- t (test) non crea effettivamente un map file e non modifica il boot sector;
- c abilita l'ottimizzazione del caricamento del kernel, velocizzandone la fase di boot;
- f *disk_tab*
indica di utilizzare il file *disk_tab* per le specifiche della geometria del disco, al posto del file di default `/etc/disk_tab`;
- i *boot_sector*
indica di utilizzare il contenuto del file *boot_sector* per essere inserito nel boot sector, al posto del contenuto del file `/boot/boot.b`;
- l indica di generare indirizzamenti lineari dei settori piuttosto che del tipo CHS;
- L indica di generare indirizzamenti LBA piuttosto che CHS;
- P {*fix*|*ignore*}
ripara (*fix*) o ignora (*ignore*) le partition table non corrette;
- s *save_file*
indica il file in cui effettuare un backup del contenuto del boot sector prima di sovrascriverlo. Per default viene utilizzato il file `boot.n`, dove *n* è un valore esadecimale che identifica univocamente la partizione di cui contiene il boot sector. Se specificato insieme a `-u`, indica il file col quale ripristinare il boot sector;
- S *save_file*
come `-s` ma se *save_file* esiste già, lo sovrascrive;
- u *device_name*
rimuove LILO, ripristinando il boot sector con quello precedente;
- U *device_name*
come `-u`, ma non controlla il *time-stamp* del boot sector;
- R *command*
indica il comando di default (*command*) per il boot loader solo per la prossima volta che questo viene eseguito. Poi il comando viene rimosso. Si tratta di un comando once-only;
- I *label*
indica di visualizzare il path del kernel avviato, subito dopo la fase di boot;
- V visualizza la versione di `lilo`;

Di seguito è riportato un esempio del file `/etc/lilo.conf`.

```
# Parte generale
boot=/dev/hda
prompt
timeout=50
# Caricamento di Linux
image=/boot/vmlinuz
label=linux
root=/dev/hda2
read-only
# Caricamento di Windows
other=/dev/hda1
label=windows
table=/dev/hda
```

Segue una breve spiegazione delle direttive contenute nell'esempio.

```
boot=/dev/hda
    indica a LILO di installarsi nel primo settore del primo disco rigido, ovvero nel
    suo MBR. Se si desidera installare LILO nel primo settore di una partizione,
    ad esempio nel primo settore della prima partizione del primo disco ATA, la
    direttiva da impartire sarebbe boot=/dev/hda1;
```

```
prompt
    indica a LILO di visualizzare il prompt (un invito ad inserire qualcosa). È richie-
    sto l'inserimento dell'etichetta (label) relativa al sistema che desidera avviare;
```

```
timeout=50
    indica a LILO di avviare automaticamente il sistema relativo alla prima etichetta
    (label) indicata nell'elenco contenuto file di configurazione (/etc/lilo.conf)
    se non viene premuto un tasto entro 50 decimi di secondo;
```

```
image=/boot/vmlinuz
    indica a LILO che ha inizio la definizione di un sistema da avviare. Si tratta di
    un'immagine del kernel (/boot/vmlinuz). In questo caso è un file che si trova
    sullo stesso filesystem di quello presente nel momento dell'avvio della macchina;
```

```
label=linux
    definisce l'etichetta relativa all'avvio del sistema che si sta definendo;
```

```
root=/dev/hda2
    indica a LILO quale partizione del disco deve essere montata come root directory
    (in questo caso /dev/hda2 cioè la seconda partizione del primo hard disk);
```

```
read-only
    indica a LILO di montare la root directory in sola lettura, in modo da permettere
    al kernel di eseguire gli opportuni controlli di integrità prima di avviare il sistema;
```

```
other=/dev/hda1
    indica a LILO che ha inizio la definizione di un altro sistema da avviare per il
    quale deve essere usato un altro boot loader che si trova nel primo settore della
    partizione specificata (in questo caso /dev/hda1);
```

```
label=windows
    definisce l'etichetta relativa all'avvio del sistema che si sta definendo;
```

```
table=/dev/hda
    indica a LILO quale MBR contiene la tabella delle partizioni (in questo caso il
    MBR del primo disco ATA).
```

Con LILO deve essere usata un'accortezza: se si cambia qualcosa all'interno della directory `/boot` (oppure si modifica o si sposta il file del kernel) è necessario ripetere l'installazione, ovvero lanciare nuovamente il comando `lilo`.

2.2 Parametri di avvio

I boot loader permettono di specificare dei parametri di avvio del sistema sulla linea di comando. Questa possibilità permette di passare al sistema dei valori secondo i quali operare opportunamente. Tali parametri possono specificare degli argomenti per il kernel (v. sez. 6.1.4) delle variabili di ambiente (v. sez. 7.3.3) o degli argomenti per il primo processo avviato dal kernel stesso (*init*) (v. sez. 2.3).

È importante sottolineare il fatto che i parametri passati al boot loader devono essere specificati senza includere spazi al loro interno, gli spazi possono essere utilizzati soltanto come separatori tra argomenti diversi. I valori che fanno parte di un elenco per uno specifico argomento, devono essere separati soltanto dal carattere ‘,’ (senza spazi).

Innanzitutto il kernel, quando viene avviato, considera per sé gli argomenti passati sulla riga di comando, che è in grado di riconoscere (es. *root*).

I parametri non riconosciuti dal kernel, con la sintassi *name=value*, vengono interpretati come l’assegnamento della variabile di ambiente *name* il cui valore può essere controllato dagli script di inizializzazione del sistema ed in base ad esso possono essere effettuate le opportune operazioni (es. *TERM=vt100*).

Tutti gli altri parametri, non riconosciuti dal kernel, né della forma *name=value* vengono passati al primo processo avviato dal kernel, che in genere è *init* (tipicamente il parametro *single* che indica a *init* di avviare il sistema in *single user mode* – v. sez. 2.4).

Una volta avviato il sistema, l’elenco di argomenti passati al boot è contenuto nel file */proc/cmdline*.

2.3 Avvio del sistema

Una volta che il boot loader è stato avviato, questo carica il kernel, lo decompime (negli elaboratori X386 questa operazione avviene in *protected mode*¹¹), e lo avvia passandogli eventuali parametri. Il kernel si preoccupa di compiere le operazioni necessarie all’avvio dell’intero sistema operativo, ovvero dei processi fondamentali per il suo utilizzo.

L’unico processo che viene avviato dal kernel è *init* tramite l’esecuzione del file */sbin/init* (man page *init(8)*), il quale poi si preoccupa di avviare gli altri.¹²

Comando: *init*
Path: */sbin/init*

SINTASSI
init [*option*] [*runlevel*]

DESCRIZIONE

option indica la modalità di funzionamento di *init*. Può assumere i seguenti valori

- a | **auto**
indica che il kernel è stato caricato con la riga di comando di default. Questo fa impostare ad *init* la variabile di ambiente *AUTOBOOT* con il valore “yes”;
- b | **emergency**
lancia direttamente una shell in modalità monoutente senza eseguire nessuno script di startup;
- s | S | **single**
modalità di avvio monoutente. In tale modalità, prima di avviare la shell in modalità monoutente, viene elaborato il file */etc/inittab* e quindi eseguiti gli script rc;

¹¹la modalità di funzionamento delle CPU X386 sono essenzialmente 2: *real mode* e *protected mode* (v. <http://www.x86.org/articles/pmbasics/>).

¹²questo è ciò che avviene generalmente su macchine configurate in maniera “standard”, poiché è possibile configurare diversamente il sistema.

-z xxx ignora l'argomento **xxx**. Aumenta la riga di comando di **init** (occupando più stack) in modo che il comando **ps** possa visualizzare il runlevel corrente;

runlevel indica la modalità di funzionamento del sistema. Può assumere uno dei seguenti valori

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

imposta il sistema con il runlevel relativo;¹³

S | s imposta il sistema in una particolare modalità single-user, ovvero **init** avvia il sistema senza l'ausilio del file **/etc/inittab**, presentando automaticamente una shell con i privilegi del *superuser* sulla console di sistema (**/dev/console**);

Il processo **init** imposta le seguenti variabili di ambiente per i processi figli (tutti i processi da esso lanciati)

PATH **/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin** è l'elenco delle directory nel quale vengono ricercati i file eseguibili specificati sulla riga di comando;

INIT_VERSION
versione di **init**;

RUNLEVEL
il valore del runlevel corrente;

PREVLEVEL
il valore del runlevel precedente;

CONSOLE
il valore deriva dal kernel. Se non esiste viene impostato a **/dev/console**;

La sequenza di avvio dei processi iniziali utilizza un meccanismo derivato da *Unix System V*¹⁴, secondo il quale **init** legge il contenuto del file **/etc/inittab** (man page **inittab(5)**) che contiene le direttive per l'avvio degli altri processi necessari al funzionamento del sistema. Tale file, come la quasi totalità dei file di configurazione dei sistemi Unix-like, è in formato testo (ASCII).

La sintassi delle righe contenute nel file **/etc/inittab** è la seguente:

id: *runlevel*: *action*: *file*

dove

id è un'etichetta che identifica univocamente la riga all'interno del file. Può essere lunga al massimo 2 caratteri;

runlevel è l'elenco dei *runlevel*¹⁵ in corrispondenza dei quali deve essere presa in considerazione la riga;

action è una direttiva che indica a **init** l'azione da intraprendere. Le possibili direttive sono:

respawn

Il *file* viene lanciato ed ogni volta che il processo da esso generato termina viene rilanciato;

wait

Il *file* viene lanciato e **init** attende che il processo da questo avviato sia terminato prima di procedere nell'elaborazione delle altre righe di **/etc/inittab**;

¹³v. sez. 2.4.

¹⁴Io Unix attuale si divide in due grandi filoni: il *System V*, degli *Unix System Laboratories* (USL) e la *Berkeley Software Distribution* (BSD). La versione USL è arrivata alla quarta revisione (revision 4) indicata come SVr4, mentre l'ultima versione BSD è la 4.4.

¹⁵v. sez. 2.4.

once Il *file* viene lanciato una sola volta;

boot Il *file* viene lanciato ignorando il runlevel corrente;

bootwait
Il *file* viene lanciato ignorando il runlevel corrente e *init* attende la terminazione del processo avviato da *file* prima di procedere nell'elaborazione delle altre righe di */etc/inittab*;

off Non esegue alcuna operazione;

ondemand
La riga viene presa in considerazione quando lo specifico *ondemand runlevel* è richiesto, senza però cambiare il runlevel corrente. Gli ondemand runlevel possibili sono **a**, **b** e **c**;

initdefault
Imposta il runlevel corrente del sistema all'avvio al valore di *runlevel*. Il campo *runlevel* deve contenere un solo valore che identifica il runlevel da impostare. Il *file* viene ignorato. Se nessuna riga specifica questa direttiva, *init* richiede all'utente di inserire il valore del runlevel con cui avviare il sistema;

sysinit
Il *file* viene lanciato ignorando il runlevel corrente. Tale riga viene presa in considerazione prima di quelle contenenti **boot**, **bootwait** e di quelle relative a runlevel specifici;

powerwait
Il *file* viene lanciato quando l'alimentazione viene a mancare. Il sistema è in grado di rilevare questo stato quando un UPS è opportunamente connesso a quest'ultimo. *init* attende la terminazione del processo lanciato da *file* prima di procedere;

powerfail
È analogo a **powerwait** tranne per il fatto che *init* non attende la terminazione del processo lanciato da *file*;

powerokwait
Il *file* viene lanciato quando l'alimentazione è stata ripristinata. Il sistema è in grado di rilevare questo stato quando un UPS è opportunamente connesso a quest'ultimo. *init* attende la terminazione del processo lanciato da *file* prima di procedere;

powerfailnow
Il *file* viene lanciato quando la batteria dell'UPS è quasi scarica e l'alimentazione generale non c'è. Il sistema è in grado di rilevare questo stato quando un UPS è opportunamente connesso a quest'ultimo;

ctrlaltdel
Il *file* viene lanciato quando *init* riceve il segnale **SIGINT**¹⁶. Questo vuol dire che è stata premuta la combinazione di tasti

Ctrl	Alt	Canc
------	-----	------

 (o

Ctrl	Alt	Del
------	-----	-----

 dipende dal tipo di tastiera);

kbrequest
Il *file* viene lanciato quando *init* riceve un segnale dal gestore (handler) della tastiera. Per far funzionare questa direttiva è opportuno segnalare al gestore della tastiera di notificare la pressione di una determinata combinazione di tasti a *init*;

file è il file (ovvero il suo path assoluto) da eseguire, nel caso in cui la direttiva lo richieda.

Al suo avvio, *init* cerca nel file */etc/inittab* una riga contenente la stringa **initdefault**, che determina il runlevel iniziale del sistema. Se tale indicazione non viene trovata (o addirittura il file */etc/inittab* non esiste), *init* ne richiede l'inserimento da tastiera.

¹⁶i segnali saranno trattati nel cap. 6.

Dopo che *init* ha lanciato tutti i processi indicati in */etc/inittab*, attende che si verifichi una delle seguenti condizioni:

- uno dei processi lanciati da *init* termina;
- *init* riceve un segnale di mancata alimentazione SIGPWR;
- *init* riceve la richiesta di cambiamento di runlevel (da *telinit*);

Quando si verifica una delle condizioni sopra elencate *init* riesamina il file */etc/inittab*. Si può anche forzare *init* a riesaminare il file */etc/inittab* con il comando

```
# telinit q
```

Se il sistema non è in single-user mode e *init* riceve il segnale SIGPWR, quest'ultimo tenta di leggere il contenuto del file */etc/powerstatus* che riporta i dettagli relativi al segnale in questione

- | | |
|---|--|
| F | (Fail) l'alimentazione è mancata e l'UPS sta erogando energia. In tal caso <i>init</i> esegue le direttive <i>powerwait</i> e <i>powerfail</i> presenti nel file <i>/etc/inittab</i> ; |
| O | (Ok) l'alimentazione è stata ripristinata. In tal caso <i>init</i> esegue la direttiva <i>powerokwait</i> presente nel file <i>/etc/inittab</i> ; |
| L | (Low) l'alimentazione è mancata e l'UPS sta esaurendo la sua energia. In tal caso <i>init</i> esegue la direttiva <i>powerfailnow</i> presenti nel file <i>/etc/inittab</i> ; |

Nel caso in cui il file */etc/powerstatus* non esista o contenga qualcosa di diverso dalle lettere sopra riportate, *init* esegue le operazioni relative al caso in cui il file */etc/powerstatus* esista e contenga la lettera F.

La gestione del segnale SIGPWR è comunque obsoleta ed è consigliabile interagire con *init* per mezzo del dispositivo */dev/initctl*.

Quando *init* riceve la richiesta di cambiamento di runlevel, invia il segnale SIGTERM a tutti i processi (che appartengono allo stesso *process group* (v. ???) che *init* aveva creato inizialmente – gli altri processi devono essere terminati manualmente) che non sono definiti nel nuovo runlevel. Quindi attende 5 secondi prima di inviare ad essi il segnale SIGKILL che ne forza la terminazione immediata.

Quando un processo lanciato da *init* termina, *init* memorizza l'evento nel system status file (*/var/run/utmp* e */var/log/wtmp*).

Un esempio del contenuto del file */etc/inittab* è riportato di seguito.

```
#
# inittab      This file describes how the INIT process should set up
# the system in a certain run-level.
#
# Author:      Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
#              Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:
```

```

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left. Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon

```

Quando *init* deve lanciare in esecuzione un file *filename*, cerca il file */etc/initscript* (man page *initscript(5)*) e se lo trova lo utilizza per lanciare in esecuzione il file *filename* passandogli le indicazioni relative alla riga di */etc/inittab* che in quel momento *init* sta considerando. Il file */etc/initscript* potrebbe essere uno script che serve per definire delle variabili di ambiente e altre impostazioni che riguardano l'interpretazione degli script della procedura di inizializzazione del sistema. Nel caso in cui */etc/initscript* non esista (situazione standard), i file sono lanciati in esecuzione direttamente da *init*.

In genere, il primo file lanciato in esecuzione da *init* è */etc/rc.d/rc.sysinit* (v. riga che si riferisce alla direttiva *sysinit* nel file */etc/inittab*) che si occupa di eseguire le seguenti operazioni:

- Montare il filesystem */proc*;
- Impostare l'ora dell'orologio del sistema (clock) a quella dell'orologio del BIOS;
- Impostare il layout della tastiera (keymap) ed i caratteri del sistema (font);¹⁷
- Attivare il funzionamento dell'interfaccia di rete;
- Inizializzare il controller USB (Universal Serial Bus);

¹⁷v. cap. 7.

- Eseguire un controllo di integrità sul filesystem nel caso si accorga che il precedente spegnimento del sistema sia avvenuto in modo non corretto.
- Inizializzare la gestione dei quota relativi alla partizione montata come root directory;¹⁸
- Inizializzare i dispositivi ISA PNP (Plug'N'Play);
- Rimontare la partizione relativa alla root directory (/) in lettura/scrittura;
- Inizializzare il LVM (Logical Volume Management);¹⁹
- Attivare lo swap;²⁰
- Avviare i dispositivi RAID;²¹
- Inizializzare il LVM per i dispositivi RAID;
- Montare le altre partizioni;
- Riconfigurare la macchina se non è configurata;
- Inizializzare le porte seriali;
- Caricare il modulo per la gestione dei nastri SCSI (se rilevati);
- Inizializzare il controller Firewire (se rilevato);
- Attivare le ottimizzazioni per l'accesso ai dischi.

Quindi vengono avviati i servizi (*daemon*) necessari per l'utilizzo del sistema e vengono aperti i terminali per far accedere gli utenti al sistema, generalmente per mezzo di un apposito script come `/etc/rc.d/rc` (v. sez. 2.5).

Alla fine dell'elaborazione di `/etc/inittab`, *init* lancia in esecuzione il file `/etc/rc.d/rc.local` (se esiste) che può essere quindi creato o modificato dall'amministratore del sistema per personalizzare la procedura di avvio.

2.4 I runlevel

runlevel

GNU/Linux ha la possibilità di “funzionare” in 10 modalità differenti, dette **runlevel**. I runlevel non sono altro che dei valori che identificano la modalità di funzionamento del sistema, come riportato di seguito

- 0** (*halt mode*) è la modalità di preparazione allo spegnimento del sistema;
- 1** (*single-user mode*) è la modalità di funzionamento con un solo utente. Praticamente il sistema permette l'accesso ad un solo utente;

Dal file `/etc/inittab` mostrato in sez. 2.3 (preso da una distribuzione Red Hat) si vede che nel runlevel 1 non viene aperto nessun terminale virtuale, ma in tal caso ci pensa `/etc/rc.d/rc.sysinit` ad aprire un terminale che permette all'utente di accedere al sistema con i diritti di superuser (non richiedendo l'autenticazione da parte dell'utente).

- 2** (*multi-user mode no NFS*) è la modalità di funzionamento in multi user (fino a 6 utenti contemporanei sulla stessa macchina fisica), analogo al runlevel 3, ma senza NFS;

¹⁸v. cap. 3.

¹⁹v. cap. 3.

²⁰v. cap. 3.

²¹v. cap. 3.

- 3** (*full multi-user mode*) è la modalità di funzionamento in multi user (fino a 6 utenti contemporanei sulla stessa macchina fisica). Praticamente il sistema apre 6 terminali virtuali;
- 4** è una modalità non definita. Può essere utilizzata per definirne una particolare;
- 5** (*full multi-user graphic mode*) è la modalità di funzionamento in multi user con avvio automatico dell'interfaccia grafica. Praticamente il sistema apre 6 terminali virtuali ed un terminale grafico;
- 6** (*reboot mode*) è la modalità di preparazione al riavvio del sistema;
- 7** non documentato;
- 8** non documentato;
- 9** non documentato;
- S** modalità speciale single-user.

Generalmente, quando un sistema GNU/Linux viene avviato in modalità grafica, il runlevel di avvio (v. riga che si riferisce alla direttiva `initdefault` nel file `/etc/inittab`) è il 5, se invece viene avviato in modalità testo (interfaccia carattere), il runlevel di avvio è il 3.

Il runlevel corrente può essere visualizzato con il comando `runlevel` (man page `runlevel(8)`).

Comando: `runlevel`
Path: `/sbin/runlevel`

SINTASSI
`runlevel` [*logfile*]

DESCRIZIONE

logfile specifica il nome del file di log da considerare (per default viene considerato il file `/var/run/utmp`);

Tale comando visualizza il runlevel precedente e quello corrente leggendoli dal *system status file*²² (`/var/run/utmp`). Se il runlevel precedente non esiste, sarà visualizzato il carattere 'N'. A tale scopo può essere utilizzato anche il comando `who -r`. I runlevel dovrebbero comunque essere anche memorizzati nelle variabili di ambiente `RUNLEVEL` e `PREVLEVEL`.

Il cambiamento di runlevel può essere effettuato anche durante il funzionamento del sistema con il comando `telinit` (man page `init(8)`).

Comando: `telinit`
Path: `/sbin/telinit`

SINTASSI
`telinit` [*option*] [*runlevel*]

DESCRIZIONE

option indica la modalità di funzionamento di `telinit`. Può assumere i seguenti valori

`-t sec` specifica l'intervallo di tempo (in secondi) che `init` deve attendere tra l'invio del segnale `SIGTERM` e quello del segnale `SIGKILL`. Se non specificato il valore di default è 5 (5 secondi);

`Q | q` indica a `init` di rielaborare tutte le direttive contenute nel file `/etc/inittab`;

`S | s` indica a `init` di portare il sistema nella speciale modalità single-user;

²²v. sez. 5.7.

U | u indica a *init* di rieseguirsi (preservando lo stato) senza la riesaminare il file */etc/inittab*. Questo avviene soltanto se il runlevel attuale è uno dei seguenti: 1, 2, 3, 4, 5, S;

runlevel indica il runlevel da impostare. Può assumere i seguenti valori

0 | 1 | 2 | 3 | 4 | 5 | 6

indica a *init* di impostare il nuovo runlevel specificato;

a | b | c

indica a *init* di rielaborare le direttive contenute nel file */etc/inittab* relativa al runlevel *a*, *b* o *c*;

2.5 I daemon

Un sistema, oltre ad eseguire i comandi impartiti dagli utenti, può mettere a disposizione degli stessi alcuni *servizi* come la posta elettronica, il server web, la gestione centralizzata degli utenti (in genere tutte cose che vengono utilizzate quando si ha a che fare con reti di computer, v. parte II).

La procedura di inizializzazione del sistema ha il compito di avviare il sistema operativo e di fermarlo, attivando e disattivando tutti i servizi necessari, cioè intervenendo nell'avvio e nella terminazione dei processi ad essi relativi. Tali processi, detti **daemon** (tradotti volgarmente “demoni”), forniscono i servizi desiderati ed hanno la caratteristica di essere avviati automaticamente all'avvio del sistema e rimanere in esecuzione in background (v. cap. 6) finché il sistema non viene spento. Niente vieta comunque che tali processi possano essere avviati o terminati manualmente in qualunque altro momento del funzionamento del sistema qualora lo si desideri.

La procedura di avvio dei daemon è un sistema automatico per l'avvio dei servizi che si desidera attivare subito dopo l'avvio del sistema (e terminare subito prima dell'arresto del sistema stesso), derivata dalla procedura di avvio di *System V*. Secondo una convenzione diffusa, per facilitare l'avvio e la terminazione dei servizi si definisce una directory specifica (*/etc/rc.d/init.d* o simile), all'interno della quale si possono inserire degli *script*²³ di gestione dei daemon che si preoccupano di gestire opportunamente l'avvio e la terminazione dei servizi. In genere per ogni daemon esiste sempre un opportuno script, detto script di controllo del daemon, che si preoccupa di lanciare in esecuzione il daemon in maniera opportuna, effettuando gli adeguati controlli, per mezzo di una sintassi semplice e “standard”. In genere lo script di controllo del daemon ha il nome del servizio da esso gestito. Convenzionalmente gli script di controllo dei daemon hanno la seguente sintassi di lancio:

```
# script_name {start|restart|stop|...}
```

dove *script_name* è il nome dello script. Gli argomenti possono essere molteplici, ma comunque quasi tutti (per non dire tutti) gli script di controllo dei servizi hanno come argomenti **start**, **stop** e **restart**, che, come sottolineano i termini, indicano rispettivamente allo script di avviare, terminare o riavviare (terminare e subito dopo riavviare) il servizio. Tali script si preoccupano anche di effettuare i controlli necessari per servire la richiesta specificata dall'argomento passato sulla riga di comando: ad esempio controllano, prima di avviare un servizio, che questo non sia già in esecuzione.

Gli script di controllo dei daemon vengono gestiti nel modo seguente (v. fig. 2.5): ad ogni runlevel è associata una directory */etc/rc.d/rcx.d* (dove *x* è il valore del runlevel relativo). In tali directory ci sono dei *symbolic link*, appositi file (v. sez. 3.6.3), che fanno riferimento ad alcuni script menzionati precedentemente. Il nome dei *symbolic link* è del tipo *Snnscript_name* e *Knnscript_name*, dove *nn* è un valore numerico a 2 cifre e *script_name* è il nome dello script che gestisce il servizio considerato. Il file che effettua la gestione automatica degli script di controllo dei daemon (*/etc/rc.d/rc* o simili) viene lanciato in esecuzione generalmente da *init* che provvede a fornirgli, nella riga di comando,

²³gli script vengono generalmente interpretati dalla shell, quindi sono scritti con la sintassi opportuna (Bash – v. cap. 7).

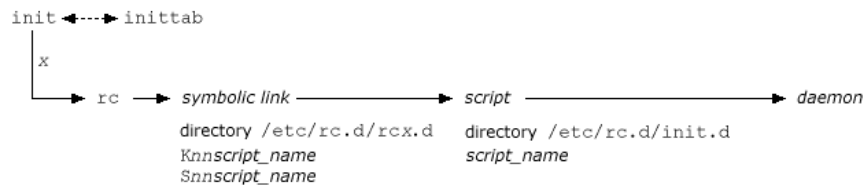


Figura 2.5: Gestione dei daemon.

il runlevel corrente (x). Tale file, che è a sua volta uno script, lancia in esecuzione, in ordine alfabetico, i file, presenti all'interno della directory `/etc/rc.d/rcx.d`, il cui nome inizia per 'K' (Kill) con l'argomento `stop`. Quindi avvia quelli il cui nome inizia per 'S' (Start) con l'argomento `start`. In questo modo vengono eseguiti i symbolic link sopra menzionati ovvero gli script di controllo dei daemon (contenuti generalmente nella directory `/etc/rc.d/init.d`), i quali avviano o terminano l'esecuzione del relativo servizio (daemon). Un esempio semplificato del contenuto del file di gestione automatica degli script di controllo dei daemon è quello riportato di seguito.

```

#!/bin/bash
#
# Sample file for starting/stopping
# services when the runlevel changes.

runlevel="$1"

# Is there an rc directory for this runlevel?
[ -d /etc/rc$runlevel.d ] || exit 0

# First, run the KILL scripts.
for i in /etc/rc$runlevel.d/K* ; do
    # Check if the file exists at all.
    [ -x "$i" ] || continue

    # Check if the daemon is already up.
    daemonname=${i#/etc/rc$runlevel.d/K??}
    [ -f /var/lock/subsys/$daemonname -o -f /var/lock/subsys/$daemonname.init ] \
        || continue

    # Bring the daemon down.
    $i stop
done

# Then run the START scripts.
for i in /etc/rc$runlevel.d/S* ; do
    # Check if the file exists at all.
    [ -x "$i" ] || continue

    # Check if the daemon is already up.
    daemonname=${i#/etc/rc$runlevel.d/S??}
    [ -f /var/lock/subsys/$daemonname -o -f /var/lock/subsys/$daemonname.init ] \
        || continue

    # Bring the daemon up.
    $i start
done

```

Da quanto detto, se ne deduce che le directory `/etc/rc.d/rc0.d` e `/etc/rc.d/rc6.d`, relative ai runlevel 0 e 6 (che portano rispettivamente allo spegnimento e al riavvio del sistema), conterranno esclusivamente file che iniziano con 'K', in quanto, in tali casi, si dovrà terminare l'esecuzione di eventuali daemon in esecuzione.

In questo modo, quando si vuole aggiungere l'avvio di un servizio non c'è bisogno di modificare nessun file, ma è sufficiente creare lo script per la gestione del relativo daemon in modo che accetti i parametri `start` e `stop` e quindi creare i symbolic link relativi a tale file con i nomi opportuni (*Snn...* e *Knn...*) in maniera da inserirli nell'ordine corretto (l'ordine alfabetico dei symbolic link è quello secondo il quale vengono avviati/arrestati i servizi) tra quelli già presenti nella directory `/etc/rc.d/rcx.d`.

Ad esempio, il servizio server HTTP è effettuato dal daemon `/usr/sbin/httpd` che è controllato tramite lo script `/etc/rc.d/init.d/httpd`. Se si desidera avviare automaticamente il servizio di server HTTP all'accensione del sistema nel runlevel 3, sarà sufficiente creare un symbolic link al file `/etc/rc.d/init.d/httpd` con il nome *Snnhttp* dove *nn* è il numero che esprime l'ordine di avvio del daemon relativo al servizio HTTP rispetto agli altri servizi. L'ordine è importante perché alcuni servizi si appoggiano ad altri che devono quindi essere avviati precedentemente. Nel caso del server HTTP è una buona idea assegnare a *nn* un valore piuttosto alto.

Di seguito è riportato, a titolo di esempio, il contenuto del file `/etc/init.d/httpd` che gestisce il daemon `httpd` che fornisce il servizio di server HTTP (server web) (v. cap. 19).

```
#!/bin/bash
#
# Startup script for the Apache Web Server
#
# chkconfig: - 85 15
# description: Apache is a World Wide Web server. It is used to serve \
#              HTML files and CGI.
# processname: httpd
# pidfile: /var/run/httpd.pid
# config: /etc/httpd/conf/httpd.conf

# Include common useful functions.
# Source function library.
. /etc/rc.d/init.d/functions

if [ -f /etc/sysconfig/httpd ]; then
    . /etc/sysconfig/httpd
fi

# This will prevent initlog from swallowing up a pass-phrase prompt if
# mod_ssl needs a pass-phrase from the user.
INITLOG_ARGS=""

# Path to the apachectl script, server binary, and short-form for messages.
apachectl=/usr/sbin/apachectl
httpd=/usr/sbin/httpd
prog=httpd
RETVAL=0

# check for 1.3 configuration
check13 () {
    CONFFILE=/etc/httpd/conf/httpd.conf
    GONE="(ServerType|BindAddress|Port|AddModule|ClearModuleList|"
    GONE="${GONE}AgentLog|RefererLog|RefererIgnore|FancyIndexing|"
    GONE="${GONE}AccessConfig|ResourceConfig)"
    if grep -Eiq "[[:space:]]*($GONE)" $CONFFILE; then
        echo
        echo 1>&2 " Apache 1.3 configuration directives found"
        echo 1>&2 " please read /usr/share/doc/httpd-2.0.40/migration.html"
        failure "Apache 1.3 config directives test"
    fi
}

start() {
```

```

        echo -n $"Starting $prog: "
        check13 || exit 1
# Start httpd as a daemon.
        daemon $httpd $OPTIONS
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && touch /var/lock/subsys/httpd
        return $RETVAL
}
stop() {
        echo -n $"Stopping $prog: "
# Stop httpd process.
        killproc $httpd
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && rm -f /var/lock/subsys/httpd /var/run/httpd.pid
}
reload() {
        echo -n $"Reloading $prog: "
        check13 || exit 1
# Send SIGHUP signal to httpd process (it causes httpd to rescan the config file).
        killproc $httpd -HUP
        RETVAL=$?
        echo
}

# See how we were called.
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    status)
        status $httpd
        RETVAL=$?
        ;;
    restart)
        stop
        start
        ;;
    condrestart)
        if [ -f /var/run/httpd.pid ] ; then
            stop
            start
        fi
        ;;
    reload)
        reload
        ;;
    graceful|help|configtest|fullstatus)
        $apachectl $@
        RETVAL=$?
        ;;
    *)
        echo $"Usage: $prog {start|stop|restart|condrestart|reload|status|fullstatus|graceful|help|configtest}"
        exit 1
esac

exit $RETVAL

```

Vari comandi utilizzati nello script riportato, come `daemon` e `killproc`, sono funzioni definite nel file `/etc/rc.d/init.d/functions`.

Il sistema di avvio automatico dei daemon, ovvero la creazione dei symbolic link che fanno riferimento ai relativi script di controllo dei servizi, è generalmente gestito da opportune applicazioni come `chkconfig` (man page `chkconfig(8)`).

Comando: `chkconfig`

Path: `/sbin/chkconfig`

SINTASSI

`$ chkconfig [option] [name {on|off|reset}]`

DESCRIZIONE

option specifica la modalità di funzionamento di `chkconfig`. Può assumere i seguenti valori

`--list [name]`

visualizza se servizio *name* è gestito da `chkconfig`. Se non è specificato *name*, elenca tutti i servizi gestiti da `chkconfig`;

`--add name`

aggiunge il servizio *name* a quelli da avviare automaticamente all'accensione del sistema, creando i relativi symbolic link nella directory `/etc/rc.d/rcx.d` (dove *x* è il runlevel considerato);

`--del name`

rimuove il servizio *name* da quelli da avviare automaticamente all'accensione del sistema, cancellando i relativi symbolic link dalla directory `/etc/rc.d/rcx.d` (dove *x* è il runlevel considerato);

`--level lev`

indica il runlevel da considerare secondo quanto specificato da *lev*, ovvero un elenco contiguo di valori di runlevel (per default viene considerato il runlevel corrente);

name {on|off|reset} indica di modificare il meccanismo di avvio relativo al servizio *name* secondo quanto specificato dall'argomento *on* (imposta l'avvio automatico), *off* (disattiva l'avvio automatico) e *reset* (reimposta l'avvio automatico secondo lo script di gestione del daemon).

Se non viene specificato nessun argomento oltre a *name*, viene controllato se il servizio *name* è impostato per essere avviato in automatico all'accensione del sistema;

Se sulla riga di comando non viene specificato nessun argomento, viene visualizzato un aiuto sommario di `chkconfig`.

Affinché un servizio sia gestibile da `chkconfig`, è necessario che lo script di gestione del daemon relativo (situato generalmente nella directory `/etc/rc.d/init.d`) contenga delle apposite righe di commento. La prima riga del file deve avere la seguente sintassi

```
# chkconfig:  def_runlevel start_level stop_level
```

dove

def_runlevel

indica l'elenco dei runlevel di default per i quali il servizio deve essere avviato automaticamente. Se è indicato un simbolo '-' significa che il servizio non deve essere avviato per default per nessun runlevel;

start_level

indica il livello di priorità (ordine) di avvio del servizio rispetto agli altri;

stop_level

indica il livello di priorità (ordine) di terminazione del servizio rispetto agli altri;

Le righe successive contengono generalmente un commento che descrive il servizio gestito dallo script. Il commento può essere esteso su più righe per mezzo del carattere di continuazione di riga utilizzato dalla shell ‘\’ (*backslash continuation*).

Un esempio delle prime righe di uno script di gestione di un daemon è riportato di seguito (tratto dallo script di gestione dei numeri pseudocasuali)

```
# chkconfig: 2345 20 80
# description: Saves and restores system entropy pool for \
#               higher quality random number generation.
```

Oltre a `chkconfig`, esistono altri comandi come `ntsysv` (man page `ntsysv(8)`) (ed altri, forniti con le varie distribuzioni di GNU/Linux), che presentano un’interfaccia più agevole per l’utente.

Comando: `ntsysv`
Path: `/usr/sbin/ntsysv`

SINTASSI
\$ `ntsysv [option]`

DESCRIZIONE

option specifica la modalità di funzionamento di `ntsysv`. Può assumere i seguenti valori

- `--back` indica di visualizzare sull’interfaccia di `ntsysv` un “pulsante” per poter ritornare all’operazione precedente;
- `--level lev` indica i runlevel per i quali configurare i servizi da lanciare automaticamente secondo quanto specificato da *lev*. I runlevel vengono specificati in una sequenza di caratteri senza spazi (per default viene considerato il runlevel corrente);

Il valore di ritorno (exit status) di `ntsysv` è riportato in tab. 2.3.

Valore	Significato
0	Operazione conclusa correttamente.
1	Operazione annullata dall’utente.
2	Si è verificato un errore.

Tabella 2.3: Exit status di `ntsysv`.

2.5.1 inetd e xinetd

Esiste anche un altro metodo di avvio automatico dei daemon, attraverso l’utilizzo del daemon *inetd* (*Internet services daemon*) o derivato. Tale daemon, detto anche *super daemon*, è stato pensato principalmente per l’avvio automatico dei daemon relativi ai servizi di rete: anziché lanciare in esecuzione i daemon relativi a tali servizi all’avvio del sistema, viene lanciato soltanto *inetd* che rimane in attesa sulle varie porte²⁴ di eventuali richieste di uno dei servizi elencati nel file di configurazione `/etc/inetd.conf`. Non appena arriva una richiesta di un servizio, *inetd* lancia in esecuzione il daemon relativo. In questo modo è possibile risparmiare risorse di sistema piuttosto che avere processi che consumano risorse e non vengono utilizzati per gran parte del tempo che il sistema rimane acceso.

²⁴v. cap. 18.

Le informazioni riportate in questa sezione presuppongono una conoscenza dei processi e dei protocolli di rete. Si raccomanda pertanto di leggere i cap. 6 e 16 prima di proseguire con la lettura di questa sezione.

Generalmente le distribuzioni di GNU/Linux forniscono un comando che estende le funzionalità di `inetd`, `xinetd` (extended `inetd`) (man page `xinetd(8)`).

Comando: `xinetd`
 Path: `/usr/sbin/xinetd`
 SINTASSI
 # `xinetd` [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di `xinetd`. Può assumere i seguenti valori:

- `-d` abilita la modalità di debug;
- `-syslog syslog_facility`
 imposta la *facility* relativa al *system log*²⁵ secondo quanto specificato da *syslog_facility*;
- `-filelog logfile`
 indica di redirigere il log degli eventi di `xinetd` nel file *logfile*;
- `-f config_file`
 indica il file di configurazione da considerare secondo quanto specificato da *config_file* (default `/etc/xinetd.conf`);
- `-pidfile pid_file`
 indica di scrivere nel file *pid_file* il PID del processo lanciato;
- `-stayalive`
 indica di rimanere in esecuzione anche se nel file di configurazione non è stato specificato nessun servizio;
- `-limit proc_limit`
 imposta il numero massimo di processi che `xinetd` può lanciare secondo quanto specificato da *proc_limit*;
- `-logprocs limit`
 imposta il numero massimo di daemon che possono essere lanciati in esecuzione per ogni utente, secondo quanto specificato da *limit*;
- `-version`
 visualizza la versione di `xinetd`;
- `-inetd.compat`
 indica di considerare anche il file di configurazione `/etc/inetd.conf` subito dopo `/etc/xinetd.conf`;
- `-cc interval`
 indica di controllare un controllo periodico del proprio stato ogni *interval* secondi;

Il processo `xinetd` effettua le operazioni elencate in corrispondenza dei seguenti segnali.

SIGHUP rilegge il file di configurazione e termina l'esecuzione dei daemon relativi a servizi non più attivi (secondo quanto specificato nel file di configurazione).

SIGQUIT
 termina la sua esecuzione.

SIGTERM
 termina l'esecuzione di tutti i daemon prima di terminare anche la sua esecuzione.

SIGUSR1
 scrive il suo stato interno (dump) nel file `/var/run/xinetd.dump`.

SIGIOT controlla la consistenza delle sue strutture dati, visualizzando quindi un messaggio relativo.

²⁵v. sez. 5.7.

Il file di configurazione di *xinetd* (*/etc/xinetd.conf*) è un file di testo che indica i servizi gestiti da *xinetd*. Contiene delle sezioni, ognuna delle quali identifica un servizio, con la seguente sintassi:

```
service service_name
{
    attribute assign_op [value] [value] [...]
    [...]
}
```

dove

service_name
è l'indicazione di un servizio gestito da *xinetd*;

attribute
indica un attributo relativo al servizio *service_name*. Gli attributi possibili sono di seguito elencati

- id** identifica univocamente un servizio (se non indicato tale attributo ha lo stesso valore di *service_name*);
- type** indica il tipo di servizio, secondo quanto specificato da *value*, che può assumere i seguenti valori
 - RPC** servizio RPC (Remote Procedure Call);
 - INTERNAL** servizio fornito direttamente da *xinetd*;
 - TCPMUX/TCPMUXPLUS** servizio avviabile per mezzo della porta TCPMUX (v. RFC 1078);
 - UNLISTED** servizio il cui daemon non si trova nella nomenclatura standard del filesystem (es. */etc/rpc* per i servizi RPC, */etc/services* per i servizi non RPC);
- flags** indica il tipo di gestione del servizio da parte di *xinetd*, secondo quanto specificato da *value*, che può assumere i seguenti valori
 - INTERCEPT** intercetta i pacchetti per verificare che provengano da indirizzi conosciuti;
 - NORETRY** non effettua nessun tentativo di rilancio nel caso in cui si verifichi un errore di lancio in esecuzione del daemon che gestisce il servizio;
 - IDONLY** accetta le connessioni soltanto se il mittente identifica l'utente;
 - NAMEINARGS** fa considerare al daemon da lanciare in esecuzione il primo argomento dell'attributo *server_args* come primo argomento della riga di comando;
 - NODELAY** imposta il flag *TCP_NODELAY* del socket relativo al servizio (se si tratta di un servizio TCP);
 - KEEPALIVE** imposta il flag *SO_KEEPALIVE* del socket relativo al servizio (se si tratta di un servizio TCP);
 - NOLIBWRAP** disabilita le chiamate alla libreria *tcp-wrap* per determinare l'accesso al servizio;
 - SENSOR** rimpiazza l'attivazione del servizio con un'apposita routine che rileva gli accessi alla specifica porta ed inserisce gli indirizzi IP dei mittenti

nell'elenco globale degli indirizzi indesiderati (*no_access list*). I pacchetti successivi che arrivano dagli indirizzi IP in tale elenco saranno scartati, finché non sarà passato il tempo specificato dall'attributo **deny_time**;

IPv4
indica che il servizio deve gestire il protocollo IPv4;

IPv6
indica che il servizio deve gestire il protocollo IPv6;

disable
indica se il servizio è disabilitato (**yes**) o meno (**no**);

socket_type
indica il tipo di socket utilizzato dal servizio, secondo quanto specificato da *value*, che può assumere i seguenti valori

stream
servizio basato su *stream*;

dgram
servizio basato su *datagram*;

raw servizio che accede direttamente a livello IP;

seqpacket
servizio che richiede una trasmissione affidabile dei *datagram*;

protocol
indica il protocollo utilizzato dal servizio (il nome del protocollo deve essere elencato nel file */etc/protocols*);

wait indica se il servizio è di tipo single-threaded (**yes**) o multi-threaded (**no**). Nel caso di servizio single-threaded, *xinetd* avvierà il daemon alla prima richiesta del servizio ed ulteriori richieste non verranno gestite (finché il daemon non terminerà la sua esecuzione). Nel caso di servizio multi-threaded ogni richiesta del servizio farà lanciare in esecuzione un'istanza del daemon relativo (ad esempio, se arrivano 3 richieste dello stesso servizio una dopo l'altra, viene lanciato in esecuzione 3 volte il relativo daemon) (v. anche **instances**);

user indica lo username dell'utente che deve lanciare in esecuzione il daemon (lo username deve essere elencato nel file */etc/passwd*);

group indica il groupname del gruppo che deve lanciare in esecuzione il daemon (il groupname deve essere elencato nel file */etc/group*);

instances
indica il numero di daemon che possono essere in esecuzione contemporaneamente per un servizio. Se non si desidera indicare un limite si può assegnare a tale attributo il valore **UNLIMITED** (default);

nice indica la priorità di esecuzione del daemon (v. man page **nice(3)**);

server indica il daemon da lanciare in esecuzione per la gestione del servizio;

server_args
indica gli argomenti da passare al daemon sulla riga di comando, tranne il nome del comando (daemon) che è nell'attributo **server**;

only_from
indica un elenco di indirizzi IP per i quali è disponibile il servizio. L'indicazione degli indirizzi IP può essere effettuata tramite la decimal dotted notation con l'utilizzo del network prefix, il nome di una rete (contenuto nel file */etc/networks*), il nome di un'interfaccia (contenuto nel file */etc/hosts* o raggiungibile per mezzo di un DNS);

no_access
indica un elenco di indirizzi IP per i quali non è disponibile il servizio. L'indicazione degli indirizzi IP può essere effettuata nello stesso modo indicato per l'attributo **only_from**. La disponibilità del servizio per un determinata macchina viene decisa dall'indirizzo IP più stringente tra quelli elencati negli attributi **only_from** e **no_access** ;

access_time
indica gli intervalli di tempo in cui il servizio deve essere disponibile. Questi sono espressi nella forma *hour:min-hour:min*, dove *hour* e *min* sono rispettivamente l'ora ed il minuto che identificano gli estremi di un intervallo;

log_type
specifica in che modo tracciare il servizio, secondo quanto specificato da *value*, che può assumere i seguenti valori

SYSLOG *facility* [*level*]
i messaggi di diagnostica vengono inviati al system log relativamente alla *facility* indicata (v. sez. 5.7);

FILE *filename* [*soft_limit* [*hard_limit*]]
i messaggi di diagnostica vengono registrati nel file *filename* (che se non esiste viene creato). Nel caso in cui il *soft_limit* venga superato, *xinetd* registrerà un evento (nel system log), mentre se viene superato l'*hard_limit* *xinetd* non scriverà più messaggi di diagnostica relativamente al servizio considerato;

log_on_success
indica le informazioni da registrare quando un daemon viene lanciato in esecuzione e quando esso termina, secondo quanto specificato da *value*, che può assumere i seguenti valori

PID registra il PID (Process Identifier)²⁶ del daemon;

HOST
registra l'indirizzo IP della macchina remota;

USERID
registra lo user id dell'utente remoto secondo quanto descritto nella RFC 1413;

EXIT
registra l'exit status del daemon o il segnale di terminazione ad esso inviato;

DURATION
registra la durata della sessione del servizio;

log_on_failure
indica le informazioni da registrare quando un daemon non può essere lanciato in esecuzione, secondo quanto specificato da *value*, che può assumere i seguenti valori

HOST
registra l'indirizzo IP della macchina remota;

USERID
registra lo user id dell'utente remoto secondo quanto descritto nella RFC 1413;

ATTEMPT
registra il fatto che il tentativo di lancio in esecuzione del daemon non è andato a buon fine;

rpc_version
indica la versione della RPC del servizio che può essere un numero o un intervallo di valori (*first_num-last_num*);

rpc_number
indica il numero del servizio RPC (non in elenco);

env
imposta le variabili di ambiente elencate (nella forma *name=value*) prima di lanciare in esecuzione il daemon;

passenv
indica le variabili di ambiente di *inetd* da pasare al daemon prima di lanciarlo in esecuzione;

port
indica il numero della porta relativa al protocollo di livello trasporto (TCP o UDP) utilizzato dal daemon;

²⁶v. cap. 6

redirect
indica di redirigere il servizio ad un altro indirizzo IP. Quando *xinetd* riceve la richiesta del servizio, esso crea un processo che stabilisce una connessione con l'indirizzo IP e la porta specificata e gli inoltra i pacchetti ricevuti. L'indicazione del destinatario della redirezione della comunicazione avviene per mezzo della sintassi (*IP_address*) (*port*);

bind | interface
indica di collegare il servizio ad una specifica interfaccia di rete presente sulla macchina. Questo può essere fatto con la sintassi (*interface.IP_address*);

banner indica di visualizzare il contenuto di un file sul monitor del sistema remoto quando viene stabilita una connessione con il servizio;

banner_success
indica di visualizzare il contenuto di un file sul monitor del sistema remoto quando l'accesso al servizio è consentito;

banner_fail
indica di visualizzare il contenuto di un file sul monitor del sistema remoto quando l'accesso al servizio non è consentito;

per_source
indica il numero massimo di istanze (o **UNLIMITED**) del daemon per ogni indirizzo IP remoto;

cps indica il limite al numero di connessioni da sistemi remoti nell'unità di tempo. Prende due argomenti: il primo indica il numero massimo di connessioni al secondo, mentre il secondo indica il numero di secondi di attesa prima di riabilitare il servizio (nel caso in cui il numero di connessioni al secondo superi il valore specificato dal primo argomento il servizio viene disabilitato e riabilitato dopo il numero di secondi specificato dal secondo argomento) (default 50 10);

max_load
indica il carico massimo (carico medio per minuto) oltre il quale *xinetd* non accetta più connessioni. Questo è espresso da un valore in virgola mobile;

groups indica se il daemon deve essere eseguito tenendo conto del gruppo a cui appartiene il relativo *effective UID*²⁷ (**yes**) o meno (**no**);

umask imposta l'*umask*²⁸ ereditata dal daemon, espressa per mezzo di un numero ottale. Se non viene specificata, i daemon avranno tutti la *umask* relativa al momento nel quale è stato lanciato in esecuzione *xinetd* in OR con il valore (22)₈;

enabled
indica l'elenco dei servizi da abilitare (nel caso in cui un servizio sia stato indicato come disabilitato – servizio **disable** o attributo **DISABLE** – il servizio non viene considerato neanche se indicato nell'elenco di questo attributo);

include
indica di utilizzare il file specificato di seguito come file di configurazione di *xinetd*;

includedir
indica di utilizzare tutti i file contenuti nella directory specificata di seguito (tranne quelli il cui nome contiene un punto '.' o finisce con il carattere tilde '~'), come file di configurazione di *xinetd*;

rlimit_as
indica il limite massimo dello spazio di indirizzamento per il processo relativo al daemon che gestisce il servizio (in byte, ma possono essere utilizzati i simboli 'K' – kibi – e 'M' – mebi – o 'UNLIMITED' – illimitato);

²⁷v. cap. 6.

²⁸v. cap. 3.

rlimit_cpu
indica il limite massimo dei secondi di CPU che il processo relativo al daemon che gestisce il servizio può utilizzare (può essere utilizzato anche il simbolo 'UNLIMITED' – illimitato);

rlimit_data
indica il limite massimo dei byte utilizzabili per la gestione dei dati relativi al processo (può essere utilizzato anche il simbolo 'UNLIMITED' – illimitato);

rlimit_rss
indica il limite massimo della parte di memoria riservata per la parte residente del processo (che non deve risentire dello swapping) (in byte, ma può essere utilizzato anche il simbolo 'UNLIMITED' – illimitato);

rlimit_stack
indica il limite massimo dello stack del processo (in byte, ma può essere utilizzato anche il simbolo 'UNLIMITED' – illimitato);

deny_time
indica l'intervallo di tempo, in minuti che deve intercorrere tra due accessi consecutivi allo stesso servizio, nel caso in cui si riesca a scavalcare il SENSOR (possono essere utilizzati anche i simboli 'FOREVER' – l'accesso è consentito soltanto una volta – e 'NEVER' – traccia l'indirizzo IP del mittente);

assign_op

è un operatore di assegnamento, e può essere = (specifica l'unico valore dell'attributo), += (aggiunge un valore all'attributo) o -= (rimuove un valore dall'attributo). gli attributi **only_from**, **no_access**, **log_on_success**, **log_on_failure** e **passenv** supportano tutti i tipi di operatori, l'operatore **env** non supporta l'operatore -= e tutti gli altri supportano soltanto l'operatore =;

Non è necessario specificare tutti gli attributi per ogni servizio, ma quelli necessari sono riportati in tab. 2.4

Attributo	Tipo di servizio
socket_type	Tutti
user	Non interno a xinetd
server	Non interno a xinetd
wait	Tutti
protocol	RPC e unlisted
rpc_version	RPC
rpc_number	unlisted RPC
port	unlisted non RPC

Tabella 2.4: Attributi necessari per i servizi avviati da **xinetd**.

Il file di configurazione può anche contenere una sezione che contiene gli attributi di default per tutti i servizi che non li specificano.

```
defaults
{
    attribute = value value ...
    ...
}
```

Alcuni attributi (**log_on_success**, **log_on_failure**, **only_from**, **no_access**, **passenv**, **disabled**, **enabled**) hanno un effetto cumulativo e ogni volta che vengono specificati accumulano il valore a quello precedente, cioè per essi, l'operatore = equivale a +=.

???

Per ogni servizio può essere specificato un grado di tracciamento (log) quando il relativo daemon serve le richieste. **xinetd** genera dei messaggi di log con la seguente sintassi

entry: *service_id data*

dove

entry è costituito da uno dei seguenti elementi

START indica l'avvio di un daemon. Il relativo messaggio di log ha la seguente sintassi

START: *service_id* [*pid=num*] [*from=IP_address*]

EXIT indica la terminazione di un daemon. Il relativo messaggio di log ha la seguente sintassi

EXIT: *service_id* [*type=num*] [*pid=num*] [*duration=num(sec)*]

dove *type* può essere **status**, seguito dal relativo exit status, o **signal**, seguito dal valore numerico del segnale che ha provocato la terminazione.

FAIL indica un tentativo di avvio di un daemon non andato a buon fine. Il relativo messaggio di log ha la seguente sintassi

FAIL: *service_id reason* [*from=IP_address*]

dove *reason* può assumere i seguenti valori

fork indica che un determinato numero di tentativi di chiamata alla system call **fork** non è andato a buon fine (il numero di tentativi è configurabile);

time il controllo sul tempo non è andato a buon fine;

address

il controllo relativo all'indirizzo non è andato a buon fine;

service.limit

indica che è stato oltrepassato il numero massimo di istanze del daemon;

process.limit

indica che è stato oltrepassato il numero di processi figli (creati con **fork**);

DATA indica un tentativo di avvio di un daemon non andato a buon fine, se il daemon supporta l'opzione **RECORD**. Il relativo messaggio di log ha la seguente sintassi

DATA: *service_id data*

dove *data* può assumere, dipendentemente dal servizio, i valori di seguito riportati

login

remote_user=remote_username local_user=local_username tty=tty

exec

remote_user=remote_username verify=status command=command

dove *status* può essere **ok** (la password è stata riconosciuta), **failed** (la password non è stata riconosciuta) e **baduser** (lo username non è stato riconosciuto);

```
shell
    remote_user=remote_username local_user=local_username command=command
```

```
finger
    string
```

dove *string* è la stringa ricevuta o una riga vuota;

USERID indica che è utilizzata l'opzione **USERID**. Il relativo messaggio di log ha la seguente sintassi

```
USERID: service_id text
```

dove *text* è la risposta dell'identificazione da parte del richiedente il servizio;

NOID indica che è utilizzata l'opzione **USERID**, è utilizzato il flag **IDONLY** ed il richiedente non si è identificato. Il relativo messaggio di log ha la seguente sintassi

```
NOID: service_id IP_address reason
```

```
service_id
    ;
```

```
data    ;
```

```
???
```

2.6 Accesso al sistema

Dopo aver avviato il sistema con tutti i servizi (demoni) necessari al suo funzionamento, la procedura di avvio del sistema crea i **virtual terminal** (VT) o *virtual console* (terminali virtuali) tramite il comando **getty** o derivati²⁹. Un terminale virtuale non esiste fisicamente, ma ha le stesse funzionalità che avrebbe un terminale fisico, ovvero consente di collegarsi con il sistema operativo e di potervi interagire.

Il processo **getty** (o derivato) si occupa di:

1. aprire la linea di comunicazione terminale e impostarne le modalità di comunicazione;
2. visualizzare un testo di “benvenuto” all'utente che si accinge ad accedere al sistema (il testo visualizzato è quello contenuto nel file `/etc/issue`) seguito dall'invito ad inserire lo *username*³⁰ (un prompt);
3. ricevere lo *username* dell'utente che vuole accedere al sistema;
4. attivare il programma per l'autenticazione di accesso (convenzionalmente si tratta di `/bin/login`), fornendogli già lo *username* (sarà poi compito di **login** richiedere l'inserimento della *password*, per verificare le credenziali dell'utente).

Per i dettagli v. cap. 5.

²⁹il comando utilizzato originariamente per questo scopo è **getty**, ma quasi tutte le distribuzioni di GNU/Linux utilizzano programmi da esso derivati come **agetty**, **mingetty** e **mgetty**.

³⁰gli utenti sono trattati in cap. 5.

2.6.1 Accesso da interfaccia carattere

Come descritto in precedenza, l'accesso al sistema da shell (metodo più classico) avviene per mezzo del processo *getty* (o derivato) che visualizza un messaggio di benvenuto del sistema, seguito da un prompt che invita l'inserimento di uno *username*, come riportato di seguito

```
Red Hat Linux release 8.0 (Psyche)
Kernel 2.4.18-18.0.0 on an i686
```

```
Zeus login: _
```

Quindi viene lanciato il processo *login* che visualizza la richiesta dell'inserimento della password, cioè

```
Password: _
```

Per motivi di sicurezza, la password inserita non viene visualizzata sullo schermo e nemmeno il cursore viene spostato verso destra man mano che la si digita sulla tastiera (è comunque possibile cancellare gli ultimi caratteri digitati premendo il tasto Backspace tante volte quanti sono i caratteri da cancellare).

Dopo aver controllato la corrispondenza delle credenziali fornite, con quelle presenti sul sistema, viene visualizzato un messaggio che riporta l'ultimo accesso effettuato dall'utente riconosciuto e viene lanciata la shell con i privilegi di quest'ultimo, la quale provvede a visualizzare uno specifico prompt. Un esempio è riportato di seguito

```
Red Hat Linux release 8.0 (Psyche)
Kernel 2.4.18-18.0.0 on an i686

Zeus login: danielle
Password:
Last login: Wed Mar 12 18:42:57 on :0
[danielle@Zeus ~] _
```

2.6.2 Accesso da interfaccia grafica

Nel caso in cui il sistema venga avviato nel runlevel 5, *init*, oltre ad avviare varie istanze di *getty* (o derivati), avvia il processo *xdm* (mediante uno script opportuno v. ultime righe del file */etc/inittab* riportato a pag. 67) che si preoccupa di gestire l'accesso al sistema mediante l'interfaccia grafica (X Window). *xdm* visualizzerà un *greeter* (grafico o a finestre standard) che richiede l'inserimento di uno username (v. fig. 2.6).

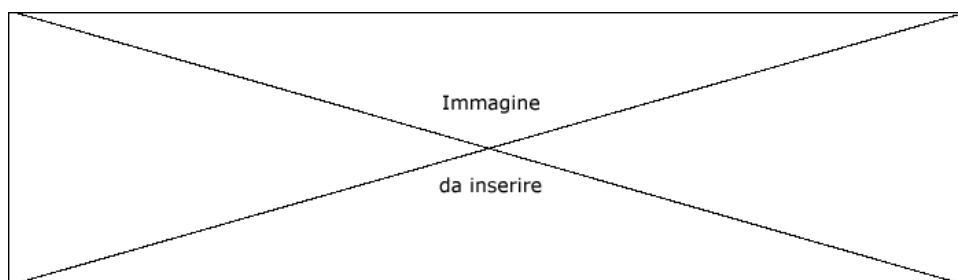


Figura 2.6: figura greeter.

Inserito lo username, lo stesso greeter richiede l'inserimento della password.

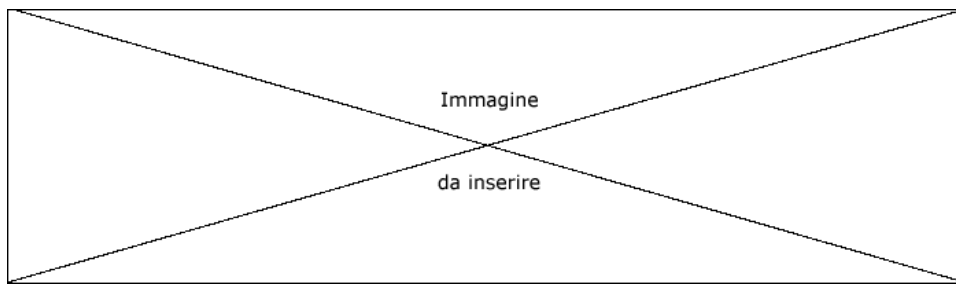


Figura 2.7: figura solo greater relativo all'inserimento password.

Per motivi di sicurezza, la password inserita non viene visualizzata in chiaro sullo schermo ma per ogni carattere inserito viene visualizzato un simbolo '*'.

Dopo aver controllato la corrispondenza delle credenziali fornite, con quelle presenti sul sistema, viene mostrato all'utente l'ambiente grafico di lavoro, quello che comunemente viene definito *desktop*.

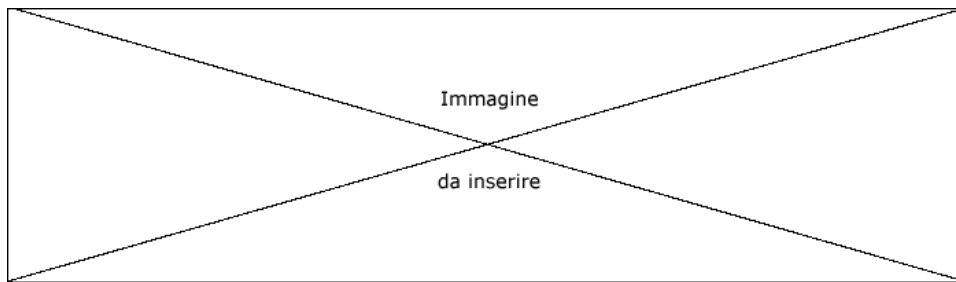


Figura 2.8: figura desktop.

2.7 Arresto del sistema

Al fine di far terminare in maniera opportuna i processi in esecuzione sul sistema stesso e per la corretta gestione del caching relativo al filesystem, è necessario arrestare GNU/Linux per mezzo del comando **shutdown** (man page **shutdown(8)**). Tale comando effettua l'arresto del sistema cambiando il *runlevel* dal valore corrente a 0 (o 6) tramite una chiamata al comando **telinit** e notificando a tutti i processi e agli utenti collegati, che il sistema sta per essere arrestato. I processi riceveranno il segnale **SIGTERM** (v. cap. 6) che indica loro di terminare l'esecuzione. Il sistema non accetterà più nessun accesso (login) da parte di altri utenti per la presenza del file **/etc/nologin**.³¹ Il comando **shutdown** è utilizzato non solo per arrestare il sistema ma anche per riavviarlo (ovvero l'arresto seguito da un successivo boot della macchina).

Comando: **shutdown**
Path: **/sbin/shutdown**

SINTASSI

shutdown [*option*] *time* [*warning-message*]

DESCRIZIONE

option indica la modalità di funzionamento di **shutdown**. Può assumere i seguenti valori:

³¹v. cap. 5.

- a indica di considerare (se esiste) il file `/etc/shutdown.allow`: il file contiene l'elenco degli utenti (uno username per riga, per un massimo di 32 utenti) a cui è permesso effettuare lo spegnimento della macchina. Se l'utente che impartisce il comando `shutdown` è uno di tali utenti o il *superuser*, il comando viene eseguito con successo, altrimenti viene visualizzato un messaggio analogo a

```
shutdown: no authorized users logged in
```

ed il comando `shutdown` terminerà senza effetto sul sistema;

- t *sec* indica di attendere *sec* secondi tra l'invio del segnale di avvertimento `SIGTERM` e quello di terminazione ai processi;
- k indica di visualizzare il messaggio *warning-message* agli utenti collegati, ma di non effettuare lo spegnimento del sistema;
- r indica di effettuare un reboot (riavvio) del sistema dopo lo spegnimento;
- h indica di arrestare definitivamente (halt) il sistema dopo lo spegnimento;
- f indica di non avviare `fsck` al riavvio del sistema (fast reboot), creando il file `/fastboot` la cui presenza viene controllata ad ogni avvio del sistema: se il file esiste non viene eseguito il comando `fsck` ed il file in questione viene quindi rimosso;
- F indica di forzare l'esecuzione di `fsck` al riavvio del sistema, creando il file `/forcefsck` la cui presenza viene controllata ad ogni avvio del sistema: se il file esiste viene eseguito il comando `fsck` con un flag che indica di forzarne l'esecuzione ed il file in questione viene quindi rimosso;
- c indica di annullare un'eventuale spegnimento del sistema in corso;

time indica quando iniziare la procedura di spegnimento del sistema. Può essere specificato con la sintassi *hh:mm* dove *hh* e *mm* sono rispettivamente l'ora ed i minuti del momento in cui iniziare la procedura di spegnimento del sistema, o con la sintassi *m+* dove *m* sono i minuti da attendere prima di avviare la procedura di spegnimento del sistema (può essere specificata anche la stringa *now* che indica di iniziare subito la procedura di spegnimento del sistema, analogamente a 0+);

warning-message è un messaggio da inviare a tutti gli utenti collegati al sistema;

È possibile arrestare il sistema anche con appositi comandi dell'interfaccia grafica, messi a disposizione dalle varie distribuzioni di GNU/Linux, che a loro volta lanciano in esecuzione il comando `shutdown`.

Il riavvio del sistema può generalmente essere effettuato anche attraverso la pressione contemporanea dei tasti `Ctrl` `Alt` `Del`. Tale combinazione di tasti viene intercettata da *init* se nel file `/etc/inittab` è presente una riga nella quale è specificata l'*action* `ctrlaltdel`, come riportato nell'esempio a pag. 67.

Se non si desidera che gli utenti (tranne il *superuser*) possano riavviare il sistema con la combinazione di tasti `Ctrl` `Alt` `Del`, si può commentare o eliminare dal file `/etc/inittab` la riga contenente l'*action* `ctrlaltdel`, oppure modificare il campo *file* della stessa riga, aggiungendo al comando `shutdown` l'opzione `-a` ed inserendo nel file `/etc/shutdown.allow` gli username dei soli utenti che hanno la possibilità di effettuare il riavvio del sistema tramite la combinazione di tasti suddetta.

2.8 Riferimenti

- “Il processo di avvio del sistema”
<http://ctdp.tripod.com/os/linux/startupman/index.html>

Capitolo 3

Il filesystem

“Chi conosce il territorio ha le maggiori possibilità di vittoria in battaglia.”
– Confucio

In questo capitolo verrà trattata la struttura del filesystem con particolare riferimento ai filesystem ext2 ed ext3. Oltre a venir definiti i file e le directory, verranno dati al lettore gli strumenti necessari alla gestione del filesystem, in modo tale da poter interagire con il sistema fin da subito al fine di prenderci pratica prima possibile.

3.1 I dispositivi di memoria di massa

I sistemi Unix-like trattano tutti i dispositivi di memoria di massa in maniera analoga, ovvero considerano il supporto di memoria come un contenitore nel quale possono essere contenute delle informazioni e permettono all’utente di accedere ad essi con gli stessi comandi. Hard disk, floppy disk, CD-ROM, flash pen, ... sono dispositivi di memoria di massa che vengono trattati tutti allo stesso modo (eccezion fatta per la scrittura su CD-ROM che ha bisogno di comandi specifici). La memorizzazione delle informazioni all’interno dei dispositivi di memoria di massa avviene, come sarà illustrato in seguito, per mezzo del filesystem in essi contenuto.

Nel seguito del presente testo non verrà fatta alcuna distinzione tra i dispositivi di memoria di massa (hard disk, floppy disk, CD-ROM, ...) poiché i metodi di accesso a questi, ovvero ai loro filesystem, sono identici (a parte la scrittura su CD-ROM).

I dispositivi di memoria di massa sono “visti” dal sistema, come dei file, detti *file di dispositivo* (*device file*)¹ e contenuti generalmente nella directory `/dev` (`/dev/hda`, `/dev/sda`, ...)². Sono opportuni programmi, i *driver*, che effettuano l’associazione tra i dispositivi ed i relativi file di dispositivo: il driver ATA associa le periferiche collegate a tale bus ai file di dispositivo `/dev/hd*`, mentre quello relativo al bus SCSI associa le periferiche di quel tipo ai file `/dev/sd*`. Altri driver, come quelli per la gestione dei masterizzatori CD ATAPI, che si collegano al bus ATA, o quelli per la gestione dei dispositivi di memorizzazione connessi al bus USB (come le flash pen), associano tali periferiche ai file `/dev/sd*`, facendole vedere al sistema come se fossero dei dispositivi SCSI.

3.2 Partizionamento del disco

Per poter essere utilizzato, un hard disk va innanzitutto preparato mediante l’operazione di *partizionamento*, che consiste nella suddivisione dello spazio totale del disco in sottoinsiemi più piccoli, detti appunto **partizioni**.³ L’esigenza di tale operazione è nata dal fatto che i BIOS ed i sistemi operativi più datati non riuscivano a gestire dischi

partizioni

¹v. sez. 3.6.3.

²v. tab. 3.1 e sez. 3.12.

³v. sez. 1.6.4.

con dimensioni superiori a 2, 4 o 8 GiB. Oggi il partizionamento dei dischi (al limite il disco è costituito da una sola partizione) viene generalmente utilizzato per suddividere il supporto di memorizzazione, in maniera tale che informazioni logicamente separate vadano ad essere memorizzate in zone fisicamente separate del disco. Inoltre, dipendentemente dal filesystem utilizzato, il partizionamento può migliorare le prestazioni dello stesso o limitare lo spazio sprecato (esso infatti dipende in maniera proporzionale dalla dimensione dei blocchi⁴ del filesystem, che può variare con la dimensione della partizione sulla quale risiede il filesystem).

Il partizionamento del disco viene effettuato per mezzo del comando `fdisk`, come descritto in sez. 1.6.4, o del comando `parted` (man page `parted(8)`).

Comando: `parted`

Path: `/sbin/parted`

SINTASSI

`parted [option] [device [command]]`

DESCRIZIONE

option indica la modalità di funzionamento di `parted`. Può assumere i seguenti valori:

`-h` | `--help`

visualizza un aiuto sommario di `parted`;

`-i` | `--interactive`

richiede indicazioni su come operare quando necessario;

`-s` | `--script`

non richiede mai indicazioni;

`-v` | `--version`

visualizza la versione di `parted`;

device indica il dispositivo (disco) da partizionare;

command indica il comando da far eseguire a `parted`. Se non indicato `parted` visualizzerà un prompt attendendo l'inserimento di un comando (l'elenco dei comandi possibili può essere ottenuto con il comando `help`, mentre `help cmd` fornisce un aiuto più dettagliato relativo al comando *cmd*);

Ogni partizione può contenere un filesystem.

3.3 I file di dispositivo

In un sistema GNU/Linux, ad ogni disco e ad ogni partizione in esso contenuta è associato un *file di dispositivo* (v. sez. 3.6.3) come riportato in tab. 3.1 (*x* è una lettera: *a* rappresenta il primo disco, *b* il secondo e così via, mentre *n* è un numero).

File	Descrizione
<code>/dev/hdx[n]</code>	Disco ATA <i>x</i> (o CD-ROM ATAPI) o <i>n</i> -esima partizione del disco.
<code>/dev/sdx[n]</code>	Disco SCSI <i>x</i> (o visto come SCSI) o <i>n</i> -esima partizione di questo.
<code>/dev/edx[n]</code>	Disco ESDI (obsoleto) <i>x</i> o <i>n</i> -esima partizione di questo.
<code>/dev/hdx[n]</code>	Disco XT (obsoleto) <i>x</i> o <i>n</i> -esima partizione di questo.
<code>/dev/scdn</code>	CD-ROM SCSI <i>n</i> o masterizzatore CD (ATAPI o SCSI).
<code>/dev/fdn</code>	Floppy disk <i>n</i> .

Tabella 3.1: File di dispositivo associati ai dischi e partizioni di un sistema GNU/Linux.

Vari dispositivi sono gestiti da GNU/Linux come periferiche SCSI (anche se praticamente non lo sono). Ad esempio, i masterizzatori CD di tipo ATAPI (Advanced Technology Attachment Packet Interface - che utilizzano il bus ATA), gli hard disk connessi via USB (come le flash pen), vengono “visti” dal sistema come fossero periferiche SCSI.

⁴v. sez. 3.5.2.

Alcuni esempi

`/dev/hda`

è il file di dispositivo associato al primo (a) dispositivo ATA del sistema;

`/dev/hdb`

è il file di dispositivo associato al secondo (b) dispositivo ATA del sistema;

`/dev/sdb`

è il file di dispositivo associato al secondo (b) dispositivo visto come SCSI dal sistema;

`/dev/hda1`

è il file di dispositivo associato alla prima (1) partizione del primo (a) disco ATA del sistema;

`/dev/hdb3`

è il file di dispositivo associato alla terza (3) partizione del secondo (b) disco ATA del sistema;

`/dev/sda2`

è il file di dispositivo associato alla seconda (2) partizione del primo (a) disco visto come SCSI dal sistema;

L'elenco delle partizioni riconosciute dal sistema è contenuto nel file `/proc/partitions`.

3.4 Inizializzazione del filesystem

Una partizione (o un dispositivo di memoria di massa) deve essere preparata per essere gestita con un determinato filesystem, ovvero, su di essa va memorizzata la struttura dati necessaria alla gestione del filesystem, attraverso un'apposita operazione di inizializzazione del filesystem (gli utenti di sistemi DOS o derivati, la conoscono sotto il nome di **formattazione**⁵). Tale operazione può essere eseguita per mezzo del comando `mkfs` *formattazione* (man page `mkfs(8)`).

Comando: `mkfs`

Path: `/sbin/mkfs`

SINTASSI

`# mkfs [option] device [block]`

DESCRIZIONE

option specifica la modalità di funzionamento di `mkfs`. Può assumere i seguenti valori

`-V` visualizza tutti i comandi che vengono eseguiti per la formattazione dello specifico filesystem;

`-t fstype` indica il tipo di filesystem secondo quanto specificato da *fstype* (v. tab. 3.2). Se non è specificato viene utilizzato il filesystem di default (ext2);

`-c` indica di effettuare un controllo sui blocchi del disco prima di formattarlo;

`-l filename` indica di leggere l'elenco dei blocchi non buoni (bad blocks) dal file *filename*;

`-v` indica di produrre un output più verboso;

È inoltre possibile indicare opzioni caratteristiche dello specifico filesystem considerato;

⁵dal nome del comando `format`.

Filesystem	Descrizione
adfs	filesystem dei sistemi RISC OS.
affs	filesystem dei sistemi AmigaOS.
autofs	automounter di GNU/Linux. “Monta” automaticamente il filesystem al momento che un utente fa accesso al relativo mountpoint e lo “smonta” se non viene utilizzato per più di un certo periodo di tempo.
coda	filesystem di rete sviluppato al CMU.
cramfs	filesystem di sola lettura.
devpts	
efs	(Extent FileSystem) filesystem sviluppato da <i>Silicon Graphics</i> .
ext2	filesystem di default di GNU/Linux.
ext3	journaled filesystem di default di GNU/Linux.
hfs	(Hierarchical FileSystem) filesystem dei sistemi <i>Apple</i> .
hpfs	(High Performance FileSystem).
iso9660	filesystem standard utilizzato nei CD-ROM (default).
jfs	(Journaling FileSystem) journaled filesystem sviluppato da <i>Sun</i> .
minix	filesystem dei sistemi minix.
msdos	filesystem dei sistemi MS-DOS (FAT).
ncpfs	implementazione del filesystem di rete NCP di <i>Novell NetWare</i> .
nfs	(Network FileSystem) filesystem di rete ideato da <i>Sun</i> .
ntfs	filesystem di sistemi Windows NT/2000/XP.
proc	filesystem virtuale di GNU/Linux per la gestione di impostazioni del sistema.
qnx4	filesystem dei sistemi QNX.
reiserfs	un journaled filesystem per GNU/Linux.
romfs	un filesystem di sola lettura per GNU/Linux.
smbfs	implementazione del filesystem networking protocol di Windows (SMB) che può essere utilizzato per il mounting di filesystem remoti.
swap	filesystem di swap di GNU/Linux.
sysv coherent xenix	
tmpfs	(temporary filesystem) filesystem virtuale che utilizza la memoria centrale.
udf	(Universal Disk Format) filesystem utilizzato nei DVD e CD multimediali.
ufs	(Unix FileSystem) filesystem dei sistemi Unix.
umsdos	filesystem di GNU/Linux che permette di installare il sistema in una partizione DOS.
vfat	filesystem dei sistemi Windows 95/98 (FAT32).
xfs	journaling filesystem sviluppato da <i>IBM</i> .

Tabella 3.2: Alcuni filesystem supportati da GNU/Linux.

device indica il dispositivo da formattare. Può essere indicato come file di dispositivo o come *mount point* (v. sez. 3.9);

block indica il numero di blocchi del dispositivo *device* da utilizzare per il filesystem. Se non è specificato viene utilizzato l'intero dispositivo;

Alle partizioni gestite dal filesystem ext2 può essere associata un'etichetta che deve essere univoca fra tutte le partizioni. Tale etichetta può essere visualizzata o modificata con il comando **e2label** (man page **e2label(8)**).

Comando: **e2label**

Path: **/sbin/e2label**

SINTASSI

e2label device [label]

DESCRIZIONE

device indica il dispositivo da considerare;

label specifica l'etichetta (formata da un massimo di 16 caratteri) da assegnare al dispositivo *device*. Se non è indicata viene visualizzata l'etichetta assegnata al dispositivo;

È buona regola utilizzare come etichetta il nome del mount point utilizzato per la partizione in questione.

3.5 La struttura del filesystem

L'insieme delle informazioni memorizzate sulla memoria di massa è organizzato secondo una struttura che va sotto il nome di **filesystem**. È necessario distinguere tra la struttura logica e quella fisica delle informazioni memorizzate: la *struttura fisica* è il modo in cui le informazioni sono scritte fisicamente sulla memoria di massa in modo tale che il sistema possa ritrovarle per poterle leggere e/o modificare; la *struttura logica* è il modo in cui l'organizzazione delle informazioni è presentata all'utente.

filesystem

La struttura fisica dipende pesantemente dal filesystem considerato, mentre tutti i filesystem Unix-like hanno una struttura logica analoga.

I sistemi Unix-like utilizzano i concetti comuni di *file* e *directory*. Un *file* è un insieme di informazioni che possono rappresentare dei dati (un testo, un'immagine, una canzone, un film, ...) o anche le istruzioni di un programma. Una *directory* è una sorta di contenitore che può contenere sia file che altre directory.

Poiché le directory possono contenere sia file che altre directory, si viene a delineare una struttura gerarchica ad *albero* dell'organizzazione dei file e delle directory, detto **directory tree** o *albero delle directory*, ovvero un particolare *grafo orientato* in cui i *nodi* (elementi del grafo) sono connessi tra loro da *freccie* che rappresentano la relazione di contenimento (una freccia che va dal nodo *A* al nodo *B* indica che il nodo *A* contiene il nodo *B* - in genere si dice che il nodo *B* è figlio del nodo *A*), in modo tale che dal nodo radice (che è l'unico nodo che non è figlio di nessun altro nodo) è possibile raggiungere qualsiasi altro nodo ognuno secondo un percorso univoco.

directory tree

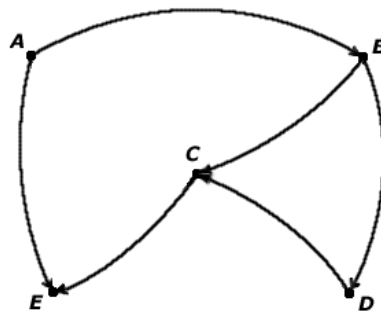


Figura 3.1: Esempio di grafo orientato.

La caratteristica fondamentale dei sistemi Unix-like è riassunta dalla frase “Everything is a file” (ogni cosa è rappresentata da un file), per cui i file rivestono un ruolo particolarmente importante per tali sistemi. Tutte le operazioni, anche quelle sui dispositivi avvengono attraverso l'ausilio di file. Ad esempio la tastiera è associata ad un file che rappresenta il buffer di input dei caratteri inviati dalla tastiera al sistema. Il monitor è rappresentato da un altro file che costituisce il buffer di output dei caratteri inviati dal sistema sullo schermo.

Un filesystem è formato quindi dall'insieme dei file e delle directory e dalla loro organizzazione, ovvero è il meccanismo che collega la struttura logica, cioè l'albero delle directory, con quella fisica, cioè i settori sul disco e la metodologia di accesso agli stessi. Un disco senza filesystem è solo un'insieme ordinato di settori a partire dal primo: le informazioni in esso contenute non sono suddivise in file e directory. Il filesystem permette inoltre l'accesso diretto alle informazioni memorizzate sulla memoria di massa (se questa è realizzata da un dispositivo che lo consente, come i dischi magnetici), ovvero

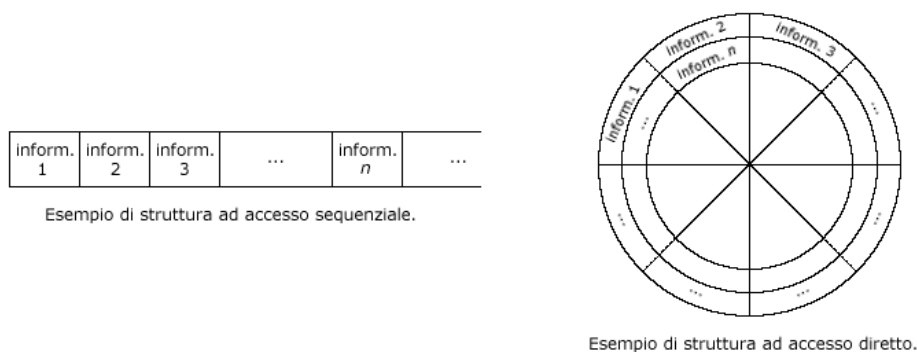


Figura 3.2: Accesso sequenziale e accesso diretto.

mette a disposizione i meccanismi per accedere ad una determinata informazione, in modo che non sia necessario accedere a tutte quelle precedenti.

Infatti, come illustrato in fig. 3.2, se il sistema di accesso al supporto di memorizzazione ha un solo grado di libertà, come può essere ad esempio un lettore di nastro magnetico, la testina di lettura/scrittura può accedere all' n -esima informazione passando prima per tutte le precedenti $n - 1$. Così, il tempo di accesso all'informazione, ovvero il tempo necessario affinché la testina di lettura/scrittura riesca a posizionarsi sull'informazione desiderata, è proporzionale alla posizione dell'informazione sul supporto magnetico. Si parla in tal caso di **accesso sequenziale** alle informazioni. Nel caso in cui il meccanismo di accesso al supporto abbia ulteriori gradi di libertà, si riduce la dipendenza del tempo di accesso alla posizione dell'informazione. In un disco magnetico, infatti, la testina di lettura/scrittura muovendosi radialmente rispetto al supporto circolare, può accedere più velocemente ad un numero più elevato di informazioni con spostamenti meccanici minori (il supporto di memorizzazione ruota intorno al proprio asse con una velocità costante). Per ontrapposizione al tipo precedentemente descritto, si parla di **accesso diretto** alle informazioni. È evidente che non si può parlare di accesso diretto vero e proprio, poiché il tempo di accesso all'informazione non è del tutto indipendente dalla posizione che l'informazione occupa sul supporto.

3.5.1 La struttura logica

La struttura logica del filesystem si compone di due tipi di oggetti, la *directory* ed il *file*, e la sua organizzazione è quella di un *albero*. I *nodi* dell'albero sono rappresentati dagli oggetti del filesystem ed in particolare i file costituiscono le *foglie* dell'albero, ovvero sono nodi terminali (non hanno nodi figli). Tutti gli oggetti del filesystem sono identificati da un nome che deve essere univoco per tutti quelli dello stesso livello (cioè figli dello stesso nodo). È opportuno sottolineare il fatto che il nome di un oggetto del filesystem, secondo la tradizione dei sistemi Unix-like, è *case sensitive*, cioè il sistema fa differenza tra le lettere maiuscole e quelle minuscole utilizzate nel nome dell'oggetto. Pertanto i nomi **Pippo** e **pippo** identificano due oggetti del filesystem (file o directory) diversi.

Mentre un file è essenzialmente una raccolta di informazioni, una directory può essere considerata come un contenitore, che può contenere sia file che altre directory.

L'albero della gerarchia del filesystem, cioè il *directory tree* (o *albero delle directory*) ha come *radice* (unico nodo che non è figlio di nessun altro) la **root directory** identificata dal simbolo *'/'* (*slash*). Questa directory contiene tutti gli altri nodi dell'albero. In ogni directory sono presenti due directory particolari: *'.'* e *'..'* che rappresentano rispettivamente la directory stessa e la directory padre (nel caso della *root directory* *'..'* coincide con *'.'* ovvero con *'/'*).

Nei sistemi Unix-like è definita anche una **working directory** (*directory di lavoro* o *directory corrente*) che indica la directory di default da considerare per determinate operazioni sul filesystem (una sorta di directory sott'intesa).

Il **path** (percorso) è l'elenco delle directory da attraversare per raggiungere un determinato nodo dell'albero (file o directory). Questo può essere di due tipi:

assoluto è il percorso per raggiungere un determinato nodo a partire dalla *root directory* (compresa);

relativo è il percorso per raggiungere un determinato nodo a partire da una determinata directory, che in genere coincide con la *working directory*.

Il simbolo utilizzato come separatore dei nomi dei nodi del filesystem è '/', lo stesso utilizzato per rappresentare la root directory. Non c'è ambiguità nello specificare il path di un nodo del filesystem, poiché il simbolo '/' viene interpretato come root directory soltanto nel caso in cui sia specificato come primo carattere del path, altrimenti è considerato come carattere di separazione dei nomi dei nodi dell'albero delle directory.

Dunque, si supponga ad esempio di voler raggiungere il file **prova** contenuto nella directory **seconda** contenuta nella directory **prima** che a sua volta è contenuta nella root directory (/). Il path assoluto del file in questione è quindi **/prima/seconda/prova**, mentre il suo path relativo alla working directory, supponendo che questa sia la directory **prima** contenuta nella root directory (cioè **/prima**), è **seconda/prova** (è come se **/prima** fosse un percorso sott'inteso).

Nei sistemi GNU/Linux, le informazioni memorizzate nel directory tree possono essere classificate in base ai seguenti criteri:⁶

Condivisibili (*shareable*) le informazioni possono essere condivise tra più computer;

Non condivisibili (*unshareable*) le informazioni non possono essere condivise tra più computer;

Statiche (*static*) le informazioni non possono essere modificate;

Variabili (*variable*) le informazioni possono essere modificate;

Tale classificazione è soltanto indicativa e può anche non essere seguita, ma seguirla può aiutare a gestire l'albero delle directory.

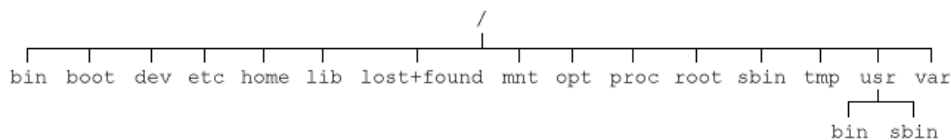


Figura 3.3: Struttura tipica del filesystem di un sistema GNU/Linux.

In generale un sistema GNU/Linux ha l'albero delle directory riportato in fig. 3.3 con le caratteristiche seguenti:

- /bin** contiene dei file eseguibili necessari per l'utilizzo del sistema ed usufruibili da tutti gli utenti (**ls**, **cp**, **mv**, **rm**, ...) – Informazioni statiche, non condivisibili;
- /boot** contiene i file necessari all'avvio del sistema (procedura di boot), come il *boot loader* ed il *kernel* – Informazioni statiche, non condivisibili;
- /dev** contiene i file relativi alle periferiche (devices), a partire dalle partizioni degli hard disk (**/dev/hda1**, **/dev/hda2**, ..., **/dev/sda1**, **/dev/sda2**, ...), ai CD ROM (**/dev/cdrom**), ai floppy disk (**/dev/fd0**), alla scheda audio (**/dev/dsp**), alle porte seriali (**/dev/ttyS0** e **/dev/ttyS1**) – Informazioni statiche, non condivisibili;

⁶l'organizzazione del directory tree segue il FHS (Filesystem Hierarchy Standard), v. <http://www.pathname.com/fhs/>.

<code>/etc</code>	contiene i file relativi alla configurazione del sistema e dei programmi, che generalmente hanno estensione <code>conf</code> o <code>cfg</code> . All'interno di questa directory si trovano, tra le altre, la directory <code>X11</code> , contenente i file di configurazione del server grafico (v. cap. 11), e la directory <code>rc.d</code> , che contiene i file per l'avvio del sistema (v. cap. 2) – Informazioni statiche, non condivisibili;
<code>/home</code>	contiene i file relativi agli utenti. Ad ogni utente è assegnata una <i>home directory</i> , che in genere è una directory che ha lo stesso nome dello <i>username</i> dell'utente, che si trova all'interno della directory <code>/home</code> . ⁷ Per esempio l'utente con username <i>pippo</i> avrà come home directory <code>/home/pippo</code> . Tale directory sarà utilizzata per salvare i file di configurazione a livello utente di vari applicativi – Informazioni variabili, condivisibili;
<code>/lib</code>	contiene le <i>librerie</i> (library) di sistema utilizzate per il funzionamento delle applicazioni e i moduli utilizzati dal kernel (<code>/lib/modules</code>) – Informazioni statiche, non condivisibili;
<code>/lost+found</code>	contiene gli eventuali file “corrotti” (danneggiati) trovati dopo un riavvio successivo ad un arresto non “adeguato” del sistema – Informazioni variabili, non condivisibili;
<code>/mnt</code>	è utilizzata come <i>mount point</i> generico per il <i>mounting</i> (v. sez. 3.9) delle periferiche che contengono dei filesystem (dischetti, CD, ...) – Informazioni variabili, non condivisibili;
<code>/opt</code>	contiene i file di applicazioni non fornite dal sistema. È un retaggio dei primi Unix e si tende a non utilizzarla;
<code>/proc</code>	è un filesystem virtuale che contiene informazioni relative ai processi in esecuzione sul sistema (non utilizza spazio fisico su disco);
<code>/root</code>	è la home directory dell'utente <i>superuser</i> che rappresenta l'amministratore del sistema (tale utente è di solito identificato dallo username <i>root</i>) – Informazioni variabili, non condivisibili.
<code>/sbin</code>	contiene dei file eseguibili necessari al funzionamento del sistema utilizzabili soltanto dall'utente <i>superuser</i> – Informazioni statiche, non condivisibili;
<code>/tmp</code>	contiene file temporanei che possono essere creati da alcuni processi. In alcuni sistemi il contenuto di tale directory può essere cancellato automaticamente allo spegnimento o all'avvio del sistema – Informazioni variabili, non condivisibili;
<code>/usr</code>	contiene i file relativi alle applicazioni installate nel sistema. Al suo interno sono contenute varie directory tra cui <code>bin</code> e <code>sbin</code> , contenenti i comandi non essenziali del sistema, <code>X11R6</code> , che contiene i programmi per il funzionamento dell'interfaccia grafica del sistema, <code>share</code> , in cui dovrebbero essere riposti i file contenenti i dati (indipendenti dall'architettura) relativi ai programmi presenti in <code>/usr/bin</code> e <code>/usr/sbin</code> , nonché i file relativi alla documentazione contenuti in <code>/usr/share/doc</code> e <code>/usr/share/man</code> – Informazioni statiche, condivisibili;
<code>/var</code>	contiene file di sistema che variano con frequenza elevata (i log di sistema <code>/var/log</code> , le caselle di posta elettronica <code>/var/spool/mail</code> , i file di stampa temporanei <code>/var/spool/lpd</code> , la <i>http root directory</i> <code>/var/www/html</code> ⁸ , ...) – Informazioni variabili condivisibili o meno (dipendentemente dalle sottodirectory).

⁷gli utenti saranno trattati nel cap. 5.

⁸l'http è trattato nel cap. 16.

3.5.2 La struttura fisica

La memoria di massa, viene generalmente suddivisa in **partizioni**, come illustrato in fig. 1.23, ognuna delle quali può contenere un filesystem. *partizioni*

Un filesystem gestisce unità minime di memorizzazione delle informazioni, dette **blocchi** (*block*), cioè tutte le sequenze di byte vengono memorizzate sulla memoria di massa in blocchi, ovvero raggruppamenti di byte, e l'accesso ai byte è effettuato dal filesystem passando per i blocchi. Ogni blocco può essere **allocato** (*allocated*), cioè contenere effettivamente delle informazioni, o **libero** (*free*), ovvero non contenere nessuna informazione. *blocchi*
allocato
libero

La scelta della dimensione dei blocchi deriva da un compromesso tra l'avere prestazioni elevate e minimizzare lo spreco di spazio medio inutilizzato sul filesystem. Blocchi più grandi permettono di mantenere le informazioni in maniera più raccolta, avendo una minore dispersione (**frammentazione esterna**) delle stesse sul disco: quando un blocco viene riempito, il sistema ne deve utilizzare un altro libero. Pertanto quando si devono ritrovare le informazioni memorizzate, più blocchi esse occupano sul disco, più "ricerche" deve effettuare il filesystem per ricercarle. Per contro, blocchi più piccoli permettono di sprecare meno spazio sul disco, in quanto, come sarà illustrato in seguito, per ogni file memorizzato sul disco si spreca mediamente la metà della dimensione di un blocco (frammentazione interna). In genere quindi, per ottimizzare il filesystem la dimensione dei blocchi da utilizzare dovrebbe essere proporzionale alla dimensione media dei file che si andranno a memorizzare sullo stesso. *frammentazione esterna*

Non si confondano i blocchi logici in cui vengono suddivise le informazioni dal filesystem, con quelli fisici utilizzati per la memorizzazione delle informazioni sulla superficie magnetica del dispositivo di memoria di massa. In genere i blocchi fisici hanno una dimensione di 512 byte, mentre i blocchi logici hanno una dimensione maggiore (comunque multiplo di quella dei blocchi fisici).

La struttura fisica per la memorizzazione delle informazioni sul filesystem, dipende pesantemente dal filesystem considerato, quindi, a titolo di esempio, nel presente testo sarà descritta soltanto quella relativa al filesystem di default di GNU/Linux: *ext2* (Second Extended Filesystem).⁹

In genere un file viene rappresentato sul filesystem da un'apposita struttura che contiene le sue caratteristiche ed i riferimenti relativi ai blocchi nei quali sono memorizzate le informazioni contenute nel file stesso (v. fig. 3.4).

Gli extent

Alcuni filesystem utilizzano come riferimenti ai blocchi delle strutture particolari, denominate **extent**, ognuno dei quali rappresenta un gruppo di blocchi fisicamente contigui. Ogni extent è una tripla di valori *(inizio, dimensione, offset)* dove *extent*

inizio indica il blocco di inizio;

dimensione

indica il numero di blocchi contigui che costituiscono l'extent;

offset indica la posizione dell'extent all'interno del file considerato;

Gli extent consentono una più efficace gestione dei blocchi del filesystem utilizzati dai file. Ad esempio, si considerino le seguenti operazioni:

1. si crea un file **pippo**;
2. si scrivono i primi 5 byte;
3. ci si sposta in avanti di 10.000 byte;

⁹da circa un anno viene utilizzato l'*ext3* (Third Extended Filesystem), che in più a *ext2* ha anche la caratteristica di *journaling*, descritta più avanti.

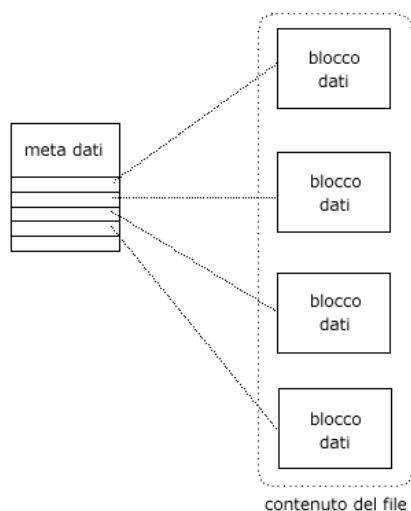


Figura 3.4: Struttura per la memorizzazione dei file sul filesystem.

4. si scrivono altri 3 byte;
5. si chiude il file;

A questo punto il sistema memorizza quanto effettuato sul filesystem. Senza utilizzare la tecnica degli extent, per la memorizzazione del contenuto del file **pip**po, viene utilizzato un numero di blocchi necessario a contenere $5 + 10.000 + 3 = 10.008$ byte, la cui maggior parte è sprecata, poiché 10.000 byte non contengono nessun valore significativo. Utilizzando gli extent invece si possono memorizzare soltanto i blocchi relativi ai 5 + 3 byte significativi (scartando gli altri 10.000), i primi 5 in un blocco riferito da un extent (ad esempio $\langle 123, 1, 0 \rangle$), e gli ultimi 3 in un altro, riferito da un altro extent (ad esempio $\langle 135, 1, 1 \rangle$) (si noti che l'indice del blocco contenente le informazioni può essere qualunque: l'ordine dei blocchi è specificato dall'offset degli extent). Si ottiene così quello che viene chiamato uno **sparse file**, ovvero un file con dei “buchi”.

sparse file

3.6 Il filesystem ext2

L'**ext2** (second extended filesystem) è il filesystem di default di GNU/Linux, ovvero il filesystem che dal 1994 è fornito con le varie distribuzioni di GNU/Linux, sebbene il primo filesystem sul quale si appoggiava GNU/Linux sia stato *minix*.

Le caratteristiche generali del filesystem ext2 sono riportate in tab. 3.3

ext2

Descrizione	Valore
Dimensione dei blocchi	1, 2 o 4 KiB
Dimensione massima del filesystem	16 TiB (con blocchi da 4 KiB)
Dimensione massima dei file	4 TiB (con blocchi da 4 KiB)
Lunghezza massima dei nomi	255 caratteri
Supporto dei symbolic link	Sì
Supporto dei fast link	Sì
Supporto ctime, mtime e atime	Sì
Spazio riservato per il superuser	Sì
Attributi estesi dei file	Sì

Tabella 3.3: Caratteristiche principali del filesystem ext2.

3.6.1 Struttura di base

Il filesystem ext2 definisce la minima unità di memorizzazione dei dati, il *blocco* (*block*), generalmente delle dimensioni di 1.024 byte, ma la sua dimensione può essere scelta tra 1.024, 2.048 o 4.096 byte in fase di creazione del filesystem.

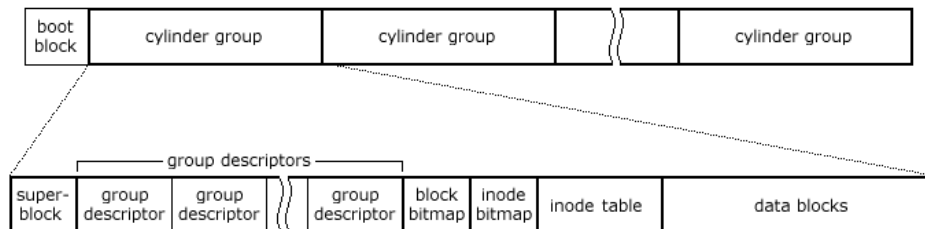


Figura 3.5: Struttura del filesystem ext2.

I blocchi vengono raggruppati in **cylinder group** (v. fig. 3.5) in modo da suddividere l'intero filesystem in entità autonome allo scopo di limitare eventuali errori che possono verificarsi nella struttura del filesystem (un errore è limitato all'interno di un cylinder group e non compromette l'intero filesystem) e per ridurre i tempi di accesso alle informazioni memorizzando i file nelle vicinanze delle directory che li contengono. Ogni cylinder group si suddivide a sua volta in

cylinder group

Superblock contiene informazioni relative all'intero filesystem, secondo le seguenti tipologie

- caratteristiche tipiche (identificazione del filesystem¹⁰, dimensioni, struttura, ...) – valori stabiliti al momento della creazione del filesystem e non possono essere più modificati, se non distruggendo il filesystem e creandone un altro (perdendo in questo modo le informazioni in esso eventualmente memorizzate);
- parametri modificabili (massimo numero di mount, massimo numero di mount prima di effettuare il controllo di errori, ...) – valori modificabili dal superuser per mezzo del comando `tune2fs`;
- stato (stato di montaggio del filesystem, numero di blocchi liberi, numero di mount correnti, ...) – informazioni relative all'uso corrente del filesystem;

Il superblock di riferimento è memorizzato a partire dal 1.024° byte del disco e la sua dimensione è 1.024 byte, ma per scopi di backup viene copiato in ogni cylinder group, anche se il kernel utilizza sempre quello di riferimento (l'originale).

La struttura del superblock è riportata di seguito

```
struct ext2_super_block
{
    __u32    s_inodes_count;        /* Inodes count */
    __u32    s_blocks_count;       /* Blocks count */
    __u32    s_r_blocks_count;     /* Superuser reserved blocks count */
    __u32    s_free_blocks_count;  /* Free blocks count */
    __u32    s_free_inodes_count;  /* Free inodes count */
    __u32    s_first_data_block;   /* First Data Block */
    __u32    s_log_block_size;    /* Block size */
    __s32    s_log_frag_size;     /* Fragment size */
    __u32    s_blocks_per_group;  /* # Blocks per group */
    __u32    s_frags_per_group;   /* # Fragments per group */
    __u32    s_inodes_per_group;  /* # Inodes per group */
}
```

¹⁰ext2 è identificato dal valore EF53_H memorizzato nel 57° e 58° byte del superblock.

```

__u32  s_mtime;                /* Mount time */
__u32  s_wtime;                /* Write time */
__u16  s_mnt_count;            /* Mount count */
__s16  s_max_mnt_count;        /* Maximal mount count */
__u16  s_magic;                /* Magic signature */
__u16  s_state;                /* File system state */
__u16  s_errors;               /* Behaviour when detecting errors */
__u16  s_pad;
__u32  s_lastcheck;            /* time of last check */
__u32  s_checkinterval;        /* max. time between checks */
__u32  s_creator_os;           /* OS */
__u32  s_rev_level;            /* Revision level */
__u16  s_def_resuid;           /* Default uid for reserved blocks */
__u16  s_def_resgid;           /* Default gid for reserved blocks */
__u32  s_reserved[235];        /* Padding to the end of the block */
}

```

Il campo `s_log_block_size` contiene la dimensione dei blocchi del filesystem espressa in termini della potenza di 2 relativa alla base di 1.024, cioè la dimensione dei blocchi, espressa in byte, è data da

$$\text{block size} = 2^{\text{s_log_block_size}} \times 1.024$$

per cui se `s_log_block_size` contiene il valore 0, significa che la dimensione dei blocchi del filesystem è 1.024 byte, se contiene il valore 1 è 2.048 byte, ...;

Group descriptor ogni *group descriptor* contiene informazioni relative al cylinder group a cui si riferisce (un riferimento alla *blocks bitmap*, uno alla *inode bitmap*, uno alla *inode table*, il numero dei blocchi liberi, quello degli inode liberi e delle directory presenti nel cylinder group).

Il numero dei group descriptor, ovvero dei cylinder group, è dato da

$$\text{Numero di group descriptor} = 1 + \frac{\text{s_blocks_count} - 1}{\text{s_blocks_per_group}}$$

dove `s_blocks_count` e `s_blocks_per_group` sono campi del superblock.

In particolare, il numero delle directory presenti nel cylinder group viene utilizzato dal filesystem alla creazione di una nuova directory. Per velocizzare gli accessi, ext2 tenta di memorizzare i file nello stesso cylinder group in cui è memorizzata la directory che li contiene, quindi la nuova directory verrà creata nel cylinder group che risulta più libero.

La struttura dei group descriptor è riportata di seguito

```

struct ext2_group_desc
{
    unsigned long bg_block_bitmap;    /* Blocks bitmap block */
    unsigned long bg_inode_bitmap;    /* Inodes bitmap block */
    unsigned long bg_inode_table;     /* Inodes table block */
    unsigned short bg_free_blocks_count; /* Free blocks count */
    unsigned short bg_free_inodes_count; /* Free inodes count */
    unsigned short bg_used_dirs_count; /* Directories count */
    unsigned short bg_pad;
    unsigned long bg_reserved[3];
}

```

Il campo `bg_inode_table` contiene il numero che identifica il primo blocco della *inode table*.

Anche i *group descriptor* vengono copiati in ogni cylinder group per scopi di backup;

Data block bitmap è una sequenza di bit, ognuno dei quali indica se il corrispondente data block del cylinder group è libero o allocato. La sua dimensione è quella di un blocco (tipicamente 1.024 byte) e quindi saranno individuati 8.192 (1.024×8) data block per ogni cylinder group;

Tipicamente, nel *data block bitmap*, tutti i bit di un byte contengono il valore 0 o 1, ovvero i byte sono 00_H o FF_H per il fatto che il kernel effettua un'ottimizzazione raggruppando i dati nei blocchi fisicamente adiacenti.

Inode bitmap analogo al *data block bitmap* ma si riferisce agli inode (che generalmente sono in numero minore dei data blocks). Per il ragionamento effettuato precedentemente, con blocchi di 1.024 byte, possono esistere al massimo 8.192 inode per cylinder group;

Inode table è la porzione del filesystem che contiene gli inode del cylinder group, ovvero strutture adibite alla memorizzazione delle proprietà degli oggetti del filesystem (file, directory, ...).

Un inode è formato da una struttura di 128 byte, che è riportata di seguito

```
struct ext2_inode
{
    unsigned short i_mode;           /* File mode */
    unsigned short i_uid;           /* Owner Uid */
    unsigned long i_size;            /* Size in bytes */
    unsigned long i_atime;           /* Access time */
    unsigned long i_ctime;           /* Creation time */
    unsigned long i_mtime;           /* Modification time */
    unsigned long i_dtime;           /* Deletion Time */
    unsigned short i_gid;            /* Group Id */
    unsigned short i_links_count;    /* Links count, max 32000 */
    unsigned long i_blocks;          /* Blocks count */
    unsigned long i_flags;           /* File flags */
    unsigned long i_reserved1;
    unsigned long i_block[15];       /* Pointers to blocks */
    unsigned long i_generation;      /* File version (for NFS) */
    unsigned long i_file_acl;        /* File ACL */
    unsigned long i_dir_acl;         /* Directory ACL */
    unsigned long i_faddr;           /* Fragment address */
    unsigned char i_frag;            /* Fragment number */
    unsigned char i_fsize;           /* Fragment size */
    unsigned short i_pad1;
    __u16 i_uid_high;                /* High order part of uid */
    __u16 i_gid_high;                /* High order part of gid */
    __u32 i_reserved2;
}
```

Data blocks è la porzione del filesystem che contiene i blocchi (*data blocks*) del cylinder group, adibiti alla memorizzazione delle informazioni contenute all'interno dei file.

Con la struttura del filesystem così suddivisa, si ha sì una leggera ridondanza di informazioni, ma se uno dei superblock o group descriptor risulta errato, esso può essere agevolmente ripristinato da una delle copie presenti nei vari cylinder group.

3.6.2 I file

In generale, un **file** contiene delle informazioni (*data*) ed è caratterizzato da alcune proprietà, come il tipo, la dimensione, i permessi di accesso, ..., ovvero quelli che vengono denominati *metadati* (*metadata*). Il filesystem ext2 memorizza i metadati di un *file*

file in un'apposita struttura, detta *index node*, o più comunemente **inode**, mentre le informazioni contenute nel file sono memorizzate nei blocchi, denominati anche *data block*. *inode*

Ogni *inode* ha la struttura riportata in sez. 3.6.1 ed il numero degli inode è fissato alla creazione del filesystem (per default viene creato un inode ogni 4.096 byte disponibili nella partizione considerata). Quando tutti gli inode saranno utilizzati (allocati) da altrettanti file, non sarà più possibile creare altri file anche se ci sono data block liberi.

In un inode ext2 vengono memorizzati i metadati di un file, tra cui

Tipo indica il tipo di file (regular file, directory, symbolic link, ...).¹¹

UID proprietario identifica l'utente proprietario del file (è il suo UID¹²);

GID proprietario identifica il gruppo proprietario del file (è il suo GID¹³);

Permessi (*mode*) sono i permessi di accesso al file (v. sez. 3.6.5);

Dimensione indica la lunghezza del file in byte.

Numero dei data block indica il numero dei data block utilizzati per memorizzare il file.

data/ora ultima modifica (*mtime*) è il riferimento al momento nel quale il contenuto del file ha subito l'ultima modifica (quella più recente);

data/ora ultimo cambiamento dei metadati (*ctime*) è il riferimento al momento nel quale l'inode relativo al file in questione ha subito l'ultima modifica, ma il suo contenuto non è stato modificato;

data/ora ultimo accesso (*atime*) è il riferimento al momento nel quale il file è stato acceduto l'ultima volta;

Numero degli hard link indica il numero degli *hard link* associati al file (v. sez. 3.6.3);

Riferimenti ai data block sono i riferimenti ai blocchi sui quali sono memorizzate le informazioni contenute nel file (v. fig. 3.6). In particolare per ogni inode possono essere memorizzati fino a 12 riferimenti a blocchi contenenti dati (*direct block*), un riferimento ad un blocco che a sua volta contiene un elenco di direct block (*indirect block*), un riferimento ad un blocco che a sua volta contiene un elenco di indirect block (*double indirect block*) ed un riferimento ad un blocco che a sua volta contiene un elenco di double indirect block (*triple indirect block*).

Ogni riferimento ai blocchi (direct o indirect) è relativo ad un blocco del filesystem, che nel caso di indirect block contiene i riferimenti (ognuno di 4 byte) ad altri blocchi.

Supponendo di avere blocchi di dimensione B un file potrà contenere al massimo $12 + \frac{B}{4} + (\frac{B}{4})^2 + (\frac{B}{4})^3$ blocchi. Considerando la dimensione dei blocchi di 1.024 byte, la dimensione massima di un file permessa da ext2 è 17.247.252.480 byte (poco più di 16 GiB) (16.843.020 blocchi), mentre con blocchi da 4.096 byte si ha una dimensione massima dei file di 4.402.345.721.856 byte (circa 4 TiB) (1.074.791.436 blocchi).

Nel caso in cui, all'interno di un inode, il riferimento ad un blocco di dati sia nullo, ovvero contenga il valore 0, indica che esso non si riferisce a nessun blocco (la numerazione dei blocchi e quella degli inode inizia da 1). Questo è il meccanismo di gestione che ext2 attua nel caso di *sparse file*¹⁴.

¹¹v. sez. 3.6.3.

¹²v. cap. 5.

¹³v. cap. 5.

¹⁴v. pag. 95.

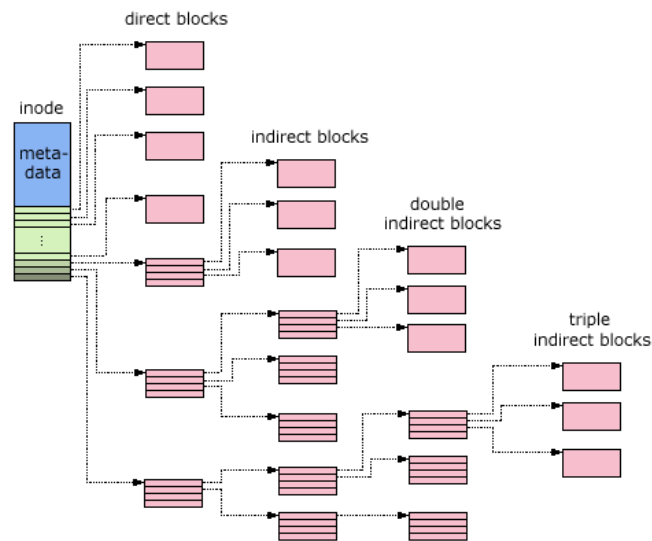


Figura 3.6: Gli inode ed i riferimenti diretti ed indiretti ai blocchi di dati.

La dimensione massima di un file è la minore tra quella consentita dal filesystem e quella consentita dal kernel. Infatti, su architetture a 32 bit, le system call messe a disposizione dal kernel per l'accesso al filesystem utilizzano parametri esprimibili su 32 bit, e quindi possono gestire file con dimensioni inferiori o uguali a $2^{31} - 1 = 2.147.483.647$ byte (2 GiB). Con il kernel 2.4 (o successivo) è stato implementato il **LFS** (Large File Support), ovvero sono state messe a disposizione delle nuove system call per l'accesso al filesystem che utilizzano parametri esprimibili su 64 bit, e quindi, teoricamente, il kernel non ha problemi a gestire file di dimensioni fino a $2^{63} - 1 = 9.223.372.036.854.775.807$ byte (circa 8 EiB), per cui il limite della dimensione massima dei file dipende essenzialmente dal filesystem.

LFS

In un inode del filesystem ext2 sono memorizzati particolari flag (o attributi) che specificano appositi comportamenti di cui il sistema deve tenere di conto per tale inode. Tali flag sono descritti di seguito

Secure Deletion indica al sistema di azzerare i relativi data block quando l'inode viene cancellato (solo per le versioni del kernel Linux 1.0 e 1.2);

No atime update indica al sistema di non aggiornare il campo `i_atime` dell'inode;

Synchronous update indica al sistema di effettuare la scrittura sincrone su tale inode (nessuna bufferizzazione);

Append only indica al sistema di permettere soltanto l'accodamento (*appending*) di informazioni per tale inode (nel caso di directory non permette la cancellazione dei file in essa contenuti);

Immutable indica al sistema di non permettere nessuna modifica ai dati e metadati relativi all'inode;

No dump indica al comando `dump` di non considerare l'inode;

Compress indica al sistema di comprimere il file relativo dopo una scrittura sullo stesso e di decomprimerlo prima di una lettura (non è nel kernel standard);

Undeletable indica al sistema di preservare i data block relativi quando l'inode viene cancellato, per poter permettere un successivo ripristino (non è nel kernel standard);

No tail-merging indica al sistema che il relativo file non deve avere un fragment nell'ultimo blocco in comune con un altro file (per i filesystem che supportano il *tail-merging*);

Journaling (ext3)¹⁵ indica al sistema di aggiornare il journal prima di effettuare una variazione sull'inode (ha senso soltanto se il filesystem è montato con le opzioni *data=ordered* o *data=writeback*);

Tali flag possono essere gestiti per mezzo dei comandi **lsattr** (**list attributes** – man page **lsattr(1)**) e **chattr** (**change attributes** – man page **chattr(1)**).

Comando: **lsattr**
 Path: **/usr/bin/lsattr**
 SINTASSI
\$ lsattr [*option*] [*file_name*]

DESCRIZIONE

Visualizza gli attributi di un inode del filesystem ext2.

option indica la modalità di funzionamento di **lsattr**. Può assumere i seguenti valori

- R** visualizza gli attributi delle directory e dei file in esse contenuti in maniera ricorsiva;
- a** visualizza tutti i file contenuti nelle directory, compresi quelli che iniziano con il carattere '.';
- d** tratta le directory alla stregua dei file, piuttosto che visualizzare il loro contenuto;
- v** visualizza la versione dei file;

file_name indica il file (o directory) di cui visualizzare gli attributi;

Comando: **chattr**
 Path: **/usr/bin/chattr**
 SINTASSI
\$ chattr [*option*] *file_name*

DESCRIZIONE

Modifica gli attributi di un inode del filesystem ext2.

option indica la modalità di funzionamento di **chattr**. Può assumere i seguenti valori

- R** modifica gli attributi delle directory e dei file in esse contenuti in maniera ricorsiva;
- V** visualizza gli attributi modificati;
- v version**
 imposta la variazione del file secondo quanto indicato da *version*;
- {+|-|=}attr**
 imposta (+), rimuove (-) o indica esattamente (=) l'elenco degli attributi specificato da *attr* (i possibili valori sono riportati in tab. 3.4);

file_name indica il file (o directory) al quale modificare gli attributi;

In particolare, i flag *Append only* e *Immutable* possono rivelarsi utili come difesa contro eventuali malintenzionati che si sono intrufolati nel sistema.

Generalmente un filesystem memorizza il contenuto di un file in un numero intero di blocchi, quindi se un file contiene una quantità di informazioni che non è un multiplo della dimensione di un blocco, parte dello spazio dell'ultimo blocco del file risulta inutilizzato, quindi mediamente si ha lo spreco della metà della dimensione di

¹⁵v. sez. 3.7.

attr	Attributo dell'inode ext2
S	Synchronous update
a	Append only
c	Compressed
i	Immutable (superuser)
d	No dump
s	Secure deletion
u	Undeletable
t	No tail-merging
j	Journaling

Tabella 3.4: Possibili attributi modificabili con `chattr`.

un blocco per ogni file (ad esempio, per memorizzare un file di 1.025 byte in un filesystem in cui la dimensione dei blocchi è 1.024 byte, vengono utilizzati 2 blocchi, ovvero 2.048 byte, di cui gli ultimi 1.023 sono inutilizzati). Per ovviare a tale inconveniente (**frammentazione interna**), è previsto l'utilizzo di porzioni di blocchi (i **fragment**), ma questa caratteristica non è attualmente utilizzata.

*frammentazione interna
fragment*

Il filesystem ext2 riserva alcuni inode per memorizzare informazioni particolari, ad esempio

- l'inode 1 contiene l'elenco dei blocchi danneggiati;
- l'inode 2 contiene la root directory del filesystem;
- l'inode 4 contiene informazioni relative all'ACL (Access Control List);
- l'inode 5 contiene informazioni relative al boot loader;
- l'inode 6 contiene informazioni relative ai file ripristinati;
- l'inode 7 contiene informazioni relative ai group descriptor;

Il primo inode utilizzato per memorizzare informazioni generiche è il numero 11.

3.6.3 I tipi di file

In GNU/Linux possono essere utilizzati vari tipi di file, ognuno identificato da uno specifico codice che il filesystem ext2 memorizza nell'inode relativo al file considerato.

Codice	Tipo di file
1	FIFO (o named pipe)
2	dispositivo a caratteri
4	directory
6	dispositivo a blocchi
10	file standard (o regular file)
12	symbolic link (o symlink)
14	socket

Tabella 3.5: Possibili tipi di file.

La tab. 3.5 riporta i tipi di file riconosciuti da GNU/Linux che sono illustrati di seguito.

I file “standard” o *regular file*

A tale categoria appartiene la maggior parte dei file presenti sui filesystem. Tale tipo rappresenta il file nella sua accezione di base, ovvero un contenitore di informazioni senza caratteristiche particolari. I file di questo tipo sono anche detti *regular file*.

Le directory

Una **directory** è un file particolare che contiene un elenco di elementi detti **dentry** (directory **entry**), ognuno dei quali è costituito da (v. fig. 3.7) *directory dentry*

- il numero di inode al quale si riferisce;
- la lunghezza della dentry;
- la lunghezza del nome del file;
- il tipo del file;
- il nome de file;

inode	entry length	entry name length	entry type	entry name

Figura 3.7: Le directory.

I numeri degli inode presenti in tale elenco, costituiscono i riferimenti agli inode relativi ai file accessibili da tale directory, cioè i file (o directory) contenuti all'interno della directory. Più in dettaglio, le dentry hanno la struttura riportata di seguito

```
struct ext2_dir_entry_2
{
    __u32    inode;           /* Inode number */
    __u16    rec_len;         /* Directory entry length */
    __u8     name_len;        /* File name length */
    __u8     file_type;       /* File type */
    char     name[EXT2_NAME_LEN]; /* File name */
}
```

Si tenga presente che la dimensione del campo **name** è variabile, ovvero contiene il nome del file (o directory) con allineamento a double word per mezzo di eventuali caratteri nulli aggiunti alla fine del nome.

??? esempio ???

Il campo **name_len** contiene la lunghezza effettiva del nome del file, ovvero il numero di caratteri dai quali esso è effettivamente costituito, mentre il campo **rec_len** indica la lunghezza dell'intera dentry. Il campo **file_type** contiene un numero che indica il tipo del file secondo quanto riportato in tab. 3.6.

file_type	Tipo di file
0	Sconosciuto
1	File standard (o regular file)
2	Directory
3	Dispositivo a caratteri
4	Dispositivo a blocchi
5	FIFO (o named pipe)
6	Socket
7	Symbolic link (o symlink)

Tabella 3.6: Possibili valori del campo **file_type** delle dentry.

È opportuno notare il fatto che il nome dei file non fa parte dei metadati (non è contenuto negli inode relativi ai file stessi), ma è memorizzato nelle dentry (nelle directory).

Quando viene specificato il percorso di un file (*path*), il kernel ricerca nelle dentry delle varie directory che costituiscono il path del file, il nome indicato per trovare di volta in volta il relativo numero di inode e per poter così accedere alle informazioni in esso contenute. Una volta raggiunto l'inode relativo al file indicato, il sistema ha i riferimenti per poter accedere alla parte delle informazioni contenute nel file stesso (v. fig. 3.8).

file path: /home/daniele/prova

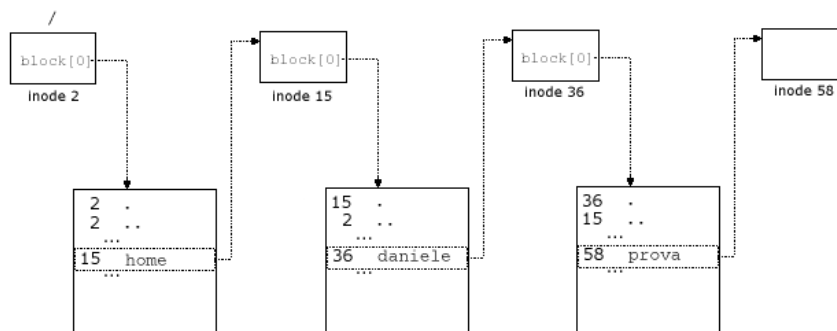


Figura 3.8: Esempio di accesso ad un file.

I link

Nei sistemi Unix-like un file è essenzialmente un inode all'interno del quale non viene memorizzato il nome del file, poiché esso non è una caratteristica intrinseca del file stesso, ma è semplicemente un'etichetta memorizzata nelle dentry che serve come riferimento per poter accedere all'inode relativo al file. Quindi, sebbene un nome di file individui un solo inode, lo stesso inode può essere individuato da più nomi di file. L'associazione tra il nome del file e l'inode a cui si riferisce, memorizzato nelle dentry, è detta **hard link** (*physical link*, *collegamento fisico* o *collegamento diretto*). Dunque, un file (inode) può avere più hard link (più nomi) che si riferiscono ad esso.

hard link

Quando viene creato un hard link, viene incrementato il contatore dei riferimenti nell'inode relativo. In questo modo il sistema tiene traccia di quanti hard link sono riferiti ad ogni inode. L'eliminazione di un hard link non implica l'eliminazione di un file, ovvero dell'inode relativo, ma un file viene effettivamente eliminato dal filesystem soltanto quando l'ultimo degli hard link ad esso relativi viene eliminato e nessun processo sta utilizzando il file (inode) in questione.¹⁶

Alla creazione di un file viene sempre creato un hard link ad esso relativo, contenente il suo primo nome, all'interno della directory dalla quale esso è direttamente accessibile.

È possibile creare hard link solo sui filesystem che li supportano e comunque sia l'inode ed il relativo link devono trovarsi sullo stesso filesystem;

In GNU/Linux è possibile creare hard link relativi soltanto a file, non a directory, poiché sarebbe altrimenti possibile creare dei riferimenti circolari (catene di collegamenti chiuse su sé stesse), complessi da gestire e soprattutto da rimuovere.

GNU/Linux permette la creazione di file particolari (come le directory) che il sistema riconosce come tali: i **symbolic link** (*symlink*, *soft link* o *collegamenti simbolici*). Essi sono dei file che contengono soltanto il path relativo al file o directory a cui si riferiscono. Quando i comandi di sistema hanno a che fare con un symbolic link essi per default

symbolic link

¹⁶il kernel tiene traccia file aperti dai vari processi e quindi i riferimenti ai relativi inode.

considerano direttamente il file a cui essi si riferiscono (ricavandolo dal loro contenuto), piuttosto che considerarli alla stregua degli altri file. Si dice in genere che i comandi “seguono” i symbolic link (*symbolic link dereference*), nel senso che si accorgono che si tratta di un symbolic link e quindi compiono l’operazione sul file o directory a cui il symbolic link si riferisce (le *system call* del kernel fanno automaticamente l’operazione di *dereference* dei symbolic link), tranne l’operazione di eliminazione (**rm** – v. sez. 4.5) che si limita a cancellare soltanto il symbolic link e non il file a cui esso si riferisce. È comunque generalmente possibile indicare ai vari comandi, con un’apposita opzione, di non seguire automaticamente i symlink ma di trattarli come tutti gli altri file.

È importante sottolineare il fatto che il sistema non tiene conto, all’interno degli inode, dei symbolic link che si riferiscono ad essi, come invece avviene per gli hard link: quindi è possibile ottenere dei *dangling link* (collegamenti penzolanti) ovvero dei symbolic link che non sono collegati ad un file o ad una directory esistente. Infatti il sistema non attua nessun controllo alla creazione di un symbolic link del fatto che il file o la directory a cui esso si riferisce esista e neanche alla cancellazione di un file (rimozione dell’ultimo hard link) il sistema tiene conto di eventuali symbolic link che si riferiscono ad esso.

Per contro, i symbolic link non hanno le restrizioni degli hard link, quindi è possibile anche creare symbolic link relativi a file o directory situati in filesystem diversi da quello in cui viene creato il link.

Ad esempio, è possibile creare un hard link al file `~/ .bashrc` con il comando

```
$ ln myhlink ~/ .bashrc
```

e così i nomi `myhlink` (nella working directory) e `.bashrc` (nella home directory) si riferiranno in seguito allo stesso identico file che sarà fisicamente rimosso dal sistema quando lo saranno entrambi i link ad esso riferiti.

È possibile creare un symbolic link al file `~/ .bashrc` con il comando

```
$ ln -s myslink ~/ .bashrc
```

anche nel caso in cui il file `~/ .bashrc` non esistesse, nel qual caso `myslink` sarebbe un *dangling link* poiché non si riferirebbe effettivamente ad un file (o directory) esistente.

Con i symbolic link possono essere creati dei riferimenti circolari (catene di collegamenti chiuse su sé stesse) che potrebbero essere problematici per certi comandi. Si supponga infatti di creare un symbolic link `/boot/myboot` alla directory `/boot` e di visualizzare il contenuto della directory `/boot` in modo da scendere ricorsivamente in tutte le sottodirectory in essa contenute, impartendo il comando

```
$ ls -r /boot
```

Così facendo il comando `ls` esaminerebbe la directory `/boot`, quindi la directory `/boot/myboot`, ovvero la directory `/boot` stessa che ovviamente contiene `/boot/myboot` e quindi esaminerebbe `/boot/myboot/myboot`, e così via, entrando in un loop infinito. In pratica ciò non genera un loop infinito poiché il sistema è limitato ad effettuare un determinato numero massimo di dereferenziazioni (seguimenti) di symbolic link quando viene effettuata un’operazione di risoluzione del nome di un oggetto del filesystem.

I link vengono creati per mezzo del comando `ln` (man page `ln(1)`).

```
Comando: ln
Path: /bin/ln
SINTASSI
$ ln [option] target [link_name]
```

DESCRIZIONE

option indica la modalità di funzionamento di `ln`. Può assumere i seguenti valori

```

-b | --backup=control
    crea una copia di backup di ogni file esistente (specificato da target).
    L'argomento control può assumere i seguenti valori
    none | off
        non effettua alcun backup;
    numbered | t
        effettua backup numerati;
    existing | nil
        effettua backup numerati, se questi esistono, altrimenti ef-
        fettua dei backup semplici;
    simple | never
        effettua dei backup semplici;
-d | -F | --directory
    crea un hard link sulla directory (soltanto il superuser);
-f | --force
    cancella i file di destinazione eventualmente esistenti;
-n | --no-dereference
    tratta target come un file normale anche se questo è un symbolic link
    ad un altro file o directory;
-i | --interactive
    chiede conferma prima di eliminare i file eventualmente esistenti;
-s | --symbolic
    crea un symbolic link (anziché un hard link);
-S suffix | --suffix=suffix
    imposta il suffisso con il quale viene salvato il file di backup (il suffisso
    di default è '~');
--target-directory=directory
    indica la directory (directory) in cui creare il link;
-v | --verbose
    visualizza il nome di ogni file prima di creare il link;
--help visualizza un aiuto sommario di ln;
--version
    visualizza la versione di ln;

target indica il file o la directory in riferimento al quale creare un link. È possibile
specificare un elenco di file o directory (separati da uno spazio) di cui si vuol
creare un link - in questo caso l'ultimo argomento deve essere una directory
che conterrà un link per ogni oggetto del filesystem specificato nell'elenco;

link_name indica il nome da assegnare al link creato (se non è specificato il link
avrà lo stesso nome del target);

```

La creazione di un link, sia hard che symbolic, non occupa molto spazio di disco, poiché non effettua la copia del contenuto del file.

Il filesystem ext2 gestisce anche i cosiddetti **fast link**, cioè speciali symbolic link *fast link* che si riferiscono ad un oggetto del filesystem il cui path ha una lunghezza inferiore a 60 caratteri: non vengono utilizzati dei blocchi per memorizzare il path dell'oggetto del filesystem al quale il symbolic (fast) link si riferisce, ma questo viene memorizzato direttamente nello spazio usualmente riservato per memorizzare i riferimenti ai blocchi contenenti i dati del file (lo spazio a disposizione è proprio $15 \times 4 = 60$ byte - v. i_block della struttura dell'inode riportata a pag. 99). I fast link sono quindi caratterizzati dal fatto che utilizzano 0 blocchi per la memorizzazione del loro contenuto.

I file speciali

Alcuni file in GNU/Linux hanno caratteristiche particolari e per questo sono detti **file speciali** *file speciali* e si dividono in *FIFO* (o *named pipe*), *file di dispositivo* e *socket* (Unix domain socket). Le FIFO ed i file di dispositivo possono essere creati con il comando **mknod** (man page **mknod(1)**) (**make inode**).

Comando: **mknod**
 Path: **/bin/mknod**

SINTASSI

```
$ mknod [option] name type [major minor]
```

DESCRIZIONE

option indica la modalità di funzionamento di **mknod**. Può assumere i seguenti valori

```
-m mode | --mode=mode      imposta i permessi (come in chmod);17
--help visualizza un aiuto sommario di mknod;
--version visualizza la versione di mknod;
```

name specifica il nome del file da creare;

type specifica il tipo di file da creare secondo quanto indicato in tab. 3.7;

type	Descrizione
b	file di dispositivo a blocchi (buffered).
c u	file di dispositivo a caratteri (unbuffered).
p	FIFO o named pipe.

Tabella 3.7: Possibili valori di *type* nel comando **mknod**.

major specifica il major number (solo per i file di dispositivo);

minor specifica il minor number (solo per i file di dispositivo);

mentre i socket possono essere creati con il comando **socket** (man page **socket(2)**).

Le FIFO o named pipe

FIFO
named pipe

Una **FIFO**, detta anche **named pipe**, è un file il cui funzionamento è simile a quello di una *pipe*, cioè come memoria di appoggio per lo scambio di informazioni tra processi. Quando i processi comunicano tra loro attraverso una FIFO, il sistema non utilizza il filesystem, ma il transito delle informazioni avviene direttamente in memoria centrale. Il file serve soltanto come riferimento univoco alla FIFO per i processi che la vogliono utilizzare per comunicare tra loro.¹⁸

Il nome FIFO deriva dal fatto che essa opera come una coda, ovvero è gestita in maniera tale che il primo dato in essa inserito è anche quello che viene estratto per primo (First In, First Out).

??? immagina ???

Una FIFO può essere creata utilizzando il comando **mkfifo** (man page **mkfifo(1)**) (**make fifo**).

Comando: **mkfifo**

Path: **/bin/mkfifo**

SINTASSI

```
$ mkfifo [option] name
```

DESCRIZIONE

option specifica la modalità di funzionamento di **mkfifo**. Può assumere i seguenti valori

```
-m mode | --mode=mode      specifica i permessi da associare al file (come chmod);
--help visualizza un aiuto sommario di mkfifo;
--version visualizza la versione di mkfifo;
```

name indica il nome del file da creare;

¹⁷v. sez. 3.6.5

¹⁸v. anche sez. 6.5.

Un processo può accedere in lettura e/o scrittura ad una FIFO, ovvero all'inode (file) presente sul filesystem, soltanto se ha i diritti per poterlo fare, come avviene per tutti gli altri file.

I file di dispositivo

I **file di dispositivo** o **device file** sono particolari file che i sistemi Unix-like associano ai dispositivi fisici. Ad esempio, ad un terminale (monitor) è associato un file di dispositivo che rappresenta per il sistema il buffer di output del terminale stesso, ovvero i caratteri scritti in quel file verranno visualizzati sullo schermo; allo stesso modo la tastiera è associata ad un file, ovvero tutto ciò che viene digitato dalla tastiera finisce in tale file ed il sistema operativo leggerà i comandi impartiti da tale file.

file di dispositivo
device file

Per poter funzionare correttamente con il sistema, ogni dispositivo necessita di un apposito software denominato **driver**. Tale software ha il compito di far “vedere” il dispositivo al sistema, nel senso che si preoccupa di gestire i meccanismi per la comunicazione con il particolare dispositivo considerato. Quindi è necessario creare anche un file di dispositivo, con le opportune caratteristiche, da associare al dispositivo stesso.

driver

Un file di dispositivo è caratterizzato da

nome è il nome del file che identifica il dispositivo (non ha nessuna influenza sul funzionamento del dispositivo);

tipo è il tipo del dispositivo. Può essere a blocchi (hard disk, CD-ROM, ...) o a caratteri (terminali, unità a nastro, ...);

major number

indica il driver che gestisce il funzionamento del dispositivo;

minor number

identifica il dispositivo per il driver;

Ad esempio

```
brw-rw---- 1 root disk 8, 0 May 5 1998 sda
```

è il file di dispositivo associato al primo disco SCSI (il nome del file è infatti **sda** e come illustrato in sez. 3.3 **sd** indica un disco SCSI ed **a** indica il primo disco – questa è soltanto la convenzione utilizzata dal sistema, ma non ha effettivamente niente a che fare con il dispositivo stesso). L'indicazione **b** come primo carattere della stringa dei permessi (all'inizio della riga, v. sez. 3.3 e 3.10) indica il fatto che il file rappresenta un dispositivo a blocchi. I valori 8 e 0 indicano rispettivamente il major e minor number del dispositivo: 8 è il driver SCSI e 0 identifica il primo disco per il driver SCSI.

Un file di dispositivo può essere creato con il comando **mknod**. Ad esempio, se si vuol creare un file di dispositivo relativo ad un controller RAID¹⁹, questo deve essere di tipo a blocchi e relativo ad uno specifico major number (es. 72). Se si desidera specificare l'intero RAID deve essere indicato il minor number 0. Il nome del file di dispositivo potrebbe essere scelto convenzionalmente come “c0d0” (controller 0, disco 0). Quindi esso potrebbe essere creato con il comando

```
# mknod /dev/c0d0 b 72 0
```

I file di dispositivo sono generalmente contenuti nella directory **/dev** (v. sez. 3.12).

¹⁹il RAID sarà trattato in sez. 3.18.

I socket

I socket²⁰ costituiscono un'interfaccia di comunicazione tra processi. I socket locali, o *Unix domain socket* (il nome deriva dal fatto che tale meccanismo di comunicazione è nato in ambiente Unix nel 1983) presi in considerazione in questo contesto, costituiscono il meccanismo di comunicazione tra processi esistenti su una stessa macchina. L'interfaccia dei socket viene attualmente utilizzata per la comunicazione tra processi esistenti anche su macchine diverse, ovvero per realizzare la comunicazione di rete (v. parte II).

Un socket rappresenta un canale di comunicazione tra due processi, nel quale si possono scrivere informazioni e dal quale si possono leggerle. L'interfaccia dei socket è caratterizzata da un'elevata genericità e flessibilità, tanto che si è diffusa in tutti gli ambienti di programmazione, anche in quelli non legati al mondo Unix.

È possibile creare dei socket locali che siano visibili sul filesystem, come per le FIFO. L'accesso a tali socket deve tener conto dei permessi relativi al file che li rappresenta. Ad esempio la creazione di un socket viene rifiutata dal sistema all'interno di una directory nella quale il processo che tenta di creare il socket non ha i diritti di lettura ed esecuzione. Allo stesso modo, per connettersi ad un socket, il processo deve avere i diritti di lettura e scrittura sullo stesso.

???

3.6.4 I formati dei file

formato

Un file contiene una sequenza di byte che assumono significati diversi dipendentemente dal modo in cui questi vengono interpretati: ciò determina il **formato** del file. Se l'interpretazione dei byte contenuti in un file deve essere fatta mediante la tabella ASCII, allora si ha a che fare con un file in *formato testo*, altrimenti si parla genericamente di file in *formato binario*.

Di seguito è riportata la visualizzazione rispettivamente di un file in formato binario (per la precisione si tratta di un file che contiene un'immagine secondo il formato **PNG** – Portable Network Graphic) e di uno in formato testo. Il file binario è il seguente

?PNG

```
IHDR0 W ?? bKGD??????? pHYs      ???tIME? 7/-0??IDATx???A
?U? ? y ?X8?hS??? AE??0?F' ? ??u>i?# #? 0??="""r ?}?? C??EgV??P??i?$$$$?ciV
L?mY?????0iV8C (?5? ?m?Z) ? ???+ ? HS?@, ??? Z???Xo'?? ò?&8?e-ç??S?c "RD??U?^
??P??1? ^ ?cIEND?B'?xtermxtermxtermxterm
```

mentre il file di testo è visualizzato in maniera comprensibile

```
Ciao,
Questo è un esempio di file di testo.
Come verrà fuori?
```

```
Ciao a tutti,
Daniele.
```

Questo deriva dal fatto che il comando utilizzato per la visualizzazione del file è **cat**, che invia i byte contenuti nel file, uno dopo l'altro, sullo standard output che li visualizza interpretandoli secondo l'ASCII.

In particolare, si parla di formato testo puro quando un file non contiene nessun carattere di controllo speciale, ovvero è composto soltanto da caratteri alfanumerici, segni di punteggiatura e poco altro. Nessuna sequenza di caratteri assume un significato particolare, ogni carattere rappresenta letteralmente quello che è: si dice in tal caso che il file non ha nessun tipo di *formattazione*. A tale tipo appartengono la totalità degli script e la quasi totalità dei file di configurazione di un sistema Unix-like.

²⁰la traduzione italiana sarebbe “presa”.

L'unica formattazione di un file in formato testo puro è relativa alla rappresentazione del carattere di fine riga: i sistemi Unix-like utilizzano il carattere ASCII LF, mentre i sistemi che derivano dal DOS usano la sequenza di caratteri CR e LF, ed i sistemi *Apple* utilizzano soltanto il carattere CR.

Il comando **file** (man page **file(1)**) fornisce indicazioni sul formato di un file effettuando tre tipi di test, nel seguente ordine

1. filesystem - controlla il valore di ritorno della system call **stat**: viene controllato se il file in questione è vuoto o è di tipo particolare (socket, symbolic link, fifo).
2. magic number - controlla il valore di particolari byte del file, detti appunto *magic number*. L'elenco dei magic number e la loro posizione relativa all'inizio del file sono contenuti nel file **/usr/share/magic**.
3. language - tenta di ricavare il tipo di lingua utilizzato nel file, se questo è in formato testo.

Il tipo di file visualizzato da **file** è quello relativo al primo test che va a buon fine. I file che non sono riconosciuti da **file**, vengono segnalati come *data file* (file contenenti dei dati non meglio specificati).

```
Comando: file
Path: /usr/bin/file
SINTASSI
$ file [option] file ...
```

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di **file**. Può assumere i seguenti valori:

- b (brief mode) non visualizza i nomi dei file in output;
- c visualizza il magic (usato per il debug);
- C crea un file **magic.mgc** contenente una prima elaborazione del file da considerare;
- f *filename* legge i nomi dei file da esaminare dal file **filename** (uno per riga);
- i visualizza le stringhe tipo MIME anziché in formato tradizionale (es. **text/plain; charset=us-ascii** al posto di **ASCII text**);
- m *list* specifica l'elenco di file (*list*) contenenti i magic number secondo la sintassi **filename1[:filename2]...** (per default è **/usr/share/magic**);
- n forza la scrittura dell'output dopo il controllo di ogni file;
- v visualizza la versione di **file**;
- z tenta di analizzare nei file compressi;
- L indica di seguire i symbolic link;
- s indica di considerare anche i file di dispositivo (che per default non vengono considerati);

file è l'elenco dei nomi dei file da esaminare, separati tra loro da uno spazio;

Ad esempio, il seguente comando controlla il tipo dei file **file.c**, **/usr/bin/file** (il file eseguibile del comando **file** stesso) e **/dev/hda** (file di dispositivo).

```
$ file file.c /usr/bin/file /dev/hda
file.c:      C program text
/usr/bin/file: ELF 32-bit LSB executable, Intel 80386, version 1,
              dynamically linked, not stripped
/dev/hda:    block special
```

Il seguente comando mostra invece informazioni più dettagliate sui file di dispositivo **/dev/hda**, **/dev/hda1**, ..., **/dev/hda10**.

```
$ file -s /dev/hda{1,2,3,4,5,6,7,8,9,10}
/dev/hda:      x86 boot sector
/dev/hda1:     Linux/i386 ext2 filesystem
/dev/hda2:     x86 boot sector
/dev/hda3:     x86 boot sector, extended partition table
/dev/hda4:     Linux/i386 ext2 filesystem
/dev/hda5:     Linux/i386 swap file
/dev/hda6:     Linux/i386 swap file
/dev/hda7:     Linux/i386 swap file
/dev/hda8:     Linux/i386 swap file
/dev/hda9:     empty
/dev/hda10:    empty
```

L'elenco dei *magic number* riconosciuti è conenuto nel file `/usr/share/magic`. In realtà ogni riga di tale file indica un controllo da effettuare ed ha la seguente sintassi

offset type message

dove

offset è un valore numerico che indica la posizione del primo byte, dall'inizio del file, di cui deve essere controllato il valore;

type è una stringa che indica il tipo di raggruppamenti di byte da considerare nel test. Può assumere i seguenti valori

- byte** un byte;
- short** due byte, nel byte order nativo del sistema;
- long** quattro byte, nel byte order nativo del sistema;
- string[*car*]** una stringa (sequenza di byte). L'eventuale *car* può assumere i seguenti valori
 - B** indica che devono esserci almeno un determinato numero di caratteri *blank* (il carattere corrispondente allo spazio);
 - b** ignora eventuali caratteri *blank*;
 - c** indica che il controllo deve essere di tipo case-insensitive;
- date** quattro byte interpretati come una data Unix-like;
- beshort** due byte, nel big-endian byte order;
- belong** quattro byte, nel big-endian byte order;
- bedate** quattro byte, nel big-endian byte order, interpretati come una data Unix-like;
- leshort** due byte, nel little-endian byte order;

message è il messaggio da visualizzare nel caso in cui il confronto specificato abbia successo;

Una riga può iniziare anche con il carattere '>', che indica ulteriori test da effettuare. Il numero di caratteri '>' indica il livello del test: una riga in cui non è presente nessun carattere iniziale '>' è considerata a livello 0 e le righe a livello n sono sotto il controllo di quella precedente e più prossima a livello $n - 1$, nel senso che un test a livello n va a buon fine, vengono effettuati anche i test di livello $n + 1$. Se il carattere che segue l'ultimo '>' è '(' il contenuto in parentesi è considerato come *indirect offset* relativo al file, ovvero viene letto il contenuto del file a partire da tale offset ed il valore letto viene considerato l'offset relativo al file a partire dal quale si deve effettuare il test. L'*indirect offset* è specificato con la seguente sintassi

(*offset*[.*length*][*plus*])

dove

offset è il valore che indica l'offset (il numero di byte) relativo al file;

length indica la quantità di byte da considerare a partire dall'offset indicato da *offset*, secondo quanto riportato in tab. 3.8 (se non indicato viene considerato sempre un valore long, cioè di 4 byte);

length	Significato
b	byte (little endian)
B	byte (big endian)
s	short (2 byte – little endian)
S	short (2 byte – big endian)
l	long (4 byte – little endian)
L	long (4 byte – big endian)

Tabella 3.8: Possibili valori di *length*.

plus è un valore che viene aggiunto a quello letto dal file a partire dall'offset indicato da *offset* ed una sintassi del tipo *+num* o *-num*, dove *num* è un valore numerico senza segno espresso in byte;

3.6.5 I permessi

Ogni oggetto del filesystem è accessibile secondo una serie di **permessi** (*mode*). Per ogni oggetto del filesystem vengono identificate le seguenti entità:

Utente proprietario (owner **user**) è l'utente proprietario dell'oggetto e come tale può specificarne i diritti di accesso;

Gruppo proprietario (owner **group**) è il gruppo proprietario dell'oggetto (del quale fa parte l'utente proprietario);

Altri (**others**) sono tutti gli altri utenti;

Quando un oggetto del filesystem viene creato, il suo creatore risulta esserne il proprietario (ed il gruppo a cui esso appartiene è il gruppo proprietario) e come tale può specificare, per ognuna delle tre entità sopra elencate, i seguenti permessi di accesso:

lettura (**read**) questo permesso dà la possibilità all'entità di poter accedere in lettura al contenuto dell'oggetto;

scrittura (**write**) questo permesso dà la possibilità all'entità di poter modificare il contenuto dell'oggetto;

esecuzione (**execute**) questo permesso dà la possibilità all'entità di poter lanciare in esecuzione l'oggetto considerato;

Per essere più precisi, ad ogni oggetto del filesystem è associata una serie di *flag*²¹ composta da 12 bit (uno per ogni flag) riportata in fig. 3.9.

I flag **suid** (*s_u* – set *UID*) e **sgid** (*s_g* – set *GID*) sono utilizzati da GNU/Linux per indicare al kernel di utilizzare uno speciale comportamento per l'esecuzione dei file (eseguibili) che hanno tali flag impostati. Generalmente, quando un file eseguibile viene lanciato in esecuzione, il processo relativo assume i privilegi dell'utente che lo ha lanciato

²¹il termine *flag* (bandiera) viene spesso utilizzato per indicare un bit che rappresenta uno stato logico (0 - stato non impostato, 1 - stato impostato).

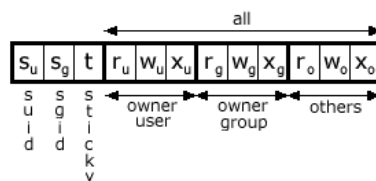


Figura 3.9: Flag relativi ai permessi associati agli oggetti del filesystem.

e del gruppo a cui quest'ultimo appartiene.²² Se il file ha il flag suid (s_u) impostato, il kernel avvia il processo relativo con i privilegi dell'utente proprietario del file piuttosto che con quelli dell'utente che lo ha lanciato in esecuzione. Analogamente se il file ha il flag sgid (s_g) impostato, il kernel avvia il processo con i privilegi del gruppo proprietario del file piuttosto che con quelli del gruppo al quale appartiene l'utente che lo ha lanciato in esecuzione.

Ad esempio, un file che ha tali flag impostati è `/bin/passwd` che è il comando utilizzato per modificare la password di un utente (file `/etc/passwd` o `/etc/shadow`). Il file che contiene le password degli utenti può infatti essere modificato soltanto da processi che hanno i privilegi del *superuser*, ma poiché `/bin/passwd` appartiene al *superuser* ed ha il flag suid impostato, quando viene lanciato da un utente, il processo relativo ha i privilegi del *superuser* e quindi può modificare il file contenente le password. Questo meccanismo permette a qualsiasi utente di poter cambiare la propria password.

Avere la possibilità di creare un processo con privilegi superiori a quelli posseduti dall'utente che lo crea comporta rischi di sicurezza e pertanto i file (programmi) che hanno tali impostazioni devono essere scritti accuratamente per non permettere all'utente che li ha lanciati di acquisire dei privilegi a lui non consentiti.

Nel caso di file non eseguibile, il flag sgid (s_g) assume un significato diverso: in tal caso GNU/Linux considera attivo per quel file il meccanismo del **mandatory locking** (se il filesystem è abilitato alla gestione del file locking – per default i filesystem montati dal sistema non sono abilitati per questo). In tal modo il sistema effettua un file locking (blocco del file) facendo sì che il file se correntemente utilizzato da un utente, sia inaccessibile ad altri: solo un utente alla volta può accedere al file (neanche il *superuser* può accedere al file bloccato in questo modo).

I flag suid e sgid assumono un significato completamente diverso per le directory. Innanzi tutto GNU/Linux ignora il flag suid (s_u) per le directory; inoltre il sistema utilizza per default la convenzione di SVr4 per la creazione di file e directory, ovvero alla creazione di un oggetto del filesystem, lo UID del proprietario viene impostato con il valore di quello relativo alla directory che lo contiene, mentre il GID del proprietario viene impostato con il valore di quello relativo al processo che lo ha creato. Impostando il flag sgid (s_g) per una directory, si indica al kernel di impostare il GID degli oggetti in essa creati con il valore relativo alla directory considerata (semantica BSD). Alla creazione di una directory, viene comunque impostato il flag sgid con il valore relativo alla directory padre.

Il flag **sticky**²³ (t) è un retaggio dei vecchi sistemi Unix. Per ottenere delle buone prestazioni, sui file utilizzati più frequentemente poteva essere impostato questo flag che faceva in modo che il sistema ponesse il segmento di codice (*code segment*) del processo relativo (v. cap. 6) nello swap (v. sez. 3.16) al momento della sua prima esecuzione e vi permaneva finché il sistema non veniva spento. Questo permetteva di ottenere delle prestazioni migliori per il file che aveva tale flag impostato. Le implementazioni della memoria virtuale attuali rendono l'utilizzo di tale flag praticamente inutile (GNU/Linux ignora l'impostazione di questo flag per i file). Per quanto riguarda le directory tale flag assume per GNU/Linux un significato diverso: i file contenuti nella directory che ha tale

²²v. cap. 6.

²³in inglese *appiccicoso*.

mandatory locking

sticky

flag impostato possono essere cancellati o rinominati soltanto dal relativo proprietario o dal proprietario della directory stessa (se tale flag non è impostato, chiunque abbia il permesso di scrittura sulla directory può cancellare o rinominare i file in essa contenuti). Tale flag è impostato, ad esempio, per la directory `/tmp`.

Quindi, ogni volta che un utente tenta di accedere ad un oggetto del filesystem (file o directory), il sistema controlla se l'utente in questione è il proprietario dell'oggetto o se fa parte del gruppo proprietario. Da ciò determina la tipologia di utente che accede all'oggetto desiderato (owner user, owner group, others). A questo punto controlla se i permessi di accesso assegnati all'oggetto del filesystem per la tipologia dell'utente in questione, concordano con la modalità di accesso allo stesso desiderata dall'utente. In caso positivo l'accesso all'oggetto è consentito all'utente in questione.

I flag r_* , w_* e x_* , ognuno per la tipologia di utente considerata (owner user, owner group, others), hanno un significato diverso se l'oggetto del filesystem è un file o ad una directory. Il flag r_* si riferisce al permesso di lettura per la tipologia di utente considerata ed ha più o meno lo stesso significato sia per i file che per le directory: per un file indica che il suo contenuto può essere visualizzato, mentre per una directory indica che può essere visualizzato l'elenco degli oggetti (file e directory) in essa contenuti. Il flag w_* si riferisce al permesso di scrittura per la tipologia di utente considerata: per un file indica che è possibile modificare il suo contenuto, mentre per una directory indica che è possibile creare, modificare o rimuovere oggetti all'interno di essa. Il flag x_* si riferisce al permesso di esecuzione per la tipologia di utente considerata: per un file indica che è possibile far eseguire il file dal sistema, mentre per una directory indica che questa può essere "attraversata", ovvero è possibile accedere ad essa ed agli oggetti in essa contenuti.

I significati delle varie combinazioni di flag (permessi) è riportata in tab. 3.9 e tab. 3.10, nelle quali è indicato con il simbolo "." il fatto che il valore del bit relativo è ininfluenza rispetto a quanto indicato in ciascuna riga: il significato si riferisce soltanto al bit (o alla combinazione di bit) il cui valore è esplicitamente indicato.

			owner user			owner group			others			Significato
s_u	s_g	t	r_u	w_u	x_u	r_g	w_g	x_g	r_o	w_o	x_o	
1	Se eseguito ha i permessi del proprietario.
.	1	.	.	.	1	Se eseguito ha i permessi del gruppo proprietario.
.	1	.	.	.	0	Il <i>mandatory locking</i> è attivato.
.	.	1	Non utilizzato.
.	.	.	1	Permesso di lettura per il proprietario.
.	.	.	.	1	Permesso di scrittura per il proprietario.
.	1	Permesso di esecuzione per il proprietario.
.	1	Permesso di lettura per il gruppo proprietario.
.	1	Permesso di scrittura per il gruppo proprietario.
.	1	.	.	.	Permesso di esecuzione per il gruppo proprietario.
.	1	.	.	Permesso di lettura per tutti gli altri.
.	1	.	Permesso di scrittura per tutti gli altri.
.	1	Permesso di esecuzione per tutti gli altri.

Tabella 3.9: Tabella riassuntiva del significato dei bit dei permessi per un file.

Il meccanismo di controllo dei permessi di accesso agli oggetti del filesystem

			owner user			owner group			others			Significato
s_u	s_g	t	r_u	w_u	x_u	r_g	w_g	x_g	r_o	w_o	x_o	
1	Non utilizzato.
.	1	Propaga il gruppo proprietario ai nuovi file creati.
.	.	1	Limita l'accesso in scrittura dei file nella directory.
.	.	.	1	Permesso di visualizzazione per il proprietario.
.	.	.	.	1	Permesso di aggiornamento per il proprietario.
.	1	Permesso di attraversamento per il proprietario.
.	1	Permesso di visualizzazione per il gruppo proprietario.
.	1	Permesso di aggiornamento per il gruppo proprietario.
.	1	.	.	.	Permesso di attraversamento per il gruppo proprietario.
.	1	.	.	Permesso di visualizzazione per tutti gli altri.
.	1	.	Permesso di aggiornamento per tutti gli altri.
.	1	Permesso di attraversamento per tutti gli altri.

Tabella 3.10: Tabella riassuntiva del significato dei bit dei permessi per una directory.

non viene attuato dal sistema nel caso in cui l'utente che effettua l'accesso sia il *superuser*²⁴: tale utente ha tutti i diritti possibili su qualunque parte del sistema.

I permessi di accesso ad un oggetto del filesystem sono impostabili con il comando `chmod` (man page `chmod(1)`) (**ch**ange **mo**de).

Comando: `chmod`
Path: `/bin/chmod`

SINTASSI

`$ chmod [option] mode file`

DESCRIZIONE

option indica la modalità di funzionamento di `chmod`. Può assumere i seguenti valori

`-c` | `--changes`

indica di visualizzare una indicazione per ogni file considerato soltanto nel caso dell'effettiva variazione dei permessi;

`-f` | `--silent` | `--quiet`

indica di non visualizzare la maggior parte dei messaggi di errore;

`-v` | `--verbose`

indica di visualizzare una indicazione per ogni file considerato;

`--reference=rfile`

indica di utilizzare il contenuto del file *rfile* per la specifica dei permessi;

`-R` | `--recursive`

indica di modificare i permessi di file e directory in modo ricorsivo (in tutto il sottoalbero);

`--help` indica di visualizzare un aiuto sommario;

`--version`

indica di visualizzare la versione del comando;

mode specifica i permessi con cui impostare il file (o directory) specificato da *file*.

Tale argomento può essere specificato in modi diversi:

²⁴v. sez. 5.

symbolic mode

(modalità simbolica) la sintassi utilizzata per specificare *mode* è la seguente

`[user][operationpermission]`

dove

user indica a quale tipo di utente fa riferimento il cambiamento di stato dell'oggetto *file*. Può assumere i seguenti valori

- u** owner user - utente proprietario dell'oggetto;
- g** owner group - gruppo proprietario dell'oggetto;
- o** others - utenti non proprietari (né appartenenti al gruppo proprietario) dell'oggetto;
- a** all - tutte le categorie di utenti;

operation indica la modalità di impostazione dei permessi indicati da *permission*. Può assumere seguenti valori

- +** indica di impostare (set) i permessi specificati da *permission* senza modificare gli altri;
- indica di rimuovere (reset) i permessi specificati da *permission* senza modificare gli altri;
- =** indica di impostare tutti i permessi come specificato da *permission*: i permessi specificati vengono considerati da impostare (set) e quelli non specificati vengono considerati da rimuovere (reset);

permission indica i permessi da assegnare a *file*. Può assumere i seguenti valori

- r** indica il permesso di lettura (read);
- w** indica il permesso di scrittura (write);
- x** indica il permesso di esecuzione (execute);
- X** indica il permesso di esecuzione (execute) soltanto se *file* è una directory oppure se è già impostato il permesso di esecuzione per altre categorie di utenti;
- s** indica l'impostazione del suid: quando *file* viene eseguito, il processo viene avviato con lo UID o GID dell'utente/gruppo proprietario;
- t** indica l'impostazione del bit sticky;

Possono essere specificate anche più impostazioni separandole con il carattere ','.

numeric mode

(modalità numerica) è un numero ottale composto da 4 cifre. Ogni cifra corrisponde ad ognuno dei gruppi di 3 bit di fig. 3.9.

file specifica il file o directory al quale modificare i permessi. Può essere specificato anche un insieme di file o directory utilizzando i metacaratteri della shell (v. cap. 7);

I permessi dei symbolic links non vengono però modificati da **chmod**, ma questo non rappresenta un problema poiché tali permessi non vengono mai considerati (al loro posto vengono presi in considerazione i permessi relativi al file o directory a cui il link si riferisce).

La proprietà di un oggetto del filesystem può essere assegnata ad un utente diverso dal creatore dell'oggetto stesso per mezzo del comando **chown** (man page **chown(1)**) (**change owner**).

Comando: **chown**

Path: **/bin/chown**

SINTASSI

\$ chown [option] [[user][:[group]]] file

DESCRIZIONE

option indica la modalità di funzionamento di **chown**. Può assumere i seguenti valori

- c | **--changes**
come **--verbose** ma visualizza soltanto le informazioni relative soltanto ai cambiamenti effettivi;
- dereference**
considera l'oggetto del filesystem riferito dal symbolic link specificato da *file*;
- h | **--no-dereference**
considera l'oggetto specificato da *file* anche se è un symbolic link;
- from=[current_owner]:[current_group]**
cambia il proprietario e/o il gruppo proprietario di *file*, soltanto se il suo proprietario e/o gruppo proprietario corrispondono a quelli specificati rispettivamente da *current_owner* e *current_group*. Se uno dei due valori viene omesso, la corrispondenza non è richiesta;
- f | **--silent** | **--quiet**
indica di non visualizzare la maggior parte dei messaggi informativi;
- reference=rfile**
utilizza il proprietario ed il gruppo proprietario di *rfile* come *user* e *group*;
- R | **--recursive**
esegue l'operazione ricorsivamente su tutto il sottoalbero delle directory la cui radice è *file*;
- v | **--verbose**
visualizza messaggi di diagnostica per ogni file (o directory) considerato;
- help** visualizza un aiuto sommario del comando **chown**;
- version**
visualizza la versione di **chown**;

user indica l'utente a cui affidare la proprietà dell'oggetto individuato da *file*;

group indica il gruppo a cui affidare la proprietà dell'oggetto individuato da *file*. Se non è specificato, ma viene specificato il simbolo ':', il gruppo di appartenenza dell'utente specificato diventa il gruppo proprietario di *file*;

file indica l'oggetto del filesystem (file o directory) al quale cambiare proprietario;

Il gruppo proprietario di un oggetto del filesystem può essere impostato tramite il comando **chgrp** (man page **chgrp(1)**) (**change group**).

Comando: **chgrp**
Path: **/bin/chgrp**

SINTASSI

\$ chgrp [option] [group] file

DESCRIZIONE

option indica la modalità di funzionamento di **chgrp**. Può assumere i seguenti valori

- c | **--changes**
come **--verbose** ma visualizza soltanto le informazioni relative soltanto ai cambiamenti effettivi;
- dereference**
considera l'oggetto del filesystem riferito dal symbolic link specificato da *file*;
- h | **--no-dereference**
considera l'oggetto specificato da *file* anche se è un symbolic link;
- f | **--silent** | **--quiet**
indica di non visualizzare la maggior parte dei messaggi informativi;
- reference=rfile**
utilizza il gruppo proprietario di *rfile* come *group*;

```

-R | --recursive
    esegue l'operazione ricorsivamente su tutto il sottoalbero delle direc-
    tory la cui radice è file;
-v | --verbose
    visualizza messaggi di diagnostica per ogni file (o directory) conside-
    rato;
--help visualizza un aiuto sommario del comando chgrp;
--version
    visualizza la versione di chgrp;

```

3.7 Journaled filesystem

Un filesystem è detto **consistente** quando i blocchi di informazioni in esso contenuti sono liberi o utilizzati, ogni blocco allocato è utilizzato da un solo file o directory e tutti i file e directory possono essere acceduti per mezzo di altre directory del filesystem. Quando un sistema GNU/Linux viene spento utilizzando gli appositi comandi messi a disposizione dal sistema, attraverso il comando **shutdown** (v. sez. 2.7), i filesystem vengono “smontati” (*unmounted*, v. sez. 3.9) in modo “pulito” (opportunamente) ed il sistema provvede ad impostare il bit *clean* per indicare la consistenza del filesystem.

consistente

Nonostante gli algoritmi sempre più sofisticati per la gestione della scrittura delle informazioni sulla memoria di massa, abbiano ampiamente ridotto, se non addirittura eliminato, la possibilità di inconsistenza del filesystem, la mancanza improvvisa dell'alimentazione elettrica costituisce di fatto un problema di possibile inconsistenza per un filesystem. Quando un sistema GNU/Linux viene spento in modo non “pulito”, ovvero senza passare dall'opportuna procedura di spegnimento (*shutdown*) del sistema, il bit *clean* non viene impostato e così al successivo riavvio, il sistema provvederà automaticamente (fin quando possibile) a verificare la consistenza del filesystem al momento del *mounting* (v. sez. 3.9), tramite **fsck** (filesystem check) (man page **fsck(8)**).

Comando: **fsck**

Path: **/sbin/fsck**

SINTASSI

```
# fsck [option] device [...] [--] [fs.option]
```

DESCRIZIONE

option specifica la modalità di funzionamento di **fsck**. Può assumere i seguenti valori

- s indica di serializzare le operazioni effettuate su più dispositivi: prima deve essere controllato un dispositivo, poi il successivo, ...;
- t *fstype*

specifica il tipo di filesystem da controllare. Può indicare anche un'elenco di filesystem separati dal simbolo “,” e se utilizzato assieme all'opzione -A verranno controllati tutti i filesystem elencati. Ad ogni filesystem può essere anteposto il prefisso ‘no’ o ‘!’ che indica di non controllare lo specifico filesystem.

Possono essere indicate anche opzioni per ogni specifico filesystem (*fs.option*) con la sintassi [!]*opts=fs.option* con l'eventuale presenza dell'operatore di negazione ‘!’;
- A controlla tutti i filesystem elencati nel file **/etc/fstab**. Il primo filesystem controllato è / (a meno che non sia specificata l'opzione -P), quindi vengono controllati gli altri filesystem nell'ordine specificato dal sesto campo delle righe del file **/etc/fstab** (se tale campo è 0 il filesystem non viene controllato affatto). Se più righe hanno lo stesso valore del sesto campo, viene tentato un controllo del filesystem in parallelo (tranne nel caso in cui i filesystem si trovino nello stesso disco fisico);
- C visualizza una barra che indica lo stato di completamento del controllo del filesystem (solo per i filesystem che lo supportano);

- N indica di non eseguire effettivamente il controllo;
- P se specificata l'opzione -A indica di controllare il filesystem / in parallelo agli altri (è un'operazione rischiosa);
- R se specificata l'opzione -A indica di non controllare il filesystem / (nel caso in cui questo sia già montato in lettura e scrittura);
- T indica di non visualizzare il titolo all'avvio;
- V visualizza un output più verboso;

device indica il dispositivo da controllare;

fs_option specifica eventuali opzioni da passare al controllore dello specifico filesystem. In genere possono essere

- a indica di effettuare una riparazione automatica del filesystem;
- r indica di effettuare una riparazione interattiva del filesystem: per ogni problema rilevato, viene richiesta una conferma a procedere con la riparazione;

I valori di ritorno di **fsck** sono interpretati bit a bit ed il significato di ogni singolo bit (dal LSB al MSB) è riportato in tab. 3.11.

bit	Descrizione
0	Gli errori presenti nel filesystem sono stati corretti.
1	Il sistema deve essere riavviato.
2	Gli errori presenti nel filesystem non sono stati corretti.
3	Errore nell'operazione.
4	Errore di sintassi.
5	Riservato.
6	Riservato.
7	Errore di libreria.

Tabella 3.11: Valori di ritorno di **fsck**.

L'inconsistenza del filesystem è dovuta al fatto che ext2 aggiorna la parte meta-data dopo aver aggiornato quella relativa ai dati (data) (non potrebbe avvenire il contrario per ragioni di performances) e l'aggiornamento della parte meta-data avviene in momenti non necessariamente immediatamente consecutivi all'aggiornamento dei dati, ma il driver effettua l'aggiornamento nei periodi "morti". Una volta che il filesystem è inconsistente, il driver relativo non è più in grado di ritrovare tutte le informazioni in esso memorizzate, pertanto si rende opportuno un meccanismo che ricontrolla l'intero filesystem e lo ripristina opportunamente (**fsck**).

Il tempo impiegato nel controllo e ripristino della consistenza di un filesystem non è da sottovalutare ed aumenta all'aumentare della capacità della memoria di massa (per una partizione delle dimensioni di 40 GiB si hanno tempi di ripristino dell'ordine di 20 minuti).

Al fine di evitare (per quanto possibile) le inconsistenze del filesystem e ridurre drasticamente i tempi di un eventuale ripristino dello stesso, viene adottata una tecnica di gestione delle informazioni che va sotto il nome di **journaling**. Tale tecnica si basa sull'utilizzo di una parte del filesystem detta *journal* o *log* (da cui il nome *journaled filesystem*) nella quale vengono memorizzate le operazioni che il sistema deve accingersi ad effettuare sul filesystem. Subito dopo il sistema effettua le modifiche sulle parti data e meta-data. Quindi il sistema elimina le informazioni precedentemente memorizzate nella parte journal. In questo modo, anche se il sistema non viene spento in modo "canonico", il ripristino del filesystem verrà effettuato controllando soltanto la parte relativa all'ultima modifica tracciata nella parte journal che non risulta essere stata ancora effettuata, non è necessario ricontrollare l'intero filesystem.

Esistono vari journaled filesystem supportati da GNU/Linux tra i quali *XFS* (???) sviluppato dalla *Silicon Graphics*, *ReiserFS* sviluppato da H. Reiser, *JFS* (Journaling FileSystem) sviluppato da IBM ed *ext3* sviluppato da S. Tweedie (uno degli autori di *ext2*) in *Red Hat*.

ext3

Il filesystem *ext3* è essenzialmente un *ext2* a cui è stata aggiunta la caratteristica di journaling. La caratteristica fondamentale di *ext3* è quella di essere compatibile al 100% con *ext2*, infatti la struttura del filesystem rimane praticamente la stessa a parte l'introduzione della parte journal. Il kernel di GNU/Linux supporta il filesystem *ext3* a partire dalla versione 2.4.16 (il kernel deve essere opportunamente compilato per poter gestire tale filesystem).

In genere i journaled filesystem annotano nel journal le modifiche prima di apportarle alla parte relativa ai meta-data, ma *ext3* ha la possibilità di annotare nel journal anche le modifiche che apporterà alla parte relativa ai dati. Ovviamente questo va a scapito delle performance ma diminuisce le probabilità di perdita di dati. La modalità di funzionamento del filesystem può essere impostata specificando l'opportuna opzione nella riga del file `/etc/fstab` (v. più avanti) che si riferisce al filesystem *ext3*. Le righe di tale file contengono infatti l'impostazione della modalità di journaling subito dopo la stringa `'ext3'`, come illustrato di seguito

```
/dev/hda3          /opt          ext3    defaults      1 0
```

dove `'defaults'` indica la modalità di default del journaling. Al suo posto possono essere specificate le seguenti modalità di funzionamento

data=journal

annota nel journal le modifiche che il sistema si appresta ad apportare nel filesystem sia alla parte relativa ai dati che a quella relativa ai meta-data (questa è la modalità che presenta le peggiori performance);

data=ordered

annota nel journal le modifiche che il sistema si appresta ad apportare nel filesystem alla parte relativa ai meta-data, ma le modifiche alla parte relativa ai dati viene effettuata prima di apportare le modifiche alla parte relativa ai meta-data (modalità di default);

data=writeback

annota nel journal le modifiche che il sistema si appresta ad apportare nel filesystem alla parte relativa ai meta-data, ma effettua le modifiche alla parte relativa ai dati in maniera asincrona rispetto alle modifiche della parte relativa ai meta-data (questa è la modalità che presenta le migliori performance);

Inoltre l'*ext3* riduce la possibilità di inconsistenza del filesystem per il fatto che l'aggiornamento della parte relativa ai meta-data e quella relativa ai dati avvengono ad intervalli di tempo minori rispetto a quelli in cui avvengono con *ext2*.

Il passaggio da *ext2* a *ext3* può essere fatto *on-the-fly* (o come si dice in gergo "a caldo") proprio perché le strutture dei due filesystem sono pienamente compatibili. Non si rende quindi necessario un backup dei dati presenti sul filesystem da aggiornare e quindi una *formattazione*²⁵ della partizione. Per convertire una partizione dal filesystem *ext2* al filesystem *ext3* è sufficiente digitare il comando

```
# tune2fs -j partition
```

dove *partition* è la partizione da convertire (es. `/dev/hda3`). Per ulteriori dettagli si veda la relativa man page (`tune2fs(8)`). È evidente che dopo aver convertito il filesystem, è necessario modificare opportunamente il file `/etc/fstab` per informare il sistema operativo che la partizione non sarà più di tipo *ext2*, ma di tipo *ext3*. Ad esempio se tale file prima della conversione conteneva la riga

```
/dev/hda3          /opt          ext2    defaults      1 2
```

questa dovrà essere modificata in

²⁵è l'operazione che prepara la memoria di massa ad essere gestita mediante un filesystem.

```
/dev/hda3          /opt          ext3      defaults      1 0
```

Si può notare il fatto che l'ultima impostazione della riga può essere tranquillamente '0' con il filesystem `ext3` poiché questa esprime la fase del processo di boot nella quale la consistenza del filesystem deve essere verificata con `fsck`. Impostandola a '0' si indica al sistema di non verificare la consistenza del filesystem con `fsck` poiché la consistenza dello stesso è ripristinabile semplicemente annullando le modifiche che sono annotate nel journal (*rollback*).

È doveroso comunque far notare che il filesystem `ext3`, poiché è stato progettato per essere totalmente compatibile con la struttura del filesystem `ext2`, non ha delle caratteristiche tipiche dei filesystem di ultima generazione come l'indicizzazione dei blocchi per mezzo di una struttura albero binario (b-tree) bilanciato che ridurrebbe i tempi di accesso agli oggetti del filesystem man mano che questo cresce, oppure la creazione dinamica degli inode (si ricorda che per `ext2` gli inode vengono creati in fase di creazione del filesystem e non è possibile aumentarne o diminuirne il numero successivamente).

3.8 Il VFS

VFS

GNU/Linux ha la possibilità di utilizzare vari filesystem (v. tab. 3.2) grazie alla presenza di un filesystem virtuale, il **VFS** (Virtual FileSystem). Questo strato di software (in gergo *abstraction layer*) presente nel kernel permette al sistema di supportare vari filesystem senza "conoscerli", infatti esso mette a disposizione del sistema un'interfaccia composta da una serie di funzioni che non fanno riferimento ad uno specifico filesystem (v. fig. 3.10). Tali funzioni chiamano poi delle opportune routine che devono essere implementate dal driver dello specifico filesystem considerato. Il tutto viene gestito a basso livello, in maniera trasparente al sistema. I driver per la gestione dei vari tipi di filesystem devono comunque rispettare determinate regole per potersi integrare con il VFS.

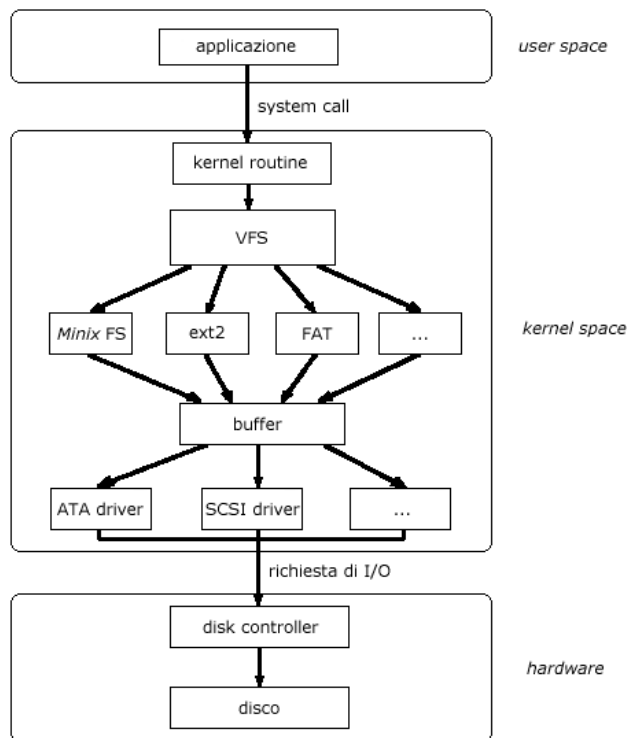


Figura 3.10: Schematizzazione del VFS.

Grazie a tale strato di astrazione, tutti i filesystem utilizzati da GNU/Linux hanno la stessa struttura logica, ovvero vengono presentati all'utente tutti nella stessa maniera.

Il VFS utilizza strutture dati che descrivono il filesystem virtuale (l'intero albero delle directory) ed i vari tipi di filesystem realmente utilizzati.

Per la gestione del filesystem virtuale, il VFS utilizza le strutture superblock e inode analoghe a quelle utilizzate da ext2.

Ogni filesystem montato (v. sez. 3.9) è rappresentato da un superblock, che tra le varie informazioni contiene quelle seguenti

Device identifier

è un numero che individua univocamente la partizione nella quale è contenuto il filesystem reale (es. per la partizione `/dev/hda1` il device identifier è 301H);

Mounted inode pointer

il riferimento al primo inode del filesystem reale;

Covered inode pointer

il riferimento al mount point nel quale il filesystem reale è montato;

Block size

la dimensione in byte dei blocchi del filesystem reale;

Superblock operations

il riferimento ad un insieme di routine utilizzate dal VFS per accedere ai superblock ed agli inode;

Filesystem type

il riferimento ad una struttura dati che descrive il tipo del filesystem reale;

Filesystem specific

il riferimento a strutture di cui necessita lo specifico filesystem reale;

Ogni oggetto del filesystem reale è rappresentato da un inode del VFS, che tra le varie informazioni contiene quelle seguenti

Device identifier

è il device identifier relativo al dispositivo o partizione nel quale è contenuto l'oggetto del filesystem a cui si riferisce l'inode in questione;

Inode number

è il numero che individua univocamente un'inode che si riferisce ad un oggetto del filesystem reale (la coppia formata dal device identifier e dall'inode number costituisce un elemento univoco all'interno del VFS);

Mode

indica i diritti di accesso per le varie categorie di utenti, per l'oggetto del filesystem rappresentato da l'inode in questione;

UID è l'UID dell'utente proprietario dell'oggetto del filesystem;

GID è il GID del gruppo proprietario dell'oggetto del filesystem;

Creation Time

???

Modification Time

???

Write Time

???

Block size

la dimensione in byte dei blocchi che costituiscono l'inode;

Inode operations

il riferimento ad un insieme di routine specifiche per la gestione del filesystem fisico;

Count

il numero di riferimenti all'inode considerato (0 significa che l'inode può essere rimosso);

Lock

indica se l'inode è bloccato o meno (ad esempio viene bloccato mentre il filesystem ci sta facendo accesso);

Dirty

indica se l'inode (o i blocchi ai quali si riferisce) è stato modificato dal VFS e quindi il filesystem reale deve aggiornarsi;

Filesystem specific information

informazioni specifiche del filesystem reale;

Più in dettaglio, la struttura di un inode del VFS è riportata di seguito

```
struct inode {
    kdev_t                i_dev;
    unsigned long         i_ino;
    umode_t               i_mode;
    nlink_t               i_nlink;
    uid_t                 i_uid;
    gid_t                 i_gid;
    kdev_t                i_rdev;
    off_t                 i_size;
    time_t                i_atime;
    time_t                i_mtime;
    time_t                i_ctime;
    unsigned long         i_blksize;
    unsigned long         i_blocks;
    unsigned long         i_version;
    unsigned long         i_nrpages;
    struct semaphore      i_sem;
    struct inode_operations *i_op;
    struct super_block    i_sb;
    struct wait_queue     i_wait;
    struct file_lock      i_flock;
    struct vm_area_struct i_mmap;
    struct page           i_pages;
    struct dquot          i_dquot[MAXQUOTAS];
    struct inode          i_next, *i_prev;
    struct inode          i_hash_next, *i_hash_prev;
    struct inode          i_bound_to, *i_bound_by;
    struct inode          i_mount;
    unsigned short        i_count;
    unsigned short        i_flags;
    unsigned char         i_lock;
    unsigned char         i_dirt;
    unsigned char         i_pipe;
    unsigned char         i_sock;
    unsigned char         i_seek;
    unsigned char         i_update;
    unsigned short        i_writecount;
    union {
        struct pipe_inode_info pipe_i;
        struct minix_inode_info minix_i;
        struct ext_inode_info ext_i;
        struct ext2_inode_info ext2_i;
        struct hpfs_inode_info hpfs_i;
        struct msdos_inode_info msdos_i;
    };
};
```



```

    struct umsdos_inode_info umsdos_i;
    struct iso_inode_info     isofs_i;
    struct nfs_inode_info     nfs_i;
    struct xiafs_inode_info   xiafs_i;
    struct sysv_inode_info    sysv_i;
    struct affs_inode_info    affs_i;
    struct ufs_inode_info     ufs_i;
    struct socket             socket_i;
    void                      *generic_ip;
} u;
};

```

Le routine di gestione del filesystem reale, hanno l'onere di tenere aggiornate le strutture del VFS.

Il kernel inoltre utilizza degli specifici buffer (*inode cache* e *directory cache*) – parti della memoria centrale (molto veloci rispetto alla memoria di massa) – che velocizzano le operazioni sul filesystem.

Quando si vuol accedere ad un file o una directory, il VFS controlla se il relativo inode è presente nella inode cache. Se lo è accede direttamente a quello, altrimenti chiama la routine che effettua l'accesso all'inode sullo specifico filesystem sul quale esso risiede (anche il filesystem specifico utilizza un proprio meccanismo di caching). Una volta recuperato, questo viene memorizzato nella inode cache in maniera tale che un eventuale ulteriore accesso ad esso avvenga senza più coinvolgere il filesystem specifico sul quale esso risiede. Se la inode cache risulta piena vengono rimossi gli inode acceduti meno di frequente per far posto ai nuovi inode da memorizzare.

Inoltre, ogni volta che si accede ad una directory, il VFS controlla che il suo contenuto non sia già stato memorizzato nella directory cache. In caso affermativo accede direttamente al suo contenuto, altrimenti chiama la routine che effettua l'accesso alla directory sullo specifico filesystem sul quale essa risiede. Una volta recuperata, il suo contenuto (le varie dentry) viene memorizzato nella directory cache, in maniera tale che un eventuale ulteriore accesso ad essa avvenga senza più coinvolgere il filesystem specifico sul quale essa risiede.

???

3.9 mount e umount

Per poter utilizzare un filesystem in un sistema Unix-like, è necessario che questo venga agganciato al directory tree, ovvero venga visto come una directory all'interno dell'albero delle directory, per mezzo del comando `mount` (man page `mount(8)`). Si parla pertanto di “montaggio” (*mounting*) del filesystem. Questa operazione informa il sistema operativo della presenza del filesystem considerato, associandolo ad una directory all'interno della struttura logica (albero delle directory), che rappresenta il **mount-point** (*punto di montaggio*) del filesystem.

mount-point

Comando: `mount`

Path: `/bin/mount`

SINTASSI

`$ mount [option] [device] [dir]`

DESCRIZIONE

option indica le opzioni di funzionamento di `mount`. Può assumere i seguenti valori

- a monta tutti i filesystem elencati nel file `/etc/fstab`;
- F monta l'elenco di filesystem in maniera parallela (non può essere utilizzato per montare filesystem in mount point dipendenti come `/usr` e `/usr/spool`);
- f (*fake*) esegue la procedura di mount senza montare nessun filesystem (utile per il debug);

- h** mostra un aiuto sommario;
- l** visualizza le etichette (v. file `/etc/fstab`) relative ai filesystem montati;
- L *label*** monta il filesystem identificato dall'etichetta *label*;
- n** monta il filesystem senza aggiornare il file `/etc/mstab`;
- O *opt*** limita il mounting dei filesystem che hanno l'opzione specificata da *opt* (che può essere anche un elenco di opzioni separate dal carattere ','). Alle opzioni può essere aggiunto il prefisso **no** per indicare che il filesystem con tale opzione non deve essere montato;
- o *optlist***
 - specifica le opzioni per il mounting del filesystem (le opzioni sono separate tra loro dal carattere ','). Le opzioni per il mounting sono
 - async** indica che tutto l'input/output (I/O) relativo al filesystem deve essere effettuato in maniera asincrona;
 - atime** aggiorna la data/ora di accesso agli inode ad ogni accesso (default);
 - auto** il filesystem può essere montato con l'opzione **-a**;
 - defaults** utilizza le opzioni di default: **rw**, **suid**, **dev**, **exec**, **auto**, **nouser** e **async**;
 - dev** interpreta i file di dispositivo presenti sul filesystem;
 - exec** permette l'esecuzione dei file eseguibili presenti sul filesystem;
 - _netdev** indica al sistema di eseguire il mounting del filesystem dopo l'attivazione dell'interfaccia di rete poiché il filesystem risiede su un dispositivo che richiede l'accesso tramite la rete;
 - noatime** non aggiorna la data/ora di accesso agli inode ad ogni accesso (per velocizzare i tempi di accesso ai file);
 - noauto** il filesystem può essere montato soltanto in maniera esplicita (l'opzione **-a** non fa il mounting del filesystem che riporta questa opzione);
 - nodev** non interpreta i file di dispositivo presenti sul filesystem;
 - noexec** non permette l'esecuzione dei file eseguibili presenti sul filesystem;
 - nosuid** rende nullo l'effetto del bit suid sugli oggetti del filesystem;
 - nouser** proibisce agli utenti (tranne al superuser) di eseguire il mounting del filesystem (default);
 - owner** come **user** ma con la restrizione che l'utente deve essere il proprietario del relativo file di dispositivo;
 - remount** esegue nuovamente il mounting del filesystem (per cambiare le opzioni di mounting senza cambiare il mount point);
 - ro** esegue il mounting del filesystem in modo da permettere l'accesso agli oggetti in esso contenuti in sola lettura (read only);
 - rw** esegue il mounting del filesystem in modo da permettere l'accesso agli oggetti in esso contenuti sia in lettura che in scrittura (read write);
 - suid** abilita l'effetto del bit suid sugli oggetti del filesystem;
 - sync** indica che tutto l'input/output (I/O) relativo al filesystem deve essere effettuato in maniera sincrona;
 - user** permette ad un utente specifico di eseguire il mounting del filesystem. L'utente che esegue il mounting viene annotato nel file `/etc/mstab` in maniera tale che lo stesso utente possa poi eseguire l'operazione inversa (unmount) su tale filesystem. Questa opzione implica **noexec**, **nosuid**, e **nodev**

a meno che non venga esplicitamente indicato diversamente nella relativa riga del file `/etc/fstab`;

users permette a tutti gli utenti di eseguire il mounting del filesystem. Questa opzione implica **noexec**, **nosuid**, e **nodev** a meno che non venga esplicitamente indicato diversamente nella relativa riga del file `/etc/fstab`;

-r monta il filesystem in sola lettura (read-only);

-s fa in modo che il montaggio non dia errore se il filesystem non supporta alcune opzioni di montaggio;

-t fstype indica il tipo di filesystem da montare secondo quanto riportato da **fstype**, che può assumere i valori riportati in tab. 3.2; Se tale opzione non è specificata oppure è specificato **autofs mount** prova a leggere l'eventuale superblocco del filesystem per riconoscerne il tipo. Se tale tentativo fallisce, **mount** tenta di leggere il file `/etc/filesystems` e se non esiste prova con il file `/proc/filesystems`; **mount** tenta di montare i filesystem dei tipi specificati in tali file.

-U uuid monta il filesystem identificato dallo UUID (Universally Unique Identifier) **uuid**;

-v imposta la modalità verbosa;

-V mostra la versione;

-w monta il filesystem in lettura e scrittura (read/wtite) (questo è la modalità di montaggio di default);

--bind rimonta un filesystem in un altro mount point in maniera tale che questo sia accessibile da più directory;

--move sposta un filesystem in un altro mount point;

device è il file di dispositivo da montare;

dir è il mount point (directory) nel quale montare il file di dispositivo indicato da **device**;

Esistono inoltre delle opzioni di mounting specifiche per ogni tipo di filesystem, per le quali si rimanda alla man page **mount(8)**.

Se nessun argomento viene specificato, **mount** visualizza il contenuto del file `/etc/mtab`.

Ad esempio, la *root directory* viene montata, all'avvio del sistema, nel mount-point `/` che rappresenta appunto la radice dell'albero delle directory.

L'albero delle directory di un sistema Unix-like può infatti essere composto da più filesystem, ognuno dei quali è montato in uno specifico mount-point. Ad esempio è possibile montare il filesystem (sempre che questo sia un filesystem di quelli supportati da GNU/Linux) contenuto nella partizione `/dev/hdb3` (terza partizione del secondo hard disk ATA) nel mount-point `/mnt/myfs`, ovvero una directory che deve esistere nel filesystem sul quale si vuole "innestare" `/dev/hdb3`; in questo modo sarà possibile accedere alla struttura di tale filesystem attraverso la directory `/mnt/myfs`. Questo modo di montare i filesystem fa sì che se un filesystem secondario (non quello montato come root directory), come ad esempio `/dev/hdb3`, per qualche motivo viene danneggiato, il filesystem principale non ne risente e semplicemente non si potrà accedere al filesystem danneggiato attraverso il suo mount-point (`/mnt/myfs`), ma il sistema continuerà a funzionare.

L'*exit status*²⁶ di **mount** ha un significato legato ai bit del valore stesso, secondo quanto riportato in tab. 3.12. Un bit impostato ad 1 implica il fatto che si è verificato l'evento ad esso correlato.

Ad esempio, il comando

```
# mount -t ext2 /dev/hda2 /root
```

²⁶v. sez. 1.9.2.

Bit	Significato
0	chiamata al comando errata o permessi utente non validi
1	errore di sistema
2	errore interno di <code>mount</code>
3	interrotto dall'utente
4	problemi di accesso al file <code>/etc/mtab</code>
5	fallimento dell'operazione di mounting
6	alcune operazioni di mounting sono state effettuate con successo

Tabella 3.12: Significato del valore di ritorno di `mount`.

monta il filesystem di tipo `ext2` contenuto nella seconda partizione del primo dispositivo ATA (hard disk) nella directory `/root`, ovvero la directory `/root` è il mount-point della seconda partizione del primo hard disk ATA. Per l'utente la cosa è trasparente: l'accesso alla directory `/root` è analogo all'accesso ad una qualunque altra directory, ma il suo contenuto è quello della seconda partizione del primo hard disk ATA. Il comando

```
# mount -t auto /dev/fd0 /mnt/floppy
```

monta il filesystem (riconoscendone il tipo in maniera automatica) contenuto nel primo floppy disk (il floppy disk non è suddiviso in partizioni) nella directory `/mnt/floppy`. L'accesso al contenuto del floppy disk avviene quindi per mezzo dell'accesso alla directory `/mnt/floppy`. Le operazioni compiute su tale directory vengono effettivamente compiute sul floppy disk. Il comando

```
# mount -t iso9660 -o ro /dev/hdc /mnt/cdrom
```

monta, consentendo soltanto l'accesso in lettura, il terzo dispositivo ATA (in questo caso si suppone si tratti di un CD-ROM) nella directory `/mnt/cdrom`. L'accesso al contenuto del CD-ROM avviene quindi per mezzo dell'accesso alla directory `/mnt/cdrom`.

La prima operazione di mounting del filesystem è eseguita dal kernel, che “monta” automaticamente la *root directory*, cioè la radice del directory tree. Il mounting di questa directory è indispensabile per poter avviare il sistema. Una volta avviata la procedura di avvio del sistema, questa poi provvede a montare gli altri filesystem indicati nel file `/etc/fstab` (man page `fstab(5)`) (filesystem **table**) che contiene le relative impostazioni per il mounting. Tale file è in formato testo e le righe al suo interno hanno la seguente sintassi:

```
fs_spec fs_file fs_type mount_option freq stage
```

dove

fs_spec indica il dispositivo per il quale eseguire il mounting del filesystem. In questo campo è possibile indicare il dispositivo (es. `/dev/hda2`), l'etichetta del volume (per i filesystem `extX`) (es. `LABEL=/boot`) o il suo UUID (es. `UUID=3e6be9de-8139-11d1-9106-a43f08d823a6`). Le due ultime possibilità rendono il sistema più flessibile alla gestione dei dischi, in quanto l'aggiunta/rimozione di un disco SCSI cambia il dispositivo ma non l'etichetta del filesystem;

fs_file indica il mount point per il filesystem (per il filesystem di `swap` questo campo dovrebbe contenere il valore `none`);

fs_type indica il tipo di filesystem (per i tipi di filesystem supportati si può vedere il file `/proc/filesystems`). Se questo campo contiene il valore `ignore`, la riga viene ignorata;

mount_option indica le opzioni per il mounting del filesystem. È composto da un elenco di parole chiavi separate dal carattere `,` come specificato in `mount(8)`;

freq indica se il filesystem ha la necessità di essere processato dal comando `dump` (man page `dump(8)`). Un valore 0 indica che il filesystem non ha bisogno di `dump`;

stage indica l'ordine in cui il filesystem deve essere controllato da `fsck` (durante la procedura di avvio). Il campo *stage* relativo al filesystem il cui mounting avviene

nel mount point ‘/’ (root directory), dovrebbe contenere il valore 1 che indica che è il primo filesystem che deve essere controllato, mentre gli altri filesystem dovrebbero avere un valore più elevato (per cercare di ottimizzare i tempi, si può cercare di far controllare più filesystem in parallelo specificando per essi lo stesso valore - in genere 2). Per i filesystem che non devono essere controllati, deve essere specificato il valore 0;

L’ordine delle righe contenute nel file `/etc/fstab` è importante poiché i comandi `fsck`, `mount` e `umount` scandiscono le righe contenute in tale file dalla prima all’ultima.

Un esempio del contenuto del file `/etc/fstab` è riportato di seguito.

```

LABEL=/                /                ext3    defaults    1 1
LABEL=/boot            /boot            ext3    defaults    1 2
none                   /dev/pts         devpts  gid=5,mode=620 0 0
none                   /proc            proc    defaults    0 0
none                   /dev/shm         tmpfs   defaults    0 0
/dev/hda5              swap             swap    defaults    0 0
/dev/cdrom              /mnt/cdrom       iso9660 noauto,owner,kudzu,ro 0 0
/dev/hda1              /mnt/windows     vfat    defaults    0 0
/dev/fd0               /mnt/floppy      auto    noauto,owner,kudzu 0 0

```

Con il file `/etc/fstab` sopra riportato, è possibile montare un filesystem indicando soltanto il mount point relativo. Ad esempio, il comando

```
# mount /mnt/floppy
```

monta il filesystem relativo al dispositivo `/dev/fd0` con le opzioni `-t auto -o noauto,owner` in maniera equivalente al comando

```
# mount -t auto -o noauto,owner /dev/fd0 /mnt/floppy
```

Un filesystem può essere anche “smontato” con il comando `umount` (man page `umount(8)`).

Comando: `umount`

Path: `/bin/umount`

SINTASSI

`$ umount [option] [device]`

DESCRIZIONE

option indica la modalità di funzionamento di `umount`. Può assumere i seguenti valori

- `-V` visualizza la versione di `umount`;
- `-h` visualizza un aiuto sommario di `umount`;
- `-v` indica di attivare la modalità verbosa;
- `-n` “smonta” il filesystem senza aggiornare il file `/etc/mtab`;
- `-r` nel caso in cui lo “smontaggio” non abbia esito positivo, tenta di “rimontarlo” in sola lettura;
- `-d` nel caso in cui il filesystem da “smontare” sia relativo ad un loop device, libera anche il loop device;
- `-a` indica di “smontare” tutti i filesystem elencati nel file `/etc/mtab` (tranne il filesystem `/proc`);
- `-t vfstype`
indica di “smontare” soltanto i filesystem del tipo indicato da *vfstype* (può specificare un elenco di tipi di filesystem separato dal simbolo ‘,’). I tipi di filesystem sono elencati in tab. 3.2. I tipi di filesystem ai quali è preposto il prefisso “no” vengono esplicitamente di non operare su tali tipi di filesystem;
- `-O opt` indica di “smontare” soltanto i filesystem che sono stati “montati” (file `/etc/fstab`) con le opzioni indicate da *opt* (può essere specificato un elenco di opzioni separate dal simbolo ‘,’). Ad ogni opzione può essere preposto il prefisso “no” che indica esplicitamente di non operare su tali filesystem;

- f indica di forzare lo “smontaggio” (ad esempio nel caso di NFS irraggiungibile);
- l indica di effettuare un *lazy unmount*, cioè di “smontare” il filesystem subito e di cancellare tutti i riferimenti al filesystem non appena questo non è più occupato (un filesystem è occupato quando alcuni file di esso sono aperti, o la working directory di qualche processo è riferita ad una directory del filesystem);

device indica il dispositivo o il filesystem da “smontare”;

Se nessun argomento viene specificato, **umount** visualizza il contenuto del file `/etc/mtab`.

I comandi **mount** e **umount** mantengono l’elenco dei filesystem correntemente montati nel file `/etc/mtab` (anche il file `/proc/mounts` ha un contenuto simile) che ha la stessa sintassi del file `/etc/fstab`.

Un esempio del contenuto del file `/etc/mtab` è riportato di seguito.

```
/dev/hda3 / ext3 rw 0 0
none /proc proc rw 0 0
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hda2 /boot ext3 rw 0 0
none /dev/pts devpts rw,gid=5,mode=620 0 0
none /dev/shm tmpfs rw 0 0
/dev/hda1 /mnt/windows vfat rw 0 0
```

In fig. 3.11 è riportato un esempio di un directory tree (ogni nodo rappresenta una directory) in cui sono evidenziati, ognuno con un proprio colore, i diversi tipi di filesystem da cui esso è composto. Come già spiegato precedentemente, l’albero delle directory di un sistema Unix-like è sempre costituito da una sola root directory, la ‘/'. Altri filesystem possono essere inseriti in tale albero, mediante il comando **mount**, per mezzo del quale la root directory di un filesystem viene inserita o “montata” in un mount-point (una directory) dell’albero. La stessa directory / viene “montata” dal kernel ed è quindi il mount-point del filesystem principale del sistema. Nella figura, le directory che rappresentano dei mount-point sono evidenziate con un cerchio più marcato rispetto alle altre.

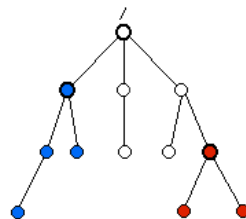


Figura 3.11: Schematizzazione di un albero delle directory formato da più filesystem.

Quindi, il montaggio dei filesystem nei mount-point permette di poter gestire contemporaneamente più filesystem differenti, come se si trattasse di un unico filesystem virtuale, poiché l’albero delle directory ha una sola radice: la directory /.

3.10 “Navigare” nel filesystem

Il filesystem può essere “navigato” immaginando di avere una posizione variabile rispetto ad esso, solidale con la *working directory*. Infatti si usano espressioni del genere “Vai nella directory ...”, “Entra nella directory ...”, “In che directory sei?”, ... riferendosi semplicemente al cambiamento della *working directory*.

La *working directory* rappresenta la directory corrente, cioè quella che in ogni istante è riferita con il carattere ‘.’. Questa directory è quella che viene utilizzata per default

da vari comandi di cui un esempio può essere `ls` (man page `ls(1)`) (`list`) che visualizza l'elenco dei file (e directory) contenuti in una directory.

Comando: `ls`
 Path: `/bin/ls`
 SINTASSI
`$ ls [option] [path]`

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di `ls`. Può assumere i seguenti valori:

- `-a` | `--all`
visualizza tutti gli oggetti del filesystem, anche quelli il cui nome inizia con il carattere `'.'`;
- `-A` | `--almost-all`
visualizza tutti gli oggetti del filesystem tranne le directory speciali `'.'` e `'..'`;
- `--author`
visualizza (nel formato esteso) anche l'autore di ogni oggetto del filesystem (potrebbe essere diverso dal proprietario);
- `-b` | `--escape`
visualizza i valori ottali corrispondenti ai caratteri non stampabili;
- `--block-size=size`
visualizza (nel formato non esteso) la dimensione dei file in base all'unità di misura dei blocchi indicata da *size* (default 1K);
- `-B` | `--ignore-backups`
non visualizza gli oggetti del filesystem il cui nome termina con il carattere `'~'`;
- `-c`
ordina l'elenco degli oggetti del filesystem in base alla data/ora relativa alla loro ultima modifica. Con `-lt` ordina l'elenco degli oggetti del filesystem in base alla data/ora relativa alla loro ultima modifica e visualizza tale data/ora. Con `-l` visualizza la data/ora dell'ultima modifica di ogni oggetto del filesystem, ma ordina gli oggetti in base al loro nome;
- `-C`
visualizza gli oggetti del filesystem ordinandoli per colonne (default);
- `--color[=when]`
indica se utilizzare i colori nella visualizzazione degli oggetti del filesystem secondo quanto specificato da *when*: **never** non usa i colori affatto (è il valore di default), **always** utilizza i colori, e **auto** utilizza i colori soltanto se l'output va su un terminale. I colori di default con i quali sono visualizzati i vari oggetti del filesystem sono riportati in tab. 3.13;

Colore	Oggetto del filesystem
bianco	file standard
verde	file eseguibili
rosso	file compressi
giallo	file di dispositivo
blu	directory
ciano	symbolic link
violetto	immagini
rosso lampeggiante	symbolic link che si riferiscono a file inesistenti

Tabella 3.13: Colori standard attribuiti agli oggetti del filesystem da `ls`.

- `-d` | `--directory`
visualizza i nomi delle directory senza visualizzarne il relativo contenuto;
- `-D` | `--dired`
genera (nel formato esteso) un output compatibile con la modalità “dired” di Emacs;

- f non effettua nessun tipo di ordinamento nella visualizzazione degli oggetti del filesystem;
- F | **--classify**
aggiunge un carattere in coda al nome dell'oggetto del filesystem visualizzato secondo quanto riportato in tab. 3.14

Oggetto del filesystem	Carattere
file eseguibile	*
directory	/
socket	=
symbolic link	@
fifo	

Tabella 3.14: Caratteri aggiunti ai nomi degli oggetti del filesystem.

- format=word**
specifica il formato di visualizzazione dell'elenco: **single-column** come -l, **vertical** come -C, **commas** come -m, **across** o **horizontal** come -x, **long** o **verbose** come -l;
- full-time**
come -l **--time-style=full-iso**;
- g come -l ma senza la visualizzazione dell'utente proprietario di ogni oggetto del filesystem;
- G | **--no-group**
non visualizza (nel formato esteso) il nome del gruppo proprietario degli oggetti del filesystem;
- h | **--human-readable**
visualizza la dimensione in forma leggibile (es. 1K, 234M, 2G) secondo potenze di 1024: 1K = 1 KiB, 1M = 1 MiB, 1G = 1 GiB;
- si** visualizza la dimensione in forma leggibile (1kB, 234MB, 2GB) secondo potenze di 1000 (Sistema Internazionale);
- H | **--dereference-command-line**
segue i symbolic link presenti sulla riga di comando;
- help** visualizza un aiuto sommario di **ls**;
- i | **--inode**
visualizza il numero dell'inode associato ad ogni oggetto del filesystem;
- I *pattern* | **--ignore=pattern**
non visualizza i file specificati da *pattern*;
- indicator-style=word**
indica il tipo di visualizzazione dei nomi degli oggetti del filesystem secondo quanto specificato da *word*: **classify** come -F e **file-type** come -p;
- k | **--kilobytes**
come **--block-size=1K**;
- l utilizza il formato esteso per la visualizzazione dell'output. Su ogni riga vengono visualizzati: i permessi relativi all'oggetto del filesystem, il numero di hard link, l'utente proprietario, il gruppo proprietario, la dimensione (per default in byte), la data/ora relative all'ultima modifica ed il nome dell'oggetto del filesystem. Per i file relativi ai dispositivi, al posto della dimensione vengono visualizzati il major ed il minor number ad esso relativi.
I permessi relativi all'oggetto del filesystem sono visualizzati per mezzo di una stringa di 10 caratteri che riporta i permessi relativi all'oggetto del filesystem: il primo carattere indica il tipo di oggetto del filesystem, i successivi tre caratteri indicano i permessi di accesso su tale oggetto per l'utente proprietario (owner user), i successivi 3 caratteri indicano i permessi di accesso su tale oggetto per il gruppo proprietario (owner group) e gli ultimi 3 caratteri indicano i permessi di accesso su tale oggetto per gli altri utenti (others). Il significato dei vari caratteri visualizzati nella stringa relativa ai permessi è

	owner user			owner group			others			Significato
-	L'oggetto del filesystem è un file “standard” (o un hard link).
l	L'oggetto del filesystem è un symbolic link.
b	L'oggetto del filesystem è un dispositivo a blocchi.
c	L'oggetto del filesystem è un dispositivo a caratteri.
d	L'oggetto del filesystem è una directory.
p	L'oggetto del filesystem è una FIFO (o named pipe).
s	L'oggetto del filesystem è uno Unix Domain Socket.
.	r	Il proprietario ha il permesso di lettura sull'oggetto.
.	.	w	Il proprietario ha il permesso di scrittura sull'oggetto.
.	.	.	x	Il proprietario ha il permesso di esecuzione sull'oggetto.
.	.	.	s	Il bit setuid è impostato ed il proprietario ha il permesso di esecuzione sull'oggetto.
.	.	.	S	Il bit setuid è impostato ma il proprietario non ha il permesso di esecuzione sull'oggetto.
.	.	.	.	r	Il gruppo proprietario ha il permesso di lettura sull'oggetto.
.	w	Il gruppo proprietario ha il permesso di scrittura sull'oggetto.
.	x	.	.	.	Il gruppo proprietario ha il permesso di esecuzione sull'oggetto.
.	s	.	.	.	Il bit setgid è impostato ed il gruppo proprietario ha il permesso di esecuzione sull'oggetto.
.	S	.	.	.	Il bit setgid è impostato ma il gruppo proprietario non ha il permesso di esecuzione sull'oggetto.
.	r	.	.	Gli altri utenti hanno il permesso di lettura sull'oggetto.
.	w	.	Gli altri utenti hanno il permesso di scrittura sull'oggetto.
.	x	Gli altri utenti hanno il permesso di esecuzione sull'oggetto.
.	t	Il bit sticky è impostato e gli altri utenti hanno il permesso di esecuzione sull'oggetto.
.	T	Il bit sticky è impostato ma gli altri utenti non hanno il permesso di esecuzione sull'oggetto.

Tabella 3.15: I flag del comando `ls`.

riportato in tab. 3.15 (il simbolo ‘-’ indica che il permesso per la relativa tipologia di utenti non è stato abilitato);

- L | **--dereference**
visualizza gli oggetti puntati dai symbolic link;
- m visualizza i nomi degli oggetti uno di seguito all'altro, separandoli con la sequenza di caratteri ‘, ’;
- n | **--numeric-uid-gid**
come -l ma l'utente ed il gruppo proprietari sono visualizzati come UID e GID (in forma numerica);
- N | **--literal**
visualizza i nomi degli oggetti senza trattare in modo particolare eventuali caratteri di controllo;
- o come -l ma senza l'indicazione del gruppo proprietario;
- p | **--file-type**
aggiunge un carattere in coda al nome dell'oggetto del filesystem visualizzato dipendentemente dal tipo, secondo quanto riportato in tab. 3.14, tranne per il carattere *.
- q | **--hide-control-chars**
visualizza un carattere ‘?’ al posto di eventuali caratteri non stampabili;

```

--show-control-chars
    visualizza gli eventuali caratteri non stampabili interpretandoli (com-
    portamento di default);
-Q | --quote-name
    visualizza i nomi degli oggetti del filesystem tra doppi apici “”;
--quoting-style=word
    imposta l'utilizzo dello stile di visualizzazione identificato da word:
    literal come -N, shell effettua il quoting dei nomi se contengono
    metacaratteri interpretabili dalla shell, shell-always effettua il quo-
    ting dei nomi comunque, c come -Q, locale come -Q tranne che i
    simboli utilizzati per il quoting sono quelli appropriati per le impo-
    stazioni locali, locale come -Q tranne che il quoting viene effettuato
    con i singoli apici, escape come -b;
-r | --reverse
    mostra l'elenco nell'ordine inverso;
-R | --recursive
    elenca anche il contenuto delle sottodirectory in maniera ricorsiva;
-s | --size
    visualizza la dimensione degli oggetti del filesystem;
-S
    ordina i nomi degli oggetti del filesystem in base alla dimensione degli
    oggetti stessi;
--sort=word
    ordina l'elenco secondo quanto specificato da word: access o atime
    come -u, extension come -X, none come -U, size come -S, status
    o ctime o use come -c, time come -t, version come -v;
-t
    ordina l'elenco degli oggetti del filesystem in base alla data/ora del-
    l'ultima modifica;
-T cols | --tabsize=cols
    imposta l'ampiezza del carattere di tabulazione, TAB (09H), secondo
    quanto specificato da cols (default 8);
--time=word
    visualizza la data/ora dell'oggetto secondo quanto specificato da word:
    atime o access o use la data/ora relativa all'ultimo accesso, ctime
    o status la data/ora relativa all'ultima modifica (default);
--time-style=word
    imposta lo stile di visualizzazione della data/ora degli oggetti del fi-
    lesystem secondo quanto specificato da word: full-iso data (aaaa-
    mm-gg) ora (hh:mm:ss.nnnnnnnnn) con la precisione del nanosecon-
    do e il fuso orario o time zone (-hhmm) secondo lo standard ISO 8601
    (es. 2001-05-14 23:45:56.477817180 -0700), iso data/ora ISO
    abbreviate (es. 2001-05-14 o 05-14 23:45), locale data/ora secon-
    do le impostazioni POSIX locali (es. Aug 14 2001 o Oct 20 16:45),
    posix-iso data/ora secondo l'impostazione locale (POSIX o ISO),
    +format data/ora secondo lo stile specificato da format secondo la
    sintassi interpretata dal comando date;
-u
    ordina l'elenco degli oggetti del filesystem in base alla data/ora re-
    lativa al loro ultimo accesso. Con -lt ordina l'elenco degli oggetti
    del filesystem in base alla data/ora relativa al loro ultimo accesso e
    visualizza tale data/ora. Con -l visualizza la data/ora dell'ultimo
    accesso di ogni oggetto del filesystem, ma ordina gli oggetti in base
    al loro nome;
-U
    nessun ordinamento: visualizza gli oggetti secondo l'ordine in cui si
    trovano fisicamente sui blocchi del filesystem;
-v
    ordina l'elenco degli oggetti del filesystem in base alla versione degli
    oggetti stessi;
--version
    visualizza la versione di ls;
-w cols | --width=cols
    imposta la larghezza dello schermo secondo quanto specificato da
    cols;
-x
    visualizza l'elenco dei nomi degli oggetti del filesystem ordinato per
    righe;

```

- X ordina l'elenco degli oggetti del filesystem in base alla loro estensione (l'insieme dei caratteri che segue l'ultimo carattere '.');
- 1 visualizza un oggetto del filesystem per riga;

path è il path che specifica la directory di cui visualizzare il contenuto. Può essere anche il nome del file di cui visualizzare le informazioni salienti. Se non è specificato viene visualizzato il contenuto della working directory.

Prima dell'elenco dei nomi degli oggetti del filesystem è visualizzata una riga del tipo

total blocks

dove *blocks* indica la porzione del filesystem occupata dagli oggetti elencati.

I colori con i quali sono visualizzati i file, sono impostati nella variabile di ambiente `LS_COLORS`. In genere tale variabile viene impostata con il comando `dircolors` (man page `dircolors(1)`).

Comando: `dircolors`

Path: `/usr/bin/dircolors`

SINTASSI

\$ dircolors [*option*] [*filename*]

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di `dircolors`.

Puo assumere i seguenti valori:

- b | --sh | --bourne-shell
imposta la variabile d'ambiente `LS_COLORS` utilizzando la sintassi della shell *Bash*;
- c | --csh | --c-shell
imposta la variabile d'ambiente `LS_COLORS` utilizzando la sintassi della *C* shell;
- p | --print-database
visualizza le impostazioni di default;
- help visualizza un aiuto sommario di `dircolors`;
- version
visualizza la versione di `dircolors`;

filename legge le impostazioni dei colori dal file *filename* (se non specificato, le impostazioni sono quelle di default);

Generalmente i file che contengono le impostazioni dei colori della visualizzazione dei file sono `/etc/DIR_COLORS`, a livello di sistema, e `~/.dir_colors`, a livello utente (v. man page `dir_colors(5)`). Questi sono file di testo che contengono delle direttive costituite da righe con la seguente sintassi

TERM *terminal-type*

indica l'inizio di una sezione di uno specifico tipo di terminale;

COLOR {*yes* | *all* | *no* | *none* | *tty*}

(solo per la distribuzione Slackware) indica se i colori di visualizzazione devono essere sempre utilizzati o meno oppure soltanto nei terminali TTY (per default è *no*);

EIGHTBIT {*yes* | *no*}

(solo per la distribuzione Slackware) indica se l'insieme di caratteri ISO 8859 a 8 bit deve essere abilitato o meno – per compatibilità 1 equivale a *yes* e 0 equivale a *no* (per default è *no*);

OPTIONS *options*

(solo per la distribuzione Slackware) aggiunge le opzioni *options* sulla riga di comando di **ls**;

NORMAL *color*

specifica il colore con il quale visualizzare il testo che non fa parte di un nome di un file, secondo quanto indicato da *color*;

FILE *color*

specifica il colore con il quale visualizzare il nome dei file standard, secondo quanto indicato da *color*;

DIR *color*

specifica il colore con il quale visualizzare il nome delle directory, secondo quanto indicato da *color*;

LINK *color*

specifica il colore con il quale visualizzare il nome dei symbolic link, secondo quanto indicato da *color*;

ORPHAN *color*

specifica il colore con il quale visualizzare il nome dei symbolic link che non si riferiscono ad un file esistente, secondo quanto indicato da *color* (se non specificato viene utilizzato il colore indicato da **LINK**);

MISSING *color*

specifica il colore con il quale visualizzare il nome dei file esistenti che hanno dei symbolic link che li riferiscono, secondo quanto indicato da *color* (se non specificato viene utilizzato il colore indicato da **FILE**);

FIFO *color*

specifica il colore con il quale visualizzare il nome delle FIFO (named pipe), secondo quanto indicato da *color*;

SOCK *color*

specifica il colore con il quale visualizzare il nome dei socket, secondo quanto indicato da *color*;

DOOR *color*

specifica il colore con il quale visualizzare il nome delle door (Solaris 2.5 e successivi), secondo quanto indicato da *color*;

BLK *color*

specifica il colore con il quale visualizzare il nome dei file di dispositivi a blocchi, secondo quanto indicato da *color*;

CHR *color*

specifica il colore con il quale visualizzare il nome dei file di dispositivi a caratteri, secondo quanto indicato da *color*;

EXEC *color*

specifica il colore con il quale visualizzare il nome dei file eseguibili, secondo quanto indicato da *color*;

LEFTCODE *color*

???

RIGHTCODE *color*

???

ENDCODE *color*

???

***ending** *color*

specifica il colore con il quale visualizzare il nome dei file che terminano in *ending*, secondo quanto indicato da *color*;

`.ext color`

specifica il colore con il quale visualizzare il nome dei file con estensione `ext`, secondo quanto indicato da `color` (obsoleto);

???

I colori sono specificati secondo i codici ISO 6429 (ANSI) supportati dalla maggior parte dei terminali attuali. Tali codici sono composti da sequenze di numeri (v. tab. 3.16) separati dal carattere ‘;’.

Codice	Colore
0	colore di default
1	colore più luminoso
4	sottolineatura
5	testo lampeggiante
30	testo nero
31	testo rosso
32	testo verde
33	testo giallo
34	testo blu
35	testo porpora
36	testo ciano
37	testo grigio
40	sfondo nero
41	sfondo rosso
42	sfondo verde
43	sfondo giallo
44	sfondo blu
45	sfondo porpora
46	sfondo ciano
47	sfondo grigio

Tabella 3.16: Codici dei colori secondo l’ISO 6492 (ANSI).

Il comando `ls` utilizza le seguenti impostazioni di default:

```
NORMAL  0
FILE    0
DIR     32
LINK    36
FIFO    31
SOCK    33
BLK     44;37
CHR     44;37
XEC     35
```

Le direttive che precedono la prima direttiva `TERM` sono valide per qualunque tipo di terminale.

Di seguito è riportato un esempio del file `/etc/DIR.COLORS`.

```
# Configuration file for the color ls utility
# This file goes in the /etc directory, and must be world readable.
# You can copy this file to .dir_colors in your $HOME directory to override
# the system defaults.

# COLOR needs one of these arguments: 'tty' colorizes output to ttys, but not
# pipes. 'all' adds color characters to all output. 'none' shuts colorization
# off.
COLOR tty

# Extra command line options for ls go here.
# Basically these ones are:
# -F = show '/' for dirs, '*' for executables, etc.
# -T 0 = don't trust tab spacing when formatting ls output.
```

```
OPTIONS -F -T 0
```

```
# Below, there should be one TERM entry for each termtype that is colorizable
TERM linux
TERM console
TERM con132x25
TERM con132x30
TERM con132x43
TERM con132x60
TERM con80x25
TERM con80x28
TERM con80x30
TERM con80x43
TERM con80x50
TERM con80x60
TERM cons25
TERM xterm
TERM rxvt
TERM xterm-color
TERM color-xterm
TERM vt100
TERM dtterm
TERM color_xterm
TERM ansi
TERM screen
```

```
# EIGHTBIT, followed by '1' for on, '0' for off. (8-bit output)
EIGHTBIT 1
```

```
# Below are the color init strings for the basic file types. A color init
# string consists of one or more of the following numeric codes:
# Attribute codes:
# 00=none 01=bold 04=underscore 05=blink 07=reverse 08=concealed
# Text color codes:
# 30=black 31=red 32=green 33=yellow 34=blue 35=magenta 36=cyan 37=white
# Background color codes:
# 40=black 41=red 42=green 43=yellow 44=blue 45=magenta 46=cyan 47=white
NORMAL 00      # global default, although everything should be something.
FILE 00        # normal file
DIR 01;34      # directory
LINK 01;36     # symbolic link
FIFO 40;33     # pipe
SOCK 01;35     # socket
BLK 40;33;01   # block device driver
CHR 40;33;01   # character device driver
ORPHAN 01;05;37;41 # orphaned symlinks
MISSING 01;05;37;41 # ... and the files they point to
```

```
# This is for files with execute permission:
EXEC 01;32
```

```
# List any file extensions like '.gz' or '.tar' that you would like ls
# to colorize below. Put the extension, a space, and the color init string.
# (and any comments you want to add after a '#')
.cmd 01;32 # executables (bright green)
.exe 01;32
.com 01;32
.btm 01;32
.bat 01;32
.sh 01;32
.csh 01;32
.tar 01;31 # archives or compressed (bright red)
```

```
.tgz 01;31
.arj 01;31
.taz 01;31
.lzh 01;31
.zip 01;31
.z 01;31
.Z 01;31
.gz 01;31
.bz2 01;31
.bz 01;31
.tz 01;31
.rpm 01;31
.cpio 01;31
.jpg 01;35 # image formats
.gif 01;35
.bmp 01;35
.xbm 01;35
.xpm 01;35
.png 01;35
.tif 01;35
```

Alcuni esempi di utilizzo di `ls`

```
[daniele@Zeus ~]$ ls /
bin  documents  initrd      misc  proc  sbin  TotemPrj
boot etc        lib          mnt   public tftpboot usr
dev  home        lost+found  opt   root  tmp   var
[daniele@Zeus ~]$ _
```

```
[daniele@Zeus ~]$ ls -l /
total 2820
[...]
drwxr-xr-x 4 root root 4096 May 13 2003 gnome
drwxr-xr-x 3 root root 4096 May 13 2003 gnome-vfs-2.0
-rw-r--r-- 1 root root 9167 Feb 4 2003 gnome-vfs-mime-magic
-rw-r--r-- 1 root root 1756 Jan 29 2003 gpm-root.conf
-rw-r--r-- 1 root root 694 May 13 2003 group
-rw----- 1 root root 682 Nov 15 2002 group-
-rw----- 1 root root 5 May 13 2003 group.lock
lrwxrwxrwx 1 root root 22 Nov 15 2002 grub.conf -> ../boot/grub/grub.conf
-r----- 1 root root 575 May 13 2003 gshadow
-rw----- 1 root root 565 Nov 15 2002 gshadow-
-rw----- 1 root root 5 May 13 2003 gshadow.lock
drwxr-xr-x 2 root root 4096 May 13 2003 gtk
drwxr-xr-x 2 root root 4096 May 13 2003 gtk-2.0
-rw-r--r-- 1 root root 17 Jul 23 2000 host.conf
-rw-r--r-- 2 root root 179 Oct 23 00:04 hosts
-rw-r--r-- 1 root root 161 Jan 13 2000 hosts.allow
-rw-r--r-- 1 root root 179 Oct 4 01:14 hosts.bak
-rw-r--r-- 1 root root 5 Feb 4 2003 hosts.canna
-rw-r--r-- 1 root root 347 Jan 13 2000 hosts.deny
drwxr-xr-x 4 root root 4096 May 13 2003 hotplug
-rw-r--r-- 1 root root 8510 Jan 26 2003 htdig.conf
drwxr-xr-x 4 root root 4096 Nov 29 19:32 httpd
-rw-r--r-- 1 root root 3376 Jan 24 2003 im_palette.pal
-rw-r--r-- 1 root root 920 Jan 24 2003 im_palette-small.pal
-rw-r--r-- 1 root root 224 Jan 24 2003 im_palette-tiny.pal
-rw-r--r-- 1 root root 5464 Jan 24 2003 imrc
```

```

-rw-r--r--  1 root    root      1247 Nov 14  2002 info-dir.rpmsave
lrwxrwxrwx  1 root    root        11 May 13  2003 init.d -> rc.d/init.d
-rw-r--r--  1 root    root      658 Mar 13  2003 initlog.conf
-rw-r--r--  1 root    root     1756 Nov 15  2002 inittab
[...]
[daniele@Zeus ~]$ _

```

???

La *working directory*, che per default è quella contenuta nella variabile di ambiente `HOME`, può essere visualizzata con il comando `pwd` (**p**rint **w**orking **d**irectory). Generalmente la shell ha un comando interno `pwd` e le relative informazioni sono ottenibili con il comando `help pwd` come descritto nel cap. 7.

Comando (bash): `pwd`

SINTASSI

`$ cd [-L|-P]`

DESCRIZIONE

`-P` visualizza la *working directory* senza i symbolic link;

`-L` forza `pwd` a seguire i symbolic link;

Digitando semplicemente

`$ pwd`

verrà eseguito il comando interno della shell. Se si desidera eseguire il comando `pwd` di sistema (man page `pwd(1)`), è necessario specificare il suo path (relativo o assoluto), come, ad esempio

`$ /bin/pwd`

Ad esempio il comando

```
[daniele@Zeus ~]$ pwd
```

dà il seguente risultato

```
/home/daniele
```

ovvero indica il fatto che la *working directory* è attualmente impostata alla *home directory* dell'utente identificato dallo username "daniele".

La *working directory* può essere variata per mezzo del comando `cd` (**c**hange **d**irectory), anch'esso un comando interno della shell.

Comando (bash): `cd`

SINTASSI

`$ cd [-L|-P] [dir]`

DESCRIZIONE

`-P` indica a `cd` di non seguire i symbolic link;

`-L` forza `cd` di seguire i symbolic link;

dir è la directory da far diventare *working directory*. La directory *dir* viene ricercata nel percorso (elenco di directory separato dal carattere ':') specificato dalla variabile di ambiente `CDPATH`, a meno che non inizi con il carattere '/'. Se la directory indicata non esiste e l'opzione della shell `cdable_vars` è impostata, `cd` considera il contenuto della variabile di ambiente indicata da *dir*.

3.11 La directory /proc

La directory `/proc` è il mount-point di un tipo di filesystem particolare, “proc”, che è un filesystem virtuale, nel senso che non occupa spazio sulla memoria di massa, il quale permette di presentare sottoforma di file e directory alcuni parametri del sistema. In questo modo è possibile gestire tali parametri direttamente dalla shell.

Il contenuto dei file contenuti all’interno di `/proc` ed in tutte le sue sottodirectory, viene aggiornato in tempo reale dal kernel e viene generalmente utilizzato dai programmi che hanno bisogno di conoscere lo stato del sistema. I comandi come `ps` o `top`²⁷, che visualizzano i processi presenti sul sistema con i loro relativi stati, attingono le informazioni da `/proc`, come anche i comandi `lspci`, `scanpci`, `pnpdump` che visualizzano le caratteristiche delle periferiche PCI o ISA riconosciute dal sistema.

Poiché `/proc` è un filesystem virtuale residente in memoria centrale (RAM), tutte le volte che il sistema viene avviato, questo viene creato con la caratteristica che la dimensione dei file e directory in esso presenti è sempre 0 e la data dell’ultima modifica è quella corrente.

All’interno di `/proc` sono contenute varie directory, alcune delle quali hanno un nome costituito soltanto da caratteri numerici come riportato di seguito

```
[root@Zeus ~]# ls /proc
1      1158  1182  408  700  8      filesystems  meminfo      sys
1084   1160  1184  425  731  822    fs           misc          sysvipc
1126   1164  1185  480  740  823    ide          modules       tty
1129   1166  12    5    744  877    interrupts   mounts        uptime
1131   1167  1215  520  753  apm     iomem        mtrr          version
1133   1169  1241  6    762  bus     ioports      net
1136   1170  1244  615  771  cmdline  irq           partitions
1139   1171  1293  647  772  cpuinfo  kcore        pci
1144   1172  159   67   773  devices  kmsg         self
1147   1174  2     671  774  dma      ksyms        slabinfo
1149   1176  3     681  775  driver   loadavg      speakup
1151   1178  4     691  776  execdomains  locks        stat
1156   1180  404   7    777  fb        mdstat       swaps
```

Ognuna di tali directory rappresenta un processo in esecuzione sul sistema ed il suo nome è l’identificatore univoco del processo, o PID (Process Identifier)²⁸. Al suo interno vi sono i seguenti oggetti

`cmdline`

è un file che contiene l’intera riga di comando che ha lanciato in esecuzione il processo;

`cwd` è un symbolic link alla (current) working directory del processo;

`environ`

è un file che contiene l’ambiente (environment) del processo, ovvero tutte le variabili di ambiente definite al lancio del processo;

`exe` è un symbolic link al file eseguibile che ha lanciato il processo;

`fd` è una directory che contiene l’elenco dei descrittori dei file aperti dal processo;

`maps` è una *named pipe* (o FIFO) che contiene lo spazio di indirizzi del processo attualmente mappato in un file. I campi in esso contenuti sono

- lo spazio di indirizzi associato al mapping;
- i permessi associati al mapping;
- l’offset dall’inizio del file a cui inizia il mapping;

²⁷v. cap. 6.

²⁸v. cap. 6.

- il dispositivo sul quale risiede il file;
- il numero dell'inode del file;
- il nome del file;

`root` è un symbolic link alla directory che è “vista” dal processo come root directory;

`status`

è un file che contiene informazioni sul processo: il suo nome, il suo stato, il suo PID, il suo UID, il suo PPID, ...

Ad esempio, il contenuto della directory `/proc/408` è riportato di seguito

```
[root@Zeus ~]# ls -l /proc/408
total 0
-r--r--r-- 1 root root 0 Mar 27 22:42 cmdline
lrwxrwxrwx 1 root root 0 Mar 27 22:42 cwd -> /
-r----- 1 root root 0 Mar 27 22:42 environ
lrwxrwxrwx 1 root root 0 Mar 27 22:42 exe -> /sbin/klogd
dr-x----- 2 root root 0 Mar 27 22:42 fd
-r--r--r-- 1 root root 0 Mar 27 22:42 maps
-rw----- 1 root root 0 Mar 27 22:42 mem
-r--r--r-- 1 root root 0 Mar 27 22:42 mounts
lrwxrwxrwx 1 root root 0 Mar 27 22:42 root -> /
-r--r--r-- 1 root root 0 Mar 27 22:42 stat
-r--r--r-- 1 root root 0 Mar 27 22:42 statm
-r--r--r-- 1 root root 0 Mar 27 22:42 status
```

Gli altri file presenti nella directory `/proc` contengono informazioni di varia natura, tra le quali le principali sono quelle riportate di seguito

`/proc/cpuinfo`

contiene informazioni sulla CPU: il tipo, il modello ed altre caratteristiche;

`/proc/devices`

contiene l'elenco dei dispositivi riconosciuti dal sistema;

`/proc/dma`

contiene i canali DMA (Direct Memory Access) correntemente utilizzati dal sistema;

`/proc/filesystems`

contiene informazioni sui filesystem configurati nel kernel;

`/proc/interrupts`

contiene l'elenco degli interrupts in uso e quante volte si sono verificati;

`/proc/ioports`

contiene l'elenco delle porte di I/O (Input/Output) correntemente in uso dal sistema;

La directory `/proc/sys` contiene vari parametri del sistema operativo;

La directory `/proc/bus` contiene le informazioni relative ai bus riconosciuti dal sistema. Al suo interno si trovano le directory `pci` e `usb` che contengono rispettivamente le informazioni relative ai bus PCI e USB. Ad esempio, nel file `/proc/bus/usb/devices` è contenuto qualcosa di simile a quanto riportato di seguito

```
T: Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=12 MxCh= 2
B: Alloc= 0/900 us ( 0%), #Int= 0, #Iso= 0
D: Ver= 1.10 Cls=09(hub ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0000 ProdID=0000 Rev= 0.00
S: Product=USB OHCI Root Hub
S: SerialNumber=c8049000
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr= 0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E: Ad=81(I) Atr=03(Int.) MxPS= 2 Iv1=255ms
```

che indica la presenza di un controller USB (S: Product=USB OHCI Root Hub), compatibile con la versione USB 1.1 (D: Ver= 1.10). Al momento dell'inserimento di un dispositivo USB il contenuto di tale file cambierà, riportando anche le caratteristiche del dispositivo riconosciuto.

3.12 La directory /dev

Questa directory contiene essenzialmente i dispositivi riconosciuti dal sistema, come

fdn unità di lettura/scrittura dei floppy disk;
hdx unità a disco ATA (hard disk, CD-ROM);
sdx unità a disco SCSI (hard disk, CD-ROM, hard disk USB);
stn unità a nastro SCSI;
cuan porta seriale;
ttySn porta seriale;
loopn è l'interfaccia di rete di loopback;²⁹
lpn stampante;
scdn masterizzatore CD (SCSI);
sgx dispositivo SCSI;

Nella directory sono presenti dei symbolic link che aiutano a ricordare il significato dei vari dispositivi.

cdrom è un symbolic link ad una unità CD-ROM (**hdx**, **sdx**);
tape è un symbolic link ad una unità a nastro (**stx**);
mouse è un symbolic link ad un mouse (**psaux**);
fax è un symbolic link ad un FAX (**ttySn**);
modem è un symbolic link ad un modem (**ttySn**);

Di seguito è riportato un estratto del contenuto della directory /dev

```
# ls -l /dev
-rw-r--r-- 1 root root 5 Nov 1 09:01 :0
crw-r--r-- 1 root root 10, 175 Aug 31 2002 agpgart
crw----- 1 daniele root 10, 134 Aug 31 2002 apm_bios
drwxr-xr-x 2 root root 4096 Oct 20 01:30 ataraid
crw----- 1 root root 10, 3 Aug 31 2002 atibm
crw----- 1 daniele root 14, 4 Aug 31 2002 audio
crw----- 1 daniele root 14, 20 Aug 31 2002 audio1
crw----- 1 daniele root 14, 7 Aug 31 2002 audioc1
brw-rw---- 1 root disk 29, 0 Aug 31 2002 aztcd
crw----- 1 daniele root 10, 128 Aug 31 2002 beep
brw-rw---- 1 root disk 41, 0 Aug 31 2002 bpcd
crw----- 1 root root 68, 0 Aug 31 2002 capi20

lrwxrwxrwx 1 root root 8 Oct 20 00:19 cdrom -> /dev/hdc
crw----- 1 daniele root 5, 1 Mar 27 22:14 console
crw----- 1 root root 1, 6 Aug 31 2002 core
drwxr-xr-x 18 root root 4096 Oct 20 01:30 cpu
```

²⁹v. cap. 17.

```

crw-rw---- 1 root    uucp      5,  64 Aug 31  2002 cua0
crw-rw---- 1 root    uucp      5,  65 Aug 31  2002 cua1
crw-rw---- 1 root    uucp      5,  66 Aug 31  2002 cua2

lrwxrwxrwx 1 root    root              3 Oct 20 01:30 fb -> fb0
crw----- 1 daniela root      29,   0 Aug 31  2002 fb0
crw----- 1 daniela root      29,   1 Aug 31  2002 fb1
crw----- 1 daniela root      29,  10 Aug 31  2002 fb10
crw----- 1 daniela root      29,  11 Aug 31  2002 fb11

lrwxrwxrwx 1 root    root      15 Oct 20 01:30 fd -> ../proc/self/fd
brw-rw---- 1 daniela floppy    2,   0 Aug 31  2002 fd0
brw-rw---- 1 daniela floppy    2,   4 Aug 31  2002 fd0d360
brw-rw---- 1 daniela floppy    2,  12 Aug 31  2002 fd0D360
brw-rw---- 1 daniela floppy    2,  16 Aug 31  2002 fd0D720
brw-rw---- 1 daniela floppy    2,   8 Aug 31  2002 fd0h1200
brw-rw---- 1 daniela floppy    2,  40 Aug 31  2002 fd0h1440
brw-rw---- 1 daniela floppy    2,  28 Aug 31  2002 fd0H1440

brw-rw---- 1 root    disk      3,   0 Aug 31  2002 hda
brw-rw---- 1 root    disk      3,   1 Aug 31  2002 hda1
brw-rw---- 1 root    disk      3,  10 Aug 31  2002 hda10
brw-rw---- 1 root    disk      3,  11 Aug 31  2002 hda11

brw-rw---- 1 root    disk      3,  64 Aug 31  2002 hdb
brw-rw---- 1 root    disk      3,  65 Aug 31  2002 hdb1
brw-rw---- 1 root    disk      3,  74 Aug 31  2002 hdb10
brw-rw---- 1 root    disk      3,  75 Aug 31  2002 hdb11

brw-rw---- 1 root    disk      7,   0 Aug 31  2002 loop0
brw-rw---- 1 root    disk      7,   1 Aug 31  2002 loop1
brw-rw---- 1 root    disk      7,  10 Aug 31  2002 loop10
brw-rw---- 1 root    disk      7,  11 Aug 31  2002 loop11

crw-rw---- 1 root    lp        6,   0 Aug 31  2002 lp0
crw-rw---- 1 root    lp        6,   1 Aug 31  2002 lp1
crw-rw---- 1 root    lp        6,   2 Aug 31  2002 lp2
crw-rw---- 1 root    lp        6,   3 Aug 31  2002 lp3
crw-r----- 1 root    root    109,   0 Oct 20 02:12 lvm
-rwxr-xr-x 1 root    root    20589 Aug 31  2002 MAKEDEV
brw-rw---- 1 root    disk      23,   0 Aug 31  2002 mcd
brw-rw---- 1 root    disk      20,   0 Aug 31  2002 mcdx
brw-rw---- 1 root    disk      9,   0 Oct 20 01:33 md0
brw-rw---- 1 root    disk      9,   1 Oct 20 01:33 md1
brw-rw---- 1 root    disk      9,  10 Oct 20 01:33 md10
brw-rw---- 1 root    disk      9,  11 Oct 20 01:33 md11

crw-rw-rw- 1 root    root      1,   3 Aug 31  2002 null

lrwxrwxrwx 1 root    root              4 Oct 20 01:30 ram -> ram1
brw-rw---- 1 root    disk      1,   0 Aug 31  2002 ram0
brw-rw---- 1 root    disk      1,   1 Aug 31  2002 ram1
brw-rw---- 1 root    disk      1,  10 Aug 31  2002 ram10

lrwxrwxrwx 1 root    root              4 Oct 20 01:30 ramdisk -> ram0
crw-r--r-- 1 root    root      1,   8 Aug 31  2002 random
drwxr-xr-x 2 root    root    4096 Oct 20 01:30 raw

brw-rw---- 1 root    disk      8,   0 Aug 31  2002 sda
brw-rw---- 1 root    disk      8,   1 Aug 31  2002 sda1
brw-rw---- 1 root    disk      8,  10 Aug 31  2002 sda10
brw-rw---- 1 root    disk      8,  11 Aug 31  2002 sda11

```

```

crw-rw----  1 root    disk      21,   0 Aug 31  2002 sg0
crw-rw----  1 root    disk      21,   1 Aug 31  2002 sg1
crw-rw----  1 root    disk      21,  10 Aug 31  2002 sg10
crw-rw----  1 root    disk      21,  11 Aug 31  2002 sg11

lrwxrwxrwx  1 root    root              3 Oct 20 01:31 sga -> sg0
lrwxrwxrwx  1 root    root              3 Oct 20 01:31 sgb -> sg1

crw-rw----  1 root    disk          9,   0 Aug 31  2002 st0
crw-rw----  1 root    disk          9,  96 Aug 31  2002 st0a
crw-rw----  1 root    disk          9,  32 Aug 31  2002 st0l

lrwxrwxrwx  1 root    root      17 Oct 20 01:31 stderr -> ../proc/self/fd/2
lrwxrwxrwx  1 root    root      17 Oct 20 01:31 stdin  -> ../proc/self/fd/0
lrwxrwxrwx  1 root    root      17 Oct 20 01:31 stdout -> ../proc/self/fd/1

lrwxrwxrwx  1 root    root          4 Oct 20 01:31 systty -> tty0

crw-rw-rw-  1 root    root          5,   0 Aug 31  2002 tty
crw--w----  1 root    root          4,   0 Aug 31  2002 tty0
crw-----  1 root    root          4,   1 Mar 27 22:14 tty1
crw--w----  1 root    tty          4,  10 Aug 31  2002 tty10
crw--w----  1 root    tty          4,  11 Aug 31  2002 tty11

crw-rw-rw-  1 root    tty          3,  48 Aug 31  2002 ttys0
crw-rw----  1 root    uucp          4,  64 Nov  3 00:14 ttyS0
crw-rw-rw-  1 root    tty          3,  49 Aug 31  2002 ttys1
crw-rw----  1 root    uucp          4,  65 Aug 31  2002 ttyS1
crw-rw----  1 root    uucp          4,  74 Aug 31  2002 ttyS10
crw-rw----  1 root    uucp          4,  75 Aug 31  2002 ttyS11

brw-rw----  1 root    disk      13,   0 Aug 31  2002 xda
brw-rw----  1 root    disk      13,   1 Aug 31  2002 xda1
brw-rw----  1 root    disk      13,  10 Aug 31  2002 xda10

crw-rw-rw-  1 root    root          1,   5 Aug 31  2002 zero

```

3.13 La directory /etc/sysconfig

Nella directory `/etc/sysconfig` sono contenuti vari file e directory che contengono informazioni e parametri di configurazione per i vari servizi. L'elenco completo dei file e del significato del loro contenuto, si trova nel file `/usr/share/doc/initscripts-version/sysconfig.txt`. Di seguito è illustrato il significato di alcuni dei file contenuti in `/etc/sysconfig` (molti dei file presenti in questa directory si riferiscono alla configurazione di servizi di rete che saranno trattati nella parte II).

- amd** contiene i parametri relativi al sistema automatico di mounting/unmounting dei filesystem, gestito dal daemon **amd**.
- apmd** contiene la configurazione del daemon **apmd** che gestisce il servizio di APM (Advanced Power Management), cioè il monitoraggio dello stato della batteria e la visualizzazione dell'avviso dell'eventuale basso livello della stessa.
- arpwatch** contiene i parametri da passare all'avvio del daemon **arpwatch**, che mantiene aggiornate le corrispondenze tra i MAC address ed i relativi indirizzi IP.
- authconfig** contiene le impostazioni relative al tipo di autenticazione che deve essere utilizzata sulla macchina (MD5, Kerberos, LDAP).

clock contiene informazioni per l'interpretazione dei valori letti dal system clock³⁰. All'interno del file sono generalmente utilizzate le seguenti direttive

UTC=*value*

dove *value* indica se il valore del system clock si riferisce all'Universal Time (**true**) o meno.

ARC=*value*

dove *value* indica se l'indicazione dell'anno ha un offset di 42 anni (**true**) o si assume la UNIX epoch (solo per piattaforme basate su microprocessore Alpha).

ZONE=*filename*

dove *filename* indica il file della zona, presente nella directory `/usr/share/zoneinfo`, di cui `/etc/localtime` è una copia.

desktop

indica il desktop manager utilizzato, come **DESKTOP="GNOME"**.³¹

dhcpcd contiene i parametri da passare all'avvio del daemon **dhcpcd**, che implementa i servizi DHCP (Dynamic Host Configuration Protocol) e BOOTP (Boot Protocol).

gpm contiene i parametri da passare all'avvio del daemon **gpm**, che implementa il servizio di gestione del mouse (per default il mouse viene associato al file di dispositivo `/dev/mouse`).

harddisks

contiene le impostazioni relative agli hard disk. Possono essere utilizzati anche i file **hardiskhd[a-h]**.

Il cambiamento delle impostazioni può rendere inaccessibili le informazioni memorizzate sugli hard disk.

hwconf

elenca i dispositivi che il sistema ha riconosciuto. (solo RedHat???). Non è consigliabile modificarlo manualmente.

i18n contiene le impostazioni relative alla lingua di default (es. **LANG="en_US"**).

identd

contiene i parametri da passare all'avvio del daemon **identd**, che ritorna lo username dell'utente al quale sono associati i processi che effettuano connessioni TCP/IP.

Alcuni servizi di rete (come i server FTP e IRC) rallentano le performances se **initd** non è in esecuzione, ma tale servizio non è necessario, quindi ai fini della sicurezza è meglio non farlo eseguire.

init contiene indicazioni sul comportamento del sistema durante la fase di avvio. Può contenere le seguenti direttive

BOOTUP=*value*

dove *value* indica se utilizzare la visualizzazione a colori dell'esito dell'avvio dei servizi (**color**), se utilizzare la visualizzazione standard più verbosa rispetto alla semplice indicazione del successo o meno dell'avvio di un servizio (**verbose**), o una visualizzazione senza formattazione ANSI.

RES_COL=*value*

dove *value* indica la colonna alla quale iniziare a visualizzare l'indicazione dell'esito dell'avvio dei servizi (default 60).

³⁰v. cap. 9.

³¹v. cap. 11.

MOVE_TO_COL=*value*

dove *value* indica la sequenza di escape che sposta il cursore alla colonna RES_COL. (default MOVE_TO_COL="echo -en \\033[\$RES_COLG]").

SETCOLOR_SUCCESS=*value*

dove *value* indica il colore con cui visualizzare gli esiti relativi al successo nell'avvio dei servizi (default SETCOLOR_SUCCESS="echo -en \\033[0;32m" – verde).

SETCOLOR_FAILURE=*value*

dove *value* indica il colore con cui visualizzare gli esiti relativi ad un errore nell'avvio dei servizi (default SETCOLOR_FAILURE="echo -en \\033[0;31m" – rosso).

SETCOLOR_WARNING=*value*

dove *value* indica il colore con cui visualizzare gli esiti relativi ad un avvertimento, ovvero qualcosa di strano si è verificato durante l'avvio dei servizi (default SETCOLOR_WARNING="echo -en \\033[0;33m" – giallo).

SETCOLOR_NORMAL=*value*

dove *value* indica il colore con cui visualizzare i messaggi sullo schermo (default SETCOLOR_NORMAL="echo -en \\033[0;39m" – grigio chiaro).

LOGLEVEL=*value*

dove *value* indica il livello di log per il kernel. Tale valore va da un minimo di 1, nel quale vengono tracciati soltanto gli eventi relativi ai kernel panic (errori irreversibili del kernel) ad 8, nel quale tutti gli eventi vengono tracciati. Il valore di default è 3. Al suo avvio, il daemon `syslogd` potrà reimpostare tale valore.

PROMPT=*value*

dove *value* indica se permettere (*yes*) o meno (*no*) l'avvio interattivo con la pressione di un determinato tasto (*hotkey*).

iptables

contiene le impostazioni di base utilizzate dal servizio di filtraggio dei pacchetti (firewalling). La modifica delle impostazioni in esso presenti può essere effettuata tramite il comando `/sbin/service iptables save`.³²

irda contiene le impostazioni relative alla configurazione dei dispositivi a infrarossi presenti sul sistema.

keyboard

contiene le impostazioni relative alla tastiera. Può contenere le seguenti direttive

KEYBOARDTYPE=*value*

dove *value* indica il tipo di tastiera: `pc` (una tastiera connessa alla porta PS/2) o `sun` (una tastiera Sun relativa al file di dispositivo `/dev/kbd`).

KEYTABLE=*filename*

dove *filename* indica il file relativo al layout di tastiera presente all'interno della directory `/usr/lib/kbd/keymaps/i386` (es. `KEYTABLE="us"`). Il nome dei file di tastiera ha generalmente l'estensione `kmap.gz`.

mouse contiene le impostazioni relative al mouse. Può contenere le seguenti direttive

FULLNAME=*value*

dove *value* indica il nome completo del tipo di mouse presente sul sistema.

MOUSETYPE=*value*

dove *value* indica il tipo di mouse: `microsoft` (un mouse *Microsoft*), `mouseman` (un mouse *MouseMan*), `mousesystems` (un mouse *Mouse Systems*), `ps/2` (un mouse PS/2), `msbm` (un mouse *Microsoft* di tipo bus), `logibm` (un mouse *Logitech* di tipo bus), `atibm` (un mouse *ATI* di tipo bus), `logitech` (un mouse *Logitech*), `mmseries` (un mouse *MouseMan* di vecchia generazione), `mmhittab` (una tavoletta grafica).

³²v. cap. ??.

XEMU3=value

dove *value* indica se emulare (**yes**) o meno (**no**) il terzo pulsante nell'interfaccia grafica. In genere i mouse più datati hanno soltanto due pulsanti, mentre GNU/Linux utilizza mouse a tre pulsanti; in tal caso il terzo pulsante viene emulato mediante una combinazione degli altri due.

XMOUSETYPE=value

dove *value* indica il tipo di mouse utilizzato nell'interfaccia grafica. Ha le stesse opzioni della direttiva **MOUSETYPE**.

DEVICE=filename

dove *filename* indica il file di dispositivo associato al mouse.

named contiene i parametri da passare all'avvio del daemon **named**, che implementa il servizio di server DNS (Domain Name System) tramite BIND (Berkeley Internet Name Domain).

netdump

contiene la configurazione relativa al servizio `/etc/init.d/netdump`, che invia i dati relativi ad operazioni e scarichi della memoria (dump) sulla rete.

Tale servizio non è tra quelli indispensabili, per cui è bene utilizzarlo soltanto se necessario.

network

contiene la configurazione relativa alla rete. Può contenere le seguenti direttive

NETWORKING=value

dove *value* indica se la rete è configurata (**yes**) o meno (**no**).

HOSTNAME=value

dove *value* indica il FQDN (Fully Qualified Domain Name) dell'interfaccia di rete (es. `hostname.domain.it`).

GATEWAY=value

dove *value* indica l'indirizzo IP del gateway.

GATEWAYDEV=value

dove *value* indica l'interfaccia connessa al gateway (es. `eth0`).

NISDOMAIN=value

dove *value* indica il nome del dominio NIS.

ntpd contiene i parametri da passare all'avvio del daemon **ntpd**, che implementa il servizio NTP (Network Time Protocol), per sincronizzare il system clock con un orologio di riferimento in rete. Le informazioni dettagliate possono essere reperite dal file `/usr/share/doc/ntp-version/ntpd.htm`.

pcmcia

contiene la configurazione relativa alla gestione dell'interfaccia PCMCIA.

radvd contiene i parametri da passare all'avvio del daemon **radvd**, che implementa il servizio di gestione delle informazioni di routing per IP versione 6, in modo che le macchine possano cambiare dinamicamente i loro router di default.

rawdevices

contiene la configurazione relativa ai dispositivi raw, come `/dev/raw/rawn`.

samba contiene i parametri da passare all'avvio dei daemon **smbd** e **nmbd**, che implementano il servizio di server SMB. Il daemon **smbd** permette la condivisione dei file verso le macchine con sistema operativo Windows, mentre il daemon **nmbd** offre il servizio di risoluzione dei nomi NetBIOS sul protocollo IP.

sendmail

contiene le impostazioni di default del daemon **sendmail**, che implementa il server di posta elettronica.

soundcard

contiene l'indicazione della scheda sonora, utilizzata da **sndconfig**. Le informazioni di configurazione della scheda sono contenute nel file **/etc/modules.conf**.

squid contiene i parametri da passare all'avvio del daemon **squid**, che implementa il servizio di proxy server. Informazioni relative alla configurazione di **squid** sono contenute in **/usr/share/doc/squid-version**.

ups contiene le informazioni relative alle UPS (Uninterruptible Power Supply) – batterie tampone – collegate al sistema.

vncservers

contiene le informazioni di avvio del server VNC (Virtual Network Computing), che implementa un sistema di gestione di desktop remoto indipendente dalla piattaforma.

xinetd

contiene i parametri da passare all'avvio del daemon **xinetd**, che implementa il servizio di TCP wrapper, ovvero gestisce l'avvio automatico dei processi che forniscono servizi Internet, qualora ne venga fatta una richiesta alla relativa porta.

???

Le aziende produttrici di distribuzioni aggiungono generalmente dei file di configurazione per il funzionamento di servizi da esse sviluppati. Ad esempio *Red Hat* introduce i file seguenti

???

firstboot

.

kudzu

.

3.14 I permessi e l'umask

Come illustrato in sez. 3.6, gli oggetti del filesystem sono caratterizzati da permessi di accesso per le varie categorie di utenti (utente proprietario, gruppo proprietario, altri). Tali permessi consentono agli utenti di operare sui relativi oggetti secondo quanto da essi specificato.

Alla creazione di un oggetto del filesystem i relativi permessi di accesso sono impostati secondo quanto specificato dal valore di **umask** (**user mask**), o maschera dei permessi, specifico per l'utente considerato. Questo valore, espresso generalmente in notazione ottale, indica quali sono i permessi che non devono essere abilitati alla creazione di un oggetto del filesystem. Il valore della maschera dei permessi può essere visualizzato con il comando interno di *Bash* **umask**.

umask

Comando: **umask**

SINTASSI

umask [*option*] [*mode*]

DESCRIZIONE

option indica la modalità di funzionamento di **umask**. Può assumere i seguenti valori

- p** visualizza il comando che imposta la maschera dei permessi come quella corrente;
- S** visualizza i permessi che verranno impostati automaticamente alla creazione di un oggetto del filesystem, secondo la maschera dei permessi corrente;

mode indica i permessi da impostare come maschera dei permessi;

Se non viene specificato niente, viene visualizzato il valore corrente di **umask**.

3.15 Limitazioni di accesso al filesystem

È possibile indicare al sistema quale deve essere considerata la root directory che determinati processi devono “vedere”, ovvero la directory corrispondente al simbolo ‘/’. Questo può essere fatto con il comando **chroot** (man page **chroot(1)**) (**change root directory**).

Comando: **chroot**
 Path: **/sbin/chroot**
 SINTASSI
chroot [*option*] *dirname* [*command*]

DESCRIZIONE

option modifica il funzionamento di **chroot**. Può assumere i seguenti valori

--help visualizza un aiuto sommario di **chroot**;
--version
 visualizza la versione di **chroot**;

dirname indica la directory da considerare come *root directory* (‘/’);

command indica il comando da eseguire. Se non è specificato, viene avviata una shell interattiva (il programma da lanciare come shell è indicato dalla variabile di ambiente **SHELL**).

Questo comando risulta molto utile nel caso in cui si voglia restringere l’accesso a determinate directory del filesystem solo a particolari utenti, cioè si vuol creare una cosiddetta **chroot jail** (prigione di chroot). Questo risulta particolarmente utile nel caso di servizi che permettono di accedere a parte del filesystem a chiunque, come ad esempio il FTP anonimo (v. cap. 19).

È evidente che creando una chroot jail è necessario che i privilegi del processo padre all’interno della jail, non siano quelli del superuser, altrimenti l’utente, avendo i privilegi di superuser, può creare una sottodirectory e quindi definire una nuova chroot jail su tale sottodirectory. In questo modo la working directory risulterebbe al di fuori della chroot jail, ovviando quindi alle restrizioni imposte.

La realizzazione di una chroot jail richiede però l’accortezza di far rimanere accessibili i comandi necessari all’utente (o il servizio “visto” da quell’utente) altrimenti certe funzionalità potrebbero risultare non più operative.

3.16 Lo swap

Nei sistemi GNU/Linux viene generalmente utilizzata una parte della memoria di massa, detta **swap**³³, per memorizzare dati temporanei che non riescono ad essere contenuti nella memoria centrale (RAM): una sorta di memoria di appoggio della memoria centrale. L’insieme formato dall’area di swap e dalla memoria centrale costituisce quella che viene comunemente denominata **virtual memory** (memoria virtuale).

Quando il kernel ha bisogno di memoria centrale da dedicare ad un’attività, parte del contenuto di quest’ultima viene “scaricato” nello swap (*swap out*). È il *virtual memory manager* (una parte del kernel) che decide quale parte della memoria deve essere scaricata, ad esempio quella acceduta meno frequentemente o non acceduta da più tempo delle altre.

Quando invece le informazioni alle quali un’attività fa riferimento sono presenti nello swap, queste devono essere ricaricate in memoria centrale (*swap in*).

La frequenza con la quale le informazioni vengono scaricate o ricaricate dallo swap può essere visualizzata con il comando **vmstat** (v. sez. 6.3.2).

Tale meccanismo può migliorare le prestazioni di macchine con una limitata quantità di memoria centrale. Lo swap, però, sebbene estenda la memoria centrale del sistema,

³³in inglese *scambiare*.

chroot jail

swap

virtual memory

non può essere considerato come parte della memoria centrale stessa in quanto i tempi di accesso allo swap sono quelli relativi alla memoria di massa, notevolmente superiori a quelli della memoria centrale.

I dati utilizzati dal kernel, come anche il codice del kernel stesso, non vengono mai scaricati dalla memoria centrale nello swap. Neanche il codice dei processi che girano in user space ha la necessità di essere scaricato nello swap poiché risiede già in una parte del filesystem dal quale è stato caricato in memoria centrale per essere eseguito. I dati relativi ai processi che girano in user space sono soggetti al meccanismo di swapping.

La dimensione dello swap dipende dall'utilizzo che si fa del sistema considerato: se si utilizza generalmente molta più memoria di quella fisicamente presente sul sistema si ha bisogno di una grande area di swap. In genere la dimensione dello swap dovrebbe essere uguale al doppio di quella della memoria centrale (RAM) e comunque un valore compreso tra 32 MiB e 2 GiB. È importante utilizzare una dimensione adeguata dello swap: non troppo piccola poiché potrebbe causare rallentamenti al sistema dovuti ad uno swapping eccessivo, né troppo grande per non avere uno spazio praticamente inutilizzato sul filesystem.

Lo swap qui presentato è di tipo *statico* nel senso che la sua dimensione è fissata quando viene creato. Esiste anche la possibilità di usare uno swap *dinamico* che incrementa o diminuisce la sua dimensione dipendentemente dal suo utilizzo da parte del sistema (**swpd** v. <http://ftp.linux.hr/pub/swpd>).

La parte di swap correntemente utilizzata dal sistema è visualizzabile con il comando **free** (v. sez. 6.3.2) o visualizzando il contenuto dei file **/proc/swaps** e **/proc/meminfo**.

Lo swap è generalmente costituito da un filesystem dedicato, in una partizione del disco. Si parla in tal caso di *swap filesystem* o *swap partition*. È comunque possibile realizzare lo swap con un file (*swap file*) od una combinazione dei due.

Per poter essere utilizzata, un'area di swap deve necessariamente essere inizializzata con il comando **mkswap** (man page **mkswap(8)**) (**make swap**).

Comando: **mkswap**

Path: **/sbin/mkswap**

SINTASSI

mkswap [option] device [size]

DESCRIZIONE

option indica la modalità di funzionamento di **mkswap**. Può assumere i seguenti valori

- c controlla che sul dispositivo *device* non vi siano errori nei blocchi prima di creare lo swap. Se vengono rilevati errori, viene visualizzato il numero di errori trovati;
- f forza la creazione dello swap anche se il comando ha poco senso (in questo modo è possibile creare aree di swap più grande del file o della partizione sulla quale risiede);
- p *page.size* imposta la dimensione delle pagine da utilizzare per la memorizzazione delle informazioni nello swap, secondo quanto specificato da *page.size* (valori tipici sono 4096 o 8192). Il suo utilizzo non è consigliato poiché potrebbe causare problemi con vecchie versioni delle librerie di sistema. La dimensione delle pagine di swap è contenuta nel file **/proc/cpuinfo**;
- v0 crea un'area di swap di versione 0 (la più vecchia). Gli ultimi 10 byte della prima pagina contengono il valore "SWAP_SPACE". Ogni bit della parte rimanente indica la presenza di una pagina utilizzabile (1) o meno (0) nell'area di swap. Poiché la prima pagina è utilizzata per contenere queste informazioni, il primo bit è 0. Quindi in tale versione è possibile utilizzare al massimo $8 \times (ps - 10) - 1$ pagine di swap (dove *ps* è la dimensione delle pagine di swap);

-v1 crea un'area di swap di versione 1 (la più recente). Gli ultimi 10 byte della prima pagina contengono il valore "SWAPSPACE2" (default);

device indica il dispositivo (partizione) o il file sul quale creare l'area di swap. Per utilizzare un file di swap è necessario crearlo con il comando

```
# dd if=device of=filename bs=block_size count=blocks
```

dove

device è il dispositivo nel quale creare il file di swap;

filename è il nome del file di swap;

block_size è la dimensione dei blocchi del file;

blocks è il numero totale dei blocchi di cui si compone il file;

È importante tenere presente che un file di swap non può essere uno sparse file (ovvero non può contenere dei buchi – *hole*), quindi non può essere utilizzato il comando **cp** per creare il file di swap;

size indica la dimensione desiderata dell'area di swap da creare in blocchi da 1024 byte. Se non viene specificata, viene utilizzata la dimensione del dispositivo o file indicato da **device**. Il suo utilizzo è sconsigliato – è rimasto per compatibilità con il passato;

L'utilizzo dello swap di GNU/Linux può essere attivato o disattivato durante il funzionamento del sistema, senza aver bisogno di effettuare un riavvio dello stesso. Questo può essere fatto per mezzo dei comandi **swapon** e **swapoff** (man page **swapon(8)**).

Comando: **swapon**

Path: **/sbin/swapon**

SINTASSI

```
# swapon [option] [filename]
```

DESCRIZIONE

option indica la modalità di funzionamento di **swapon**. Può assumere i seguenti valori

-h visualizza un help sommario di **swapon**;

-V visualizza la versione di **swapon**;

-s visualizza l'utilizzo della swap per ogni dispositivo;

-a attiva lo swap per i dispositivi indicati con "sw" (swap) nel file **/etc/fstab**. I dispositivi già montati come swap non vengono considerati;

-e se utilizzato con **-a** non visualizza un messaggio di errore nel caso un dispositivo non esista;

-p priority

imposta la priorità di swap-out secondo quanto specificato da **priority** (un valore numerico compreso tra 0 e 32767 – i valori più elevati indicano priorità più alta). Le aree di swap vengono utilizzate nell'ordine indicato dalla loro priorità (dalla priorità più alta a quella più bassa). Tale operazione può essere effettuata anche specificando l'opzione **pri=priority** nel file **/etc/fstab**;

filename indica il file di swap da considerare;

Comando: **swapoff**

Path: **/sbin/swapoff**

SINTASSI

```
# swapoff [option] [filename]
```

DESCRIZIONE

option indica la modalità di funzionamento di **swapon**. Può assumere i seguenti valori

- h visualizza un help sommario di **swapon**;
- V visualizza la versione di **swapon**;
- a disattiva lo swap per i dispositivi indicati con “**sw**” (swap) nel file **/etc/fstab**. I dispositivi già montati come swap non vengono considerati;

filename indica il file di swap da considerare;

Le aree di swap vengono considerate dal sistema al suo avvio, se indicate nel file **/etc/fstab**. Tra le opzioni specifiche può essere indicata l'opzione **pri** che specifica la priorità da assegnare all'area di swap considerata (v. opzione **-p priority** del comando **swapon**). Di seguito è riportata, ad esempio, la parte di un file **/etc/fstab** relativa alle direttive di montaggio dell'area di swap. Essa mostra due partizioni di swap (**/dev/hda2** e **/dev/hdc3**) con priorità più elevata (5) e due file di swap (**/work/swapfile.1** e **/work/swapfile.2**) con priorità più bassa (3).

```
# swap space in device partitions
/dev/hda2      none    swap    pri=5,defaults 0 0
/dev/hdc3      none    swap    pri=5,defaults 0 0

# swap space in swap files
/work/swapfile.1 none    swap    pri=3,defaults 0 0
/work/swapfile.2 none    swap    pri=3,defaults 0 0
```

In questo modo, quando il sistema avrà bisogno di effettuare operazioni di swapping, utilizzerà prima di tutto le partizioni **/dev/hda2** e **/dev/hdc3**. Nel momento in cui questo spazio viene esaurito, saranno utilizzati i file **/work/swapfile.1** e **/work/swapfile.2**.

L'utilizzo di più aree di swap (partizioni o file) su dischi differenti ha l'effetto di *striping* dei dischi, analogo a quello del RAID 0 (v. sez. 3.18), ovvero le informazioni vengono memorizzate su tutti i dischi. È dunque una buona idea avere più aree di swap (con la stessa priorità) se si hanno a disposizione più dischi su canali indipendenti. Così il kernel può utilizzare le aree di swap effettivamente in parallelo, senza perdere in prestazioni.

3.17 Il quota

Il sistema GNU/Linux permette di specificare dei limiti all'utilizzo della memoria di massa: il numero di inode che un utente (o gruppo) può possedere (esserne proprietario), ed il numero di blocchi disponibili per un utente (o gruppo). In questo modo si può limitare agli utenti l'utilizzo di parte di un filesystem. Questo può essere gestito con il meccanismo del **quota**, che però necessita che il kernel sia opportunamente predisposto (compilato) per il suo funzionamento. quota

Per attivare la gestione del quota su un filesystem, è necessario montare il filesystem in questione con l'opzione **usrquota** o **grpquota** (o anche entrambi) che indicano rispettivamente di abilitare il quota per utente o per gruppo.

La gestione del quota si basa sui file **quota.user** e **quota.group**, presenti nella directory principale del filesystem considerato (ovvero il relativo mount point). Tali file di gestione del quota, servono sia per la memorizzazione delle impostazioni (configurazione), che per la memorizzazione delle informazioni relative alla gestione dello stesso. In effetti esistono due versioni dei file di gestione del quota ed i file precedentemente indicati si riferiscono alla versione 1 (la più vecchia), mentre i corrispondenti file relativi alla versione 2 (la più recente) sono **aquota.user** e **aquota.group**. I file con estensione **.user** si riferiscono al quota gestito per utenti, mentre quelli con estensione **.group** sono relativi al quota per gruppi.

Quindi, si può effettuare un controllo della gestione del quota con il comando **quotacheck** (man page **quotacheck(8)**) (questo è necessario la prima volta per preparare i file di

configurazione del quota). All'avvio, **quotacheck** esamina ogni filesystem per il quale è abilitato il quota e si costruisce una tabella in cui riporta l'utilizzo dello stesso da parte dei utenti (e dei gruppi). Quindi confronta tale tabella con i dati presenti nei file di configurazione del quota. Se incontra una differenza, i file vengono aggiornati. È consigliabile far eseguire **quotacheck** sui filesystem effettivamente non utilizzati.

Comando: **quotacheck**
 Path: **/sbin/quotacheck**
 SINTASSI
quotacheck [*option*] [*filesystem*]

DESCRIZIONE

option specifica la modalità di funzionamento **quotacheck**. Può assumere i seguenti valori

- b** effettua un backup dei file di quota prima di scrivere altri dati;
- v** visualizza informazioni sul progresso di **quotacheck**;
- d** abilita la modalità di debug: vengono visualizzati molti messaggi durante il funzionamento del comando;
- f** forza il controllo dei filesystem sui quali è stato abilitato il quota;
- M** indica di effettuare un controllo del filesystem in modalità di lettura e scrittura se l'operazione di remount non va a buon fine (da usare soltanto nel caso in cui non ci sono processi che scrivono sul filesystem durante il controllo);
- m** non tentare di effettuare il remount del filesystem in modalità sola lettura (v. **-M**);
- i** abilita la modalità interattiva: per default se **quotacheck** rileva un errore termina la sua attività; in questo modo invece richiede se procedere;
- n** indica di considerare valida la prima entry relativa all'utente o il gruppo a cui è stato assegnato il quota (nei file di quota non corretti potrebbero esserci più entry relative allo stesso utente o gruppo);
- F *format_name***
 controlla il quota secondo il formato specificato da *format_name* (per default il controllo del quota effettua una rilevazione automatica del formato). I valori possibili sono riportati in tab. 3.17;

Formato	Descrizione
vfsold	quota versione 1.
vfsv0	quota versione 2.
rpc	quota su NFS.
xfs	quota su XFS.

Tabella 3.17: Formati possibili dei file di gestione del quota.

- a** controlla tutti i filesystem non NFS elencati in **/etc/mtab**;
- R** se utilizzato con **-a**, tutti i filesystem ad eccezione del root filesystem (**/**) vengono controllati;

filesystem indica il filesystem sul quale effettuare il controllo relativo al quota;

Dunque è possibile attivare il quota con il comando **quotaon** (man page **quotaon(8)**).

Comando: **quotaon**
 Path: **/sbin/quotaon**
 SINTASSI
quotaon [*option*] *filesystem*

DESCRIZIONE

option specifica la modalità di funzionamento **quotaon**. Può assumere i seguenti valori

- a attiva il quota per tutti i filesystem (non NFS) elencati in **/etc/fstab** che hanno l'opzione **usrquota** o **grpquota**;
- v visualizza un messaggio per ogni filesystem per il quale è stato attivato il quota;
- u attiva il quota per gli utenti (default);
- g attiva il quota per i gruppi;
- p visualizza lo stato attuale del quota (on o off);
- f esegue **quotaon** come se fosse stato lanciato **quotaoff**;

filesystem specifica il filesystem da considerare;

Analogamente, è possibile disabilitare il quota con il comando **quotaoff** (man page **quotaon(8)**).

Comando: **quotaoff**

Path: **/sbin/quotaoff**

SINTASSI

quotaoff [*option*] *filesystem*

DESCRIZIONE

option specifica la modalità di funzionamento **quotaoff**. Può assumere i seguenti valori

- F *format_name*
disattiva il quota per i filesystem con il cui formato dei file di configurazione del quota è specificato da *format_name* (v. tab. 3.17);
- a disattiva il quota per tutti i filesystem (non NFS) elencati in **/etc/fstab** che hanno l'opzione **usrquota** o **grpquota**;
- v visualizza un messaggio per ogni filesystem per il quale è stato disattivato il quota;
- u disattiva il quota per gli utenti (default);
- g disattiva il quota per i gruppi;
- p visualizza lo stato attuale del quota (on o off);
- x *delete*
(solo per XFS) libera lo spazio del filesystem utilizzato per memorizzare le informazioni sul quota;
- x *enforce*
(solo per XFS) disabilita i limiti sul filesystem relativi al quota;

filesystem specifica il filesystem da considerare;

Quindi, una volta attivato il quota è necessario configurarlo con il comando **edquota** (man page **edquota(8)**) o **setquota** (man page **setquota(8)**).

Comando: **edquota**

Path: **/usr/sbin/edquota**

SINTASSI

edquota [*option*] [*username*]

DESCRIZIONE

option specifica la modalità di funzionamento **edquota**. Può assumere i seguenti valori

- r | -n
configura il quota su sistemi remoti;
- u configura il quota per un utente (default);
- g configura il quota per un gruppo;

-p *protoname*
 duplica la configurazione del prototipo del quota indicato da *protoname* per gli utenti o i gruppi specificati;

-F *format_name*
 configura il quota secondo il formato specificato da *format_name* (i formati possibili sono elencati in tab. 3.17). Se non indicato viene effettuato un riconoscimento automatico del formato dei file di configurazione del quota;

-f *filesystem*
 configura il quota soltanto per il filesystem indicato da *filesystem*;

-t
 imposta il *grace period* per i filesystem per i quali è abilitato il quota;

username specifica l'utente o il gruppo per il quale si vuole configurare il quota (possono essere specificati anche un elenco di utenti o gruppi separati da uno spazio);

Ad esempio il comando

```
# edquota -u daniele
```

Visualizza le impostazioni contenute nel file `quota.user` relative allo username “daniele”, ovvero qualcosa di analogo a quanto riportato di seguito

```
Quotas for user daniele:
/dev/hda3: blocks in use: 2594, limits (soft = 5000, hard = 6500)
          inodes in use: 356, limits (soft = 1000, hard = 1500)
```

dove

blocks in use indica il numero di blocchi (in KiB) utilizzati dall'utente sullo specifico filesystem (partizione) in oggetto;

inodes in use indica il numero di inode (in KiB) utilizzati dall'utente sullo specifico filesystem (partizione) in oggetto;

limits indica i limiti di utilizzo dello specifico filesystem per l'utente considerato. Esistono due limiti: un *soft limit* ed un *hard limit*. Il *soft limit* può essere oltrepassato dall'utente per un periodo di tempo limitato, detto *grace period*. Scaduto tale periodo di tempo viene considerato l'*hard limit*. Soltanto la modifica ai valori relativi ai limiti è considerata da `edquota`.

Impostando i limiti a 0 indica che non si deve effettuare nessuna gestione dei quota per l'utente considerato.

Ad esempio, il comando

```
# edquota -t
```

Visualizza le impostazioni relative al *grace period*, ovvero qualcosa di analogo a quanto riportato di seguito

```
Time units may be: days, hours, minutes, or seconds
Grace period before enforcing soft limits for users:
/dev/hda2: block grace period: 5 days, file grace period: 5 days
```

dove

block grace period è il *grace period* utilizzato per i blocchi;

file grace period è il *grace period* utilizzato per i file (inode).

Per attivare il quota automaticamente all'avvio del sistema, è possibile modificare il file `/etc/fstab`³⁴ inserendo le opportune opzioni nelle righe relative ai filesystem per i quali si desidera abilitare il quota.

Quindi si può attivare il quota con uno script avviato durante la procedura di boot del sistema. Lo script sarà analogo a quello riportato di seguito

```
# Check quota and then turn quota on.
```

```
if [ -x /usr/sbin/quotacheck ]
then
    echo "Checking quotas..."
    /usr/sbin/quotacheck -avug
    echo " Done."
fi
if [ -x /usr/sbin/quotaon ]
then
    echo "Turning on quota..."
    /usr/sbin/quotaon -avug
    echo " Done."
fi
```

Non è una cattiva idea far controllare periodicamente (magari una volta alla settimana) il quota, inserendo il comando `quotacheck` nel `crontab`³⁵.

Per avere un dettaglio sull'utilizzo dei vari filesystem per i quali è abilitato il quota, da parte dei vari utenti o gruppi, si può utilizzare il comando `repquota` (man page `repquota(8)`).

Comando: `repquota`

Path: `/usr/sbin/repquota`

SINTASSI

`repquota` [*option*] *filesystem*

DESCRIZIONE

option specifica la modalità di funzionamento `repquota`. Può assumere i seguenti valori

- a visualizza un report di utilizzo dei filesystem elencati in `/etc/mtab` per i quali è abilitato il quota e sono utilizzati;
- v visualizza un report di utilizzo dei filesystem elencati in `/etc/mtab` per i quali è abilitato il quota;
- t tronca la visualizzazione degli username e groupname a 9 caratteri.
- n non effettua la risoluzione dei nomi relativi agli UID e GID;
- s visualizza lo spazio utilizzato, il numero di inode utilizzati ed i limiti con unità di misura più appropriate rispetto a quelle di default;
- F *format_name* visualizza un report relativo ai file di configurazione del quota nel formato indicato da *format_name* (v. tab. 3.17);
- g visualizza un report relativo al quota dei gruppi;
- u visualizza un report relativo al quota degli utenti (default);

filesystem specifica il filesystem da considerare;

Ogni utente può ottenere le statistiche relative al suo utilizzo del quota con il comando `quota` (man page `quota(1)`).

Comando: `quota`

Path: `/usr/bin/quota`

³⁴v. sez. 3.9

³⁵v. cap. 9

SINTASSI

quota *[option]* *[user]*

DESCRIZIONE

option specifica la modalità di funzionamento **quota**. Può assumere i seguenti valori**-F** *format_name*visualizza le statistiche relative al quota secondo il formato specificato da *format_name* (v. tab. 3.17);**-g** visualizza le statistiche relative al quota per il gruppo specificato da *user*;**-u** visualizza le statistiche relative al quota per l'utente specificato da *user* (default);**-v** visualizza le statistiche relative al quota anche per i filesystem che non sono stati ancora utilizzati;**-s** visualizza lo spazio utilizzato, il numero di inode utilizzati ed i limiti con unità di misura più appropriate rispetto a quelle di default;**-q** visualizza le statistiche relative ai filesystem per i quali i limiti del quota sono già stati superati;*user* specifica lo username o il groupname di cui si desiderano visualizzare le statistiche relative al quota (solo il superuser può specificare un utente o gruppo qualunque). Se non specificato è considerato l'utente che ha impartito il comando (o il gruppo di cui questi fa parte);

Il comando **warnquota** (man page **warnquota(8)**) controlla l'utilizzo dei filesystem secondo le impostazioni del quota ed invia dei messaggi via e-mail agli utenti che abbiano raggiunto i limiti di utilizzo dello spazio per essi previsto.

Comando: **warnquota**Path: **/usr/sbin/warnquota**

SINTASSI

warnquota *[option]*

DESCRIZIONE

option specifica la modalità di funzionamento **warnquota**. Può assumere i seguenti valori**-F** *format_name*specifica il formato del quota: **vfsvold** (versione 1), **vfsv0** (versione 2), **rpc** (quota su NFS) o **xfs** (quota su filesystem XFS);**-q** *filename*specifica il file contenente la descrizione dei filesystem (default **/etc/quotatab**);**-c** *filename*specifica il file contenente la configurazione di **warnquota** (default **/etc/warnquota.conf**);

3.18 RAID

Allo scopo di incrementare l'affidabilità della memoria di massa, che costituisce l'archivio delle informazioni del sistema, possono essere utilizzate varie tecniche che generalmente vanno sotto il nome di **RAID** (Redundant Array of Independent Disks)³⁶. Le tecniche si basano sull'utilizzo di due o più dischi in modo tale che se uno dei dispositivi si guasta, le informazioni sono comunque accessibili attraverso gli altri. Il sistema operativo "vede" un'unica unità logica composta da più unità fisiche.

Esistono vari livelli di RAID, illustrati di seguito, di cui i più utilizzati sono lo 0 il 3 ed il 5.

³⁶ad oggi anche Redundant Array of Inexpensive Disks visto il costo sempre decrescente dei dischi magnetici.

RAID 0 - Striping

I blocchi di informazioni sono suddivisi in parti più piccole (*stripes*) ed ognuna di esse è scritta (possibilmente) in un disco diverso (v. fig. 3.12). In questo modo le performance di lettura e scrittura delle informazioni vengono notevolmente incrementate rispetto all'utilizzo normale dei dischi, in quanto, se il canale lo permette, è possibile leggere/scrivere le informazioni contemporaneamente dai/sui dischi (le migliori prestazioni si ottengono infatti quando le informazioni sono memorizzate sui dischi ognuno di quali utilizza un proprio controller).

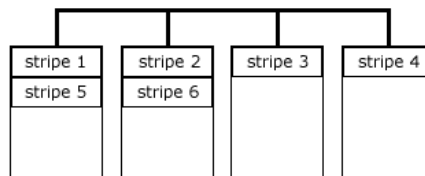


Figura 3.12: Schematizzazione del RAID 0.

Per questo tipo di tecnica sono richieste almeno 2 unità disco.

Per com'è realizzato, il guasto di un disco implica la perdita delle informazioni memorizzate sull'intero insieme di dischi.

È utile per macchine che hanno bisogno di performance elevate ma non di tipo mission critical.

RAID 1 - Mirroring and Duplexing

I blocchi di informazioni sono memorizzati su coppie di dischi³⁷ (v. fig. 3.13). Per ottenere le migliori performance è necessario che il controller sia in grado di effettuare una scrittura contemporanea su ogni coppia di dischi, altrimenti le performance in scrittura si dimezzerebbero rispetto a quelle di un singolo disco poiché si dovrebbero effettuare due scritture delle stesse informazioni sequenzialmente su due dischi.

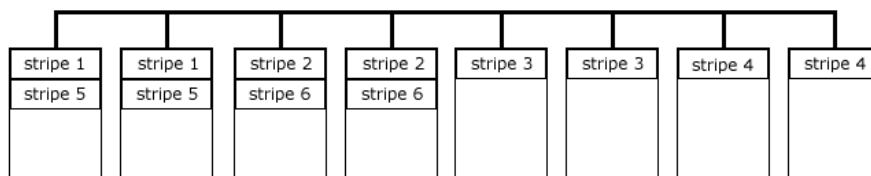


Figura 3.13: Schematizzazione del RAID 1.

Per questo tipo di tecnica sono richieste almeno 2 unità disco.

Il reperimento dell'informazione richiede due letture (una per disco) e pertanto il tempo di lettura risulta doppio rispetto a quello che si ha normalmente per un disco.

Il guasto di un disco non implica la perdita dei dati memorizzati sull'intero insieme di dischi (fault tolerance). In tal caso la fase di ripristino dell'insieme di dischi consiste nella copia delle informazioni contenute nel disco di mirror di quello guasto. Con più di due unità disco, se si guastano più dischi contemporaneamente è probabile che si riesca comunque a non perdere le informazioni memorizzate sull'insieme dei dischi (questo è vero finché non si guastano due dischi in mirror tra loro).

È utile per macchine che necessitano essenzialmente di un sistema fault tolerance ma non di elevate prestazioni.

³⁷le coppie di dischi devono avere la stessa capacità di memorizzazione.

RAID 2 - ECC

L'insieme dei dischi è suddiviso in due sottoinsiemi: uno destinato a contenere le informazioni (data disk) e l'altro a contenere i codici di controllo per la correzione degli errori calcolati a partire dalle informazioni stesse (ECC disk). Tale codice effettua il controllo sugli eventuali errori nella memorizzazione delle informazioni e permette la correzione dell'errore stesso on-the-fly. Le informazioni ed i codici di controllo sono memorizzati nei relativi dischi utilizzando la tecnica dello striping (RAID 0) (v. fig. 3.14).

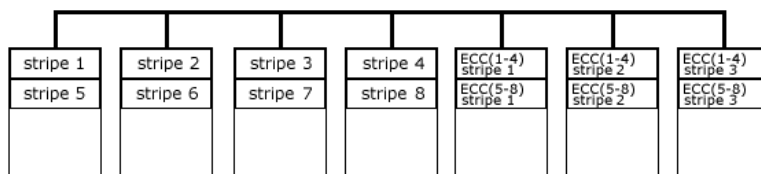


Figura 3.14: Schematizzazione del RAID 2.

La fase di scrittura ha l'overhead del calcolo del codice di controllo ECC.

Il guasto di un disco non implica la perdita delle informazioni memorizzate sull'insieme dei dischi.

RAID 3 - Parallel ECC

L'insieme dei dischi è suddiviso in due sottoinsiemi: uno destinato a contenere le informazioni (data disk) e l'altro a contenere i codici di controllo per la correzione degli errori calcolati a partire dalle informazioni stesse (ECC disk). Tale codice effettua il controllo sugli eventuali errori nella memorizzazione delle informazioni e permette la correzione dell'errore stesso on-the-fly. Le informazioni sono memorizzate nei data disk utilizzando la tecnica dello striping (RAID 0) (v. fig. 3.15).

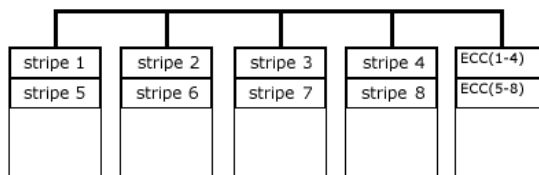


Figura 3.15: Schematizzazione del RAID 3.

Per questo tipo di tecnica sono richieste almeno 3 unità disco.

La fase di scrittura ha l'overhead del calcolo del codice di controllo ECC.

Il guasto di un disco non implica la perdita delle informazioni memorizzate sull'insieme dei dischi.

RAID 4

L'insieme dei dischi è suddiviso in due sottoinsiemi: uno destinato a contenere le informazioni (data disk) e l'altro a contenere i codici di controllo per la correzione degli errori calcolati a partire dalle informazioni stesse (ECC disk). Tale codice effettua il controllo sugli eventuali errori nella memorizzazione delle informazioni e permette la correzione dell'errore stesso on-the-fly. Ogni blocco di informazione è memorizzata in un disco diverso dei data disk (non si utilizza lo striping) (v. fig. 3.16).

Per questo tipo di tecnica sono richieste almeno 3 unità disco.

Il guasto di un disco non implica la perdita delle informazioni memorizzate sull'insieme dei dischi, anche se il ripristino delle informazioni è piuttosto complesso.

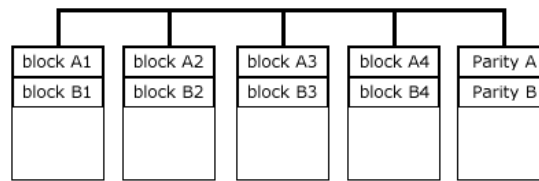


Figura 3.16: Schematizzazione del RAID 4.

RAID 5 - Distributed Parity

Ogni blocco di informazione è memorizzato in un disco diverso (no striping). I codici di controllo per la correzione degli errori sono memorizzati sugli stessi dischi in cui sono memorizzati anche le informazioni stesse, in maniera da distribuire al massimo sia le informazioni che i codici di controllo degli errori (v. fig. 3.17).

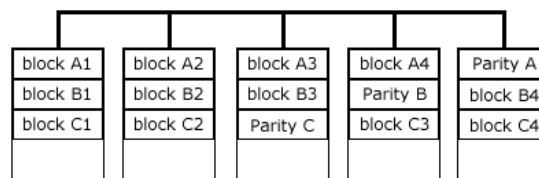


Figura 3.17: Schematizzazione del RAID 5.

Per questo tipo di tecnica sono richieste almeno 3 unità disco.

Il guasto di un disco non implica la perdita delle informazioni memorizzate sull'insieme dei dischi, anche se il ripristino delle informazioni è piuttosto complesso.

RAID 6

Si tratta essenzialmente di un'estensione del RAID 5 al quale viene aggiunto un secondo sistema per la generazione di codici di controllo per la correzione degli errori (two-dimensional parity). Le informazioni vengono memorizzate nello stesso modo del RAID 5 (v. fig. 3.18).

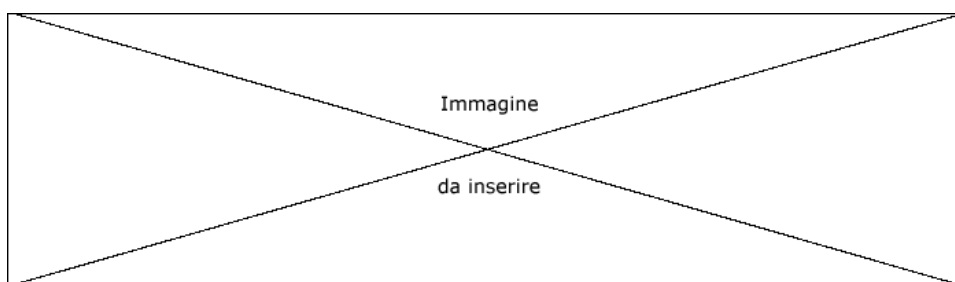


Figura 3.18: Schematizzazione del RAID 6.

Il guasto di uno o più dischi (contemporaneamente) non implica la perdita delle informazioni memorizzate sull'insieme dei dischi, anche se il ripristino delle informazioni è piuttosto complesso.

È ottimo per applicazioni di tipo mission critical.

RAID 10

Si tratta di un insieme di gruppi di dischi, su ognuno di quali viene applicata la

tecnica RAID 1. Sull'insieme dei gruppi viene invece applicata la tecnica RAID 0 (v. fig. 3.19).

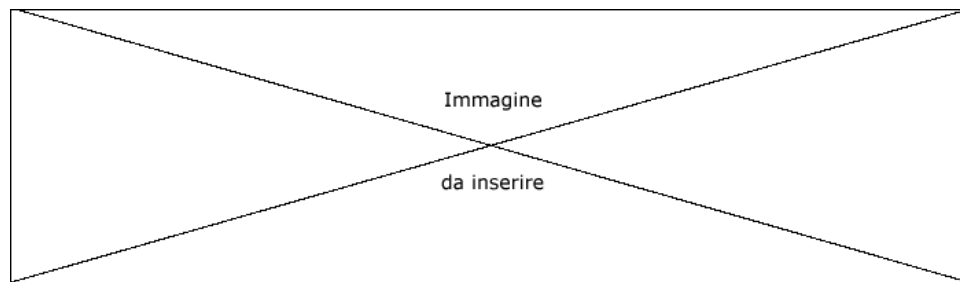


Figura 3.19: Schematizzazione del RAID 10.

Ha lo stesso tipo di fault tolerance del RAID 1 ed il suo stesso overhead.

Il RAID 0 (striping) fra i gruppi di dischi permette di aumentarne le prestazioni.

In particolari situazioni le informazioni memorizzate non sono perse anche nel caso di guasti su più dischi contemporaneamente.

RAID 53

Anche detto RAID 03 poiché si tratta di un insieme di gruppi di dischi, su ognuno di quali viene applicata la tecnica RAID 3. Sull'insieme dei gruppi viene invece applicata la tecnica RAID 0 (v. fig. 3.20).

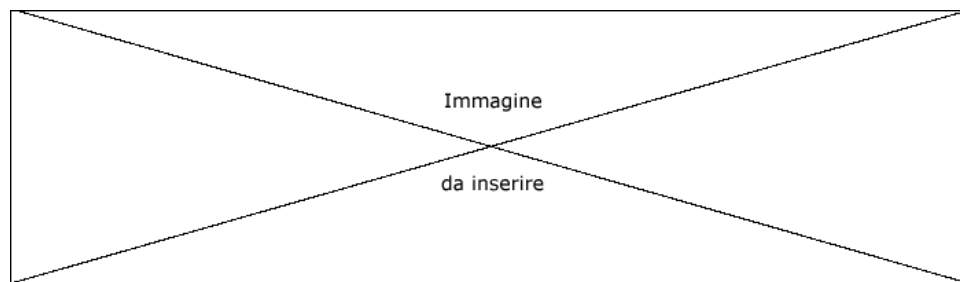


Figura 3.20: Schematizzazione del RAID 53.

Ha lo stesso tipo di fault tolerance del RAID 3 ed il suo stesso overhead.

Il RAID 0 (striping) fra i gruppi di dischi permette di aumentarne leggermente le prestazioni.

RAID 0+1

Si tratta di un insieme di gruppi di dischi, su ognuno dei quali viene applicata la tecnica RAID 0. Sull'insieme dei gruppi viene invece applicata la tecnica RAID 1 (v. fig. 3.21).

Ha lo stesso tipo di fault tolerance del RAID 5 e l'overhead del RAID 1 (mirroring).

È opportuno notare che la gestione RAID è indipendente dal filesystem utilizzato per la memorizzazione delle informazioni sulle partizioni dei dischi fisici che lo compongono. ???

Nei sistemi GNU/Linux, il RAID può essere di tipo hardware o software. Il RAID hardware affida la gestione della tecnica di memorizzazione ad uno specifico controller (hardware), che presenta al sistema operativo un unico disco composta dall'insieme dei dischi in RAID. Il RAID software è la gestione di più unità a dischi secondo la tecnica RAID, effettuata dal sistema operativo, senza la necessità di controller dedicati. Vista l'elevata velocità dei processori attuali, il RAID software è una valida alternativa

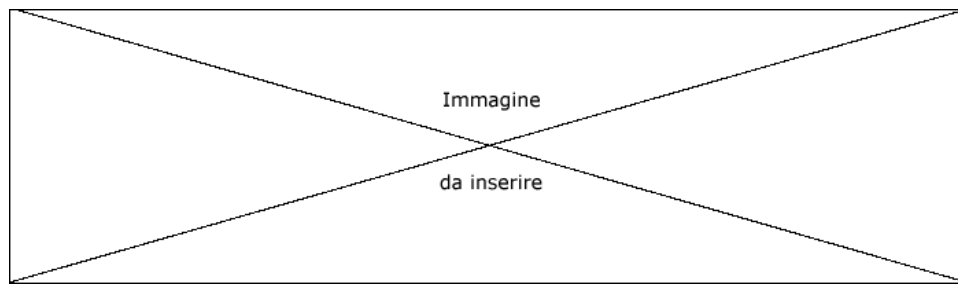


Figura 3.21: Schematizzazione del RAID 0+1.

economica ai sistemi RAID hardware. Il sistema operativo non conosce e non deve conoscere da cosa sia costituito il disco che vede, ma si limita a gestirlo come fosse un'unica unità a dischi, che nel caso di RAID software viene generalmente riferita come **md device** (*md* sta per **m**ultiple **d**evice).

md device

GNU/Linux è in grado di gestire a livello software il **RAID 0**, il **RAID 1**, il **RAID 4** ed il **RAID 5**. Inoltre, poiché le unità a disco gestite in RAID sono “viste” dal sistema come un'unità logica a blocchi (come un disco), è possibile utilizzare la tecnica RAID sui dischi già gestiti in RAID per ottenere le varie tecniche combinate come il RAID 10.

Oltre alle modalità RAID sopra elencate, è possibile gestire i dischi in una modalità detta **linear mode**, per mezzo della quale è possibile far “vedere” al sistema più dischi fisici come una sola unità logica, facendoglieli utilizzare in maniera sequenziale: le informazioni verranno scritte sul secondo disco quando il primo sarà stato riempito, poi sarà la volta del terzo, e così via. L'unico vantaggio è rappresentato dal fatto che se più utenti accedono alle informazioni presenti sui dischi gestiti in questo modo, si può avere un leggero incremento delle prestazioni nel caso in cui le informazioni accedute dagli utenti si trovino fisicamente su dischi diversi.

linear mode

GNU/Linux introduce anche il concetto di *spare disk*. Uno **spare disk** è un disco che viene associato all'unità RAID, ma non è destinato a contenere le informazioni. Questo viene usato soltanto nel caso in cui il sistema rileva che uno degli altri dischi non funziona più e quindi lo *spare disk* viene utilizzato automaticamente per ricostruire le informazioni presenti sul disco non più funzionante. In questo modo, il RAID software può ripristinare una situazione di rottura di un disco.

spare disk

Per riconoscere se il sistema è in grado di supportare la gestione del RAID software, è sufficiente controllare se esiste il file `/proc/mdstat`. Esso contiene le informazioni relative alla gestione dei dischi in RAID, come riportato di seguito.

```
Personalities :
read_ahead not set
unused devices: <none>
```

Esso riporta le **Personalities**, ovvero le modalità RAID correntemente utilizzate e le unità a disco attualmente non utilizzate.

La gestione dei dispositivi RAID è possibile in GNU/Linux tramite i pacchetti `raidtools` (`mkraid`, `raidstart`, `raidstop`, `lsraid`, ...) e `mdadm` (`mdadd`, `mdrun`, `mdstop`, ...). I comandi del pacchetto `raidtools`, presuppongono che le impostazioni relative alla gestione RAID siano contenute nel file `/etc/raidtab`. Ogni riga del file contiene una specifica direttiva, tra quelle possibili, riportate di seguito.

raiddev device

definisce una sezione relativa all'unità logica RAID *device*;

nr-raid-disks count

indica il numero dei dischi fisici gestiti in RAID, secondo quanto specificato da *count* (il numero massimo di dischi fisici è 12);

nr-spare-disks *count*

indica il numero degli spare disk assegnati al RAID, secondo quanto specificato da *count* (il numero massimo di spare disk è 12). Gli spare disk possono essere utilizzati soltanto con il RAID 4 ed il RAID 5;

persistent-superblock *value*

indica se l'unità logica RAID utilizza (1) o meno (0) un superblock persistente, cioè una piccola porzione dei vari dischi fisici che compongono il RAID che aumenta la riconoscibilità di tali dischi al kernel. Questa caratteristica è necessaria se i dischi in RAID devono essere riconosciuti automaticamente dal sistema in fase di boot;

Per poter gestire i dischi in RAID software al boot, è necessario che sui relativi hard disk vengano create partizioni di tipo *Linux RAID (FDH)*.

parity-algorithm *which*

indica il tipo di algoritmo per il calcolo della parità del RAID 5, secondo quanto specificato da *which* (*left-asymmetric*, *right-asymmetric*, *left-symmetric* e *right-symmetric* – *left-symmetric* è quello che offre le migliori prestazioni);

chunk-size *size*

imposta la dimensione della stripe a *size* KiB (*size* deve comunque essere una potenza di 2 ed al massimo 4 MiB). Valori tipici vanno da 4 (4 KiB) a 128 (128 KiB);

device *dev*

aggiunge il file di dispositivo *dev* alla gestione RAID relativa all'unità logica specificata dalla precedente direttiva *raiddev*. Tale direttiva deve essere necessariamente seguita da una delle direttive *raid-disk*, *spare-disk* o *parity-disk*;

raid-disk *index*

indica la posizione del disco fisico all'interno della gestione RAID;

spare-disk *index*

indica la posizione del disco fisico all'interno del gruppo di spare disk assegnati al RAID;

parity-disk *index*

indica la posizione del disco fisico all'interno del gruppo dei dischi in RAID utilizzati per la memorizzazione della parità;

failed-disk *index*

indica la posizione del disco fisico all'interno del gruppo di failed disk assegnati al RAID;

Di seguito è riportato un esempio del contenuto del file */etc/raidtab* in cui sono definite due unità a disco RAID, l'una (*/dev/md0*) di tipo RAID 0 e l'altra (*/dev/md1*) di tipo RAID 5. La */dev/md0* è formata da 2 dischi fisici, */dev/hda1* e */dev/hdb1*, mentre */dev/md1* è formata da 3 dischi fisici, */dev/sda1*, */dev/sdb1* e */dev/sdc1* oltre ad uno spare disk (*/dev/sdd1*).

```
raiddev /dev/md0
    raid-level          0
    nr-raid-disks       2
    persistent-superblock 0
    chunk-size          8
    device              /dev/hda1
    raid-disk           0
    device              /dev/hdb1
    raid-disk           1

raiddev /dev/md1
    raid-level          5
```



```

nr-raid-disks      3
nr-spare-disks     1
persistent-superblock 1
parity-algorithm   left-symmetric
device             /dev/sda1
raid-disk          0
device             /dev/sdb1
raid-disk          1
device             /dev/sdc1
raid-disk          2
device             /dev/sdd1
spare-disk         0

```

Il comando `mkraid` (man page `mkraid(8)`) inizializza un insieme di dischi RAID.

Comando: `mkraid`
 Path: `/sbin/mkraid`

SINTASSI

`mkraid` [*option*] [*device*]

DESCRIZIONE

option specifica la modalità di funzionamento `mkraid`. Può assumere i seguenti valori

```

-c filename | --configfile filename
    specifica il file (filename) da utilizzare come file di configurazione
    (default /etc/raidtab);
-f | --force
    forza l'inizializzazione dei dischi anche se essi contengono già delle
    informazioni;
-h | --help
    visualizza un aiuto sommario di mkraid;
-o | --upgrade
    aggiorna i dischi gestiti con le versioni precedenti del RAID software
    con quella corrente, senza perdita delle informazioni già presenti sui
    dischi;
-V | --version
    visualizza la versione di mkraid;

```

device specifica il file di dispositivo relativo all'insieme dei dischi gestiti in RAID (unità logica);

Ad esempio, il comando

```
# mkraid /dev/md0
```

inizializza il dispositivo `/dev/md0` ad essere gestito in RAID. L'effettivo riconoscimento del dispositivo può essere verificato visualizzando il contenuto del file `/proc/mdstat`. A questo punto tale dispositivo può essere montato, incluso nel file `/etc/fstab`, utilizzato, ...

Il comando `raidstart` (man page `raidstart(8)`) attiva la gestione del RAID software su di una unità.

Comando: `raidstart`
 Path: `/sbin/raidstart`

SINTASSI

`raidstart` [*option*] [*device*]

DESCRIZIONE

option specifica la modalità di funzionamento **raidstart**. Può assumere i seguenti valori

- a | --all
applica il comando a tutte le configurazioni specificate nel file di configurazione (/etc/raidtools);
- c *filename* | --configfile *filename*
specifica il file (*filename*) da utilizzare come file di configurazione (default /etc/raidtab);
- h | --help
visualizza un aiuto sommario di **raidstart**;
- V | --version
visualizza la versione di **raidstart**;

device specifica il file di dispositivo relativo all'insieme dei dischi gestiti in RAID (unità logica) da abilitare;

Ad esempio, il comando

```
# raidstart /dev/md1
```

attiva la gestione per l'unità RAID /dev/md1.

Il comando **raidstop** (man page **raidstop(8)**) disattiva la gestione del RAID software su di una unità.

Comando: **raidstop**
Path: /sbin/raidstop
SINTASSI
raidstop [*option*] [*device*]

DESCRIZIONE

option specifica la modalità di funzionamento **raidstop**. Può assumere i seguenti valori

- a | --all
applica il comando a tutte le configurazioni specificate nel file di configurazione (/etc/raidtools);
- c *filename* | --configfile *filename*
specifica il file (*filename*) da utilizzare come file di configurazione (default /etc/raidtab);
- h | --help
visualizza un aiuto sommario di **raidstop**;
- V | --version
visualizza la versione di **raidstop**;

device specifica il file di dispositivo relativo all'insieme dei dischi gestiti in RAID (unità logica) da disabilitare;

Ad esempio, il comando

```
# raidstart /dev/md2
```

disattiva la gestione per l'unità RAID /dev/md2.

Il comando **raid0run** (man page **raid0run(8)**) attiva la gestione del RAID 0 e linear mode su di una unità e senza superblock. Questo è un comando obsoleto.

Il comando **raidreconf** (man page **raidreconf(8)**) permette di riconfigurare i dischi gestiti in RAID seguendo le direttive contenute in un file di configurazione creato dall'utente con indicazioni analoghe a quelle contenute in /etc/raidtab.

Comando: **raidreconf**
Path: /sbin/raidreconf

SINTASSI

```
# raidreconf [option]
```

DESCRIZIONE

option specifica la modalità di funzionamento **raidreconf**. Può assumere i seguenti valori

- e *dev* | --export *dev*
specifica il dispositivo fisico (*dev*) da eliminare dalla gestione in RAID;
- i *dev* | --import *dev*
specifica il dispositivo fisico (*dev*) da aggiungere alla gestione in RAID;
- m *dev* | --mddev *dev*
specifica il dispositivo RAID (*dev*) da modificare;
- n *configfile* | --new *configfile*
specifica il nome del nuovo file di configurazione;
- o *configfile* | --old *configfile*
specifica il nome del vecchio file di configurazione (quello corrente);
- h | --help
visualizza un aiuto sommario di **raidreconf**;
- V | --version
visualizza la versione di **raidreconf**;

device specifica il file di dispositivo relativo all'insieme dei dischi gestiti in RAID (unità logica) da disabilitare;

Ad esempio, il comando

```
# raidreconf -o /etc/raidtab -n /etc/newraidtab -m /dev/md0
```

riconfigura il RAID **/dev/md0** configurato correntemente come specificato in **/etc/raidtab**, secondo quanto specificato in **/etc/newraidtab**, cercando di mantenere intatte le informazioni già memorizzate sul RAID.

Il comando **lsraid** (man page **lsraid(8)**) può essere utilizzato per ottenere informazioni sui dispositivi RAID attivi o meno.

Comando: **lsraid**

Path: **/sbin/lsraid**

SINTASSI

```
# lsraid [option] [device]
```

DESCRIZIONE

option specifica la modalità di funzionamento **raidreconf**. Può assumere i seguenti valori

- A Seleziona la modalità Array-based. Visualizza l'elenco dei RAID a cui appartengono i dischi fisici trovati;
- a *dev* considera il RAID *dev* nella ricerca delle informazioni;
- D Seleziona la modalità Disk-based. Visualizza l'elenco dei dischi fisici che fanno parte dei RAID trovati;
- d considera il disco fisico *dev* nella ricerca delle informazioni;
- f visualizza soltanto i dispositivi fisici sui quali sono stati rilevati dei problemi (nella modalità Array-based);
- g visualizza soltanto i dispositivi fisici sui quali non sono stati rilevati dei problemi (nella modalità Array-based);
- h | --help
visualizza un aiuto sommario di **lsraid**;
- l visualizza i superblock dei dischi fisici (nella modalità Disk-based).

- p ricerca i dischi fisici nel file `/proc/partitions` (non può essere utilizzata con le opzioni `-a` e `-d`).
- R visualizza le informazioni in un formato utilizzabile nel file `/etc/raidtab`.
- s visualizza soltanto gli spare disk (nella modalità Array-based);

Ad esempio, il comando

```
# lsraid -A -a /dev/md0
```

visualizza l'elenco dei dispositivi fisici relativi al RAID `/dev/md0`. Il comando

```
# lsraid -A -d /dev/sda1
```

visualizza il RAID a cui appartiene il disco fisico `/dev/sda1`. Il comando

```
# lsraid -A -f -a /dev/md0
```

visualizza i dispositivi fisici appartenenti al RAID `/dev/md0` sui quali è stato rilevato un problema. Il comando

```
# lsraid -D -l -a /dev/md0
```

visualizza il superblock di ognuno dei dischi fisici che appartengono al RAID `/dev/md0`.

Parallelamente al pacchetto `raidtools` può essere utilizzato il pacchetto `mdadm` che basa il suo funzionamento sul file di configurazione `/etc/mdtab` (corrispondente al file `/etc/raidtab` del pacchetto `raidtools`) che contiene righe con la seguente sintassi

```
md_dev mode,[c],[f],[crc32] dev1 dev2 ...
```

dove

md_dev

è il file di dispositivo corrispondente all'unità logica da gestire in RAID;

mode specifica la modalità RAID (`linear`, `raid0` o `raid1`);

c indica la dimensione della stripe. Può essere espresso con la sintassi *sizek* per indicare direttamente i KiB (non utilizzabile nella modalità `linear`);

f indica il numero massimo di fault (non utilizzabile nelle modalità `linear` e `raid0`);

crc32 è un codice di controllo aggiunto da `mdcreate`;

devn file di dispositivo relativi ai dischi fisici gestiti in RAID;

I comandi `mdadd` (man page `mdadd(8)`) e `mdcreate` (man page `mdcreate(8)`) permettono di configurare gli insiemi dei dischi da gestire in RAID inserendo le righe all'interno del file `/etc/mdtab`.

Comando: `mdadd`

Path: `/sbin/mdadd`

SINTASSI

```
# mdadd [option] [md_device] [real_device ...]
```

DESCRIZIONE

option specifica la modalità di funzionamento `mdadd`. Può assumere i seguenti valori

- a applica il comando all'elenco dei dispositivi contenuto nel file `/etc/mdtab`;
- r avvia il dispositivo md dopo che sono stati aggiunti i dispositivi reali;

-V visualizza la versione di **mdadd**;

md_device specifica il dispositivo virtuale formato dall'insieme di dischi;

real_device specifica un dispositivo fisico (un singolo disco);

??? esempio ???

Comando: **mdcreate**

Path: **/sbin/mdcreate**

SINTASSI

mdcreate [*option*] *personality md_device real_device ...*

DESCRIZIONE

option specifica la modalità di funzionamento **mdcreate**. Può assumere i seguenti valori

-cn imposta la dimensione delle stripes a $2^n \times \text{PAGE_SIZE}$. La dimensione può essere anche espressa in KiB aggiungendo in coda al valore *n* il simbolo 'k';

-fn imposta il massimo fault number ad *n*;

md_device specifica il dispositivo virtuale formato dall'insieme di dischi;

real_device specifica un dispositivo fisico (un singolo disco) da inserire nell'unità RAID;

??? esempio ???

Il comando **mdrun** (man page **mdrun(???)**) avvia la gestione delle unità RAID.

Comando: **mdrun**

Path: **??*/mdrun**

SINTASSI

mdrun [*option*] [*md_device*]

DESCRIZIONE

option specifica la modalità di funzionamento **mdrun**. Può assumere i seguenti valori

-a applica il comando all'elenco dei dispositivi contenuto nel file **/etc/mdtab**;

-px avvia il dispositivo nella modalità specificata da *x* (v. tab. 3.18);

x	Descrizione
1	Avvia il dispositivo in linear mode.
0	Avvia il dispositivo in RAID 0 mode.
1	Avvia il dispositivo in RAID 1 mode.

Tabella 3.18: Modalità di avvio dei dispositivi virtuali.

-cn imposta la dimensione del *chunk*;

-fn imposta il massimo livello di tolleranza agli errori a *n*;

-V visualizza la versione di **mdrun**;

md_device specifica il dispositivo virtuale formato dall'insieme di dischi;

??? esempio ???

Il comando **mdstop** (man page **mdstop(8)**) disabilita la gestione delle unità RAID.

Comando: **mdstop**

Path: **??*/mdstop**

SINTASSI

mdstop [*option*] [*md_device*]

DESCRIZIONE

option specifica la modalità di funzionamento **mdstop**. Può assumere i seguenti valori

- a applica il comando all'elenco dei dispositivi contenuto nel file **/etc/mdtab**;
- V visualizza la versione di **mdstop**;

md_device specifica il dispositivo virtuale dormato dall'insieme di dischi;

??? esempio ???

Il comando **mdop** (man page **mdop(8)**) abilita o disabilita un disco fisico alla gestione RAID 1.

Comando: **mdop**

Path: **???/mdop**

SINTASSI

mdop [*option*] [*md_device*] [*real_device*]

DESCRIZIONE

option specifica la modalità di funzionamento **mdop**. Può assumere i seguenti valori

- a applica il comando all'elenco dei dispositivi contenuto nel file **/etc/mdtab**;
- i toglie un dispositivo reale dall'appartenenza ad un dispositivo md (RAID 1);
- v aggiunge un dispositivo reale ad un dispositivo md (RAID 1);
- V visualizza la versione di **mdop**;

md_device specifica il dispositivo virtuale dormato dall'insieme di dischi;

real_device specifica un dispositivo fisico (un singolo disco);

???

3.19 LVM

Per GNU/Linux esiste anche la possibilità di gestire più dischi come se fossero un unico disco logico. Per questo è stato sviluppato il **LVM** (Logical Volume Manager) che fa parte del kernel di GNU/Linux a partire dalla versione 2.4.

Il LVM permette una gestione dei dischi a più alto livello rispetto alla tradizionale visione di dischi e partizioni, permettendo un'amministrazione molto flessibile della memoria di massa.

Una delle prime difficoltà nell'installazione di un sistema operativo come GNU/Linux è la decisione del partizionamento dei dischi, in quanto deve essere effettuata una stima a priori dell'utilizzo del filesystem, tanto che spesso si tende a realizzare un'unica partizione nella quale memorizzare tutte le informazioni. Infatti se si partiziona lo spazio del disco in modo da creare più filesystem (uno per ogni directory particolare come **/home**, **/usr**, ...), esaurito lo spazio per un particolare filesystem non è più possibile utilizzare la directory relativa (e tutte le sue sottodirectory) anche se il disco ha magari molto spazio libero nel suo complesso (in altre partizioni). Il problema è dovuto al fatto che la dimensione delle partizioni dei dischi viene impostata alla loro creazione e non può essere più modificata se non perdendo l'intero contenuto della partizione stessa.

Questo problema è risolto dal LVM poiché lo stesso permette di definire dei dischi logici (o virtuali) detti **volumi** le cui dimensioni possono essere modificate (aumentate o ridotte) in qualunque momento.

3.19.1 Concetti di base

Il LVM introduce i concetti di **Volume Group** (VG), **Physical Volume** (PV), **Logical Volume**, **Physical Extent** (PE) e **Logical Extent** (LE) come illustrato in fig. 3.22 e riportato di seguito.

LVM

volumi

Volume Group
Physical Volume
Logical Volume
Physical Extent
Logical Extent

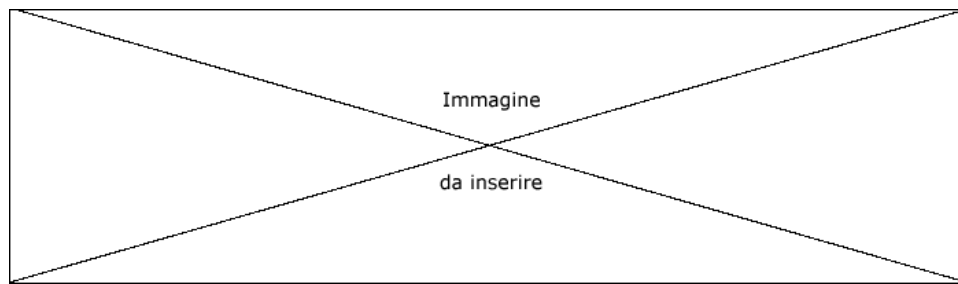


Figura 3.22: Struttura del LVM.

Physical Volume è un dispositivo di memoria di massa visibile dal sistema. Può essere una partizione di un disco, un vero e proprio disco o un insieme di dischi in RAID;

Logical Volume è un disco logico, ovvero un oggetto che il LVM rende visibile al sistema come se fosse un disco. Può essere paragonato ad una tradizionale partizione di un disco;

Volume Group è il concetto a livello di astrazione più elevato. È il collegamento tra i Physical Volume ed i Logical Volume;

Logical Extent ogni logical volume è suddiviso in sezioni (logical extent), la cui dimensione è la stessa per tutti i logical volume appartenenti allo stesso volume group;

Physical Extent ogni physical volume è suddiviso in sezioni (physical extent). la cui dimensione è la stessa di quella dei logical extent del volume group;

Per chiarire ulteriormente il significato dei concetti introdotti si consideri l'esempio seguente: si supponga di avere un volume group VG1 con una logical extent di 4 MiB. In VG1 sono state inserite le partizioni `/dev/hda1` e `/dev/hdb1` alle quali sono stati assegnati rispettivamente i nomi PV1 e PV2. Si supponga che PV1 e PV2 abbiano dimensioni diverse e tali per cui risultino composti rispettivamente da 99 e 248 physical extent (si ricordi che la dimensione di ogni physical extent è la stessa di quella del logical extent del volume group). Dunque è possibile creare un logical volume LV1 di dimensioni comprese tra 1 e 347 (99 + 248) physical extent, cioè tra 4 MiB e 1388 MiB. Alla creazione del logical volume, viene creata una corrispondenza tra i logical extent ed i physical extent, in modo tale che il logical extent 1 potrebbe essere corrispondente al physical extent 51 di PV1 cosicché le informazioni memorizzate nei primi 4 MiB del LV1 siano effettivamente memorizzate nel 51° physical extent di PV1.

È possibile scegliere il tipo di relazione con cui il LVM effettua la corrispondenza tra i logical extent ed i physical extent: **linear mapping** o **striped mapping**.

linear mapping
striped mapping

Linear mapping il LVM assegna ad un intervallo di logical extent un intervallo di physical extent fisicamente adiacenti;

Striped mapping il LVM assegna ad un intervallo di logical extent un intervallo di physical extent di ogni physical volume disponibile, in modo da cercare di incrementare le prestazioni del logical volume. È importante sottolineare il fatto che i logical volume creati con questa caratteristica non possono essere aumentati di dimensione oltre il numero totale dei physical volume sui quali sono stati inizialmente mappati;

Il LVM permette di creare degli snapshot (fotografie) ovvero di creare una copia di backup della situazione di un logical volume in un determinato istante, senza necessariamente dover le applicazioni che girano sul sistema.

Se il LVM è attivo, esistono vari file in `/proc/lvm` corrispondenti ai physical volume, logical volume e volume group, oltre al file `/proc/lvm/global` che riporta al suo interno un report sullo stato del LVM stesso.

3.19.2 Utilizzo

Il file di configurazione relativo ai dischi gestiti in LVM è `/etc/lvmtab`, che può essere modificato con appositi comandi, decritti di seguito.

Per poter utilizzare LVM, i dischi devono essere inizializzati per la creazione dei physical volume (PV), con il comando `pvcreeate` (man page `pvcreeate(8)`).³⁸

Comando: `pvcreeate`

Path: `/sbin/pvcreeate`

SINTASSI

`# pvcreeate [option] device [device ...]`

DESCRIZIONE

option specifica la modalità di funzionamento di `pvcreeate`. Può assumere i seguenti valori

`-d` | `--debug`

imposta la modalità di debug (più verbosa);

`-f` | `--force`

forza la creazione del PV senza nessuna conferma;

`-s size` | `--size size`

specifica la dimensione del PV (per default viene considerata la dimensione totale del *device* specificato);

`-y` | `--yes`

risponde automaticamente “sì” (yes) a tutte le domande;

`-h` | `--help`

visualizza un aiuto sommario di `pvcreeate`;

`-v` | `--verbose`

visualizza più messaggi informativi;

`-V` | `--version`

visualizza la versione di `pvcreeate`;

device specifica il dispositivo (disco o partizione) da inizializzare come physical volume (PV);

L'*exit status* di `pvcreeate` assume il significato riportato in tab. 3.19.

Valore	Significato
0	tutto ok
1	nessun physical volume specificato sulla riga di comando
2	errore durante la rimozione della riga di <code>lvmtab</code> relativa al physical volume
3	errore durante l'impostazione della struttura del physical volume
4	errore durante la scrittura della struttura del physical volume
5	identificatore del tipo di partizione errato
6	nome del physical volume errato
7	errore durante il reperimento della dimensione del physical volume
95	il driver/modulo non presente nel kernel
96	versione del protocollo di I/O non valida
97	errore di locking relativo al volume manager
98	file di configurazione non corretto
99	comando non corretto

Tabella 3.19: Significato dell'*exit status* di `pvcreeate`.

LVM funziona solo su partizioni di tipo `8EH`.

Ad esempio

```
# pvcreeate /dev/hdb
```

³⁸i dischi devono essere precedentemente partizionati.

crea un descrittore di volume group nel secondo disco ATA, mentre

```
# pvcreate /dev/hda2
```

crea un descrittore di volume group nella seconda partizione del primo disco ATA.

Quindi è necessario creare un volume group (VG) con il comando **vgcreate** (man page **vgcreate(8)**).

Comando: **vgcreate**

Path: **/sbin/vgcreate**

SINTASSI

```
# vgcreate [option] VGname PV [PV ...]
```

DESCRIZIONE

option specifica la modalità di funzionamento di **vgcreate**. Può assumere i seguenti valori

```
-A {y|n} | --autobackup {y|n}
```

specifica se effettuare automaticamente (y) o meno (n) il backup dei metadata di VG dopo che sono cambiati (y è il default);

```
-d | --debug
```

imposta la modalità di debug (più verbosa);

```
-h | --help
```

visualizza un aiuto sommario di **vgcreate**;

```
-l MaxLV | --maxlogicalvolumes MaxLV
```

imposta il numero massimo di logical volume (LV) per il volume group (VG) specificato (al massimo 256);

```
-p MaxPV | --maxphysicalvolumes MaxPV
```

imposta il numero massimo di physical volume (PV) per il volume group (VG) specificato (al massimo 256);

```
-s size | --physicalextentsize size
```

imposta la dimensione per le physical extent (PE). Può essere specificato un suffisso per indicare il moltiplicatore da considerare (se non specificato la dimensione indicata è considerata espressa in MiB);

```
-v | --verbose
```

visualizza più messaggi informativi;

```
--version
```

visualizza la versione di **vgcreate**;

VGname specifica il nome da associare al volume group (VG) da creare;

PV indica il physical volume (PV) che costituiranno il VG;

??? exit code **vgcreate** ???

Ad esempio

```
# vgcreate my_volume_group /dev/hda1 /dev/hdb1
```

crea un VG col nome **my_volume_group** relativo ai physical volume creati precedentemente nelle partizioni **/dev/hda1** e **/dev/hdb1**.

Un volume group può essere eliminato con il comando **vgremove** (man page **vgremove(8)**).

Comando: **vgremove**

Path: **/sbin/vgremove**

SINTASSI

```
# vgremove [option] VGname [VGname ...]
```

DESCRIZIONE

option specifica la modalità di funzionamento di **vgremove**. Può assumere i seguenti valori

```

-d | --debug
    visualizza dei messaggi utili per il debug;
-h | --help
    visualizza un aiuto sommario di vgremove;
-v | --verbose
    visualizza dei messaggi di informazioni;

```

VGname specifica il nome del volume group (VG) da eliminare;

??? exit status **vgremove** ???

Le informazioni relative alla gestione di un VG da parte del LVM possono essere visualizzate con il comando **vgdisplay** (man page **vgdisplay(8)**).

??? man page **vgdisplay** ???

??? Esempio ???

È possibile aggiungere ad un VG uno o più physical volume, precedentemente inizializzati con **pvccreate**, con il comando **vgextend** (man page **vgextend(8)**).

??? man page **vgextend** (/sbin/**vgextend**) ???

Ad esempio

```
# vgextend my_volume_group /dev/hdc1
```

aggiunge il PV **/dev/hdc1** al VG **my_volume_group**.

Un physical volume può essere rimosso dal VG con il comando **vgreduce** (man page **vgreduce(8)**).

??? man page **vgreduce** (/sbin/**vgreduce**) ???

Ad esempio

```
# vgreduce my_volume_group /dev/hda1
```

rimuove il PV **/dev/hda1** dal VG **my_volume_group**.

Le informazioni relative alla gestione di un PV da parte del LVM possono essere visualizzate con il comando **pvddisplay** (man page **pvddisplay(8)**).

??? man page **pvddisplay** (/sbin/**pvddisplay**) ???

Ad esempio

```
# pvddisplay /dev/hda1
```

visualizza le informazioni relative al physical volume **/dev/hda1**, che potrebbero essere qualcosa di analogo a quanto riportato di seguito

```

--- Physical volume ---
PV Name           /dev/hda1
VG Name           myvg
PV Size           1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV#               1
PV Status          available
Allocatable        yes (but full)
Cur LV           1
PE Size (KByte)    4096
Total PE           499
Free PE            0
Allocated PE       499
PV UUID            Sd44tK-9IRw-SrMC-M0kn-76iP-iftz-OVSen7

```

Si può creare quindi un logical volume con il comando **lvcreate** (man page **lvcreate(8)**).

??? man page **lvcreate** (/sbin/**lvcreate**) ???

Ad esempio

```
# lvcreate -L1500 -ntestlv testvg
```

crea un LV con nome “testlv” della dimensione di 1500 MiB utilizzando il linear mapping, a cui è associato il dispositivo rappresentato dal file `/dev/testvg/testlv`.

Un logical volume può essere eliminato (soltanto se è stato precedentemente “smontato” - v. sez. 3.9) con il comando `lvremove` (man page `lvremove(8)`).

??? man page `lvremove` ???

Ad esempio

```
# lvremove /dev/myvg/homevol
```

elimina, dietro conferma, il LV relativo al dispositivo `/dev/myvg/homevol`.

La dimensione di un LV può essere variata con il comando `lvextend` (man page `lvextend(8)`).

??? man page `lvextend` (/sbin/`lvextend`) ???

Ad esempio

```
# lvextend -L12G /dev/myvg/homevol
```

ridimensiona lo spazio del LV associato al dispositivo `/dev/myvg/homevol` fino al raggiungimento di un totale di 12 GiB, mentre

```
# lvextend -L+1G /dev/myvg/homevol
```

estende la dimensione del LV associato al dispositivo `/dev/myvg/homevol` di 1 GiB.

Una volta esteso il logical volume è necessario ampliare anche la dimensione del filesystem. Questa operazione dipende dal tipo di filesystem utilizzato. Ad esempio per il filesystem `ext2` (o `ext3`) esiste il comando `e2fsadm` (man page `e2fsadm(8)`).

??? man page `e2fsadm` (/sbin/`e2fsadm`) ???

Ad esempio

```
# e2fsadm -L+1G /dev/myvg/homevol
```

è analogo alla sequenza di comandi

```
# lvextend -L+1G /dev/myvg/homevol
```

```
# resize2fs /dev/myvg/homevol
```

Prima di ridurre le dimensioni di un LV è importante effettuare un ridimensionamento del filesystem, altrimenti si rischia di perdere le informazioni in esso contenute.

Ad esempio

```
# e2fsadm -L-1G /dev/myvg/homevol
```

riduce la dimensione del LV associato al dispositivo `/dev/myvg/homevol` di 1 GiB.

L’avvio e l’arresto di LVM può essere eseguito con i comandi `vgscan` (man page `vgscan(8)`) e `vgchange` (man page `vgchange(8)`).

??? man page `vgscan` ??? ??? /sbin/`vgscan`

??? man page `vgchange` ??? ??? /sbin/`vgchange`

Avvio

```
# vgscan # vgchange -ay
```

Arresto

```
# vgchange -an
```

3.20 RAM disk

Parte della memoria centrale (RAM) può essere utilizzata per la memorizzazione di file e directory, come se fosse parte della memoria di massa, con la differenza che i tempi di accesso alle informazioni è notevolmente inferiore rispetto a quello relativo alla memoria di massa e che il suo contenuto viene perduto con lo spegnimento della macchina. La gestione di tale porzione di memoria va sotto il nome di *RAM disk*.

La gestione del RAM disk può essere effettuato secondo due metodologie diverse. La prima (la più datata) utilizza i file di dispositivo `/dev/ram*` come fossero dei dispositivi esterni. I comandi seguenti

```
# mkfs -t ext2 /dev/ram1
# mount /dev/ram1 /mnt/ramdisk
```

formattano il dispositivo `/dev/ram1` (prima partizione del RAM disk) secondo il filesystem ext2 e lo montano nel mount point `/mnt/ramdisk` (che ovviamente deve esistere come directory).

La seconda metodologia, più recente, utilizza un filesystem particolare, il *ramfs*. Il comando seguente

```
# mount -t ramfs none /mnt/ramdisk
```

monta il ramdisk nel mount point `/mnt/ramdisk`. L'utilizzo del filesystem ramfs permette di sottrarre alla RAM la parte effettivamente richiesta per la memorizzazione dei file, in maniera automatica. Il filesystem ramfs permette le seguenti opzioni di montaggio

```
maxsize=size
    indica la massima dimensione del filesystem (del ramdisk) in KiB (default 0,
    nessun limite, fino all'occupazione di tutta la RAM);

maxfilesize=size
    indica la massima dimensione di un file in KiB (default 0, nessun limite);

maxdentries=num
    indica il massimo numero di directory (default maxsize/4 - 0 indica nessun
    limite);

maxinodes=num
    indica il massimo numero di inode (default 0, nessun limite). Questo valore non
    può superare maxdentries;
```

3.21 Copia e ripristino di partizioni

Per effettuare una copia di un'intera partizione per poterla successivamente ripristinare, esistono degli appositi software che non accedono alle informazioni presenti sulla memoria di massa secondo la struttura logica del filesystem, ma lo fanno a livello di blocchi. In questo modo il contenuto di una partizione viene copiato mantenendo addirittura la posizione di singoli blocchi in cui sono memorizzate le informazioni. Si parla infatti di *immagine* della partizione.

Si supponga infatti di avere un hard disk in cui vi siano delle partizioni (sicuramente almeno una ci sarà) e si desidera effettuare una copia di una o più partizioni per poterla ripristinare in un secondo momento oppure per crearne una copia su un altro hard disk. Con i software sopra menzionati è possibile creare un file per ogni partizione contenente l'immagine della partizione, ovvero le informazioni memorizzate nella partizione stessa. Il file contenente l'immagine della partizione può essere successivamente ripristinato in una partizione dello stesso o di un altro hard disk (v. fig. 3.23).

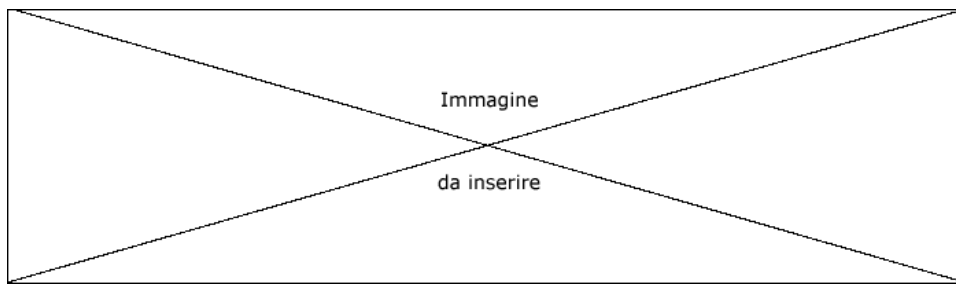


Figura 3.23: Gestione delle partizioni.

3.21.1 Partition Image

*Partition Image*³⁹ è un'applicazione che permette la creazione di immagini di partizioni ed il ripristino delle stesse. Le partizioni delle quali deve essere creata l'immagine non devono essere montate, pertanto l'applicazione mette a disposizione due dischetti di avvio con i quali effettuare il boot del sistema. Questi permettono di avviare un sistema operativo GNU/Linux minimale che permette di operare su un RAM disk, senza montare nessuna partizione dell'hard disk. In questo modo è possibile creare l'immagine della partizione desiderata, salvandola magari in un file all'interno di un'altra partizione (la partizione sulla quale avviene il salvataggio dell'immagine della partizione deve essere comunque montata).

L'applicazione viene lanciata con il comando **partimage**.

Comando: **partimage**

Path: **/sbin/partimage**

SINTASSI

partimage [option] [action [device] filename]

DESCRIZIONE

option indica la modalità di funzionamento di **partimage**. Può assumere i seguenti valori:

-zn | --compress[=n]
indica il livello di compressione relativo al file che conterrà l'immagine della partizione, dove **n** può assumere i valori riportati in tab. 3.20;

n	Significato
0	nessuna compressione (molto veloce ma file immagini molto grandi)
1	compressione con gzip (veloce e file immagini piccoli) (default)
2	compressione con bzip2 (piuttosto lento ma file immagini molto piccoli)

Tabella 3.20: Livello di compressione.

-c | --nocheck
indica di non effettuare il controllo sulla partizione prima di creare l'immagine;

-o | --overwrite
indica di sovrascrivere l'immagine esistente senza conferma;

-d | --nodesc
indica di non richiedere una descrizione per l'immagine della partizione;

-Vn | --volume[=n]
indica di suddividere l'immagine su più file, dove **n** indica la dimensione massima dei file in kB;

-w | --waitvol
indica di attendere una conferma ad ogni cambio di file;

³⁹v. <http://www.partimage.org>.

-e | **--erase**
 indica di eliminare i blocchi vuoti durante il ripristino;
-m | **--allowmnt**
 indica di procedere anche se la partizione è montata (pericoloso);
-M | **--nombr**
 indica di non creare una copia del MBR nel file contenente l'immagine della partizione;
-h | **--help**
 visualizza un aiuto sommario di **partimage**;
-v | **--version**
 visualizza la versione di **partimage**;
-i | **--compilinfo**
 visualizza le opzioni utilizzate nella compilazione di **partimage**;
-fn | **--finish[=n]**
 indica l'azione da intraprendere se l'operazione ha avuto esito positivo, dove *n* può assumere i valori riportati in tab. 3.21;

n	Significato
0	wait (non intraprendere nessuna azione)
1	halt (arresta il sistema)
2	reboot (riavvia il sistema)
3	quit

Tabella 3.21: Livello di compressione.

-b | **--batch**
 indica di eseguire **partimage** in modo non interattivo;
-y | **--nosync**
 indica di non sincronizzare il disco al termine dell'operazione (pericoloso);
-saddr | **--server[=addr]**
 indica l'indirizzo IP (*addr*) del server;
-pn | **--port[=n]**
 indica il numero di porta (*n*) per il collegamento al server;
-gn | **--debug[=n]**
 indica il livello di verbosità (*n*) del debug (default 1);
-n | **--nossl**
 indica di disabilitare SSL⁴⁰ nella comunicazione via rete;
-S | **--simulate**
 indica di simulare l'operazione (solo per il ripristino dell'immagine);
-aopt | **--automnt[=opt]**
 indica le opzioni (*opt*) per il mount;
-Uusr | **--username[=usr]**
 indica lo username con il quale effettuare l'autenticazione al server;
-Ppass | **--Password[=pass]**
 indica la password per l'autenticazione al server;

action indica l'operazione da compiere. Può assumere i seguenti valori:

save salva le informazioni contenute nella partizione (immagine della partizione) nel *filename*;
restore
 ripristina la partizione con l'immagine salvata nel file *filename*;
restmbr
 ripristina il MBR con quello contenuto nell'immagine salvata nel file *filename*;
imginfo
 visualizza le informazioni relative all'immagine della partizione salvata nel file *filename*;

device è il file di dispositivo relativo alla partizione di cui creare l'immagine o da ripristinare;

⁴⁰v. `chapters:???`

filename è il nome del file che contiene o dovrà contenere l'immagine della partizione;

L'interfaccia visualizzata è analoga a quella riportata in fig. 3.24, nella quale si può selezionare la partizione di cui creare l'immagine e il percorso nel quale salvare il file relativo. È anche possibile scegliere la partizione da ripristinare ed il file contenente l'immagine con la quale ripristinare la partizione.

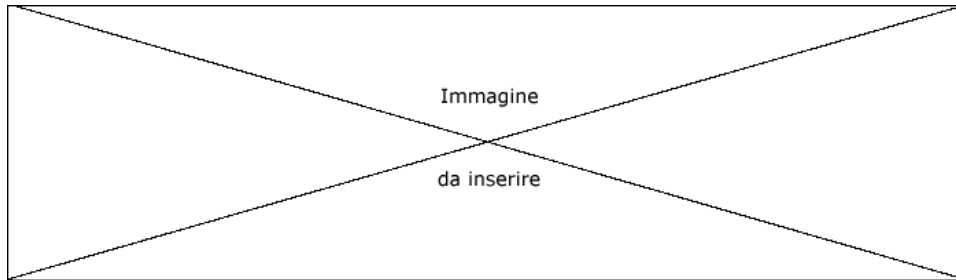


Figura 3.24: Esempio di Partition Image.

Partition Image permette anche di salvare/leggere i file relativi alle immagini delle partizioni all'interno di un altro computer che funge da server. A tale scopo, sul computer server deve essere in esecuzione *partimaged* che permette la comunicazione per default attraverso la porta 4025. In questo modo è possibile gestire le immagini delle partizioni in maniera centralizzata su un'unica macchina in rete.

Comando: *partimaged*
Path: */sbin/partimaged*

SINTASSI

partimaged [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di *partimaged*. Può assumere i seguenti valori:

```
-h | --help
    visualizza un aiuto sommario di partimaged;
-v | --version
    visualizza la versione di partimaged;
-pn | --port[=n]
    indica il numero di porta (n) sulla quale partimaged deve rimanere
    in ascolto;
-gn | --debug[=n]
    indica il livello di verbosità (n) del debug (default 1);
-rdir | --chroot[=dir]
    indica chroot per aumentarne la sicurezza;
-D | --daemon
    avvio in modalità daemon (in background);
-L | --noloin
    disabilita il login dai client;
```

È una buona idea eseguire un controllo del filesystem relativo alle partizioni delle quali creare le immagini. Ciò può essere fatto con il comando *e2fsck*. Ad esempio

```
# e2fsck -f /dev/hda1
```

controlla il filesystem relativo alla prima partizione del primo disco ATA (*/dev/hda1*). Eventualmente si può effettuare anche un backup delle partition table del disco di destinazione. Ad esempio

```
# dd if=/dev/hda of=backup-hda.mbr count=1 bs=512
```

salva il MBR del disco (primi 512 byte del primo disco ATA) nel file `backup-hda.mbr` e

```
# sfdisk -d /dev/hda > backup-hda.sf
```

salva la partition table relativa alle partizioni estese contenute nel disco (primo disco ATA) nel file `backup-hda.sf`.

All'occorrenza, tali informazioni possono essere ripristinate rispettivamente con i seguenti comandi

```
# dd if=backup-hda.mbr of=/dev/hda
```

```
# sfdisk /dev/hda < backup-hda.sf
```

Il MBR non viene di norma influenzato dalle operazioni di creazione delle immagini e dal ripristino delle stesse, ma *Partition Image* permette di ripristinare il MBR dal file contenente l'immagine di una partizione o addirittura soltanto il boot loader (al posto dell'intero MBR). Se necessario si può comunque ripristinare il boot loader dopo aver ripristinato le immagini delle partizioni, avviando il sistema da dischetto e quindi seguendo le istruzioni del boot loader desiderato.

È opportuno osservare che il disco sul quale devono essere ripristinate le immagini delle partizioni precedentemente create, deve essere preventivamente partizionato (v. sez. 3.2).

3.22 Comandi utili

In questa sezione saranno elencati alcuni comandi utili per la gestione del filesystem. Il comando `df` (`disk free` - man page `df(1)`) ad esempio visualizza la quantità di memoria di massa ancora disponibile sul sistema.

Comando: `df`

Path: `/bin/df`

SINTASSI

```
$ df [option] [file]
```

DESCRIZIONE

option specifica la modalità di funzionamento **quota**. Può assumere i seguenti valori

```
-a | --all
```

include anche i filesystem costituiti da 0 blocchi;

```
-B size | --block-size=size
```

indica di utilizzare la dimensione dei blocchi del filesystem specificata da *size*, secondo la sintassi

```
[n]udm
```

dove

n indica un valore numerico (se omissso viene considerato il valore 1);

udm indica l'unità di misura come riportato in tab. 3.22;

Size	Descrizione
kB	1.000 byte
K	1.024 byte
MB	1.000 kB
M	1.024 K

Tabella 3.22: Unità di misura specificabili per le dimensioni dei blocchi.


```

-h | --human-readable
    visualizza le dimensioni in un formato più leggibile, come 1K, 234M,
    2G (potenze di 2);
-H | --si
    visualizza le dimensioni secondo i multipli del sistema metrico deci-
    male (potenze di 10);
-i | --inodes
    visualizza le informazioni relative agli inode piuttosto che ai blocchi;
-k
    come --block-size=1K;
-l | --local
    visualizza le informazioni relative ai filesystem locali (non remoti);
--no-sync
    non lancia in esecuzione sync prima di visualizzare le informazioni
    (default);
-P | --portability
    utilizza il formato POSIX per l'output;
--sync lancia in esecuzione sync prima di visualizzare le informazioni;
-t fstype | --type=fstype
    visualizza le informazioni solo per i filesystem di tipo fstype;
-T | --print-type
    visualizza il tipo di filesystem;
-x fstype | --exclude-type=fstype
    visualizza le informazioni solo per i filesystem non di tipo fstype;
--help visualizza un aiuto sommario di df;
--version
    visualizza la versione di df;

```

file indica di visualizzare le informazioni relative ai filesystem che contengono il file indicato da *file* (può essere anche un elenco di file separati da uno spazio);

Il comando **du** (**disk usage** – man page **du(1)**) visualizza l'utilizzo del disco per ogni directory e file.

Comando: **du**

Path: **/usr/bin/du**

SINTASSI

\$ du [option] [filename]

DESCRIZIONE

option indica la modalità di funzionamento di **du**. Può assumere i seguenti valori

```

-a | --all
    visualizza il totale dei file (non solo delle directory);
--block-size=size
    imposta la dimensione dei blocchi a size byte;
-b | --bytes
    visualizza le dimensioni in byte;
-c | --total
    visualizza anche un totale complessivo;
-D | --dereference-args
    considera i file a cui si riferiscono i symbolic link presenti nella riga
    di comando;
-h | --human-readable
    visualizza le dimensioni secondo i multipli del sistema binario (K =
    kibibyte, M = mebibyte, G = gibibyte);
-H | --si
    visualizza le dimensioni secondo i multipli del sistema decimale (K =
    kilobyte, M = megabyte, G = gigabyte);
-k | --kilobytes
    come --block-size=1024;

```

```

-l | --count-links
    considera più volte la dimensione di un file se si tratta di un hard
    link;
-L | --dereference
    considera i file a cui si riferiscono i symbolic link;
-m | --megabytes
    come --block-size=1048576;
-S | --separate-dirs
    non include nello stesso totale la dimensione delle sottodirectory;
-s | --summarize
    visualizza solo un totale per ogni argomento;
-x | --one-file-system
    non considera directory su filesystem diversi;
-X path | --exclude-from=path
    non considera i file contenuti in qualsiasi path specificato da path
    (???);
--exclude=path
    non considera i file contenuti nel path specificato da path;
--max-depth=n
    la ricerca dei file viene effettuata per un massimo di n livelli di
    sottodirectory rispetto alla riga di comando;
--help visualizza un aiuto sommario di du;
--version
    visualizza la versione di du;
filename indica di visualizzare l'occupazione dei dischi dovuto al file o directory
    specificato;

```

Il comando **sync** (man page **sync(1)**) forza la scrittura delle informazioni presenti nella cache sui dischi.

```

Comando: sync
Path: /bin/sync

SINTASSI
$ sync [option]

```

DESCRIZIONE

option indica la modalità di funzionamento di **sync**. Può assumere i seguenti valori

```

--help visualizza un aiuto sommario di sync;
--version
    visualizza la versione di sync;

```

```

Comando: badblocks
Path: /sbin/badblocks

```

```

SINTASSI
# badblocks [option] device [last_block] [start_block]

```

DESCRIZIONE

option indica la modalità di funzionamento di **badblocks**. Può assumere i seguenti valori

```

-b block_size
    indica la dimensione dei blocchi (in byte) secondo quanto specificato
    da block_size;
--help visualizza un aiuto sommario di sync;
--version
    visualizza la versione di sync;

```

???

??? man page /sbin/dumpe2fs ???

???

3.23 Riferimenti

- B. Nguyen, *Linux Filesystem Hierarchy*
<http://tldp.org/LDP/Linux-Filesystem-Hierarchy/Linux-Filesystem-Hierarchy.html>
- Informazioni sul filesystem *ext2*
<http://e2fsprogs.sourceforge.net/ext2intro.html>
- Informazioni sul filesystem *ext3*
<http://www.redhat.com/support/wpapers/redhat/ext3/index.html>
<http://people.spoiled.org/jha/ext3-faq.html>
- “Linux Swap Space Mini-HOWTO”
<http://www.xenotime.net/linux/swap-mini-howto.txt>
- “Chrooting daemons and system processes HOW-TO”
<http://www.networkdweebs.com/chroot.html>
- Informazioni sul RAID
<http://linas.org/linux/raid.html>
<http://www.acnc.com/raid.html>
- “Software-RAID HOWTO”
<http://ostenfeld.dk/~jakob/Software-RAID.HOWTO>

Capitolo 4

Operazioni su file e directory

“Ciò che dobbiamo imparare a fare lo impariamo facendolo.”
– Aristotele

In questo capitolo saranno trattati i comandi relativi alla gestione dei file e directory e sarà riservata una sezione per la trattazione delle espressioni regolari, che sono utilizzate per la ricerca del testo contenuto all’interno di un file. ???

Quando si ha a che fare con un file, la prima operazione che un software compie è detta apertura del file (da cui la dizione “aprire un file”) poiché la funzione di sistema che permette l’accesso ai file è `open()`. Un file può essere aperto nella modalità di accesso desiderata: *lettura*, nel caso in cui si sia interessati ad accedere al contenuto del file senza alcuna modifica, *scrittura*, nel caso in cui si sia interessati a sovrascrivere il contenuto del file o ad aggiungere altre informazioni nello stesso, *lettura/scrittura*, nel caso in cui si voglia poter accedere ad informazioni già presenti nel file e poterle anche cambiare).

4.1 Creazione di file

Un file può essere creato in vari modi. Da programma si utilizzano apposite chiamate alle librerie di sistema (`open()`, `fopen()`) per aprire un file in scrittura, che, nel caso in cui il file non esista, provvedono a crearlo. Da shell è invece possibile creare un file con il comando `touch` (man page `touch(1)`).

Comando: `touch`
Path: `/bin/touch`
SINTASSI
`$ touch [option] [file]`

DESCRIZIONE

option è l’insieme delle opzioni che modificano il comportamento di `touch`. Può assumere i seguenti valori:

- `-a` modifica soltanto la data/ora dell’ultimo accesso (e non anche quella dell’ultima modifica) del file;
- `-c` | `--no-create` indica di non creare nessun file;
- `-d string` | `--date=string` valuta la stringa *string* e la utilizza al posto della data/ora corrente;
- `-m` modifica soltanto la data/ora dell’ultima modifica (e non anche quella dell’ultimo accesso) del file;
- `-r file` | `--reference=file` utilizza la data/ora del file *file*, invece della data/ora corrente per la creazione del file;

```

-t timestamp
    utilizza la data/ora espressa da timestamp nel formato [CCYY]MMDDhhmm[.ss]
    per la creazione del file;
--time=time
    utilizza la data/ora specificata time atime, mtime ???;
--help
    visualizza un aiuto sommario di touch;
--version
    visualizza la versione di touch;

file
    specifica il nome del file da creare;

```

???

La creazione di un file può essere effettuata anche per mezzo di una redirectione dell'output di un comando (v. cap. 7) oppure impartendo comandi che effettuano copie di file (v. sez. 4.3).

Un file di testo può essere creato aprendo il file con un text editor (v. sez. 4.7) quando il file in questione non esiste.

4.2 Creazione di directory

La creazione di una directory vuota può essere effettuata con il comando **mkdir** (man page **mkdir(1)**) (**make directory**).

```

Comando: mkdir
Path: /bin/mkdir

SINTASSI
$ mkdir [option] [dirname]

```

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di **mkdir**. Può assumere i seguenti valori:

```

-m mode | --mode=mode
    imposta i permessi della nuova directory (v. chmod);
-p | --parents
    non dà errore nel caso in cui la directory esista ???;
-v | --verbose
    visualizza un messaggio per ogni directory creata;
--help
    visualizza un aiuto sommario di mkdir;
--version
    visualizza la versione di mkdir;

```

dirname specifica il nome della directory da creare;

???

Le directory possono essere create anche con altri comandi che effettuano ad esempio la copia di altre directory (v. sez. 4.3);

4.3 Copia di file o directory

In GNU/Linux è possibile copiare i file (speciali o meno) e le directory con il comando **cp** (man page **cp(1)**) (**copy**).

```

Comando: cp
Path: /bin/cp

SINTASSI
$ cp [option] source dest

```

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di **cp**. Può assumere i seguenti valori:

- a | **--archive**
come **-dpR**;
- b | **--backup[=*control*]**
effettua un backup di ogni file di destinazione esistente;
- copy-contents**
copia il contenuto dei file speciali in caso di copia ricorsiva;
- P | **--no-dereference**
copia i symbolic link così come sono (non copia il file al quale essi si riferiscono);
- f | **--force**
cancella i file di destinazione esistenti prima di rimpiazzarli con i file da copiare;
- i | **--interactive**
chiede conferma prima di sovrascrivere file esistenti;
- H |
considera i symbolic link sulla linea di comando;
- l | **--link**
crea dei symbolic link anziché copiare i file;
- L | **--dereference**
segue sempre i symbolic link (considera i file ai quali essi si riferiscono);
- p |
come **--preserve=mode,ownership,timestamps**;
- preserve[=*attr_list*]**
mantiene gli attributi dei file o directory da copiare, specificati da *attr_list* (v. tab. 4.1). Per default vengono mantenuti gli attributi relativi a **mode,ownership,timestamps**;

Keyword	Significato
all	tutti gli attributi.
links	vengono copiati i link (non i file ai quali si riferiscono).
mode	i permessi di accesso.
ownership	l'utente ed il gruppo proprietari.
timestamps	le date/ore.

Tabella 4.1: Permessi mantenibili da **cp**.

- no-preserve[=*attr_list*]**
non mantiene gli attributi dei file o directory da copiare, specificati da *attr_list*;
- parents**
aggiunge il percorso specificato da *source* alla directory; ???
- R | -r | **--recursive**
copia i file e directory presenti in tutto il sottoalbero indicato da *source* (in maniera ricorsiva);
- remove-destination**
elimina ogni file esistente prima di rimpiazzarlo; ???
- reply={yes|no|query}**
specifica come gestire il caso in cui esista già un file o directory con lo stesso nome di quello specificato da *source*: **yes** indica di rimpiazzare il file esistente (il contenuto file esistente sarà perduto - comportamento di default), **no** indica di non rimpiazzare il file esistente (il file non verrà copiato affatto), **query** indica di richiedere all'utente se rimpiazzare o meno il file esistente;
- sparse=*when***
indica se copiare come sono o meno i file contenenti lunghe serie di zero (v. tab. 4.2) ;
???
- strip-trailing-slashes**
elimina tutti gli eventuali slash (/) presenti nella parte finale del nome dei file o directory da copiare;

when	Significato
auto	viene creato un file di tipo <i>sparse</i> solo se il sorgente è stato riconosciuto automaticamente come tale (default).
always	viene creato un file di tipo <i>sparse</i> solo se il sorgente è stato riconosciuto come tale.
never	.

Tabella 4.2: Copia dei file di tipo *sparse* da *cp*.

```

-s | --symbolic-link
    crea symbolic link anziché copiare i file;
-S | --suffix=suffix
    indica il suffisso (suffix) da utilizzare per i file di backup (per default
    è "~");
--target-directory=directory
    sposta i file indicati da source nella directory directory;
-u | --update
    copia soltanto i file specificati da source più nuovi rispetto a quelli
    di destinazione o non esistenti nella directory di destinazione;
-v | --verbose
    visualizza dei messaggi di informazione;
-x | --one-file-system
    effettua la copia rimanendo all'interno dello stesso filesystem;
--help visualizza un aiuto sommario di cp;
--version
    visualizza la versione di cp;

```

source specifica il nome del file o della directory da copiare (sorgente);

dest specifica il nome del file o della directory in cui devono essere copiate le informazioni (destinazione);

???

Quando un file viene copiato, a meno che non venga esplicitamente specificata l'opzione ??, viene effettuata una copia del file stesso, ovvero esisteranno sul filesystem due file che hanno lo stesso contenuto ma accessibili con path diversi.

4.4 Spostamento o rinominazione di file o directory

Un oggetto del filesystem può essere spostato o rinominato per mezzo del comando *mv* (man page *mv*(1)) ((*move*)). Lo spostamento di un file o una directory consiste nel modificare la directory di appartenenza dell'inode ad esso associato, in modo che il suo contenuto non sia più raggiungibile con il path attuale, ma con un nuovo path. La rinominazione consiste nel cambiamento del nome associato al file o alla directory: anche questa operazione fa sì che il contenuto del file o directory sia raggiungibile attraverso un nuovo path.

Comando: *mv*

Path: */bin/mv*

SINTASSI

\$ *mv* [*option*] *source dest*

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di *mv*. Può assumere i seguenti valori:

```

-b | --backup[=control]
    effettua un backup di ogni file di destinazione esistente;
-f | --force
    non chiede conferma di sovrascrittura dei file (come --reply=yes);

```



```

-i | --interactive
    richiede una conferma di sovrascrittura dei file esistenti (come --reply=query);
--reply={yes|no|query}
    specifica come gestire il caso in cui esista già un file o directory
    con lo stesso nome di quello specificato da source: yes indica di
    rimpiazzare il file esistente (il contenuto file esistente sarà perduto
    - comportamento di default), no indica di non rimpiazzare il file
    esistente (il file non verrà spostato/rinominato affatto), query indica
    di richiedere all'utente se rimpiazzare o meno il file esistente;
--strip-trailing-slashes
    elimina tutti gli eventuali slash (/) presenti nella parte finale del
    nome dei file o directory da spostare/rinominare;
-S | --suffix=suffix
    indica il suffisso (suffix) da utilizzare per i file di backup (per default
    è "~");
--target-directory=directory
    sposta i file indicati da source nella directory directory;
-u | --update
    sposta/rinomina soltanto i file specificati da source più nuovi rispetto
    a quelli di destinazione o non esistenti nella directory di destinazione;
-v | --verbose
    visualizza dei messaggi di informazione;
--help visualizza un aiuto sommario di mv;
--version
    visualizza la versione di mv;
source nome del file (o della directory) da spostare/rinominare;
dest nuovo nome del file (o della directory) o nome della directory in cui spostare
    il file indicato da source;

```

???

4.5 Cancellazione di file o directory

Un file può essere cancellato, cioè rimosso dal filesystem, con il comando **rm** (man page **rm(1)**) (**remove**). In questo modo, l'inode ad esso relativo viene reso nuovamente disponibile al filesystem che lo può utilizzare per memorizzarci un altro file.

Comando: **rm**
 Path: **/bin/rm**
 SINTASSI
\$ rm [option] file

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di **rm**. Può assumere i seguenti valori:

```

-d | --directory
    elimina la directory indicata da file anche se non è una directory
    vuota (solo per il superuser);
-f | --force
    non chiede conferma prima della rimozione del file o directory indi-
    cato da file e non dà errore nel caso in cui file si riferisca ad un file
    o directory inesistente;
-i | --interactive
    richiede una conferma di rimozione del file;
-r | --recursive
    elimina tutti i file e directory presenti all'interno della directory
    specificata da file;
-v | --verbose
    visualizza dei messaggi di informazione;

```

```

--help visualizza un aiuto sommario di rm;
--version
    visualizza la versione di rm;
source nome del file (o della directory) da spostare/rinominare;
dest nuovo nome del file (o della directory) o nome della directory in cui spostare
    il file indicato da source;

```

???

In certi casi è possibile recuperare il contenuto di un file cancellato con il comando **rm**, per cui se si desidera eliminare un file senza possibilità alcuna di recupero del suo contenuto, è necessario utilizzare il comando **shred** (man page **shred(1)**), che prima di eliminare il file, sovrascrive il suo contenuto.

```

Comando: shred
Path: /usr/bin/shred
SINTASSI
$ shred [option] file

```

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di **shred**. Può assumere i seguenti valori:

```

-f | --force
    cambia eventualmente (se possibile) i permessi al file nel caso in cui
    l'accesso in scrittura non sia consentito;
-n num | --iterations=num
    sovrascrive il file num volte anziché il numero di volte di default (25);
-s size | --size=size
    sovrascrive il file portandolo alla dimensione specificata da size (i
    suffissi k, M e G sono accettati);
-u | --remove
    rimuove il file dopo averlo sovrascritto;
-v | --verbose
    visualizza dei messaggi di informazione;
-x | --exact
    non arrotonda la dimensione del file fino al riempimento di tutto
    l'ultimo blocco;
-z | --zero
    esegue un'ulteriore sovrascrittura dei byte contenuti nel file con il
    valore 0;
-
    esegue lo shredding del file indicato sullo standard output (termina-
    le);
--help visualizza un aiuto sommario di shred;
--version
    visualizza la versione di shred;

```

file nome del file da sovrascrivere/cancellare;

Il funzionamento di **shred** si basa sull'assunzione che il sistema sovrascriva le informazioni direttamente sul supporto magnetico (disco). Questo potrebbe non essere effettuato dai sistemi/dischi di ultima generazione come i journaled filesystem, RAID, NFS, filesystem compressi, ...

???

4.6 Visualizzazione del contenuto di un file

Il contenuto di un file di testo può essere visualizzato sia con il comando **cat** (man page **cat(1)**) che per mezzo di appositi programmi, i **text viewer** (visualizzatori di testo), i quali suppongono che il file sia in formato testo, ovvero che non abbia alcun tipo di *formattazione*.

Comando: **cat**
 Path: **/bin/cat**

SINTASSI
\$ cat [option] [file]

DESCRIZIONE

option indica le opzioni di funzionamento di **cat**. Può assumere i seguenti valori:

- A** | **--show-all**
 come **-vET**;
- b** | **--number-nonblank**
 numera le righe non vuote visualizzate sullo schermo;
- e** come **-vE**;
- E** | **--show-ends**
 visualizza il carattere '\$' alla fine di ogni riga;
- n** | **--number**
 numera le righe visualizzate sullo schermo;
- s** | **--squeeze-blank**
 raggruppa le righe vuote consecutive in un'unica riga;
- t** come **-vT**;
- T** | **--show-tabs**
 visualizza il carattere TAB come '^I';
- v** | **--show-nonprinting**
 visualizza i caratteri non stampabili con la notazione ^ (tranne per i
 caratteri LF e TAB);
- help** visualizza un aiuto sommario di **cat**;
- version**
 visualizza la versione di **cat**;

file specifica il file da visualizzare;

Se non viene specificato nessun file o **file** è costituito dal carattere '-', viene considerato lo standard input.

Ad esempio, il seguente comando

```
$ cat ~/.bashrc
```

visualizza il contenuto del file **~/.bashrc**, cioè quello riportato di seguito.

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
```

Anche i comandi **head** (man page **head(???)**) e **tail** (man page **tail(???)**) permettono di visualizzare le prime o ultime *n* righe di un file.

??? man page head ???

??? man page tail ???

I text viewer più utilizzati sui sistemi Unix-like sono **more** e **less**. Questi permettono la visualizzazione di file in formato testo, e forniscono all'utente gli strumenti per poter scorrere il contenuto del file e soffermarsi sulla parte che gli interessa.

Comando: **more**
 Path: **/bin/more**

SINTASSI
\$ more [option] file

DESCRIZIONE

option indica le opzioni di funzionamento di **more**. Può assumere i seguenti valori:

- num* indica il numero di linee (*num*) dello schermo;
- d* visualizza dei messaggi di aiuto tipo ‘Premere spazio per continuare, q per uscire’ e ‘Premere h per le istruzioni’ anziché avvertire l’utente con un segnale acustico in caso di comando non riconosciuto;
- l* annulla il comportamento di default che fa fermare la visualizzazione al carattere ASCII FF (salto pagina);
- f* abilita il conteggio delle linee logiche anziché quelle relative allo schermo (le righe che contengono più di 80 caratteri non vengono considerate come righe separate);
- p* indica di non effettuare lo scroll del testo, ma di cancellare lo schermo e mostrare quindi nuovamente il contenuto del testo;
- c* indica di non effettuare lo scroll del testo, ma di riscrivere il contenuto dello schermo a partire sempre dalla prima riga;
- s* raggruppa le linee vuote consecutive in una sola riga vuota;
- u* annulla la sottolineatura del testo;
- +/*string* specifica la ricerca della stringa *string* all’interno del testo prima di visualizzarlo;
- +/*num* inizia la visualizzazione dalla riga indicata dal numero *num*;

file è il file (comprensivo eventualmente di path relativo o assoluto) da visualizzare.

Il comando **more** è dotato anche di comandi interattivi che vengono impartiti per mezzo dell’ultima riga dello schermo che non viene utilizzata per visualizzare il testo del file specificato. I comandi interattivi si basano su quelli di **vi** (v. più avanti) e sono riportati in tab. 4.3.

Comando	Descrizione
H ?	visualizza l’elenco dei comandi interattivi (help)
[n] Spacebar	visualizza le successive <i>n</i> righe del file (per default <i>n</i> è il numero di righe dello schermo)
[n] Z	visualizza le successive <i>n</i> righe del file ed <i>n</i> diventa il nuovo default (per default <i>n</i> è il numero di righe dello schermo)
[n] Return	visualizza le successive <i>n</i> righe del file ed <i>n</i> diventa il nuovo default (per default <i>n</i> è 1)
[n] D	visualizza le successive <i>n</i> righe del file ed <i>n</i> diventa il nuovo default (per default <i>n</i> è 11)
Q Shift Q	esce da more
[n] S	visualizza le successive <i>n</i> righe del file (per default <i>n</i> è 1)
[n] F	visualizza l’ <i>n</i> -esima schermata successiva del file (per default <i>n</i> è 1)
[n] B	visualizza l’ <i>n</i> -esima schermata precedente del file (per default <i>n</i> è 1)
,	visualizza la posizione a cui è iniziata l’ultima ricerca nel testo
=	visualizza il numero della riga corrente
[n] /pattern	ricerca l’ <i>n</i> -esima occorrenza dell’espressione regolare ¹ indicata da <i>pattern</i> (per default <i>n</i> è 1)
[n] N	ricerca la prossima <i>n</i> -esima occorrenza dell’espressione regolare indicata dal precedente comando /pattern (per default <i>n</i> è 1)
!cmd !:cmd	esegue il comando <i>cmd</i> in una subshell (shell lanciata come processo figlio)
V	lancia in esecuzione un <i>editor</i> (v. più avanti) alla linea corrente (l’editor da lanciare è specificato nella variabile di ambiente VISUAL , se non esiste viene lanciato quello specificato nella variabile di ambiente EDITOR e se anche questo non esiste viene lanciato vi)
: [n]n	visualizza il successivo <i>n</i> -esimo file (per default <i>n</i> è 1)
: [n]p	visualizza il precedente <i>n</i> -esimo file (per default <i>n</i> è 1)
:f	visualizza il nome del file e la linea corrente
.	ripete il comando precedente

Tabella 4.3: Comandi di **more**.

Ad esempio, il comando

```
$ more ~/.bashrc
```

visualizzerà una schermata analoga a quella seguente

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
```

```
BLACK="\[\033[0;30m\"
RED="\[\033[0;31m\"
GREEN="\[\033[0;32m\"
BROWN="\[\033[0;33m\"
BLUE="\[\033[0;34m\"
PURPLE="\[\033[0;35m\"
CYAN="\[\033[0;36m\"
LIGHTGRAY="\[\033[0;37m\"
```

```
DARKGRAY="\[\033[1;30m\"
LIGHTRED="\[\033[1;31m\"
LIGHTGREEN="\[\033[1;32m\"
YELLOW="\[\033[1;33m\"
--More--(62%)
```

in cui è visualizzato il contenuto del file `.bashrc` presente nella home directory di ogni utente.

Il file `.bashrc` qui presentato è stato opportunamente modificato per aumentarne il numero di righe in modo tale da oltrepassare il numero di righe dello schermo, altrimenti `more` terminerebbe subito dopo averlo visualizzato (e non apparirebbe nessuna indicazione della percentuale di visualizzazione del file).

Per uscire da `more` si può premere il tasto Q.

Il comando `less` (man page `less(1)`) nasce come contrapposizione a `more` e ne estende notevolmente le funzionalità.

```
Comando: less
Path: /usr/bin/less

SINTASSI
$ less [option] file
```

DESCRIZIONE

option indica le opzioni di funzionamento di `less`. Può assumere i seguenti valori:

- ? | **--help**
visualizza un aiuto sintetico del comando `less`;
- a | **--search-skip-screen**
fa in modo che le ricerche vengano effettuate dopo l'ultima riga visualizzata sullo schermo;
- bn | **--buffers=n**
specifica il numero di *buffer* che `less` deve utilizzare per visualizzare ogni file (i buffer sono della dimensione di 1 KB e per default vengono utilizzati 10 buffer per file);
- B | **--auto-buffers**
per default, quando i dati da visualizzare sono letti da una *pipe*², i buffer vengono allocati automaticamente (per quanto ce n'è bisogno).

²v. cap. 6

Questa opzione disabilita l'allocazione automatica in modo tale che vengano allocati soltanto quelli specificati dall'opzione `-b`.

- `-c` | `--clear-screen`
disabilita lo scroll delle righe, ovvero fa in modo che lo schermo venga riscritto a partire dalla prima linea quando la visualizzazione dei dati deve essere aggiornata;
- `-C` | `--CLEAR-SCREEN`
come `-c` con la differenza che lo schermo viene cancellato prima di essere riscritto;
- `-d` | `--dumb`
non fa mostrare il messaggio di errore nel caso in cui il terminale non possieda dei requisiti fondamentali per il funzionamento di `less`;
- `-e` | `--quit-at-eof`
indica a `less` di terminare la seconda volta che incontra la fine del file (EOF: end of file) (per default si può uscire da `less` soltanto premendo i tasto `Q`);
- `-E` | `--QUIT-AT-EOF`
indica a `less` di terminare la prima volta che incontra la fine del file (EOF: end of file);
- `-f` | `--force`
indica a `less` di aprire un file non di testo e non fa visualizzare il messaggio di avvertimento del fatto che il file che si sta per aprire non è un file di testo;
- `-F` | `--quit-if-one-screen`
indica a `less` di terminare se l'intero file può essere visualizzato in una riga dello schermo;
- `-g` | `--hilite-search`
per default `less` evidenzia (highlight) tutte le stringhe trovate nell'ultima ricerca effettuata. Con questa opzione viene evidenziato soltanto la stringa corrente;
- `-G` | `--HILITE-SEARCH`
indica a `less` di non evidenziare nessuna delle stringhe trovate nell'ultima ricerca effettuata;
- `-hn` | `--max-back-scroll=n`
specifica il numero massimo di linee di scroll all'indietro. Se di deve scrollare all'indietro più di `n` linee lo schermo viene scrollato in avanti;
- `-i` | `--ignore-case`
indica a `less` di non fare differenza tra maiuscole e minuscole nelle ricerche (questa direttiva viene ignorata se nel *pattern* di ricerca sono state specificate delle lettere maiuscole);
- `-I` | `--IGNORE-CASE`
come `-i` ma non fa differenza tra maiuscole e minuscole neanche se nel *pattern* ci sono delle lettere maiuscole;
- `-jn` | `--jump-target=n`
specifica una linea dello schermo dove deve essere posizionata la linea "bersaglio". La linea bersaglio indica che le operazioni di ricerca o salti alle linee sono riferite a partire da questa.
- `-Jn` | `--status-column`
visualizza una colonna di stato al bordo sinistro dello schermo (solo se utilizzata in contemporanea a `-w` o `-W`).
- `-kfilename` | `--lesskey-file=filename`
indica a `less` di leggere ed interpretare il file *filename* come se fosse un file di *lesskey*;
- `-Kcharset`
indica a `less` di utilizzare il set di caratteri specificato da *charset* invece di quello definito nelle variabili di ambiente `JLESSCHARSET` o `LESSCHARSET`;
- `-m` | `--long-prompt`
indica a `less` di utilizzare un prompt che indica la percentuale del file visualizzato (il prompt di default di `less` è il carattere `'`);
- `-M` | `--LONG-PROMPT`
come `-m` ma ancora più verboso;

```

-n | --line-numbers
    indica a less di non visualizzare i numeri di riga;
-N | --LINE-NUMBERS
    indica a less di visualizzare i numeri di riga all'inizio di ogni riga;
-o filename | --log-file=filename
    indica a less di copiare il suo input nel file filename (solo se l'input
    proviene da una pipe). Se filename esiste chiede conferma per la
    sovrascrittura;
-O filename | --LOG-FILE=filename
    come -o ma non chiede conferma;
-ppattern | --pattern=pattern
    indica a less di iniziare la visualizzazione a partire dalla prima
    occorrenza di pattern all'interno del file;
-Pprompt | --prompt=prompt
    imposta i vari prompt di less secondo il seguente schema:
    -Pstring
        imposta il prompt di default (short prompt) con la stringa
        string;
    -Pmstring
        imposta il medium prompt con la stringa string;
    -PMstring
        imposta il long prompt con la stringa string;
    -Phstring
        imposta il prompt della schermata di aiuto con la stringa
        string;
    -P=string
        imposta il messaggio visualizzato col comando =, con la stringa
        string;
-q | --quiet | --silent
    indica a less di operare in modo silenzioso (non viene emesso nessun
    segnale acustico a meno che non si digiti un comando errato);
-Q | --QUIET | --SILENT
    indica a less di operare in modo totalmente silenzioso (non viene
    emesso nessun segnale acustico);
-r | --raw-control-chars
    indica a less di visualizzare i caratteri di controllo (non visualizzabili)
    utilizzando la caret notation, per esempio il carattere ASCII 01H
    (ottenibile con i tasti Ctrl A) è visualizzato come ^A;
-R | --RAW-CONTROL-CHARS
    come -r ma tenta di tener conto della visualizzazione dei caratteri
    di controllo per quanto sia possibile;
-s | --squeeze-blank-lines
    indica a less di raggruppare le linee vuote consecutive in un'unica
    linea vuota;
-S | --chop-long-lines
    indica a less di tagliare le linee alla larghezza dello schermo, anziché
    visualizzare il resto di una linea su quella successiva;
-ttag | --tag=tag
    indica a less di "editare" il file (aprirlo in modifica) contenente il
    tag specificato. A tale scopo deve esserci un file tags nella working
    directory creato per mezzo del comando ctags (man page ctags(1));
-Ttagsfile | --tag-file=tagsfile
    indica a less di utilizzare il file tagsfile anziché il file tags;
-u | --underline-special
    indica a less di visualizzare i caratteri ASCII BS e CR;
-U | --UNDERLINE-SPECIAL
    indica a less di visualizzare i caratteri ASCII BS, TAB e CR;
-V | --version
    indica a less di visualizzare il numero di versione;
-w | --hilite-unread
    indica a less di evidenziare temporaneamente le nuova linea visualizzata
    sullo schermo per effetto di uno scroll in avanti. L'evidenziazione
    viene eliminata al successivo comando impartito dall'utente;

```

-W | **--HILITE-UNREAD**
 indica a **less** di evidenziare temporaneamente le nuove linee visualizzate sullo schermo.

-XXX indica a **less** di utilizzare i caratteri di marcatura per rappresentare i caratteri non riconosciuti (per default i caratteri non riconosciuti sono visualizzati col loro valore binario).

-xn | **--tabs=*n***
 indica a **less** di considerare il carattere TAB come *n* spazi (per default *n* è 8);

-X | **--no-init**
 disabilita l'invio delle stringhe di inizializzazione al terminale;

-yn | **--max-forw-scroll=*n***
 imposta il numero massimo di linee di scroll in avanti al valore *n*. Se si avanza per più di *n* righe, al posto dello scroll viene riscritto interamente l'intero schermo;

-zn | **--window=*n***
 imposta la dimensione dell'area "scrollabile" a *n* righe (per default l'area scrollabile è tutto lo schermo);

-Z dà la precedenza alla codifica giapponese SJIS rispetto alla UJIS (per default viene data precedenza alla codifica UJIS);

-"cc | **--quotes=*cc***
 cambia il carattere di *quoting* (delimitatori) dei nomi dei file³;

-~ | **--tilde**
 per default **less** visualizza le linee oltre la fine del file con il simbolo ~, ma con questa indicazione tali linee saranno semplicemente vuote;

-#n | **--shift=*n***
 specifica il numero di default delle posizioni di cui effettuare lo scroll laterale (nel caso in cui *n* sia 0, lo scroll laterale viene effettuato con la metà della larghezza dell'intero schermo);

-- specifica la fine delle opzioni sulla riga di comando: gli argomenti specificati di seguito vengono interpretati come nomi di file;

+string la stringa *string* è interpretata come comando interattivo per **less**;

file indica il file (eventualmente comprensivo di path) da visualizzare;

Come **more**, anche **less** è dotato di comandi interattivi che vengono impartiti per mezzo dell'ultima riga dello schermo che non viene utilizzata per visualizzare il testo del file specificato. I comandi interattivi si basano su quelli di **vi** (v. più avanti) e sono riportati in tab. 4.4.

Una volta lanciato **less**, viene visualizzato il contenuto del file specificato. L'ultima riga dello schermo viene utilizzata per impartire i comandi e l'utente può utilizzare su questa le combinazioni di tasti riportate in tab. 4.7.

Ad esempio, il comando

```
$ less ~/.bashrc
```

visualizzerà una schermata analoga a quella seguente

³v. cap. 7.

Comando	Descrizione
H	visualizza l'elenco dei comandi interattivi (help)
[n]{ Spacebar F }	scrolla in avanti <i>n</i> righe (per default <i>n</i> è il numero di righe dello schermo)
[n]Z	come Spacebar , ma se <i>n</i> è specificato diviene la nuova dimensione della finestra visibile
[n]{ Return E J }	visualizza le successive <i>n</i> righe del file (per default <i>n</i> è 1)
[n]D	scrolla in avanti <i>n</i> righe (per default <i>n</i> è la metà del numero di righe dello schermo). Se <i>n</i> è specificato, diventa il nuovo valore di default
[n]B	scrolla indietro <i>n</i> righe (per default <i>n</i> è il numero di righe della finestra)
[n]W	come B , ma se specificato <i>n</i> diventa la nuova dimensione della finestra
[n]{ Y K }	scrolla indietro <i>n</i> righe (per default <i>n</i> è 1)
[n]U	scrolla indietro <i>n</i> righe (per default <i>n</i> è la metà del numero di righe dello schermo). Se <i>n</i> è specificato, diventa il nuovo valore di default
[n]Esc) →	scrolla verso destra di <i>n</i> colonne (per default <i>n</i> è la metà del numero di colonne dello schermo)
[n]Esc (←	scrolla verso sinistra di <i>n</i> colonne (per default <i>n</i> è la metà del numero di colonne dello schermo)
R	aggiorna la visualizzazione dello schermo
Shift R	aggiorna la visualizzazione dello schermo scartando i dati presenti nel buffer
Shift F	scrolla in avanti riprovando a leggere il file anche se è stata raggiunta la fine
[n]G	visualizza la riga <i>n</i> (per default <i>n</i> è 1)
[n]Shift G	visualizza la riga <i>n</i> (per default <i>n</i> è l'ultima riga del file)
[n]{ P % }	visualizza la porzione del file che corrisponde alla percentuale <i>n</i> % (<i>n</i> deve essere compreso tra 0 e 100)
[n]{	se il carattere '{' appare nella prima riga visualizzata sullo schermo, visualizza la riga contenente la relativa '}'. Se sulla prima riga sono presenti più caratteri '{', viene considerato l' <i>n</i> -esimo
[n]}	se il carattere '}' appare nell'ultima riga visualizzata sullo schermo, visualizza la riga contenente la relativa '{'. Se sull'ultima riga sono presenti più caratteri '}', viene considerato l' <i>n</i> -esimo
[n](come { , ma applicato al simbolo '('
[n])	come } , ma applicato al simbolo ')'
[n][come { , ma applicato al simbolo '['
[n]]	come } , ma applicato al simbolo ']'
[n]Esc Ctrl F x y	come { , ma utilizza la coppia di caratteri <i>x</i> e <i>y</i> come delimitatore di inizio e fine
[n]Esc Ctrl B x y	come } , ma utilizza la coppia di caratteri <i>x</i> e <i>y</i> come delimitatore di inizio e fine
M x	marca la posizione corrente con la lettera minuscola <i>x</i>
' x	ritorna alla posizione precedentemente marcata con la lettera <i>x</i> (se <i>x</i> è ô o \$ viene visualizzato rispettivamente l'inizio o la fine del file)

Tabella 4.4: Comandi di less (I parte).

```
# .bashrc
```

```
# User specific aliases and functions
```

```
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
/home/daniele/.bashrc (END)
```

in cui è visualizzato il contenuto del file **.bashrc** (standard) presente nella home directory di ogni utente.

Per uscire da **less** si può premere il tasto **Q**.

Comando	Descrizione
<code>[n]/pattern</code>	ricerca nel file l' <i>n</i> -esima occorrenza del <i>pattern</i> (per default <i>n</i> è 1). La ricerca avviene dalla seconda riga visualizzata sullo schermo (in avanti) (v. tab. 4.6)
<code>[n]?pattern</code>	ricerca nel file l' <i>n</i> -esima occorrenza del <i>pattern</i> (per default <i>n</i> è 1). La ricerca avviene dalla riga precedente alla prima visualizzata sullo schermo (indietro) (v. tab. 4.6)
<code>[n]</code>	ripete la ricerca precedente trovando l' <i>n</i> -esima successiva occorrenza del <i>pattern</i>
<code>[n]</code>	ripete la ricerca precedente trovando l' <i>n</i> -esima precedente occorrenza del <i>pattern</i>
<code>[n]</code>	ripete la ricerca precedente trovando l' <i>n</i> -esima successiva occorrenza del <i>pattern</i> estendendo la ricerca all'elenco di tutti i file specificati nella linea di comando
<code>[n]</code>	ripete la ricerca precedente trovando l' <i>n</i> -esima precedente occorrenza del <i>pattern</i> estendendo la ricerca all'elenco di tutti i file specificati nella linea di comando
	annulla l'evidenziazione delle occorrenze del <i>pattern</i>

Tabella 4.5: Comandi di `less` (II parte).

Carattere	Significato
!	ricerca le righe che non contengono il <i>pattern</i> specificato di seguito
*	ricerca il <i>pattern</i> in tutti i file elencati nella linea di comando
@	ricerca il <i>pattern</i> a partire dalla prima riga del primo file (o ultima riga dell'ultimo file) dell'elenco fornito nella linea di comando
	evidenzia le stringhe che soddisfano il <i>pattern</i> , contenute nella parte del file visualizzata sullo schermo
	non interpreta le espressioni regolari (v. cap. ??)

Tabella 4.6: Caratteri particolari all'inizio del *pattern* di ricerca.

Combinazione	Significato
	muove il cursore di una posizione verso sinistra
	muove il cursore di una posizione verso destra
	muove il cursore di una parola verso sinistra
	muove il cursore di una parola verso destra
	muove il cursore all'inizio della riga
	muove il cursore alla fine della riga
	cancella il carattere alla sinistra del cursore
	cancella il carattere sulla posizione del cursore
	cancella la parola alla sinistra del cursore
	cancella la parola sulla posizione del cursore
	richiama il comando precedente
	richiama il comando successivo
	completa il nome del file alla sinistra del cursore (ciclando su tutti i nomi degli eventuali file che lo completano)
	come ma cicla in ordine inverso
	completa il nome del file alla sinistra del cursore (riportando sull riga di comando tutti i nomi dei file che eventualmente lo completano)
	cancella l'intera linea di comando

Tabella 4.7: Combinazioni dei tasti sulla linea di comando di `less`.

Per una spiegazione più dettagliata si consiglia comunque di consultare la relativa man page.

Un file non testo (binario) può essere visualizzato con i comandi `od` (man page `od(1)`) e `hexdump` (man page `hexdump(1)`) ed un altro comando utile per visualizzare i caratteri testuali contenuti in un file binario è `strings` (man page `strings(1)`).

Comando: `od`

Path: `???/od`

SINTASSI

\$ od [*option*] *file*

DESCRIZIONE

option indica le opzioni di funzionamento di **od**. Può assumere i seguenti valori:

-A ???;

???

Comando: **hexdump**

Path: **/usr/bin/hexdump**

SINTASSI

\$ **hexdump** [*option*] *file*

DESCRIZIONE

option indica le opzioni di funzionamento di **hexdump**. Può assumere i seguenti valori:

- b visualizza il valore dei byte contenuti nel file *file* utilizzando il sistema di numerazione ottale;
- c visualizza il valore dei byte contenuti nel file *file* utilizzando la codifica ASCII;
- d visualizza il valore dei byte contenuti nel file *file* utilizzando il sistema di numerazione decimale;
- e *format*
specifica il formato da utilizzare per la visualizzazione del contenuto del file secondo quanto indicato da *format*;
- f *format_file*
specifica il file (*format_file*) dal quale leggere il formato da utilizzare per la visualizzazione del contenuto del file *file*;
- n *length*
specifica il massimo numero di byte da visualizzare;
- o visualizza il valore dei byte contenuti nel file *file* a coppie, utilizzando il sistema di numerazione ottale;
- s *offset*
specifica di non visualizzare i primi *offset* byte del file *file*;
- v indica di visualizzare esplicitamente il valore di tutti i byte contenuti nel file *file* (altrimenti le righe che contengono gli stessi dati di quelle precedenti sono visualizzate con dei caratteri '*');
- x visualizza il valore dei byte contenuti nel file *file* a coppie, utilizzando il sistema di numerazione esadecimale;

???

Il contenuto dei file oggetto⁴ (il risultato della compilazione di un file) può essere visualizzato con i comandi **nm** (man page **nm(1)**) e **objdump** (man page **objdump(1)**).

4.7 Modifica del contenuto di un file

Esistono programmi, i **text editors** (editor di testo), che oltre a visualizzare il contenuto di un file generico, permettono di modificarlo. I text editors più utilizzati sono **vi** o **vim** (**vi improved** - che è la sua evoluzione) e **emacs** (man page **vi(1)** o **vim(1)** e **emacs(1)**). *text editors*

Comando: **vim**

Path: **/usr/bin/vim**

SINTASSI

\$ **vim** [*option*] *file*

DESCRIZIONE

⁴v. sez. 1.9.

option specifica le opzioni di funzionamento di **vim**. Può assumere i seguenti valori:

- +*[num]*** indica a **vim** di posizionare il cursore sulla *num*-esima riga del primo file specificato (se *num* non è specificato, il cursore viene posizionato sull'ultima riga del file);
- +/*pattern*** indica a **vim** di posizionare il cursore sulla prima occorrenza di *pattern*;
- +*command*** indica a **vim** di eseguire il comando *command* (interpretato come un comando *Ex*) dopo la lettura del primo file;
- c *command*** come **+*command***;
- cmd *command*** come **-c *command***, ma il comando viene eseguito prima di considerare qualsiasi file *vimrc*;
- b** apre i file in modalità binaria;
- C** imposta la modalità compatibile a **vi**, anche se esiste il file *.vimrc*;
- d** imposta la modalità *diff*. Visualizza le differenze tra i file indicati (v. *vimdiff*);
- e** imposta la modalità *Ex*;
- f** (per l'interfaccia grafica) non esegue il fork dalla shell da cui è lanciato;
- F** imposta la modalità di scrittura Farsi, da destra verso sinistra (se è stato compilato con le opportune opzioni);
- g** lancia l'interfaccia grafica (se è stato compilato con le opportune opzioni);
- h | --help** visualizza un aiuto sommario di **vim**;
- H** imposta la modalità di scrittura Hebrew (ebraico), da destra verso sinistra (se è stato compilato con le opportune opzioni);
- i *filename*** indica di utilizzare il file *filename* al posto del file */.viminfo*;
- L** come **-r**;
- l** imposta la modalità *Lisp*;
- m** disabilita la modifica dei file (sola lettura);
- N** disabilita la piena compatibilità con **vi**;
- n** indica di non utilizzare nessun file di swap;
- o[*n*]** indica di aprire *n* finestre;
- R** imposta la modalità di sola lettura (il contenuto dei file visualizzato può essere modificato, ma il salvataggio dello stesso viene impedito, se non è esplicitamente indicato);
- r** elenca i file visualizzati;
- r *filename*** indica di utilizzare il file *filename* per il ripristino di un file in modifica durante un crash di **vim**;
- s** imposta la modalità *silenziosa* (solo per la modalità *Ex*);
- s *filename*** il file *filename* viene letto ed interpretato come se i comandi in esso contenuti fossero digitati dalla tastiera;
- T *terminal*** specifca il nome del terminale che si sta utilizzando;
- u *filename*** indica di leggere le informazioni di inizializzazione dal file *filename*;
- U *filename*** indica di utilizzare i comandi contenuti nel file *filename* per l'inizializzazione dell'interfaccia grafica;
- V** indica di visualizzare più informazioni sui file;
- v** imposta la modalità *vi* (se è lanciato con il comando **ex**);

```

-w filename
    tutti i comandi impartiti sono memorizzati nel file filename (se il
    file esiste i nuovi comandi vengono aggiunti);
-W filename
    come -w ma se il file esiste viene prima cancellato;
-x
    i file vengono salvati in modo cifrato (viene richiesto l'inserimento
    di una chiave per la cifratura);
-Z
    imposta delle restrizioni sul funzionamento di vim (come per i
    comandi che iniziano con 'r');
--
    indica esplicitamente la fine delle opzioni (utilizzato per l'aper-
    tura di file che iniziano con '-');
--version
    visualizza la versione di vim;
--remote
    indica di collegarsi in remoto ad un server vim;
--serverlist
    elenca i server vim che riesce a trovare;
--servername name
    specifica il server vim da utilizzare;
--serversend keys
    invia i comandi specificati da keys;
--socketid id
    utilizza il meccanismo GtkPlug per lanciare gvim in un'altra fi-
    nestra (funziona solo con le librerie gtk);
--echo-wid
    visualizza l'identificativo della finestra (Window ID) relativa al-
    l'interfaccia grafica;

```

file specifica il file da aprire. Può essere uno dei seguenti valori

```

filename
    un nome di file;
filelist
    un elenco di file separati da uno spazio;
-
    il file da aprire viene letto dallo standard input, mentre i comandi
    sono ricevuti dallo standard error che deve essere un terminale;
-t tag
    il file da aprire e la posizione iniziale del cursore dipende da tag, una
    sorta di etichetta;
-q [errorfile]
    imposta la modalità quickFix. Il file errorfile viene letto e visualiz-
    zato il primo errore (se il file viene omesso, viene considerato il file
    errors.vim);

```

Ulteriori informazioni possono essere ottenute da vim con il comando **help** che ha la seguente sintassi

```
:help [subject]
```

dove *subject* indica l'argomento di cui si desiderano ulteriori informazioni. L'elenco dei file relativi alla documentazione di vim può essere ottenuto con il comando

```
:help doc-file-list
```

L'editor vim ha caratteristiche diverse dipendentemente dal nome del comando utilizzato per lanciarlo. Possono infatti essere utilizzati i seguenti comandi

vim la modalità classica;

ex avvia in modalità *Ex*. È possibile ritornare alla modalità classica con il comando `":vi"`;

view apre i file in sola lettura;

gvim | **gview**
avvia l'interfaccia grafica di vim;

rvim | **rview** | **rgvim** | **rgview**
avvia l'interfaccia grafica di vim con alcune restrizioni;

??? Esempio ???

Comando: **emacs**
 Path: **/usr/bin/emacs**
 SINTASSI
\$ emacs [option] file

DESCRIZIONE

???

Per una spiegazione dettagliata delle numerose opzioni di entrambi gli editor menzionati si rimanda alle relative man page.

???

pico
 nano

4.8 Ricerca di file o nel contenuto di file

Spesso capita di non sapere se esiste un file all'interno di un filesystem o di non ricordare dove è stato creato. Per ricercare un file all'interno del filesystem può essere utilizzato il comando **find** (man page **find(1)**).

Comando: **find**
 Path: **/usr/bin/find**
 SINTASSI
\$ find [path ...] [expression]

DESCRIZIONE

path indica il percorso della directory sulla quale effettuare la ricerca e *expression* è un'espressione che indica il tipo di ricerca da effettuare.

???

Poiché **find** è un comando molto potente ma con una sintassi forbita, per dettagli ulteriori si rimanda alla relativa man page.

Il comando **which** (man page **which(???)**) invece permette di ricercare un file nelle varie directory elencate nella variabile di ambiente **PATH**.

??? man page **which** ???

Poiché il percorso di ricerca di **which** è lo stesso di quello utilizzato dalla shell nell'interpretazione dei comandi (v. cap. 7), **which** è un comando che può essere utile quando in corrispondenza del lancio di un comando si ha come risposta un messaggio del tipo: ???

Un altro comando utile di GNU/Linux è **grep** (man page **grep(1)**). Si tratta di un comando molto complesso che permette di effettuare ricerche di sequenze di byte all'interno di un file.

Comando: **grep**
 Path: **/bin/grep**
 SINTASSI
\$ grep [option] pattern [file]

DESCRIZIONE

dove *pattern* è la stringa da ricercare all'interno del file *file*.

Per le opzioni (*option*) si rimanda direttamente alla relativa man page.

???

Quindi, se si desiderano ricercare tutti i file che contengono una determinata stringa è sufficiente digitare il comando

????

4.9 Differenze tra file

```
diff
  diff3
  pdiff
```

4.10 Ordinamento

```
sort
```

4.11 Archiviazione e compressione

```
bzip2
  ???
```

4.12 Le espressioni regolari

Esistono delle sintassi standard che definiscono dei significati particolari ad alcune sequenze di caratteri, che vengono utilizzate in particolar modo da strumenti che effettuano ricerche di stringhe in un testo. Tali sequenze di caratteri prendono il nome di **espressioni regolari** o *regular expression* (RE).

espressioni regolari

Esistono due sintassi diverse per l'interpretazione delle espressioni regolari: la sintassi di base (o elementare) BRE (Basic Regular Expression) e la sintassi estesa ERE (Extended Regular Expression). La sintassi di base delle espressioni regolari sui sistemi Unix-like è identificata dall'acronimo SRE (Simple regular Expression), ma per ovviare problemi legati alla localizzazione si tende sempre a far riferimento a BRE o ERE.

Il significato delle varie sequenze di caratteri, secondo la sintassi BRE sono specificate di seguito

.	identifica un singolo carattere;
^	identifica l'inizio della riga;
\$	identifica la fine della stringa;
[<i>set</i>]	identifica i caratteri contenuti nell'insieme specificato da <i>set</i> . Un insieme può essere specificato dall'elenco degli elementi (es. 1257 rappresenta l'insieme dei caratteri 1, 2, 5 e 7), da un intervallo (es. a-d rappresenta l'insieme degli elementi a, b, c e d) o dall'indicazione di una classe predefinita di elementi (es. [:alpha:] rappresenta l'insieme di tutti i caratteri alfabetici);
<i>a</i> ?	indica nessuna o una occorrenza del carattere <i>a</i> ;
<i>a</i> +	indica una o più occorrenze del carattere <i>a</i> ;
<i>a</i> *	indica nessuna o più occorrenze del carattere <i>a</i> ;
<i>a b</i>	indica l'occorrenza del carattere <i>a</i> o <i>b</i> ;
<i>a</i> { <i>n</i> }	esattamente <i>n</i> volte il carattere <i>a</i> ;
<i>a</i> { <i>n</i> ,}	almeno <i>n</i> volte il carattere <i>a</i> ;
<i>a</i> { <i>n</i> , <i>m</i> }	da <i>n</i> a <i>m</i> volte il carattere <i>a</i> ;

Nelle espressioni regolari elementari, i metacaratteri '?', '+', '{', '|', '(' e ')' perdono il loro significato speciale. Al loro posto possono essere utilizzati gli stessi simboli preceduti dal *backslash* '\

Le espressioni regolari sono *case sensitive*, ovvero distinguono le maiuscole dalle minuscole.

???

4.13 Riferimenti

???

Capitolo 5

Utenti ed accesso al sistema

“Gli uomini mutano sentimenti e comportamenti con la stessa rapidità con cui si modificano i loro interessi.”

– A. Schopenhauer

In questo capitolo viene trattato tutto ciò che è inerente agli utenti di un sistema GNU/Linux, a partire dalla creazione degli *account* alla definizione dei gruppi e alla procedura di autenticazione (procedura di accesso) del sistema.

5.1 User account

Una persona può accedere ad un sistema diventandone un **utente**. Ogni utente è identificato univocamente dal sistema da un valore numerico detto **UID** (User IDentifier). A tale valore è associata una coppia di valori alfanumerici: lo *username* e la *password*. Lo **username** (nome dell'utente) non è altro che un alias alfanumerico dello UID, che l'utente può più facilmente memorizzare per fornirlo al sistema qualora gli venga richiesto (per esempio ciò avviene al momento dell'accesso al sistema stesso, cioè durante la procedura di accesso al sistema o *procedura di login*). La **password** è una “parola d'ordine” che solo l'utente interessato dovrebbe conoscere (mentre lo username almeno l'amministratore del sistema lo conosce). Questa serve per garantire l'autenticità dell'utente dichiarato con lo username per la procedura di accesso al sistema (v. sez. 5.9).

utente

UID

username

password

Per mantenere la compatibilità con alcuni vecchi programmi è opportuno che lo *username* sia composto al massimo da otto caratteri. Inoltre, questo dovrebbe essere composto da lettere non accentate e numeri: qualunque altro simbolo, compresi i segni di punteggiatura, potrebbero creare problemi di vario tipo.

Per ogni utente è anche definita una **home directory**, cioè la directory che sarà la *working directory* (la directory corrente) di default per l'utente subito dopo che questi è stato autenticato dal sistema. Tale directory è indicata dal carattere tilde ‘~’. In genere la home directory è costituita dalla directory */home/username*. Ad un utente è anche assegnata una **default shell**, ovvero uno specifico interprete dei comandi che accoglie l'utente subito dopo che questo è stato autenticato dal sistema (l'interfaccia a caratteri assegnata all'utente)¹.

home directory

default shell

L'insieme composto dallo UID, lo username, la password e gli altri dettagli relativi ad un utente, costituisce lo **user account** dell'utente presso il sistema. Dunque, un utente può accedere ad un sistema soltanto se ha uno *user account* (o semplicemente *account*) su quel sistema.

user account

Gli user account sono memorizzati nel file */etc/passwd* (man page *passwd(5)*) che è un file in formato testo le cui righe hanno la seguente sintassi:

¹la *shell* sarà trattata nel cap. 7.

username:password:UID:GID:description:home directory:shell

dove

username

è il nome dell'utente, ovvero l'alias alfanumerico con cui l'utente si presenta al sistema quando vuole accedervi;

password

è la password (parola d'ordine) dell'utente in forma cifrata (se vuota, la password è nulla);

UID è l'identificativo univoco dell'utente (User IDentifier);

GID è l'identificativo univoco del gruppo (Group IDentifier) di default di cui l'utente fa parte²;

description

è una descrizione dell'utente (in genere si tratta del nome e cognome dell'utente);

home directory

è la *working directory* di default dell'utente;

shell

è l'interprete dei comandi di default (*default shell*) per l'utente (se vuota viene utilizzato `/bin/sh`). In realtà questo è il path assoluto del file da eseguire una volta che l'utente è stato autenticato.

È importante notare che, per motivi di sicurezza, la password contenuta in `/etc/passwd` non è in chiaro (plain text), ma è cifrata col sistema di cifratura utilizzato dal sistema (in genere MD5)³. In questo modo chiunque arrivi a leggere il contenuto del file `/etc/passwd` non vede la password degli utenti e non è neanche in grado di risalirvi a partire dalla sua cifratura (questo dipende dal metodo utilizzato per cifrare la password).

Se *password* è un asterisco '*', significa che l'utente in questione non può accedere al sistema attraverso la procedura di autenticazione che in genere avviene tramite *login*.

Tra tutti gli utenti ce n'è uno particolare che il sistema riconosce come *amministratore*, quello il cui UID è 0. Questi, sui sistemi Unix-like, è chiamato **superuser**, ed in genere gli viene assegnato lo username "root" e la home directory `/root`. Il *superuser* ha accesso indiscriminato a tutte le risorse del sistema, ovvero può accedere a tutto il filesystem in lettura/scrittura e lanciare qualunque comando. Il sistema non fa nessun controllo per l'accesso alle risorse effettuato dal *superuser*, pertanto l'accesso al sistema in qualità di tale utente deve essere ridotto al minimo indispensabile.

La creazione di un utente consta dei seguenti passi:

1. Aggiungere una riga al file `/etc/passwd` relativa al nuovo utente;
2. Aggiungere (eventualmente) una riga al file `/etc/group` relativo ad nuovo gruppo per il nuovo utente;
3. Se esiste il file `/etc/shadow` aggiungere una riga relativa al nuovo utente⁴;
4. Se esiste il file `/etc/gshadow` aggiungere (eventualmente) una riga al nuovo gruppo per il nuovo utente;
5. Creare la *home directory* del nuovo utente (`/home/username`);
6. Copiare il contenuto della *skeleton directory* (in genere la directory `/etc/skel`) in `/home/username`;

²i gruppi sono trattati in sez. 5.2

³v. sez. 5.3.

⁴la gestione delle *shadow password* è trattata in sez. 5.4

7. Attribuire come proprietario di `/home/username` l'utente appena creato e come gruppo proprietario il gruppo di default a cui appartiene l'utente appena creato;
8. Attribuire gli opportuni permessi alla *home directory* (lettura, scrittura ed esecuzione per il proprietario);

È compito dell'amministratore far in modo che non vi siano due o più *user account* con lo stesso *username* poiché il sistema non sarebbe, in tal caso, capace di decidere a quale UID si riferisce lo username fornito dall'utente in fase di autenticazione.

Uno user account può essere creato con il comando **useradd** (man page **useradd(???)**).

Comando: **useradd**

Path: `???`/**useradd**

SINTASSI

useradd [*option*] *username*

DESCRIZIONE

option indica la modalità di funzionamento di **useradd**. Può assumere i seguenti valori:

`???` ;

username ;

`???`

che per mezzo delle opportune *opzioni* (v. man page **useradd(8)**) permette di specificare lo UID, la password, la home directory e la shell di default per l'utente che si desidera creare.

Per la creazione di uno user account, **useradd** utilizza le direttive riguardanti le caratteristiche delle utenze a livello di sistema (casella di posta elettronica, la lunghezza massima della password, ...) contenute in `/etc/login.defs` (man page **login.defs(5)**) ed il file di configurazione `/etc/default/useradd` che contiene le impostazioni di default per la creazione degli user account (la radice delle home directory, la shell di default, la skeleton directory di default, ...). La **skeleton directory** è una directory il cui contenuto viene copiato all'interno della home directory di un utente alla sua creazione. In genere tale directory è `/etc/skel`⁵.

skeleton directory

La modifica dell'account di un utente può essere effettuata per mezzo del comando **usermod** (man page **usermod(8)**) e l'eliminazione di uno user account può essere effettuata con il comando **userdel** (man page **userdel(8)**).

Comando: **usermod**

Path: `???`/**usermod**

SINTASSI

usermod [*option*] *username*

DESCRIZIONE

option indica la modalità di funzionamento di **usermod**. Può assumere i seguenti valori:

`-c comment`

;

username ;

`???`

⁵il nome **skel** sta per *skeleton*, cioè scheletro. Questa directory infatti rappresenta lo scheletro di una generica home directory.

Comando: **userdel**
 Path: ???/userdel

SINTASSI
userdel [*option*] *username*

DESCRIZIONE

option indica la modalità di funzionamento di **userdel**. Può assumere i seguenti valori:

-r ;

username ;

???

Ogni utente può modificare in qualunque momento la propria *password* per mezzo del comando **passwd** (man page **passwd(1)**).

Comando: **passwd**
 Path: ???/passwd

SINTASSI
\$ passwd [*option*] [*username*]

DESCRIZIONE

option indica la modalità di funzionamento di **passwd**. Può assumere i seguenti valori:

-k ;

username ;

???

Il superuser può modificare la password di qualunque utente.

Un utente può anche cambiare la propria *default shell* tramite il comando **chsh** (man page **chsh(1)**).

Comando: **chsh**
 Path: ???/chsh

SINTASSI
\$ chsh [*option*] [*username*]

DESCRIZIONE

option indica la modalità di funzionamento di **chsh**. Può assumere i seguenti valori:

-s shell

;

username ;

???

Il superuser può modificare la shell di default di qualunque utente.

La default shell deve essere una shell esistente, possibilmente elencata in **/etc/shells** (man page **shells(5)**).

Esiste la possibilità di modificare il contenuto del campo *description* di uno user account in **/etc/passwd**, per mezzo del comando **chfn** (man page **chfn(1)**).

Comando: **chfn**
 Path: **???**/**chfn**
 SINTASSI
\$ chfn [*option*] [*username*]

DESCRIZIONE

option indica la modalità di funzionamento di **chfn**. Può assumere i seguenti valori:
-f *full_name*
 ;
username ;

???

Il superuser può modificare le informazioni relative a qualunque utente.

Le informazioni richieste da **chfn** vengono scritte nello user account in **/etc/passwd** all'interno del campo *description* separate tra loro con una virgola.

Poiché le informazioni personali possono essere delicate, come ad esempio il numero telefonico dell'abitazione di un utente, quando si tratta di utenze presso elaboratori raggiungibili attraverso una rete estesa, come *Internet*, occorre prestare attenzione a ciò che si sta facendo: occhi indiscreti potrebbero arrivare a leggere tali informazioni.

Un utente può visualizzare il proprio *username* attraverso il comando **whoami** (man page **whoami(1)**).

Comando: **whoami**
 Path: **???**/**whoami**
 SINTASSI
\$ whoami [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di **whoami**. Può assumere i seguenti valori:
--help ;

???

Un utente può visualizzare lo username che ha utilizzato al momento dell'accesso al sistema con il comando **logname** (man page **logname(1)**) che ha la seguente sintassi

Comando: **logname**
 Path: **???**/**logname**
 SINTASSI
\$ logname [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di **logname**. Può assumere i seguenti valori:
--help ;

???

È importante notare la differenza tra lo username attuale e quello al momento dell'accesso al sistema. Nella maggior parte dei casi tali valori coincidono ma, come sarà illustrato nella sez. 5.5, un utente può temporaneamente impersonarne un altro ed in tal caso i due valori sono diversi.

Ad ogni utente viene automaticamente assegnata dal sistema una casella di posta elettronica⁶ rappresentata dal file **/var/mail/username** (o **/var/spool/mail/username**

⁶la posta elettronica è trattata nel cap. 16.

a seconda della distribuzione GNU/Linux considerata) in cui vengono accumulati i messaggi di posta elettronica a lui diretti.

5.2 Group account

Come già accennato precedentemente, un utente appartiene sempre ad almeno un *gruppo*. Un **gruppo** rappresenta un insieme di utenti. L'appartenenza di un utente ad un gruppo gli concede particolari diritti su determinati file e directory.

Ogni gruppo è identificato univocamente dal sistema da un valore numerico detto **GID** (Group Identifier). A tale valore è associato un alias alfanumerico: il **groupname** (nome del gruppo), più facilmente memorizzabile. Ad ogni gruppo sono associati gli utenti che vi appartengono.

L'insieme composto dal GID, il groupname, l'eventuale password e l'elenco degli utenti che vi appartengono, costituisce il **group account** del gruppo presso il sistema.

I group account sono memorizzati nel file `/etc/group` (man page `group(5)`) che è un file in formato testo le cui righe hanno la seguente sintassi:

```
groupname:password:GID:member_list
```

dove

groupname

è il nome del gruppo;

password

è la password del gruppo in forma cifrata (se è nulla soltanto gli utenti elencati in *elenco utenti* possono accedere al gruppo);

GID è l'identificativo univoco del gruppo (Group Identifier);

member_list

è l'elenco degli utenti appartenenti al gruppo (membri).

Un group account viene creato con il comando

```
# groupadd [option] groupname
```

che per mezzo delle opportune *option* (v. man page `groupadd(8)`) permette di specificare il GID per il gruppo che si desidera creare.

La modifica del group account può essere effettuata per mezzo del comando `groupmod` (man page `groupmod(8)`).

Comando: `groupmod`

Path: `??/groupmod`

SINTASSI

```
# groupmod [option] groupname
```

DESCRIZIONE

option indica la modalità di funzionamento di `groupmod`. Può assumere i seguenti valori:

```
-g GID
```

```
;
```

```
groupname ;
```

???

e l'eliminazione di un group account può essere effettuata con il comando `groupdel` (man page `groupdel(8)`).

Comando: **groupdel**
 Path: ???/groupdel
 SINTASSI
groupdel groupname

DESCRIZIONE

groupname ;

???

Un utente può visualizzare l'elenco dei gruppi di cui fa parte con il comando **groups** (man page **groups(1)**).

Comando: **groups**
 Path: ???/groups
 SINTASSI
\$ groups [option] [username]

DESCRIZIONE

option indica la modalità di funzionamento di **groups**. Può assumere i seguenti valori:

--help ;

username ;

???

Il superuser può visualizzare l'elenco dei gruppi di cui fa parte un qualunque utente.

Un utente può visualizzare il proprio UID e GID (corrente) con il comando **id** (man page **id(1)**).

Comando: **id**
 Path: ???/id
 SINTASSI
\$ id [option] [username]

DESCRIZIONE

option indica la modalità di funzionamento di **id**. Può assumere i seguenti valori:

-a ;

username ;

???

Il superuser può visualizzare lo UID e GID di qualunque utente.

È evidente che il GID di riferimento di un utente è un valore che può cambiare se l'utente fa parte di più di un gruppo. Al momento dell'accesso al sistema all'utente viene assegnato il GID di default, quello contenuto nello user account, ma l'utente può cambiarlo, cambiando il gruppo corrente di riferimento con il comando **newgrp** (man page **newgrp(1)**).

Comando: **newgrp**
 Path: ???/newgrp
 SINTASSI
\$ newgrp [groupname]

DESCRIZIONE

groupname ;

???

In questo modo l'utente dichiara al sistema di essere considerato da quel momento in poi, come membro di *gruppo*. Il cambiamento da un *gruppo A* ad un *gruppo B* può essere effettuato soltanto dagli utenti membri del *gruppo B* e/o dagli utenti che conoscono la password del *gruppo B* (ovviamente membri del *gruppo A*).

L'accesso al gruppo tramite richiesta di *password* per un utente non elencato in *member_list* (utente non membro del gruppo), ovvero la gestione delle password relative ai gruppi, può non funzionare a causa dello scarso interesse in tale caratteristica da parte delle varie distribuzioni di GNU/Linux. Infatti, l'idea di poter aggiungere una password ai gruppi, in modo che gli utenti estranei al gruppo che la conoscono possano essere considerati come membri del gruppo stesso (tramite *newgrp*) è piuttosto discutibile, poiché una password è "sicura" solo se conosciuta da una sola persona; nel momento in cui la stessa password è conosciuta da un gruppo di persone, la sua diffusione diventa (a causa della natura umana) incontrollabile. Ha poco senso pertanto fidarsi della presentazione di una password comune per l'accesso di un utente ad un gruppo.

5.3 La cifratura della password

Per motivi di sicurezza il sistema non memorizza le *password* in chiaro (o *plain text*), cioè così come sono digitate dalla tastiera, ma vengono cifrate (camuffate) tramite opportuni algoritmi (programmi) di cifratura⁷.

Per aumentare il livello di sicurezza della password è bene non scrivere la stessa da nessuna parte o se la si scrive riporla in un luogo sicuro (lontano da occhi indiscreti). È buona norma utilizzare una password che abbia la minima correlazione con l'utente che la utilizza, non darla mai a nessuno, per nessun motivo ed assicurarsi che nessuno stia guardando la tastiera mentre la si digita. Inoltre è importante non inserire la password in un sistema che non si conosce ed è consigliabile cambiarla periodicamente.

algoritmi di cifratura

Gli **algoritmi di cifratura** utilizzati attualmente da GNU/Linux sono DES e MD5. L'algoritmo DES (Data Encryption Standard)⁸ accetta password composte da un massimo di *otto caratteri* mentre MD5 (Message-Digest algorithm)⁹ non pone limiti al riguardo.

Un sistema di autenticazione basato sulla conservazione di una password cifrata ha una debolezza di fondo: conoscendo la stringa cifrata e l'algoritmo che la genera, si può determinare, per mezzo di un elevato numero di tentativi, una password conforme a quella originale (due password in chiaro sono "conformi" se entrambe – fissato l'algoritmo di cifratura – danno luogo alla stessa password cifrata). Un sistema che consente l'utilizzo di password con un massimo di otto caratteri è molto debole oggi, perché tutte le combinazioni possibili possono essere provate in tempi brevi, anche con un elaboratore di media potenza di calcolo.

La cifratura avviene tramite la funzione **crypt** (man page **crypt(3)**). La scelta dell'algoritmo per la cifratura delle password viene effettuata in base al secondo parametro passato a **crypt**.

La funzione **crypt** utilizza sempre l'algoritmo di cifratura DES a meno che il secondo parametro (*salt*) non sia la stringa "\$1\$"; in tal caso viene utilizzato l'algoritmo MD5.

⁷in realtà le password non vengono semplicemente camuffate, ma viene effettuata un'operazione irreversibile su di esse, in modo tale che non sia possibile ricondursi alla password in chiaro (v. cap. ??).

⁸DES è un algoritmo di cifratura a blocchi di 64 bits con una chiave a 56 bits, sviluppato da IBM negli anni '70.

⁹MD5 è un algoritmo creato nel 1991 da R. Rivest e descritto nella RFC 1321. È utilizzato come funzione hash, cioè converte un messaggio in un predefinito numero di cifre (128 bits): un *message digest*.

5.4 Il meccanismo delle shadow password

Oltre alla normale gestione degli user account, esiste un meccanismo di gestione detto **shadow password** che si basa sul principio di nascondere le password cifrate ai processi che non hanno i privilegi del superuser. Infatti, come è stato illustrato in sez. 5.1, le password sono contenute nel file `/etc/passwd` (nei sistemi in cui la gestione delle shadow password non è attiva), leggibile a tutti i tipi di utenti poiché questo contiene, oltre alle password, delle informazioni relative agli utenti che alcuni applicativi hanno bisogno di conoscere per il loro corretto funzionamento.

shadow password

Il problema nasce dal fatto che conoscendo la password, anche in forma cifrata, è possibile riuscire a generare una password che dia la stessa cifratura (attacchi per dizionario)¹⁰. Il meccanismo delle shadow password è tale per cui le password vengono effettivamente rimosse dal file `/etc/passwd` ed inserite in un altro file a cui può vi si può accedere soltanto avendo i privilegi del superuser. Così facendo, l'accesso in lettura al file `/etc/passwd` non implica la conoscenza delle password cifrate.

Ovviamente, l'utilizzo del meccanismo delle shadow password richiede che alcuni programmi, quelli che devono accedere alle password per l'autenticazione degli utenti, siano predisposti per questo. Ormai tutte le distribuzioni di GNU/Linux adottano tale gestione.

Il meccanismo delle shadow password implica la presenza del file `/etc/shadow` in parallelo al file `/etc/passwd`, che non conterrà più le password cifrate dei vari utenti, ma al loro posto comparirà semplicemente una 'x'. Nel file `/etc/shadow` (man page `shadow(5)`) sono presenti delle righe con la sintassi seguente:

```
username:password:last_modify:min_validity:max_validity:warning:expiring:end:reserved
```

dove

<i>username</i>	è l'alias alfanumerico relativo all'utente (nome dell'utente);
<i>password</i>	è la password cifrata dell'utente (tolta al file <code>/etc/passwd</code>);
<i>last_modify</i>	è la data in cui è stata modificata la password per l'ultima volta;
<i>min_validity</i>	è il numero di giorni di validità minima della password: entro questo tempo, l'utente non può cambiare la propria password;
<i>max_validity</i>	è il numero di giorni di validità massima della password: prima che trascorra questo periodo di tempo, l'utente deve cambiare la password;
<i>warning</i>	è il numero di giorni, prima della scadenza della password, durante i quali l'utente viene avvisato della necessità di modificarla;
<i>expiring</i>	è la durata massima di validità dello user account dopo che la password è scaduta;
<i>end</i>	è la data di scadenza dello user account;
<i>reserved</i>	riservato per usi futuri.

I campi che rappresentano una data possono contenere un numero intero che indica il numero di giorni trascorsi dal 01/01/1970, mentre quelli che rappresentano una durata, possono contenere un numero intero che esprime una quantità di giorni.

Con il meccanismo delle shadow password è possibile gestire anche la *durata* della password degli utenti, come illustrato in fig. 5.1, per mezzo del comando **chage** (**change age**) (man page `chage(1)`).

¹⁰v. cap. ??.

Comando: **chage**
 Path: **???**/chage
 SINTASSI
chage [*option*] *username*

DESCRIZIONE

option indica la modalità di funzionamento di **chage**. Può assumere i seguenti valori:

-m *mindays*
 ;

username ;

???

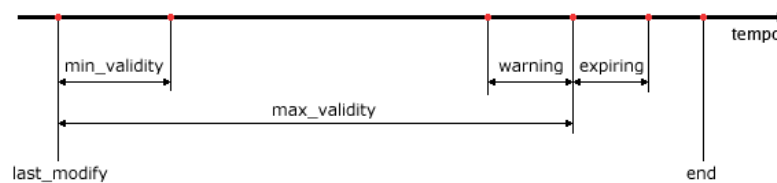


Figura 5.1: La gestione della scadenza della password con il meccanismo della shadow password.

Anche la gestione dei gruppi è leggermente diversa da quanto illustrato in sez. 5.2: analogamente a quanto accade per gli utenti, il file **/etc/group** viene privato delle password dei gruppi ed a questo viene affiancato il file **/etc/gshadow** che contiene delle righe con la sintassi seguente:

groupname : *password* : *groupadmins* : *groupmembers*

dove

groupname

è l'alias alfanumerico relativo al gruppo (nome del gruppo);

password

è la password cifrata del gruppo (tolta al file **/etc/group**);

groupadmins

è l'elenco, separato da virgole, degli amministratori del gruppo;

groupmembers

è l'elenco, separato da virgole, degli utenti (non amministratori) membri del gruppo;

Vengono dunque introdotte le figure degli *amministratori dei gruppi* che possono aggiungere o rimuovere gli utenti membri all'interno di un gruppo.

5.4.1 Attivare/disattivare il meccanismo delle shadow password

Il sistema delle shadow password può essere abilitato o disabilitato tramite opportuni comandi. Il comando **pwconv** (man page **pwconv(8)**) permette di convertire il file **/etc/passwd** standard in una coppia di file **/etc/passwd** e **/etc/shadow**, togliendo dal primo le password cifrate.

Comando: **pwconv**
 Path: **???**/pwconv
 SINTASSI
pwconv

???

Il programma funziona anche se il file `/etc/shadow` esiste già: in tal caso serve per allineare correttamente i file `/etc/passwd` e `/etc/shadow`. Si avvale della configurazione contenuta in `/etc/login.defs` (man page `login.defs(5)`) per stabilire i periodi di validità delle password.

Il comando `grpconv` (man page `grpconv(8)`) permette di convertire il file `/etc/group` normale in una coppia di file `/etc/group` e `/etc/gshadow`, togliendo dal primo le eventuali password cifrate.

Comando: `grpconv`

Path: `??/grpconv`

SINTASSI

`grpconv`

???

Il programma funziona anche se il file `/etc/gshadow` esiste già: in tal caso serve per allineare correttamente i file `/etc/group` e `/etc/gshadow`.

Il comando `pwunconv` svolge l'operazione inversa a quella di `pwconv`, ovvero trasferisce le password cifrate dal file `/etc/shadow` nel file `/etc/passwd`, eliminando poi il file `/etc/shadow`. La sintassi è la seguente

Comando: `pwunconv`

Path: `??/pwunconv`

SINTASSI

`pwunconv`

???

Il programma è in grado di funzionare correttamente anche se parte delle utenze si trovano già solo nel file `/etc/passwd`.

Il comando `grpunconv` svolge l'operazione inversa a quella di `grpconv`, ovvero trasferisce le eventuali password cifrate dal file `/etc/gshadow` nel file `/etc/group`, eliminando poi il file `/etc/gshadow`. La sintassi è la seguente

Comando: `grpunconv`

Path: `??/grpunconv`

SINTASSI

`grpunconv`

???

Il programma è in grado di funzionare correttamente anche se parte delle utenze si trovano già solo nel file `/etc/group`.

???

`vipw`

`vigr`

???

5.5 Impersonare un altro utente

Nei sistemi Unix-like esiste la possibilità di cambiare identità, ovvero un utente può impersonarne un altro. Questo rappresenta generalmente un problema di sicurezza, in quanto c'è la possibilità che un utente generico possa acquisire i privilegi del superuser impersonandolo. Il comando che rende possibile questa operazione è `su` (man page `su(1)`). In realtà tale comando lancia una shell con i privilegi di un determinato utente.

Comando: **su**
 Path: **/bin/su**

SINTASSI

\$ su [*option*] [*username*] [*argument*]

DESCRIZIONE

option indica la modalità di funzionamento di **su**. Può assumere i seguenti valori:

- | **-l** | **--login**
 lancia una *shell di login*¹¹;
- c *command* | **--command=command**
 passa il comando *command* alla shell lanciata;
- f | **--fast**
 la shell con l'opzione -f (csh o tcsh);
- m | -p | **--preserve-environment**
 mantiene le variabili di ambiente del processo padre (la shell da cui viene lanciato);
- s *shell* | **--shell=shell**
 lancia la shell *shell*;
- help** mostra un aiuto sommario del funzionamento di **su**;
- version**
 mostra la versione di **su**;

username è l'utente da impersonare (se non è specificato si intende il superuser).

argument è il parametro che viene passato alla shell.

Il comando **su** permette ad un *utente A* di diventare temporaneamente un altro (*utente B*), avviando una shell con i privilegi dell'*utente B*. Per poter diventare l'*utente B*, l'*utente A* deve conoscere la password dell'*utente B* (a meno che l'*utente A* non sia il superuser, nel qual caso non è richiesto l'inserimento di alcuna password). Una volta impersonato l'*utente B* è possibile concludere l'attività in veste di tale utente terminando la shell relativa con il comando **exit**, ritornando ad essere l'*utente A*.

Per poter funzionare, il file **/bin/su** deve essere di proprietà del superuser ed avere il bit SUID attivato¹²

sudo
 ???

5.6 Casi particolari

In un sistema GNU/Linux sono generalmente definiti degli utenti fittizi (non si tratta di utenti veri e propri poiché per essi non è prevista la possibilità di accedere al sistema) per la gestione di alcuni processi. Inoltre, per come sono gestiti gli utenti in un sistema GNU/Linux, esiste la possibilità di gestire utenti diversi con lo stesso UID o lanciare un file eseguibile, non una shell, all'accesso di un utente al sistema.

5.6.1 System user e system group

Come sarà illustrato in cap. 6, i processi che girano su un sistema GNU/Linux hanno (in genere) i privilegi dell'utente che li ha lanciati in esecuzione. Pertanto, per poter dare certi privilegi a processi particolari per il sistema ma allo stesso tempo ridurre al minimo il numero di processi lanciati con i privilegi del superuser, in un sistema GNU/Linux sono definiti degli utenti particolari (fittizi). Si tratta di **system user** (utenti di sistema), nel senso che servono soltanto per lanciare alcuni processi, ma non hanno la possibilità di accedere al sistema (il relativo campo *password* nel file **/etc/passwd** è '*').

Di conseguenza, anche **/etc/group** contiene group account particolari: i **system group** (gruppi di sistema). Segue la descrizione di alcuni di questi utenti e gruppi.

system user

system group

¹¹la *shell di login* sarà trattata nel cap. 7.

¹²v. cap. 3.

bin è il gruppo fittizio proprietario dei file eseguibili (binary). In genere il superuser fa parte del gruppo *bin* in modo tale che questi eseguibili appartengono al gruppo *bin*, mentre l'utente proprietario resta il superuser (*root*).

tty è il gruppo fittizio proprietario dei file di dispositivo¹³ relativi ai canali di comunicazione per i terminali;

disk è il gruppo fittizio proprietario dei file di dispositivo relativi alle unità a disco;

floppy è il gruppo fittizio proprietario dei file di dispositivo relativi alle unità a dischetti;

nobody è l'utente fittizio corrispondente ad un utente generico non identificato, senza nessun privilegio particolare. In genere viene usato per l'accesso a NFS da parte del superuser.

5.6.2 Gestione particolare delle utenze

Alcuni accorgimenti nella gestione degli utenti e dei gruppi possono essere utili in situazioni particolari.

Utente con funzione specifica

Potrebbe essere utile creare un utente che appena accede al sistema, quest'ultimo esegua un programma specifico. Per far ciò è sufficiente creare uno user account a cui, al posto della shell di default, si associa un file eseguibile¹⁴. Si ricordi che la working directory al momento dell'esecuzione del file eseguibile è quella impostata come home directory per l'utente in questione. Un esempio può essere rappresentato dal seguente user account che si riferisce all'utente *prova* che può accedere al sistema senza specificare alcuna password. Questi ha uno UID pari a 505 ed un GID uguale a 100, una descrizione uguale a "Utente di prova", la sua home directory è */tmp* e la shell di default è */usr/local/bin/prova* ovvero il file eseguibile da lanciare in esecuzione.

```
prova:505:100:Utente di prova:/tmp:/usr/local/bin/prova
```

Utenti con lo stesso UID

Potrebbe essere utile creare più user account con lo stesso UID. In tal caso ogni utente ha un proprio username ed una propria password, però tutti i file apparterrebbero ad un utente immaginario che rappresenta tutti gli utenti che hanno lo stesso UID. Questa è una specie di gestione di gruppi di utenti. È evidente che tale configurazione sottintende il fatto che si sia in un ambiente in cui esiste una certa fiducia nel comportamento reciproco. Un esempio può essere rappresentato dai seguenti user account che si riferiscono tutti all'utente *tutti* (UID = 1000).

```
tutti:*:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
alfa:34gdf6r123455:1000:1000:Utente del gruppo:/home/tutti:/bin/sh
bravo:e445gsdfr2124:1000:1000:Utente del gruppo:/home/tutti:/bin/sh
charlie:t654df7u72341:1000:1000:Utente del gruppo:/home/tutti:/bin/sh
```

Si potrebbe anche aver bisogno di utilizzare diversi nominativi-utente per accedere allo stesso elaboratore e gestire attività differenti, pur mantenendo lo stesso utente fisico (cioè lo stesso UID). In questo modo l'utente potrebbe avere a disposizione diverse directory personali, una per ogni progetto che conduce.

¹³i file di dispositivo sono trattati in cap. 3.

¹⁴i file eseguibili sono trattati in cap. 3

```
tizio:34gdf6r123455:1000:1000:Tizio Tizi:/home/tizio:/bin/sh
alfa:34gdf6r123455:1000:1000:Tizio Tizi prog. Alfa:/home/alfa:/bin/sh
bravo:34gdf6r123455:1000:1000:Tizio Tizi prog. Bravo:/home/bravo:/bin/sh
charlie:34gdf6r123455:1000:1000:Tizio Tizi prog. Charlie:/home/charlie:/bin/sh
```

5.7 La registrazione degli eventi

5.7.1 Il system log

system log

GNU/Linux, come tutti i sistemi multiutente, permette il controllo degli accessi al sistema per mezzo della registrazione degli eventi. Il sistema tiene traccia degli eventi “importanti” tramite un registro: il **system log** (*syslog* o *log di sistema*). Il system log è costituito essenzialmente da uno o più files (*/var/log/messages*, ...), detti files di log di sistema, gestiti dal daemon *syslogd* (man page *syslogd(8)*) che in genere è lanciato in esecuzione dalla procedura di avvio del sistema.

Comando: *syslogd*
Path: *???*/*syslogd*

SINTASSI
syslogd [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di *syslogd*. Può assumere i seguenti valori:

??? ;

???

Al suo avvio, il processo *syslogd* legge il file di configurazione */etc/syslog.conf* (man page *syslog.conf(5)*) che contiene le direttive che indicano in che modo devono essere gestiti gli eventi da registrare nel system log. È composto da righe con la seguente sintassi:

facility.priority action

dove

facility specifica il servizio che produce il messaggio: per esempio i programmi relativi alla posta (mail) elettronica registrano gli eventi nel system log (se usano *syslogd*) con la *facility mail*. Può essere una delle seguenti parole chiave (in ordine alfabetico):

auth (obsoleto) si riferisce al sistema di gestione sicurezza/autorizzazione;

authpriv

si riferisce al sistema di gestione sicurezza/autorizzazione privato;

cron si riferisce ai servizi di gestione dell'orario del sistema (clock) – in genere si riferisce ai daemon *cron* e *at*¹⁵;

daemon si riferisce a un daemon;

kern si riferisce al kernel;

¹⁵la schedulazione dei processi è trattata nel cap. 6.

localn (con *n* che va da 0 a 7) riservati per uso interno;

lpr si riferisce alla stampante;

mail si riferisce al servizio di posta elettronica;

mark è utilizzata soltanto per uso interno;

news si riferisce al servizio di news;

security (obsoleta) ha la stessa funzione di **auth**;

syslog si riferisce al daemon *syslogd*;

user si riferisce ad un'applicazione generica che non gira con un privilegio particolare;

uucp si riferisce al servizio UUCP (Unix to Unix CoPy)¹⁶.

facility può essere anche '*' in modo da identificare tutti i servizi, oppure può essere composto da un elenco di servizi separati da virgole.

priority definisce l'importanza (il livello di gravità) del messaggio da inserire nel system log. Può essere una delle seguenti parole chiave (in ordine di importanza decrescente):

emerg il sistema è inutilizzabile;

panic (obsoleta) ha lo stesso effetto di **emerg**;

alert è necessario prendere provvedimenti immediatamente;

crit condizione critica;

err errore;

error (obsoleta) ha lo stesso effetto di **err**;

warning avvertimento;

warn (obsoleta) ha lo stesso effetto di **warning**;

notice nota significativa;

info informazione;

debug messaggio per il debug.

In generale, *priority* rappresenta un livello di priorità ed implica l'inclusione dei messaggi che si riferiscono a quel livello, insieme a tutti quelli dei livelli superiori (più importanti). Per indicare esclusivamente un livello di priorità, occorre fare precedere *priority* dal simbolo di uguale '='.

priority può essere composto da un elenco di livelli separati da virgole. Un livello può essere escluso ponendogli anteriormente un punto esclamativo '!'.

action è l'indicazione della destinazione dei messaggi. Il tipo di destinazione è indicato dal primo carattere di *action* che può assumere i valori sotto elencati:

- / indica come destinazione il path assoluto di un file ('/' fa parte del path assoluto e rappresenta la root directory);
- | indica come destinazione il path assoluto di una pipe con nome¹⁷;
- @ indica come destinazione un elaboratore remoto, che ricevendo tali messaggi li inserirà nel proprio sistema di registrazione;
- * (un unico carattere) si intende che i messaggi debbano essere inviati sullo schermo del terminale di tutti gli utenti connessi in quel momento;

¹⁶è un sistema store-and-forward per il trasferimento di posta elettronica che utilizza chiamate telefoniche periodiche (v. i file di documentazione in */usr/doc/uucp* e */usr/info/uucp*).

¹⁷le pipe sono trattate in cap. 3

- se il primo carattere è diverso da quelli sopra indicati, si intende che si tratti di un elenco di utenti (separati da virgole) a cui inviare i messaggi sullo schermo del terminale, se questi stanno accedendo in quel momento.

Elenchi di *facility* e *priority* possono essere specificati sulla stessa riga separati da un punto e virgola ‘;’.

Un esempio del contenuto del file `/etc/syslog.conf` è riportato di seguito.

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none    /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                    /var/log/secure

# Log all the mail messages in one place.
mail.*                                        /var/log/maillog

# Log cron stuff
cron.*                                        /var/log/cron

# Everybody gets emergency messages
*.emerg                                       *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                              /var/log/spooler

# Save boot messages also to boot.log
local7.*                                     /var/log/boot.log
```

La consultazione del system log è di fondamentale importanza per riuscire a capire a cosa sia dovuto il malfunzionamento di qualche servizio.

I messaggi possono essere inseriti nel system log anche per mezzo del comando `logger` (man page `logger(1)`).

```
Comando: logger
Path: ???/logger

SINTASSI
$ logger [option] [message]
```

DESCRIZIONE

option indica la modalità di funzionamento di `logger`. Può assumere i seguenti valori:

```
??? ;
```

```
message ;
```

```
???
```

Esiste un apposito daemon per la gestione dei messaggi inviati del kernel di GNU/Linux, `klogd` (man page `klogd(8)`).

Comando: **klogd**
 Path: **???**/klogd

SINTASSI
klogd [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di **klogd**. Può assumere i seguenti valori:

??? ;

message ;

???

Questo, di norma, viene avviato dalla procedura di avvio del sistema, subito dopo *syslogd*.

5.7.2 Altri file di log

In genere gli accessi al sistema vengono “loggati” (memorizzati in un file di log) nel file **/var/run/utmp** (man page **utmp(5)**), detto anche *system status file*. Questo non è un file di testo, ma per la lettura delle informazioni in esso contenute è necessario utilizzare degli appositi comandi. Tuttavia, è possibile che gli utenti presenti effettivamente nel sistema siano in numero maggiore di quelli che risultano leggendo tale file, a causa del fatto che non tutti i programmi usano il metodo di registrazione fornito attraverso questo file. Solitamente, è la procedura di inizializzazione del sistema a prendersi cura di questo file, azzerandolo o ricreandolo, a seconda della necessità.

Il file **/var/log/wtmp** ha una struttura analoga a quella di **/var/run/utmp** e serve per conservare la registrazione degli accessi e della loro conclusione (login-logout). Questo file non viene creato automaticamente: se manca, la conservazione delle registrazioni all'interno del sistema non viene effettuata. Viene aggiornato da *init* e anche dal programma che si occupa di gestire la procedura di accesso al sistema (login).

Anche il file **/var/log/lastlog** viene utilizzato da per registrare gli ultimi accessi al sistema.

Il comando **logrotate** semplifica l'amministrazione dei log, permette di comprimere, rimuovere ed inviare il log via e-mail oltre ad eseguire una “rotazione” di file con vari criteri. Il suo file di configurazione è **/etc/logrotate.conf**. In genere le regole di gestione dei singoli log sono inserite nella directory **/etc/logrotate.d/**. e gli script che eseguono **logrotate** sono inseriti nel **crontab** (v. cap. 9).

Esistono diversi strumenti per analizzare i log e verificare se contengono informazioni critiche o se sono stati in qualche modo modificati (traccia di una potenziale intrusione esterna). Il comando **logwatch** fornisce un sistema di monitoring dei log personalizzabile ed estendibile che permette l'analisi del system log e la notifica via e-mail all'amministratore. Il suo file di configurazione è **/etc/log.d/conf/logwatch.conf**. Nella directory **/etc/log.d/conf/services** ci sono le configurazioni per i diversi servizi i cui log possono essere processati da **logwatch**, mentre nella directory **/etc/log.d/conf/logfiles** ci sono le configurazioni sui log relativi ai servizi e nella directory **/etc/log/scripts** ci sono i filtri predefiniti per analizzare diversi servizi e logfiles.

5.8 La libreria PAM

La libreria **Linux-PAM** (Linux Pluggable Authentication Modules) o più semplicemente **PAM** (v. <http://www.kernel.org/pub/linux/libs/pam>) è un'insieme di moduli o *shared library* (librerie condivise)¹⁸ che permettono all'amministratore del sistema di

Linux-PAM

¹⁸le shared library sono trattate in cap. 3.

scegliere il metodo per l'autenticazione degli utenti. In questo modo è possibile cambiare il meccanismo di autenticazione senza dover necessariamente ricompilare nessuna applicazione. La maggior parte dei sistemi GNU/Linux utilizzano tale libreria per l'autenticazione degli utenti.

È comunque necessario che le applicazioni che effettuano l'autenticazione degli accessi al sistema siano state adattate per funzionare con i Linux-PAM.

In generale, un'applicazione che gestisce l'autenticazione degli utenti deve essere realizzata (compilata) per uno specifico meccanismo di autenticazione, per esempio nei sistemi Unix-like tradizionali l'applicazione richiede l'inserimento di una password che viene verificata con quella memorizzata nello user account del sistema relativo all'utente in questione.

Sfortunatamente, l'incremento della velocità di calcolo delle macchine e l'introduzione delle reti hanno reso questo meccanismo di autenticazione, una volta sicuro, vulnerabile agli attacchi.

Con la libreria PAM si separa lo sviluppo del software che effettua l'autenticazione dal meccanismo di autenticazione stesso. Questo avviene per mezzo di una libreria di funzioni che una applicazione può utilizzare per richiedere l'autenticazione di un utente. La libreria PAM si compone dei file seguenti¹⁹

`/lib/libpam.so.*` file che permettono alle applicazioni di agganciarsi alla libreria PAM;

`/lib/security/pam*.so` i moduli che forniscono il meccanismo di autenticazione.

La libreria PAM è configurata tramite il file `/etc/pam.conf` (o una serie di file di configurazione contenuti nella directory `/etc/pam.d` – questo approccio è quello più recente) che fornisce le direttive per l'autenticazione degli accessi al sistema. I moduli PAM sono contenuti generalmente nella directory `/lib/security` e sono oggetti caricabili dinamicamente (v. man page `dlopen(3)`).

In genere l'applicazione che effettua la procedura di accesso al sistema esegue i seguenti passi:

1. Verifica che l'utente sia veramente chi dice di essere (autenticazione) – in genere questo viene effettuato richiedendo la password dell'utente e verificando che questa coincida con quella memorizzata nel relativo user account sul sistema;
2. Fornisce all'utente il servizio richiesto (in genere lancia una shell con i privilegi dell'utente).

Il passo 1 è quello delegato alla libreria PAM, che si prende quindi l'onere di verificare l'identità dell'utente (autenticazione) e di fornire il risultato all'applicazione che rimane l'interfaccia con l'utente.

L'applicazione (v. fig. 5.2) si interfaccia con la libreria PAM e non è necessario che conosca il meccanismo di autenticazione. La libreria PAM consulta il contenuto del file di configurazione appropriato e carica i moduli necessari.

Sono i moduli che si preoccupano di mettere in atto il meccanismo di autenticazione. I dati richiesti all'utente e le relative risposte, sono scambiati tra l'applicazione ed i moduli attraverso un opportuno meccanismo di dialogo fornito dalla libreria PAM.

L'associazione tra il meccanismo di autenticazione scelto e l'applicazione che necessita di autenticazione è effettuata con specifiche direttive nell'opportuno file di configurazione della libreria PAM.

¹⁹il simbolo asterisco '*' utilizzato all'interno dei nomi dei files sta ad indicare una qualunque sequenza di caratteri (v. cap. 3).

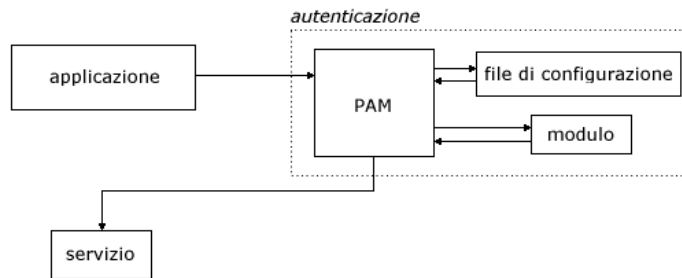


Figura 5.2: Schema di funzionamento della procedura di login con la libreria PAM.

5.8.1 Configurazione

Il file di configurazione della libreria PAM è in formato testo e contiene delle righe con la seguente sintassi

service-name module-type control-flag module-path args

dove

service-name è il nome del servizio associato alla riga corrente del file di configurazione.

In genere coincide con il nome dell'applicazione che fornisce il servizio (per esempio `ftpd`, `su`, ...). Esiste un particolare *service-name* riservato per definire il meccanismo di autenticazione di default: `other` (il meccanismo di autenticazione specificato in questo modo viene utilizzato soltanto nel caso che per un determinato servizio non ne sia stato specificato uno apposito).

module-type è l'indicazione del tipo di modulo. Può assumere uno dei seguenti valori possibili:

auth è il tipo di modulo che fornisce due aspetti dell'autenticazione degli utenti: prima verifica che l'utente sia chi dice di essere, richiedendo all'applicazione la password (o altro) per l'identificazione, poi assegna i privilegi all'utente;

account è il tipo di modulo che esegue la gestione degli account relativa agli aspetti che non si basano sull'autenticazione (per esempio la restrizione degli accessi ad un servizio basata sull'ora del giorno, sul numero massimo di utenti ai quali è permesso accedere, ...).

session è il tipo di modulo che si preoccupa di eseguire le operazioni necessarie subito prima che l'utente acceda ad un servizio o subito dopo che questi ha concluso il suo accesso (la registrazione delle operazioni in un file di log, il *mounting* di directory, ...);

password è il tipo di modulo richiesto per l'aggiornamento del *token* di autenticazione associato all'utente (tipicamente esiste un modulo per ogni tipo di autenticazione basato sul meccanismo "challenge/response").

control-flag è l'indicazione del tipo di reazione che la libreria PAM deve avere nel caso di successo o fallimento del modulo *module-path*. Poiché più moduli possono essere eseguiti in cascata, questo definisce l'importanza relativa di ogni modulo: l'applicazione non deve conoscere il dettaglio del successo/insuccesso dei singoli moduli, ma riceve un esito che la informa del successo/insuccesso dell'insieme di moduli ad essa associati. L'ordine di esecuzione dei moduli è quello in cui questi appaiono nel file di configurazione. Il *control-flag* può essere definito utilizzando una delle sintassi riportate di seguito

- una parola chiave che indica l'importanza associata all'esito di uno specifico modulo. Le possibili parole chiave, in ordine di importanza decrescente sono riportate di seguito

required

indica che il successo del modulo è richiesto affinché l'intero cascata dei moduli abbia successo. L'applicazione riceverà l'esito del fallimento soltanto quando tutti i moduli hanno eseguito il loro compito.

requisite

identico a **required**, tranne per il fatto che se uno dei moduli fallisce, tale risultato viene immediatamente ritornato all'applicazione (senza procedere con gli eventuali altri moduli in cascata). Questo è utilizzato in genere quando il sistema di autenticazione avviene attraverso un canale insicuro;

sufficient

indica che nel caso in cui nessuno dei moduli **required** precedenti abbia fallito, i moduli di questo livello non devono essere invocati. Il fallimento dei moduli di questo livello non è ritenuto fondamentale per l'esito dell'intera catena dei moduli;

optional

indica che l'esito del modulo non deve influire sull'esito dell'intera catena. Ovviamente, in caso di esito indefinito per gli altri moduli, questo ne influenza l'esito definitivo.

- una sintassi più complessa (più recente) che dà maggiore flessibilità sul controllo sul processo di autenticazione. La sintassi è la seguente:

[*value1=action1 value2=action2 ...*]

dove

valuen indica il valore di ritorno del metodo e può assumere uno dei seguenti valori:

```

success
open_err
symbol_err
service_err
system_err
buf_err
perm_denied
auth_err
cred_insufficient
authinfo_unavail
user_unknown
maxtries
new_authtok_reqd
acct_expired
session_err
cred_unavail
cred_expired
cred_err
no_module_data
conv_err
authtok_err
authtok_recover_err
authtok_lock_busy
authtok_disable_aging

```

```

try_again
ignore
abort
authtok_expired
module_unknown
bad_item
default

```

actionn indica il contributo dell'esito di un singolo modulo a quello dell'intera catena. Può essere un numero intero positivo o assumere uno dei seguenti valori:

ignore indica che l'esito del modulo non contribuisce all'esito dell'intera catena.

bad indica che il fallimento del modulo deve essere considerato determinante per il fallimento dell'intera catena.

die è equivalente a **bad** tranne per il fatto che l'esecuzione della catena dei moduli viene immediatamente interrotta e l'esito è ritornato direttamente all'applicazione.

ok indica che l'esito del modulo deve influire direttamente all'esito dell'intera catena (se il valore degli esiti dei precedenti moduli era PAM.SUCCESS, l'esito di questo modulo sovrascrive quello precedente, ma se il valore degli esiti dei precedenti moduli indicava il fallimento, l'esito di questo modulo non viene utilizzato per sovrascrivere quello precedente).

done è equivalente a **ok** tranne per il fatto che l'esecuzione della catena dei moduli viene immediatamente interrotta e l'esito è ritornato direttamente all'applicazione.

reset indica di cancellare l'esito della catena fino a quel punto e riavviare il calcolo dell'esito dal prossimo modulo in cascata.

Quando è utilizzato un numero positivo *m*, questo indica che i successivi *m* moduli dello stesso tipo devono essere saltati (non devono essere invocati).

Ognuna delle parole chiavi della sintassi più vecchia ha un'espressione equivalente nella sintassi più recente, come riportato in tab. 5.1

Vecchia	Nuova
required	[success=ok new_authtok_reqd=ok ignore=ignore default=bad]
requisite	[success=ok new_authtok_reqd=ok ignore=ignore default=die]
sufficient	[success=done new_authtok_reqd=done default=ignore]
optional	[success=ok new_authtok_reqd=ok default=ignore]

Tabella 5.1: Equivalenze tra vecchia e nuova sintassi di *control-flag*.

module-path è il nome del file (modulo) da invocare. Può essere un path assoluto (inizia per '/') o relativo alla directory `/lib/security` (default directory della libreria PAM).

args è una lista di argomenti che vengono passati a *module-path* quando invocato. Gli argomenti dipendono dal modulo specifico e quello non riconosciuti vengono ignorati dal modulo stesso ma l'errore viene comunque registrato nel system log.

In modo molto più flessibile rispetto al singolo file, la configurazione della libreria PAM può essere fatta attraverso l'utilizzo di files contenuti nella directory `/etc/pam.d`. In genere il file di configurazione riporta lo stesso nome del file (applicazione) che gestisce il servizio. La sintassi delle righe dei files di configurazione contenuti nella directory `/etc/pam.d` è la seguente

module-type control-flag module-path args

Come si può notare è analoga a quella relativa a `/etc/pam.conf` con la differenza che *service-name* non è presente, poiché il file in questione si riferisce già ad uno specifico servizio (applicazione).

La configurazione dei moduli PAM può essere gestita in due modi diversi, a discrezione del sistema (della distribuzione GNU/Linux):

- Se la directory `/etc/pam.d` esiste, vengono utilizzati i file di configurazione in essa contenuti, altrimenti viene utilizzato il file `/etc/pam.conf`;
- Si utilizzano in cascata il file relativo al modulo in questione contenuto nella directory `/etc/pam.d` e `/etc/pam.conf` in modo tale che le direttive presenti nel file di configurazione specifico (in `/etc/pam.d`) sovrascrivano quelle presenti nel file `/etc/pam.conf`.

È da notare che le parole chiavi specifiche delle direttive PAM, sono *case insensitive*, cioè non fa differenza tra lettere maiuscole e minuscole e le direttive possono essere espresse su più righe inserendo come carattere di continuazione tra una riga e l'altra il simbolo `'\'` (*backslash*).

Configurazione di default

Se un sistema deve essere sicuro, è opportuno che abbia delle direttive di default (*OTHER*) ragionabilmente sicure come segue:

```
#
# default; deny access
#
OTHER    auth      required    pam_deny.so
OTHER    account   required    pam_deny.so
OTHER    password  required    pam_deny.so
OTHER    session   required    pam_deny.so
```

Anche se le direttive sopra riportate sono necessarie per una configurazione ad elevata sicurezza, per sistemi mal configurati potrebbero non permettere l'accesso al sistema a nessun utente. È da tenere presente inoltre che il modulo `pam_deny.so` non registra nessun evento nel system log, quindi è difficile riuscire a capire dove sta il problema in caso di sistema mal configurato.

Per avere un avvertimento nel caso di applicazione non configurata, è conveniente aggiungere le seguenti linee in testa a quelle sopra indicate.

```
#
# default; wake up! This application is not configured
#
OTHER    auth      required    pam_warn.so
OTHER    password  required    pam_warn.so
```

Analogamente, per un sistema che utilizza il sistema di configurazione tramite i file contenuti nella directory `/etc/pam.d`, la corrispondente impostazione di default di quella

sopra riportata è la seguente (quello elencato è il contenuto del file di configurazione del servizio specifico)

```
#
# default configuration: /etc/pam.d/other
#
auth    required    pam_warn.so
auth    required    pam_deny.so
account required    pam_deny.so
password required    pam_warn.so
password required    pam_deny.so
session required    pam_deny.so
```

La configurazione per la gestione “standard” dei sistemi Unix-like è la seguente:

```
#
# default; standard UN*X access
#
OTHER   auth        required    pam_unix.so
OTHER   account     required    pam_unix.so
OTHER   password    required    pam_unix.so
OTHER   session     required    pam_unix.so
```

Generalmente questo rappresenta il punto di inizio per la maggior parte delle applicazioni. Per applicazioni specifiche potrebbe essere necessaria una configurazione leggermente diversa. Per esempio il servizio FTP²⁰ con l’abilitazione per l’accesso all’utente *anonymous* richiede un file di configurazione analogo al seguente

```
#
# ftpd; add ftp-specifics. These lines enable anonymous ftp over
# standard UN*X access (the listfile entry blocks access to
# users listed in /etc/ftpusers)
#
ftpd    auth        sufficient    pam_ftp.so
ftpd    auth        required      pam_unix_auth.so use_first_pass
ftpd    auth        required      pam_listfile.so \
onerr=succeed item=user sense=deny file=/etc/ftpusers
```

La seconda riga è necessaria poiché le direttive di default sono ignorate da un’applicazione (in questo caso `ftpd`) se ci sono delle righe in `/etc/pam.conf` che riguardano la specifica applicazione. È da notare anche l’uso di `use_first_pass` come argomento da passare al modulo PAM `pam_unix_auth.so` che indica al modulo di non richiedere la password, ma di prendere in considerazione quella già ottenuta dal modulo `pam_ftp.so`.

²⁰il servizio FTP è trattato in cap. 16.

Problemi di configurazione

Nel caso in cui i files di configurazione della libreria PAM andassero perduti, il sistema non permetterebbe l'accesso a nessuno. Per risolvere questo problema, è opportuno accedere al sistema in single user mode (run level 1) e correggere il problema. La sequenza dei comandi di seguito riportati può risolvere il problema riportando la configurazione della libreria PAM a quella “standard” dei sistemi Unix-like.

```
# cd /etc
# mv pam.conf pam.conf.orig
# mv pam.d pam.d.orig
# mkdir pam.d
# cd pam.d
```

quindi creare un file `/etc/pam.d/other` contenente le seguenti righe:

```
auth      required      pam_unix.so
account   required      pam_unix.so
password  required      pam_unix.so
session   required      pam_unix.so
```

5.9 La procedura di login

La procedura di accesso al sistema, ovvero la *procedura di login* è il meccanismo necessario agli utenti per accedere al sistema. Questa è il primo programma che incontra in genere l'utente che vuole accedere ad un sistema ed è anche l'applicazione più delicata dal punto di vista della sicurezza, infatti deve avere dei privilegi particolari per poter accedere agli user account e pertanto non dovrebbe mai “andare in crash” (*crashare*) cioè terminare in modo anomalo.

La procedura di login in genere è affidata al processo *login* che si preoccupa di richiedere la password all'utente e confrontarla con quella presente sul sistema. Nel caso in cui lo user account dell'utente in questione contenga la stessa password di quella specificata, l'utente ha il permesso di accedere al sistema. Il meccanismo di autenticazione può essere anche gestito in altro modo. Come è stato accennato precedentemente, affinché una persona possa accedere ad un sistema, è necessario che sul sistema stessa ci sia definito uno user account relativo alla persona che vuole accedere. Questo comporta la definizione di uno user account relativo alla persona da far accedere su ogni sistema a cui quella persona può accedere. Se ciò può essere fatto senza troppo sforzo su un paio di computer, nell'ambiente domestico, risulta molto laborioso in una rete di computer (dove solitamente le macchine sono più di un paio). In tal caso si rende necessaria una centralizzazione degli user account e quindi un sistema centralizzato per l'autenticazione degli utenti stessi. Cioè, si delega ad un solo computer della rete il compito di gestire gli account utenti di tutta la rete: gli utenti che possono accedere alla rete, lo potranno così fare da qualunque computer connesso alla rete stessa, senza che l'amministratore della rete provveda a definire tutti gli user accounts su ogni singolo computer. I meccanismi di gestione ed autenticazione centralizzata degli utenti più utilizzati nei sistemi Unix-like sono NIS e LDAP, ma questi verranno trattati nel cap. 16.

5.9.1 Accesso al terminale

Come accennato nel cap. 2, l'accesso al sistema da parte di un utente, per mezzo di un dispositivo che utilizza la porta seriale come un terminale, un terminale virtuale o un modem, viene gestito dal processo *getty* o *agetty* (molte distribuzioni utilizzano derivati di *getty* come *mingetty* poiché risultano molto meno pesanti per il sistema, ovvero occupano meno risorse, ma gestiscono soltanto i terminali virtuali). Questo si preoccupa di

1. aprire la linea di comunicazione terminale e impostarne le modalità di comunicazione.
2. visualizzare un testo di “benvenuto” (*issue* o *login banner*) all'utente che si accinge ad accedere al sistema (il testo visualizzato è, generalmente, quello contenuto nel file */etc/issue*) seguito dall'invito (prompt) ad inserire lo *username*;
3. ricevere lo *username* dell'utente che vuole accedere al sistema;
4. attivare il programma per l'autenticazione dell'utente (convenzionalmente si tratta di */bin/login*), fornendogli già lo *username* (sarà poi compito di *login* richiedere l'inserimento della *password*, per verificare che l'utente sia proprio chi dice di essere).

Il processo *getty* è lanciato con il comando *getty* (man page *getty(1)*).

??? man page *getty* ???

Comando: *getty*

Path: */etc/getty*

SINTASSI

\$ *getty* [*option*] [*line*] [*speed* [*type* [*lined*]]]

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di *getty*. Può assumere i seguenti valori:

- d *defaults.file*
(default */etc/conf.getty*);
- a ;
- h non forza l'hung up (disconnessione) della linea;
- r *delay*
indica di attendere *delay* secondi dopo aver ricevuto il primo carattere (-r0 indica di non attendere);
- t *timeout*
indica di terminare l'esecuzione se nessuno user name è stato inserito entro *timeout* secondi dalla visualizzazione del prompt di login;
- w *waitfor*
indica di attendere la stringa specificata da *waitfor* prima di continuare;
- c *gettydefs.file*
effettua un controllo sulla correttezza delle indicazioni presenti nel file *gettydefs.file*, secondo la sintassi del file */etc/gettydefs*;

line ;

speed è un'etichetta che si riferisce ad una riga del file */etc/gettydefs* che definisce la velocità iniziale (baud rate), il prompt, la velocità finale e le impostazioni della linea di comunicazione. Se non è specificato, o se *speed* si riferisce ad un'etichetta inesistente, viene considerata la prima riga del file */etc/gettydefs*. Se il file */etc/gettydefs* non esiste, viene impostata la una velocità iniziale di 9600 baud;

type è una stringa che identifica il tipo di terminale connesso alla linea di comunicazione (i tipi dei terminali riconosciuti dal sistema sono elencati nel file */etc/termcap*);

lined indica il comportamento da tenere in relazione alla linea di comunicazione (il valore di default è `LDISCO`);

All'avvio il processo *getty* tenta di leggere il contenuto del file `/etc/conf.getty.line` (nel caso in cui tale file non esista viene considerato al suo posto il file `/etc/conf.getty`) che dovrebbe contenere opportune indicazioni sul funzionamento di *getty*, secondo la seguente sintassi

name=value

dove *name* può assumere i seguenti valori

LOGIN in tal caso *value* rappresenta il nome del file (completo di path assoluto) lanciato in esecuzione per la gestione delle credenziali di accesso al sistema (per default viene lanciato `/bin/login`), al quale verrà passato (sulla riga di comando) lo username dell'utente;

INIT in tal caso *value* rappresenta una stringa di inizializzazione da inviare sulla linea di comunicazione prima che *getty* la utilizzi;

ISSUE in tal caso *value* rappresenta una stringa da visualizzare come testo di benvenuto al posto del testo contenuto nel file `/etc/issue`. Se *value* inizia con il carattere '/', allora *getty* visualizzerà come messaggio di benvenuto il testo contenuto nel file specificato da *value*;

CLEAR in tal caso *value* rappresenta una stringa del tipo 'YES' o 'NO' che indica rispettivamente se *getty* deve cancellare o meno lo schermo prima di visualizzare il messaggio di benvenuto (per default *getty* cancella lo schermo);

HANGUP in tal caso *value* rappresenta una stringa del tipo 'YES' o 'NO' che indica rispettivamente se *getty* deve disconnettere o meno la linea durante l'inizializzazione della stessa (per default *getty* disconnette la linea);

WAITCHAR in tal caso *value* rappresenta una stringa del tipo 'YES' o 'NO' che indica rispettivamente se *getty* deve attendere o meno un carattere sulla linea prima di continuare (per default *getty* non attende);

DELAY in tal caso *value* rappresenta il numero di secondi che *getty* deve attendere dopo aver ricevuto il primo carattere (per default *getty* non ha nessun tempo di attesa);

TIMEOUT in tal caso *value* rappresenta il numero di secondi dalla visualizzazione del prompt di login, entro il quale deve essere inserito uno username, altrimenti *getty* termina (per default *getty* attende l'inserimento di uno username per un tempo indefinito);

CONNECT in tal caso *value* rappresenta una stringa da inviare sulla linea di comunicazione per stabilire la connessione; ???

???

Il file di configurazione delle varie linee di comunicazione gestite da *getty* è `/etc/gettydefs` (man page `gettydefs(5)`). Questo è un file di testo che ha una struttura particolare: è composto da righe necessariamente seguite da una riga vuota (soltanto una); al solito, le righe che iniziano con il carattere cancelletto '#' sono ignorate. Le righe vuote non sono però ignorate: dopo una riga contenente una voce, si deve trovare esattamente una riga vuota; se dovessero essercene di più, la lettura del file verrebbe interrotta, ignorando di fatto le voci successive. Le righe sono suddivise in campi, secondo la sintassi seguente

label#initial_opt#final_opt#prompt#next_label




dove

label è un'etichetta che identifica univocamente la riga ed è il riferimento per il parametro *speed* del comando *getty*. Convenzionalmente, il nome da cui è composto *label* contiene un riferimento alla velocità relativa alla comunicazione. Generalmente tale campo è seguito immediatamente dal simbolo '#' in modo da non includere spazi superflui come facenti parte dell'etichetta stessa;

initial_opt è una stringa contenente una serie di opzioni che definiscono l'impostazione della linea, prima che questa venga utilizzata (v. *termio(7)*), a meno che *getty* abbia ricevuto l'indicazione di un tipo di terminale dalle cui caratteristiche estrapolare tale informazione;

final_opt è una stringa contenente una serie di opzioni (analogamente a *initial_opt*) che definiscono l'impostazione della linea subito prima che venga avviato il programma per l'autenticazione (in genere */bin/login*);

prompt è la stringa da utilizzare come invito (prompt) della procedura di accesso. Questa stringa non sostituisce il messaggio di benvenuto (issue), ma si aggiunge a questo, in coda. Generalmente si tratta semplicemente della stringa indquoteslogin:. La stringa in questione preserva gli spazi e può contenere sequenze di controllo che poi sono interpretate da *getty*. Generalmente, *getty* ammette l'uso degli stessi codici che possono essere inseriti nel file */etc/issue*;

next_label è l'etichetta che si riferisce ad una riga alternativa. Generalmente, quando *getty* riceve un carattere *break* (generato dalla pressione dei tasti   o ) , cerca di gestire la linea nel modo definito dalla riga identificata da questo nome. Per evitare problemi con gli spazi, questo nome inizia immediatamente dopo il simbolo '#'.

Il processo *agetty* è lanciato tramite il comando *agetty* (man page *agetty(8)*).
 ??? man page *agetty* ???

Il processo *mingetty* è lanciato dal comando *mingetty* (man page *mingetty(8)*) che è una versione minimale di *getty* per l'accesso al sistema attraverso console virtuali (virtual console). È particolarmente indicato per risparmiare memoria e non si appoggia a nessun file di configurazione, a parte il messaggio di benvenuto letto dal file */etc/issue*.

```
Comando: mingetty
Path: ???/mingetty

SINTASSI
# mingetty [option] vc
```

DESCRIZIONE

option è l'insieme delle opzioni che modificano il comportamento di *mingetty*. Può assumere i seguenti valori:

```
--noclear
    Non ripulisce lo schermo prima di avviare la procedura di accesso;
--long-hostname
    Visualizza il nome completo dell'elaboratore all'atto dell'attivazione
    della procedura di accesso;
```

vc è la virtual console da attivare.

5.9.2 login

Il programma *login* (man page *login(1)*) permette l'autenticazione per l'accesso di un utente al sistema.

??? man page *login* ???

In genere non dovrebbe essere possibile utilizzare direttamente tale comando, ma il suo lancio è appannaggio del programma di gestione del terminale (*getty* o derivati) dopo che questo ha ottenuto lo *username* dall'utente.

Al momento dell'avvio del processo *login*, questo richiede lo *username*, se questo non era già stato passato come parametro sulla riga di comando al lancio di *login*, e la relativa *password*.

La password inserita viene cifrata e confrontata con quella presente nello user account relativo all'utente identificato dallo *username* inserito. Se le due password coincidono, l'utente è autenticato e quindi abilitato all'accesso al sistema.

La cifratura della password avviene tramite la funzione di libreria *crypt* (man page *crypt(5)*), aggiungendo in testa alla stringa da cifrare 2 caratteri ("salt").

Al termine della procedura di autenticazione *login* visualizza

- la data e l'ora dell'ultimo accesso dell'utente in questione (se esiste il file */var/log/lastlog*);
- l'avviso della presenza di posta elettronica, se esistono messaggi non ancora letti (la casella di posta elettronica di un utente è costituita dal file */var/mail/optusername*);
- il messaggio del giorno (contenuto nel file */etc/motd*).

Esistono dei file che il processo *login* prende in considerazione per determinare l'accesso al sistema.

Affinché il superuser possa accedere al sistema, è necessario che il terminale (TTY) dal quale si intende accedere sia elencato all'interno del file */etc/securetty*, cioè sia considerato un terminale "sicuro" (la procedura di accesso da tale sistema è ritenuta non contraffatta). I tentativi di accesso come superuser che provengono da terminali non elencati nel file precedentemente indicato, vengono annotati all'interno del *system log*. Un esempio del contenuto del file */etc/securetty* è riportato di seguito

```
vc/1
vc/2
vc/3
vc/4
vc/5
vc/6
vc/7
vc/8
vc/9
vc/10
vc/11
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
tty9
tty10
tty11
```

I nomi dei terminali vengono indicati facendo riferimento ai file di dispositivo²¹ relativi, senza l'indicazione del prefisso */dev/*. L'esempio precedente mostra un elenco di terminali che comprende i terminali virtuali o console virtuali (*vc/1*, ..., *vc/11*), ed i terminali standard (*tty1*, ..., *tty11*). Nel caso in cui il file */etc/securetty* non esista, il superuser può accedere al sistema da qualunque terminale.

²¹i file di dispositivo sono trattati nel cap. 3.

Nei sistemi in cui non viene utilizzata la libreria PAM, il file `/etc/usertty`, se esiste, specifica ulteriori restrizioni di accesso al sistema per i vari utenti. Il file consiste di una serie di sezioni di tre tipi:

CLASSES definisce le classi dei sistemi e terminali;

GROUPS definisce i terminali ed i sistemi accessibili per gruppo di utenti;

USERS definisce i terminali ed i sistemi accessibili per ogni utente.

Per ulteriori dettagli v. `login(1)`.

Se esiste il file `~/.hushlogin`, viene eseguito un accesso silenzioso (hush), nel senso che vengono disattivati:

- la visualizzazione della data e dell'ora dell'ultimo accesso effettuato da parte dell'utente che ha il file `.hushlogin` nella sua home directory;
- il controllo per la presenza di messaggi di posta elettronica;
- la visualizzazione del messaggio del giorno.

Se esiste il file `/etc/nologin`, `login` ne visualizza il contenuto sullo schermo e non viene consentito l'accesso al sistema a nessun utente tranne al superuser. Ciò può servire per impedire l'accesso al sistema, tipicamente quando si intende chiuderlo o mantenerlo.

L'accesso al sistema viene registrato nel file `/var/log/lastlog` l'indicazione dell'utente, del giorno e dell'ora a cui è avvenuto l'accesso e il terminale da cui l'accesso è stato effettuato. Tale file raccoglie le informazioni relative all'ultimo accesso al sistema effettuato da ogni utente. Il formato del file non è leggibile direttamente, ma può essere letto tramite il comando `lastlog` (v. sez. 5.10).

Al termine della procedura di accesso viene avviata la default shell indicata nello user account dell'utente. Nel caso in cui questa non sia stata indicata, viene eseguito il file `/bin/sh`, ovvero la shell di default generica. Se nello user account non è indicata la home directory dell'utente, viene utilizzata la root directory `/`.

5.10 Comandi utili

Si possono visualizzare gli username degli utenti che stanno accedendo al sistema con il comando `users` (man page `users(1)`) oppure con il comando `w` (man page `w(1)`) che in più mostra anche informazioni sulle loro attività.

Comando: `users`

Path: `??/users`

SINTASSI

`$ users [file]`

DESCRIZIONE

file ;

???

Comando: `w`

Path: `??/w`

SINTASSI

`$ w [username]`

DESCRIZIONE

username ;

???

Un comando analogo è `who` (man page `who(1)`).

Comando: `who`

Path: `???`/`who`

SINTASSI

`$ who [option] [file] [am i]`

DESCRIZIONE

option ;

???

La visualizzazione dell'ultimo accesso al sistema effettuato da ogni utente può essere effettuata con il comando `lastlog` (man page `lastlog(8)`)

Comando: `lastlog`

Path: `???`/`lastlog`

SINTASSI

`# lastlog [-u username] [-t days]`

DESCRIZIONE

`-u` specifica di visualizzare l'ultimo accesso al sistema dell'utente il cui username è *username*;

`-t` specifica di visualizzare gli ultimi accessi al sistema effettuati da meno di *days* giorni.

???

Possono essere visualizzati anche tutti gli accessi al sistema memorizzati nel file `/var/log/wtmp` con il comando `last` (man page `last(1)`).

Comando: `last`

Path: `???`/`last`

SINTASSI

`$ last [option]`

DESCRIZIONE

option ;

???

5.11 Riferimenti

???

Capitolo 6

Il kernel ed i processi

“Chi cavalca la tigre non può scendere.”
– (Proverbio cinese)

In questo capitolo viene trattato il kernel di GNU/Linux, ovvero Linux, ed i passi necessari alla sua compilazione, la gestione dei processi, dal lancio alla terminazione degli stessi e la comunicazione tra i processi.

6.1 Il kernel

Senza il software, un computer è soltanto un mucchio di circuiti elettronici (hardware). Quindi si può pensare all’hardware come al corpo di un computer, ovvero la sua parte tangibile, ed al software come l’anima dello stesso, ciò che gli dà vita, lo fa funzionare.

Come già anticipato nel cap. 1, il **kernel** è un insieme di programmi che coordinano i vari processi e fornisce a questi le routine necessarie per accedere all’hardware. Esso effettua un’astrazione del sistema fisico, presentandone uno virtuale. La maggior parte dei computer può funzionare con più di un sistema operativo ed ognuno di questi può avere un’interfaccia utente diversa. Il *kernel* rappresenta quindi il costituente fondamentale del sistema operativo sebbene esso sarebbe inutile senza le librerie di sistema o l’interfaccia utente (shell¹).

Il kernel in genere si occupa principalmente di gestire

- la memoria;
- i processi;
- l’accesso all’hardware del sistema;
- il file system;
- l’accesso alle risorse da parte dei processi.

Se le risorse del sistema fossero infinite, molte delle operazioni svolte dal sistema operativo potrebbero essere considerate inutili, superflue. Una delle caratteristiche principali di un sistema operativo è quella di utilizzare una contenuta quantità di memoria (memoria fisica) ma far “vedere” alle applicazioni una quantità di memoria molto più grande (teoricamente illimitata): la memoria virtuale. Il meccanismo si basa sul fatto che la memoria viene suddivisa dal kernel in parti più piccole e più “maneggevoli”, dette *pagine*, che vengono scaricate sulla memoria di massa (hard disk) quando non servono più e ricaricate da quest’ultima in memoria centrale quando se ne presenta la necessità (tecnica dello swap²).

I programmi che controllano i vari dispositivi hardware (o periferiche) sono detti **device driver** (o più semplicemente *driver*) e costituiscono la parte principale del ker-

¹la shell sarà trattata nel cap. 7.

²v. sez. 3.16.

nel di GNU/Linux. Ad esempio, il driver relativo al controllo del disco ATA (IDE) è responsabile della lettura/scrittura dei dati al disco (spostamento delle testine, lettura dei blocchi da disco, gestione della cache del disco, ...). In genere il device driver dipende fortemente dal tipo di hardware che deve controllare, per questo per particolari dispositivi è necessario avere il relativo driver. Come altre parti del sistema operativo, essi operano in un ambiente privilegiato, ovvero hanno privilegi particolari per l'accesso all'hardware e pertanto un loro malfunzionamento può causare dei problemi al sistema, fino a bloccarlo irrimediabilmente (l'unica soluzione in tal caso è il riavvio della macchina).

I sistemi Unix-like basano il loro funzionamento sulla distinzione fra *kernel-space* e *user-space*. Questi sono due ambienti distinti nei quali vengono eseguiti i processi: il **kernel-space** è quello riservato all'esecuzione del kernel, mentre lo **user-space** è quello riservato a tutti gli altri processi. L'accessibilità completa all'hardware è possibile soltanto in kernel space, mentre i processi che sono eseguiti in user-space devono necessariamente utilizzare le routine messe a disposizione del kernel per poter accedere all'hardware.

kernel-space
user-space

6.1.1 Le versioni

La release del kernel di GNU/Linux ha la stessa forma utilizzata dal software sviluppato per tale sistema, quella riportata in sez. 1.9, cioè

MajorNumber.MinorNumber.RevisionNumber[-Build]

con la particolarità che se il *MinorNumber* è pari, il kernel è considerato *stabile*, pronto per essere utilizzato su sistemi in produzione; se è dispari è considerato *in fase di sviluppo* e quindi può essere utilizzato con molta cautela (certe funzionalità potrebbero bloccare il sistema). Le release stabili derivano sempre da quelle in sviluppo il cui *MinorNumber* è diminuito di 1. Ad esempio la release stabile 2.4.X deriva dalla release di sviluppo 2.3.X. In genere nel kernel stabile si tende soltanto alla manutenzione ed alla correzione di errori (bug), ma non all'introduzione di nuove funzionalità, le quali vengono sviluppate nella versione del kernel in fase di sviluppo che, una volta reso stabile, diverrà la release stabile successiva.

In genere nei sistemi GNU/Linux il kernel è costituito soltanto da un file */boot/loader/linuz* (dove *loader* è una directory con il nome del boot loader utilizzato). Questo file è anche detto **immagine** del kernel. In questo modo tutte le funzionalità del kernel risiedono all'interno di un unico file e pertanto si parla di **kernel monolitico**. Esiste però la possibilità di incorporare le funzionalità specifiche dall'*immagine* del kernel in altri files detti **moduli**. Si parla in tal caso di **kernel modulare**. In questo modo il kernel gestisce l'utilizzo dei moduli stessi: quando è necessario carica l'opportuno modulo in memoria e lo scarica quando non serve più. Un kernel monolitico ha il vantaggio di essere compatto e contenuto tutto in un file, ma allo stesso tempo è rigido e non consente di liberare risorse quando queste non servono. Per contro il kernel modulare è più dispersivo e difficile da gestire (fa uso di altri file oltre a quello del kernel di base) ma anche più flessibile di quello monolitico (può liberare le risorse scaricando dalla memoria i moduli non utilizzati).

immagine
kernel monolitico

moduli
kernel modulare

Al momento dell'inizio della scrittura del presente testo la versione più recente del kernel Linux è la 2.4.

6.1.2 La compilazione

Essendo un programma che deve essere eseguito come programma di base (senza potersi avvalere dell'ausilio di altri programmi già in esecuzione) è necessario che il kernel sia un programma comprensibile al computer, ovvero in linguaggio macchina. Vista la complessità della scrittura di un programma come il kernel, risulterebbe ancora più complesso scriverlo direttamente in linguaggio macchina. Per questo, anche se parti più critiche sono scritte in assembly (un linguaggio di programmazione che molto si

avvicina al linguaggio macchina), la maggior parte del kernel è scritto in un linguaggio di programmazione a più alto livello (ma non troppo): il C. Dunque l'immagine del kernel è il risultato di una compilazione dei suoi sorgenti.

Poiché il kernel deve offrire anche performance elevate, non sarebbe pensabile di utilizzare un programma interpretato, che per la presenza dell'interpretazione delle istruzioni avrebbe performance decisamente inferiori all'analogo compilato.

Poiché GNU/Linux è un sistema opensource, i sorgenti del kernel sono distribuiti insieme al sistema ed inoltre, poiché si tratta di free software è possibile modificare e ricompilare il kernel – sì, proprio così! – adattandolo al proprio hardware e personalizzandolo secondo le proprie specifiche esigenze.

Poiché questa operazione potrebbe compromettere il funzionamento dell'intero sistema (si ricordi che il kernel è il nucleo di base del sistema) si consiglia di farlo seguendo opportune indicazioni che comunque permetteranno di ripristinare facilmente la situazione precedente.

Per la compilazione del kernel è necessario avere installato sul sistema i tool per la compilazione, cioè i pacchetti necessari per compilare i sorgenti C (compilatore e librerie), nonché i file sorgenti del kernel.

La procedura di seguito descritta non modificherà né distruggerà il kernel già presente sul sistema, ma si limiterà a produrre un altro kernel. In un sistema GNU/Linux è possibile infatti avere più immagini del kernel sul filesystem e scegliere in fase di boot quale kernel avviare.

1. Reperimento dei file sorgenti del kernel.

I file sorgenti del kernel di GNU/Linux possono essere reperiti in vari modi. Bisogna fare molta attenzione a reperire i sorgenti del kernel opportuno, poiché, in genere, le varie distribuzioni di GNU/Linux modificano il kernel di base, o come si dice in gergo il *vanilla kernel*, secondo le proprie esigenze. Ad esempio, i sorgenti del kernel di una distribuzione RedHat sono contenuti in pacchetti rpm presenti nella directory RedHat/RPMS sui CD della distribuzione, che viene generalmente montata come `/mnt/cdrom/RedHat/RPMS`, e possono quindi essere installati sul sistema con i seguenti comandi

```
# cd /mnt/cdrom/RedHat/RPMS
# rpm -i kernel-headers*.rpm
# rpm -i kernel-source*.rpm
# rpm -i dev86*.rpm
# rpm -i bin86*.rpm
```

I file `bin86*.rpm` sono richiesti soltanto per i vecchi sistemi GNU/Linux.

2. Copia del file di configurazione

Se si vuole compilare un nuovo kernel con le stesse impostazioni di uno eventualmente già compilato in precedenza, si può copiare il file di configurazione.

```
# cd /usr/src/linux
# cp ../linux-old-tree/.config .
```

o alternatively utilizzare il comando `make oldconfig` che risponderà a tutte le varie domande di `make`, con i valori contenuti nel file `.config` presente nella working directory.

3. Pulire l'ambiente di compilazione

```
# cd /usr/src/linux
# make clean
# make mrproper
```

Mr Proper è un prodotto finlandese per pulizie, che probabilmente ha ispirato Linus Torvalds.

Comando: **make**
 Path: **/usr/bin/make**
 SINTASSI
\$ make [option]

DESCRIZIONE

option indica la modalità di funzionamento di **make**. Può assumere i seguenti valori

??? ???;

???

4. Configurare i parametri di compilazione

La configurazione dei parametri di configurazione può essere effettuata sia da shell che dall'ambiente grafico.

Per avviare l'ambiente grafico si può utilizzare il comando

```
# startx
```

Quindi, dopo aver aperto un terminale (dall'ambiente grafico), si possono digitare i seguenti comandi

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

A questo punto verrà visualizzata una finestra per l'impostazione dei parametri di compilazione del kernel.

Alternativamente, da una shell non grafica, si possono configurare i parametri di compilazione con il comando

```
# make menuconfig
```

A questo punto apparirà sullo schermo una sorta di finestra per l'impostazione dei parametri di compilazione del kernel.

Se tale comando non funziona, si deve utilizzare la sua versione più vecchia, con il comando

```
# make config
```

che visualizzerà una serie di domande che servono per impostare i parametri di compilazione del kernel.

In generale è consigliabile, dove possibile (specialmente nel caso di configurazione dall'ambiente grafico), impostare la configurazione come quella precedente, cioè con i valori contenuti nel file **/usr/src/linux/.config** (dall'ambiente grafico premere il pulsante Load Configuration from File), per poi eventualmente modificare soltanto le impostazioni che interessano.

- È di fondamentale importanza impostare il corretto tipo di CPU presente sul sistema per poter ottenere un'immagine del kernel funzionante sul proprio sistema.
- Select SMP support - whether single CPU or multiple CPUs

- Filesystems - È consigliabile impostare Windows95 Vfat, MSDOS, NTFS come parte del kernel e non come moduli esterni.
- Abilitare il supporto per i moduli (Loadable kernel modules support) in modo da far così gestire dinamicamente al kernel il caricamento/scaricamento dalla memoria dei driver necessari (on-the-fly). Per maggiori dettagli si consiglia di consultare le seguenti man page: `lsmod(?)`, `insmod(?)`, `rmmmod(?)`, `depmod(?)` e `modprobe(?)` (contenute nei pacchetti `modutils*`).

Alla fine dell'operazione di impostazione della configurazione scegliere il salvataggio dei dati, in modo tale che la configurazione venga scritta nel file `/usr/src/linux/.config`.

5. Dipendenze

```
# make dep
```

6. Assegnare un nome all'immagine del kernel

È consigliabile assegnare un nome all'immagine del kernel da creare in modo tale che questa sia differente da altre eventualmente presenti sul sistema e non interferisca pertanto con le altre. Una buona idea è quella di identificare l'immagine del kernel con la sua versione. Per far questo è sufficiente modificare il contenuto del file `/usr/src/linux/Makefile` con un qualsiasi text editor (in questo caso è utilizzato `vi`)

```
# cd /usr/src/linux
# vi Makefile
```

e modificare la riga che contiene la direttiva `EXTRAVERSION`. Per esempio cambiare la riga

```
EXTRAVERSION = -2.4.2XXXXXXX
```

con

```
EXTRAVERSION = -2.4.15MyKernel.10Jan2003
```

7. Moduli

Questo passo è necessario soltanto se è stato abilitato il supporto per i moduli (Loadable module support) al passo 3, altrimenti si incorrerà in un messaggio di errore analogo a “unresolved symbols” durante la fase di caricamento del kernel.

```
# cd /usr/src/linux
# make modules
# make modules_install
```

Questi comandi copieranno anche i moduli nella directory `/lib/modules`. Si può comunque impostare il percorso di ricerca dei moduli utilizzato da `insmod` nel file `/etc/modules.conf`.

8. Compilazione dei sorgenti

Prima di procedere con la compilazione dei sorgenti si può visualizzare il contenuto del file `/usr/src/linux/arch/i386/config.in` che contiene delle informazioni sulla compilazione dei sorgenti del kernel. Quindi si procede con la compilazione

```
# cd /usr/src/linux
# nohup make bzImage &
```

Il comando `nohup` (man page `nohup(1)`) lancia in esecuzione un processo immune al segnale `SIGHUP`.

```
Comando: nohup
Path: /usr/bin/nohup
SINTASSI
# nohup [option] [command] [args]
```

DESCRIZIONE

option indica la modalità di funzionamento di `nohup`. Può assumere i seguenti valori

```
--help visualizza un aiuto sommario di nohup;
--version visualizza la versione di nohup;
```

command è il comando che deve essere lanciato ed il cui relativo processo deve ignorare i segnali `SIGHUP`. Al relativo processo viene inoltre assegnata una priorità aumentata di un valore 5 e pertanto il sistema continuerà la sua esecuzione anche se l'utente si scollega (effettua un *logout*). Se il canale di output è un terminale (standard output), questo, assieme al canale di error, viene automaticamente rediretto nel file `nohup.out` o, nel caso in cui non sia possibile, nel file `~/nohup.out`;

args sono gli eventuali argomenti da passare al comando `command`;

Per controllare il progredire dell'ultimo comando impartito (in esecuzione in background per la presenza di '&'), si può utilizzare il seguente comando

```
# tail -f nohup.out
```

Alla fine di queste operazioni l'immagine del kernel sarà contenuta nel file `/usr/src/linux/arch/i386/boot`

9. Copia dell'immagine del kernel in `/boot`

Il file contenente l'immagine del kernel deve essere copiato nella directory `/boot` altrimenti non potrà essere caricato durante la fase di boot del sistema.

```
# cp /usr/src/linux/arch/i386/boot/bzImage /boot/bzImage.myker.26mar2001
```

Per i kernel più recenti (versione 2.4.0 o superiore) possono essere creati automaticamente i file `/boot/initrd*.img` ed inserite le opportune direttive nel file di configurazione del boot loader, con i comandi seguenti

```
# make modules_install
# make install
```

10. Configurazione del boot loader

LILO

GRUB

???

11. Reboot

Riavviare la macchina con il comando

```
# shutdown -r now
```

Alla presentazione dell'interfaccia del boot loader selezionare l'avvio del nuovo kernel. se tutto funziona correttamente il lavoro è finito, altrimenti si può riavviare nuovamente il sistema e far avviare al boot loader il kernel precedente (si ricordi che il kernel precedente è rimasto intatto), in modo da poter utilizzare il sistema ed eventualmente rivedere i passi precedentemente effettuati per "aggiustare" l'immagine del nuovo kernel.

12. Creare un floppy di avvio

Se il nuovo kernel funziona, si può creare un floppy per l'avvio del sistema col nuovo kernel.

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

v. anche `mkbootdisk(?)`.

13. Pulire l'ambiente di compilazione

Una volta completato il tutto, si può ripulire l'ambiente di compilazione liberando spazio sul filesystem.

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

???

6.1.3 L'avvio

Quando viene avviato, il kernel effettua un controllo dell'hardware del sistema, durante il quale visualizza una serie di messaggi sullo schermo come quelli riportati di seguito (alcune parti variano dipendentemente dall'hardware presente sul sistema, altre sono uguali sostanzialmente per tutti i sistemi GNU/Linux):

```
Linux version 2.4.18-18.8.0 (bhcompile@daffy.perf.redhat.com) (gcc version 3.2 2
0020903 (Red Hat Linux 8.0 3.2-7)) #1 Thu Nov 14 00:10:29 EST 2002
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 00000000000009fc00 (usable)
 BIOS-e820: 00000000000009fc00 - 0000000000000a0000 (reserved)
 BIOS-e820: 0000000000000f0000 - 000000000000100000 (reserved)
 BIOS-e820: 000000000000100000 - 00000000017ffd0000 (usable)
 BIOS-e820: 00000000017ffd0000 - 00000000017fff0000 (ACPI data)
 BIOS-e820: 00000000017fff0000 - 000000000180000000 (ACPI NVS)
 BIOS-e820: 00000000ffff0000 - 0000000100000000 (reserved)
OMB HIGHMEM available.
383MB LOWMEM available.
On node 0 totalpages: 98301
zone(0): 4096 pages.
zone(1): 94205 pages.
zone(2): 0 pages.
Kernel command line: ro root=LABEL=/
Initializing CPU#0
Detected 501.114 MHz processor.
Speakup v-1.00 CVS: Tue Jun 11 14:22:53 EDT 2002 : initialized
Console: colour VGA+ 80x25
Calibrating delay loop... 996.88 BogoMIPS
Memory: 382252k/393204k available (1314k kernel code, 8388k reserved, 989k data,
172k init, 0k highmem)
Dentry cache hash table entries: 65536 (order: 7, 524288 bytes)
Inode cache hash table entries: 32768 (order: 6, 262144 bytes)
Mount cache hash table entries: 8192 (order: 4, 65536 bytes)
ramfs: mounted with options: <defaults>
ramfs: max_pages=48053 max_file_pages=0 max_inodes=0 max_dentries=48053
Buffer cache hash table entries: 32768 (order: 5, 131072 bytes)
Page-cache hash table entries: 131072 (order: 7, 524288 bytes)
CPU: Before vendor init, caps: 0387f9ff 00000000 00000000, vendor = 0
CPU: L1 I cache: 16K, L1 D cache: 16K
```

```

CPU: L2 cache: 512K
CPU: After vendor init, caps: 0387f9ff 00000000 00000000 00000000
CPU serial number disabled.
Intel machine check architecture supported.
Intel machine check reporting enabled on CPU#0.
CPU:      After generic, caps: 0383f9ff 00000000 00000000 00000000
CPU:      Common caps: 0383f9ff 00000000 00000000 00000000
CPU: Intel Pentium III (Katmai) stepping 02
Enabling fast FPU save and restore... done.
Enabling unmasked SIMD FPU exception support... done.
Checking 'hlt' instruction... OK.
POSIX conformance testing by UNIFIX
mtrr: v1.40 (20010327) Richard Gooch (rgooch@atnf.csiro.au)
mtrr: detected mtrr type: Intel
PCI: PCI BIOS revision 2.10 entry at 0xf0720, last bus=1
PCI: Using configuration type 1
PCI: Probing PCI hardware
PCI: Using IRQ router PIIX [8086/7110] at 00:04.0
Limiting direct PCI/PCI transfers.
isapnp: Scanning for PnP cards...
isapnp: No Plug & Play device found
speakup: initialized device: /dev/synth, node (MAJOR 10, MINOR 25)
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
apm: BIOS version 1.2 Flags 0x0b (Driver version 1.16)
Starting kswapd
VFS: Diskquotas version dqot_6.5.0 initialized
pty: 2048 Unix98 ptys configured
Serial driver version 5.05c (2001-07-08) with MANY_PORTS MULTIPORT SHARE_IRQ SER
IAL_PCI ISAPNP enabled
ttyS0 at 0x03f8 (irq = 4) is a 16550A
ttyS1 at 0x02f8 (irq = 3) is a 16550A
Real Time Clock Driver v1.10e
block: 736 slots per queue, batch=184
Uniform Multi-Platform E-IDE driver Revision: 6.31
ide: Assuming 33MHz system bus speed for PIO modes; override with idebus=xx
PIIX4: IDE controller on PCI bus 00 dev 21
PIIX4: chipset revision 1
PIIX4: not 100% native mode: will probe irqs later
    ide0: BM-DMA at 0xb800-0xb807, BIOS settings: hda:DMA, hdb:DMA
    ide1: BM-DMA at 0xb808-0xb80f, BIOS settings: hdc:DMA, hdd:DMA
hda: IBM-DJNA-370910, ATA DISK drive
hdb: MAXTOR 6L060J3, ATA DISK drive
hdc: MATSHITADVD-ROM SR-8583, ATAPI CD/DVD-ROM drive
hdd: SAMSUNG CD-R/RW SW-408B, ATAPI CD/DVD-ROM drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
ide1 at 0x170-0x177,0x376 on irq 15
blk: queue c03afd84, I/O limit 4095Mb (mask 0xffffffff)
blk: queue c03afd84, I/O limit 4095Mb (mask 0xffffffff)
hda: 17803440 sectors (9115 MB) w/1966KiB Cache, CHS=1108/255/63, UDMA(33)
blk: queue c03afec0, I/O limit 4095Mb (mask 0xffffffff)
blk: queue c03afec0, I/O limit 4095Mb (mask 0xffffffff)
hdb: 117266688 sectors (60041 MB) w/1819KiB Cache, CHS=7299/255/63, UDMA(33)
ide-floppy driver 0.99.newide
Partition check:
    hda: hda1 hda2 hda3
    hdb: hdb1 hdb2 hdb3 hdb4 < hdb5 hdb6 >
Floppy drive(s): fd0 is 1.44M
FDC 0 is a post-1991 82077
NET4: Frame Diverter 0.46
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize

```

```

ide-floppy driver 0.99.newide
md: md driver 0.90.0 MAX_MD_DEVS=256, MD_SB_DISKS=27
md: Autodetecting RAID arrays.
md: autorun ...
md: ... autorun DONE.
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 4096 buckets, 32Kbytes
TCP: Hash tables configured (established 32768 bind 65536)
Linux IP multicast router 0.06 plus PIM-SM
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
RAMDISK: Compressed image found at block 0
Freeing initrd memory: 126k freed
VFS: Mounted root (ext2 filesystem).
Journalled Block Device driver loaded
kjournald starting. Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
Freeing unused kernel memory: 172k freed
usb.c: registered new driver usbdevfs
usb.c: registered new driver hub
usb-uhci.c: $Revision: 1.275 $ time 00:32:08 Nov 14 2002
usb-uhci.c: High bandwidth mode enabled
PCI: Found IRQ 9 for device 00:04.2
PCI: Setting latency timer of device 00:04.2 to 64
usb-uhci.c: USB UHCI at I/O 0xb400, IRQ 9
usb-uhci.c: Detected 2 ports
usb.c: new USB bus registered, assigned bus number 1
hub.c: USB hub found
hub.c: 2 ports detected
usb-uhci.c: v1.275:USB Universal Host Controller Interface driver
usb.c: registered new driver hiddev
usb.c: registered new driver hid
hid-core.c: v1.8.1 Andreas Gal, Vojtech Pavlik <vojtech@suse.cz>
hid-core.c: USB HID support drivers
mice: PS/2 mouse device common for all mice
hub.c: USB new device connect on bus1/1, assigned device number 2
usb-uhci.c: interrupt, status 2, frame# 521
EXT3 FS 2.4-0.9.18, 14 May 2002 on ide0(3,3), internal journal
Adding Swap: 819304k swap-space (priority -1)
usb_control/bulk_msg: timeout
kjournald starting. Commit interval 5 seconds
EXT3 FS 2.4-0.9.18, 14 May 2002 on ide0(3,2), internal journal
EXT3-fs: mounted filesystem with ordered data mode.
kjournald starting. Commit interval 5 seconds
EXT3 FS 2.4-0.9.18, 14 May 2002 on ide0(3,65), internal journal
EXT3-fs: mounted filesystem with ordered data mode.
kjournald starting. Commit interval 5 seconds
EXT3 FS 2.4-0.9.18, 14 May 2002 on ide0(3,66), internal journal
EXT3-fs: mounted filesystem with ordered data mode.
usb_control/bulk_msg: timeout
usb-uhci.c: interrupt, status 3, frame# 331
input0: USB HID v1.00 Joystick [Microsoft SideWinder Force Feedback 2 Joystick]
on usb1:2.0
parport0: PC-style at 0x378 [PCSPPP,TRISTATE]
ohci1394: pci_module_init failed
ip_tables: (C) 2000-2002 Netfilter core team
8139too Fast Ethernet driver 0.9.25
PCI: Found IRQ 5 for device 00:0a.0
PCI: Setting latency timer of device 00:0a.0 to 64
divert: allocating divert_blk for eth0
eth0: RealTek RTL8139 Fast Ethernet at 0xd8858000, 00:e0:4c:39:68:d7, IRQ 5
eth0: Identified 8139 chip type 'RTL-8139C'

```

```

eth0: Setting 100mbps full-duplex based on auto-negotiated partner ability 45e1.
Creative EMU10K1 PCI Audio Driver, version 0.19, 00:36:43 Nov 14 2002
PCI: Found IRQ 10 for device 00:0b.0
PCI: Setting latency timer of device 00:0b.0 to 64
emu10k1: EMU10K1 rev 5 model 0x20 found, IO at 0xa800-0xa81f, IRQ 10
ac97_codec: AC97 codec, id: 0x5452:0x4123 (TriTech TR A5)
ide-floppy driver 0.99.newide
hdc: ATAPI 32X DVD-ROM drive, 512kB Cache
Uniform CD-ROM driver Revision: 3.12
hdd: ATAPI 32X CD-ROM CD-R/RW drive, 2048kB Cache

[...]
```

La prima riga mostra la versione del kernel (in questo caso 2.4.18) del compilatore interno (gcc) e della versione del sistema (distribuzione). Le righe successive indicano la quantità di memoria presente sul sistema, gli argomenti passati al kernel³ dal boot loader (in questo caso `ro root=LABEL=`), la frequenza di clock a cui lavora la CPU (in questo caso 501.114 MHz), le proprietà della console (colori, 80 colonne per 25 righe) ed un indicatore della potenza di calcolo della CPU (in questo caso 996.88 BogoMIPS – più elevato è il numero di BogoMIPS, più “potente” è la CPU). Poi si prosegue con il riconoscimento dei bus (PCI e USB) e delle periferiche (dischi e controller IDE, scheda audio, scheda di rete) ad essi connesse ed il controllo del filesystem (EXT3).

Il comando `dmesg` (man page `dmesg(8)`) permette di visualizzare esattamente gli stessi messaggi che il kernel mostra al suo avvio.

```
??? man page dmesg(8) ???
```

6.1.4 Gli argomenti

Generalmente Linux (il kernel di GNU/Linux) riesce a riconoscere automaticamente i dispositivi presenti sul sistema, ma può essere necessario specificare opportune impostazioni per i vari dispositivi. Linux accetta un elevato numero di argomenti (parametri) specificabili sulla riga di comando, che gli possono essere passati direttamente dal boot loader (v. sez. 2.2).

Gli argomenti sono specificati per mezzo della seguente sintassi

```
keyword[=value1[, value2[, ...[, value11]]]]
```

dove *keyword* è un nome che indica univocamente la parte del kernel per i quali sono stati specificati i valori *value1*, ..., *value11* (al massimo possono essere specificati 11 valori per ogni argomento). Di seguito sono illustrati gli argomenti riconosciuti dal kernel

`root=device`

specifica il dispositivo (filesystem) da montare come root directory (per default viene considerato il dispositivo sul quale è stato realizzato), secondo quanto riportato in tab. 6.1. Tale impostazione è memorizzata nel file immagine del kernel (v. `rdev`);

device	Descrizione
<code>/dev/hdan ... hddn</code>	<i>n</i> -esima partizione su dischi ATA o CD-ROM ATAPI.
<code>/dev/sdan ... sden</code>	<i>n</i> -esima partizione su dischi SCSI o masterizzatori CD ATAPI.
<code>/dev/xdan ... xdbn</code>	<i>n</i> -esima partizione su dischi XT.
<code>/dev/fdn</code>	<i>n</i> -esima unità floppy disk (con <i>n</i> che parte da 0).
<code>/dev/nfs</code>	indica che la root directory deve essere montata tramite NFS.
<code>/dev/ram</code>	è un disco virtuale in memoria centrale (RAM).

Tabella 6.1: Indicazione dei dispositivi per il kernel di GNU/Linux.

³v. sez. 6.1.4.

rootflags=fs.option

indica delle opzioni di mounting del root filesystem, secondo quanto specificato da *fs.option*;

rootfstype=fstype

indica il tipo di filesystem da considerare nell'operazioni dei mounting della partizione associata alla root directory, secondo quanto specificato da *fstype* (per default il kernel prova a montare la partizione secondo un proprio elenco di filesystem: ext2, minix, ...). Può essere costituito da un elenco di filesystem separati dal carattere ',';

ro indica di montare il root filesystem in sola lettura (in modo che i programmi di controllo come **fsck** possano tranquillamente controllare la consistenza del filesystem senza preoccuparsi del fatto che potrebbero esserci file acceduti in scrittura da altri processi). Tale impostazione è memorizzata nel file immagine del kernel (v. **rdev**);

rw indica di montare il root filesystem in lettura e scrittura. Tale impostazione è memorizzata nel file immagine del kernel (v. **rdev**);

nfsroot=option

indica quale macchina, directory e opzioni NFS utilizzare per il root filesystem (deve essere specificato anche **root=/dev/nfs**);

ip=address | nfsaddr=address

nel caso si voglia montare un root filesystem tramite NFS, indica l'indirizzo IP dell'interfaccia da utilizzare per la comunicazione in rete. Se tale opzione non è specificata, il kernel tenta di configurarla tramite i protocolli RARP e BOOTP;⁴

ramdisk_start=offset

indica l'offset a cui inizia il kernel; ???

load_ramdisk=num

indica se deve essere effettuato il caricamento di un disco virtuale in RAM (ramdisk) secondo quanto specificato da *num* (1 sì, 0 no - default);

prompt_ramdisk=num

indica se visualizzare una richiesta per l'eventuale inserimento di un floppy disk contenente l'immagine di un ramdisk, secondo quanto specificato da *num* (1 sì, 0 no - default);

ramdisk_size=max_size

indica la dimensione massima della memoria RAM da poter essere utilizzata come ramdisk (default 4096, cioè 4MiB);

ramdisk_blocksize=size

indica la dimensione dei blocchi del ramdisk (questo potrebbe migliorare le prestazioni);

ramdisk=max_size

(obsoleto) analogo a **ramdisk_size**. Tale impostazione è memorizzata nel file immagine del kernel (v. **rdev**);

noinitrd

indica di rendere i dati di **initrd** disponibili attraverso il file **/dev/initrd**, che può essere letto prima di rilasciare la RAM al sistema;

cachesize=size

indica esplicitamente la dimensione in kiB della cache di secondo livello relativa alla CPU, anziché farla riconoscere direttamente al kernel;

⁴v. cap. ??.

mem=value

può indicare esplicitamente la quantità di RAM presente sul sistema (dove *value* specifica la quantità di memoria), oppure il fatto che non si vuole utilizzare la caratteristica della page table di 4MiB (dove *value* è `nopentium`). La quantità di memoria può essere specificata nei seguenti modi

- un valore esadecimale che indica il numero massimo di indirizzi di RAM (es. `mem=0x1000000` indica 16 MiB di RAM);
- un valore che indica la dimensione della RAM utilizzando i suffissi ‘k’ o ‘K’ per i KiB e ‘m’ o ‘M’ per i MiB (es. `mem=128m` indica 128 MiB di RAM);

memfrac=values

indica le zone in cui è frazionata la RAM. Sui sistemi basati su microprocessore 386, la memoria è suddivisa in zone che corrispondono al DMA (Direct Memory Access): “normale” per la memoria da 16 MiB fino a 1 GiB e “highmem” per la memoria oltre 1 GiB. L’elenco di valori *values* determina la quantità di memoria da preservare libera. La quantità di memoria da ritenere libera è data dalla memoria divisa per il relativo numero specificato da *values*. Il valore di default è `memfrac=32,128,128`;

swap=value

specifica il tipo di gestione della memoria virtuale, secondo il valore di *value* (`MAX_PAGE_AGE`, `PAGE_ADVANCE`, `PAGE_DECLINE`, `PAGE_INITIAL_AGE`, `AGE_CLUSTER_FRACT`, `AGE_CLUSTER_MIN`, `PAGEOUT_WEIGHT`, `BUFFEROUT_WEIGHT`);

buff=value

specifica il tipo di gestione della memoria virtuale, secondo il valore di *value* (`MAX_BUFF_AGE`, `BUFF_ADVANCE`, `BUFF_DECLINE`, `BUFF_INITIAL_AGE`, `BUFFEROUT_WEIGHT`, `BUFFERMEM_GRACE`);

acpi=value

indica se la gestione ACPI (Advanced Configuration & Power Interface) deve essere disabilitata (*value* può assumere soltanto il valore `off`);

console=value

indica di utilizzare uno specifico dispositivo come console (in genere la console è il primo terminale virtuale), sulla quale vengono visualizzati i messaggi durante il boot. Ad esempio si può utilizzare una porta seriale come console (su un sistema sul quale non è collegato uno schermo) specificando `console=ttyS1,9600` (in questo caso si tratta della seconda porta seriale con trasmissione a 9600 baud);

debug indica di visualizzare nella console tutti i messaggi del kernel (generalmente il kernel visualizza soltanto i messaggi più importanti). La stessa operazione può essere effettuata utilizzando un’opzione di `klogd`;

decnet=area,node

permette di specificare l’area (*area*) ed il nodo (*node*) di DECnet;

devfs=value

specifica come utilizzare `devfs` (*value* può assumere il valore `only` o `mount`);

gpt indica di utilizzare la gestione della partition table EFI (Extensible Firmware Interface) GUID.⁵;

idle=value

indica al kernel che nei cicli di attesa (*idle*) deve controllare il flag di rischedulazione (*value* = `poll`), piuttosto che attendere un interrupt (default). Questo può incrementare le prestazioni di un sistema SMP (con il costo di un incremento di energia consumata);

⁵i computer che si basano su architettura *Intel Itanium*, utilizzano l’EFI piuttosto che il BIOS, che gestisce un sistema di partizionamento dei dischi detto GUID che supporta dischi con dimensioni fino a 18 EB e fino a 128 partizioni.

init=value

indica al kernel il path del programma da lanciare come primo processo (in genere viene lanciato `/sbin/init`, ma può essere specificato di lanciare qualunque altro programma, ad esempio la shell – `init=/bin/sh` – qualora `/sbin/init` o un altro processo da questo lanciato sia danneggiato);

isapnp=value

può assumere la forma `isapnp=read_port,reset,skip_pci_scan,verbose`;

isapnp_reserve_dma=[n1[,n2[,...[,nN]]]]

indica di non utilizzare per i dispositivi Plug & Play i canali DMA (Direct Memory Access) elencati;

isapnp_reserve_io=io1,size1[,io2,size2][,...][,ioN,sizeN]

indica di non utilizzare per i dispositivi Plug & Play le zone di I/O identificate dalle coppie `io1,size1`, dove `io1` è l'indirizzo di inizio e `size1` la dimensione;

isapnp_reserve_irq=n1[,n2[,...[,nN]]]

indica di non utilizzare per i dispositivi Plug & Play gli interrupt specificati dall'elenco;

isapnp_reserve_mem=mem1,size1[,mem2,size2][,...][,memN,sizeN]

indica di non utilizzare per i dispositivi Plug & Play le zone di memoria identificate dalle coppie `mem1,size1`, dove `mem1` è l'indirizzo di inizio e `size1` la dimensione;

kbd-reset

indica al kernel di effettuare un reset sul controller della tastiera al suo avvio (in genere il kernel non effettua tale operazione poiché dovrebbe pensarci il BIOS);

lockd.tcpport=value

specifica la porta TCP per effettuare le operazioni lockd di NFS;

lockd.udpport=value

specifica la porta UDP per effettuare le operazioni lockd di NFS;

maxcpus=value

specifica il numero massimo di CPU da attivare in modalità SMP (Symmetric Multi-Processors): specificare 0 equivale ad indicare l'argomento `nosmp`;

mca-pentium

indica al kernel di evitare di effettuare il test per il riconoscimento del coprocessore matematico (che potrebbe dare problemi su architettura a Microchannel);

md=md_device_num,raid_level,chunk_size_factor,fault_level,dev0[,dev1[,...[,devN]]]

specifica il disco, gestito in RAID software⁶, sul quale risiede il kernel, dove `md_device_num` è il numero del dispositivo da considerare (0 indica `md0`, 1 `md1`, ...), `raid_level` è il tipo di RAID (-1 indica il linear mode e 0 lo striped mode), `chunk_size_factor` è la dimensione del chunk per il RAID 0 ed il RAID 1, il `fault_level` è il numero massimo di errori per il RAID 1. L'elenco `dev0`, ..., `devN` indica i dispositivi gestiti in RAID (v. anche l'argomento `raid`);

nmi_watchdog=value

specifica di abilitare (`value≠0`) o meno (`value=0`) il controllore degli interrupt non mascherabili (Non Maskable Interrupt Watchdog), supponendo che il kernel sia stato opportunamente compilato per supportare l'I/O APIC (Advanced Programmable Interrupt Controller ???). Se attivato, l'NMI Watchdog controlla che il contatore degli interrupt venga incrementato (che indica un funzionamento normale del sistema): se tale valore non viene incrementato, esso suppone che il sistema si sia bloccato e crea un dump con informazioni di diagnostica;

⁶v. sez. 3.18.

- no387** indica al sistema di ignorare l'eventuale presenza di un coprocessore matematico. Questo permette di evitare il verificarsi di bug dovuti ad alcuni coprocessori matematici *Intel* X387 quando vengono utilizzati in protected mode a 32 bit;
- no-hlt** indica al kernel di non inviare mai l'istruzione assembly HLT (halt) alla CPU. Questo permette di far funzionare correttamente alcune famiglie di processori *Intel* X486DX-100 che presentano un problema quando eseguono l'istruzione HLT che li fa permanere in uno stato di blocco indefinito, non riuscendo a riattivarsi a seguito di un interrupt;
- no-scroll**
indica al sistema di disabilitare le caratteristiche di scrolling, che renderebbero difficoltoso l'utilizzo dei terminali Braille;
- noapic** indica al kernel SMP di non utilizzare alcune caratteristiche avanzate APIC (Advanced Programmable Interrupt Controller ???);
- noht** indica al kernel di disabilitare l'hyper-threading ??? sui processori *Intel* che hanno questa caratteristica;
- noisapnp**
indica al kernel di disabilitare la gestione Plug & Play per le periferiche sul bus ISA;
- nomce** indica al kernel di disabilitare il controllo automatico di inconsistenza dei processori. Alcuni nuovi processori possiedono tale caratteristica che se attivata periodicamente esegue un controllo su anomalie di funzionamento del processore (ad esempio se la CPU va in surriscaldamento): se viene riscontrata un'anomalia viene lanciata un'eccezione (Machine Check Exception) che arresta il sistema;
- nosmp** indica al kernel SMP di operare come se stesse girando su un sistema a singolo processore;
- noresume**
indica al kernel di effettuare un nuovo boot nel caso di ripristino (resume) da uno stato sospeso (suspend). Un sistema può essere abilitato a porsi in uno stato sospeso, per limitare il consumo di energia (caso frequente soprattutto nei computer portatili): con questa indicazione il kernel ignora il fatto che il sistema si trova in uno stato sospeso e, anziché ripristinarlo, riavviando i processi dallo stato in cui si trovavano nel momento in cui sono stati sospesi, esegue un nuovo boot del sistema (questo perché alcuni sistemi non sono in grado di ripristinarsi);
- notsc** indica al kernel di non utilizzare il TSC (TimeStamp Counter) ??? anche se la CPU ne è provvista;
- nofxsr** indica al kernel di non utilizzare nessun meccanismo di velocizzazione reattivo al calcolo in virgola mobile, anche se il processore lo supporta;
- panic=value**
specifica il timeout (in secondi) dopo il quale il sistema deve tentare di riavviarsi automaticamente in seguito al verificarsi di un errore fatale di sistema: un *kernel panic*. Il valore di default è 0 che indica di attendere indefinitamente il riavvio manuale. Ad esempio **panic=30** specifica al kernel di tentare un riavvio del sistema dopo 30 secondi nel caso che si sia verificato un kernel panic. Tale impostazione può essere modificata anche durante il funzionamento del sistema, modificando il contenuto del file `/proc/sys/kernel/panic`;
- pirq=value**
specifica delle informazioni sul bus PCI relativamente agli interrupt (IRQ) per i kernel che girano su sistemi SMP;

profile=value

specifica il contatore per il profiling⁷ della CPU (tipicamente è impostato a 2). Il risultato del profiling è contenuto nel file `/proc/profile`;

quiet indica al kernel di essere meno verboso possibile (soltanto i messaggi critici verranno visualizzati sullo schermo). In questo modo non verranno visualizzati i messaggi informativi di riconoscimento dei dispositivi durante il boot;

raid=value

specifica l'utilizzo di un disco RAID per il caricamento del kernel: al momento *value* può assumere soltanto il valore `noautodetect` (v. anche l'argomento `md`);

reboot=value

specifica il tipo di reboot da effettuare in caso di riavvio della macchina tramite la combinazione di tasti `[Ctrl][Alt][Del]`. Per default viene effettuato un reboot a freddo (*cold reboot*), cioè un reset completo (il BIOS effettua il controllo della memoria, ...) piuttosto che un reboot a caldo (*warm reboot*), ovvero non un reset completo (il BIOS non effettua il controllo della memoria, ...). *value* può assumere i valori riportati in tab. 6.2 (più valori possono essere specificati se ha senso)

value	Descrizione
b	BIOS reboot.
c	cold reboot.
h	hard reboot.
sn	SMP reboot (<i>n</i> indica la CPU che deve gestire il reboot).
w	warm reboot.

Tabella 6.2: Possibili tipi di reboot.

reserve=iobase,extent[,iobase,extent]...

indica al kernel di non considerare per i test gli indirizzi di porte di I/O specificati (*iobase* è l'indirizzo della porta di I/O di base e *extent* è lo spazio di indirizzi relativo). Questa direttiva è utile nei casi in cui, con determinato hardware (ad esempio alcune schede di rete Ethernet), il sistema si blocca in fase di boot. Il kernel non testerà gli intervalli degli indirizzi delle porte di I/O specificati, ritenendoli già utilizzati dalla periferica "reserved". Ad esempio, se la riga di boot contiene i seguenti argomenti

```
reserve=0x300,32 blah=0x300
```

il kernel non effettuerà nessun test nell'intervallo di porte di I/O 300_H-31F_H tranne che per il driver 'blah';

resume=filename

specifica il file che deve contenere le informazioni relative ai processi quando il sistema va nello stato sospeso (suspend) per poterli ripristinare (resume) successivamente;

pci=value

imposta le caratteristiche del bus PCI. I possibili valori assunti da *value* sono i seguenti

assign-busses

indica al kernel di assegnare sempre tutti i numeri del bus PCI, non considerando eventuali impostazioni del firmware;

⁷il profiling è il meccanismo di indagine per capire come poter ottimizzare le prestazioni di un software.

bios | **nobios**
indica di effettuare i test sul bus PCI utilizzando (**bios** – default) o meno (**nobios**) il BIOS;

conf1 | **conf2**
indica di utilizzare la configurazione Type 1 o Type 2 nel caso in cui sia attivata la modalità diretta (*direct mode*) del bus PCI (questa opzione disabilita automaticamente il BIOS per effettuare i test sul bus PCI – v. **nobios**);

irqmask=*value*
specifica una bit mask per l’assegnamento automatico degli interrupt (IRQ) per il bus PCI;

lastbus=*value*
specifica il valore dell’ultimo bus PCI che deve essere controllato dal kernel;

noacpi indica al kernel di non utilizzare il meccanismo di inoltro IRQ dell’ACPI durante la configurazione del bus PCI;

nopeer indica al kernel di disabilitare il peer bridge fixup (nel caso in cui ci siano peer host bridge, controlla i bus PCI subito dopo ognuno di essi);

nosort indica al kernel di non ordinare i dispositivi PCI durante la fase di test;

off indica al kernel di disabilitare il test relativo al bus PCI (con questa impostazione, i driver che fanno uso di funzioni PCI per il rilevamento e l’inizializzazione dei dispositivi potrebbero non funzionare);

usepirqmask
indica al kernel di considerare la bit mask degli IRQ memorizzata nella tabella PIR del BIOS durante la fase di test del bus PCI (non ha effetto se è abilitato l’inoltro IRQ dell’ACPI);

rom indica al kernel di assegnare uno spazio di indirizzi alle ROM di espansione;

video=*value*
imposta le caratteristiche del driver della scheda video (*frame buffer*). La sintassi per mezzo della quale viene indicato è la seguente

video=*name:option1[,option2][,...]*

dove *name* è il nome del driver relativo al frame buffer, e *optioni* sono le eventuali opzioni specifiche per il driver considerato. I possibili valori assunti da *name* sono i seguenti

map imposta la console con la mappatura (mapping) del frame buffer: l’elenco di opzioni che seguono specificano la mappatura relativa alle varie console (la *n*-esima opzione si riferisce all’*n*-esima console);

scrollback
imposta la quantità di memoria allocata per lo *scrollback buffer* secondo quanto specificato dall’unica opzione che segue il carattere ‘:’: questa esprime il numero di byte riservati per la memorizzazione dello *scrollback buffer* (l’opzione può essere anche specificata come un numero seguito dalla lettera ‘k’ che indica che il valore numerico è espresso in kilobyte);

vc imposta il frame buffer per le virtual console indicate dall’unica opzione che segue il carattere ‘:’: questa indica la prima o l’intervallo di console virtuali (espresso con due numeri separati dal carattere ‘-’) da associare al frame buffer;

max_scsi_luns=*value*
specifica il massimo valore di LUN (Logical Unit Number) per il quale effettuare il test sul bus SCSI;

max_scsi_luns=value

specifica il valore (compreso tra 1 ed 8) massimo di LUN (Logical Unit Number) per il quale effettuare il test sul bus SCSI (per default il kernel effettua il test del bus SCSI soltanto per i dispositivi con LUN=0);

scsi_logging=value

specifica il livello di log (un valore numerico hce indica la verbosità del log) per gli eventi relativi al bus SCSI (**error**, **scan**, **mlqueue**, **mlcomplete**, **llqueue**, **llcomplete**, **hlqueue**, **hlcomplete**) in fase di boot;

st=value

indica al kernel di impostare le unita a nastro SCSI in maniera specifica. Tale parametro può essere espresso mediante la seguente sintassi

st=buf_size[, write_threshold[, max_bufs]]

dove

buf_size

è la dimensione del buffer che può essere impostata con un valore espresso in kB (per default è 32, ma può arrivare a 16.348);

write_threshold

è la soglia oltre la quale il buffer viene riversato sul dispositivo. È espresso in kB (per default è 30);

max_bufs

indica il numero massimo di buffer. Esso dipende dal numero di dispositivi riconosciuti (per default è 2);

scsi_ctrl=value

specifica le opzioni per un determinato controller SCSI. I valori possibili per **scsi_ctrl** sono riportati in tab. 6.3;

scsi_ctrl	Driver
aha152x	Adaptec aha151x, aha152x, aic6260, aic6360, SB16-SCSI.
aha1542	Adaptec aha1540, aha1542.
aic7xxx	Adaptec aha274x, aha284x, aic7xxx.
advansys	AdvanSys SCSI Host Adaptors.
in2000	Always IN2000 Host Adaptor.
AM53C974	AMD AM53C974 based hardware.
BusLogic	ISA/PCI/EISA <i>BusLogic</i> SCSI Hosts.
eata	EATA SCSI Cards.
tmc8xx	<i>Future Domain</i> TMC-8xx, TMC-950.
fdomain	<i>Future Domain</i> TMC-16xx, TMC-3260, AHA-2920.
ppa	<i>IOMEGA</i> Parallel Port / ZIP drive.
ncr5380	NCR5380 based controllers.
ncr53c400	NCR53c400 based controllers.
ncr53c406a	NCR53c406a based controllers.
pas16	<i>Pro Audio Spectrum</i> .
st0x	<i>Seagate</i> ST-0x.
t128	<i>Trantor</i> T128.
u14-34f	<i>Ultrastor</i> SCSI cards.
wd7000	<i>Western Digital</i> WD7000 cards.

Tabella 6.3: Alcuni driver SCSI supportati dal kernel Linux.

hdx=value

indica al kernel di gestire il disco ATA specificato da x (da a ad h) secondo quanto specificato da **value**, che può assumere i seguenti valori:

noprobe

indica al kernel di non testare la presenza del disco;

none indica al kernel che il disco non è presente;

nowerr indica al kernel di ignorare il bit **WRERR_STAT** per il disco;

cdrom indica al kernel che il disco è un'unità CD-ROM;

cyl, head, sect

specifica la geometria del disco (cilindri, testine, settori);

autotune

indica al kernel di tentare di impostare automaticamente la massima velocità di comunicazione possibile con il disco;

index=value

indica al kernel di gestire l'interfaccia IDE (ATA) specificata da *x* (da 0 a 3) secondo quanto specificato da *value*, che può assumere i valori di seguito riportati (per default vengono considerate le impostazioni **ide0=0x1f0** e **ide1=0x170**):

noprobe

indica al kernel di non testare/usare l'interfaccia;

base[,ctl[,irq]]

specifica l'indirizzo di base dell'interfaccia (in genere **0x1f0** o **0x170**), quello della relativa porta di controllo (nel caso in cui non sia specificato viene considerato uguale a **base+0x206**) e il numero di IRQ;

autotune

indica al kernel di tentare di impostare automaticamente la massima velocità di comunicazione possibile con l'interfaccia;

noautotune

indica al kernel di non tentare di impostare automaticamente la massima velocità di comunicazione possibile con l'interfaccia (questa impostazione è quella di default per varie interfacce);

serialize

indica al kernel di non sovrapporre operazioni su **index** e su **index+1**;

interface

specifica un particolare tipo di interfaccia IDE (ATA), secondo quanto riportato in tab. 6.4 (valito soltanto per **ide0**). Per particolari interfacce IDE PCMCIA può essere necessario indicare l'argomento **ide2=0x180,0x386**;

interface	Descrizione
dtc2278	probe/support DTC2278 interface.
ht6560b	probe/support HT6560B interface.
cmd640_vlb	REQUIRED for VLB cards with the CMD640 chip (not for PCI – automatically detected).
qd6580	probe/support qd6580 interface.
ali14xx	probe/support ali14xx chipsets (ALI M1439/M1445).
umc8672	probe/support umc8672 chipsets.

Tabella 6.4: Alcune particolari interfacce IDE riconosciute da Linux.

hd=cyl, head, sect

specifica la geometria del disco MFM/RLL/Standard ST-506 (cilindri, testine, settori);

xd=type,irq,iobase,dma

specifica alcune impostazioni dei dischi XT: *type* specifica il costruttore del disco (0=generico, 1=DTC, 2,3,4=Western Digital, 5,6,7=Seagate, 8=OMTI), *irq* specifica l'interrupt number, *iobase* l'indirizzo della porta di base e *dma* il canale DMA (per default *type*=2, *irq*=5, *iobase*=0x320, *dma*=3);

xd_geo=cyl,head,sect

specifica la geometria dei dischi XT (cilindri, testine, settori);

sound_driver=value

specifica i parametri di configurazione per la scheda audio, secondo quanto riportato nelle tab. 6.5 6.6 e 6.7;

sound_driver	Descrizione
snd-dummy	Dummy soundcard.
snd-mpu401	mpu401 UART.
snd-mtpav	MOTU Midi Timepiece.
snd-serial	Serial UART 16450/16550 MIDI.
snd-virmidi	Dummy soundcard for virtual rawmidi devices.
snd-ad1816a	ADI SoundPort AD1816A.
snd-ad1848	Generic driver for AD1848/AD1847/CS4248.
snd-als100	Avance Logic ALS100.
snd-azt2320	Aztech Systems AZT2320 (and 2316).
snd-cmi8330	C-Media's CMI8330.
snd-cs4231	Generic driver for CS4231 chips.
snd-cs4232	Generic driver for CS4232 chips.
snd-cs4236	Generic driver for CS4235/6/7/8/9 chips.
snd-dt019x	Diamond Technologies DT-019x.
snd-es1688	Generic ESS AudioDrive ESx688.
snd-es18xx	Generic ESS AudioDrive ES18xx.
snd-gusclassic	Gus classic.
snd-gusextreme	Gus extreme.
snd-gusmax	Gus Max.
snd-interwave	Interwave.
snd-interwave-stb	Interwave.
snd-opl3sa2	Yamaha OPL3SA2.
snd-opti93x	OPTi 82c93x based cards.
snd-opti92x-cs4231	OPTi 82c92x/CS4231.
snd-opti92x-ad1848	OPTi 82c92x/AD1848.
snd-es968	ESS AudioDrive ES968.
snd-sb16	SoundBlaster 16.
snd-sbawe	SoundBlaster 16 AWE.
snd-sb8	Old 8 bit SoundBlaster (1.0, 2.0, Pro).
snd-sgalaxy	Sound galaxy.
snd-wavefront	Wavefront.

Tabella 6.5: Alcune particolari schede audio ISA riconosciute dal driver ALSA.

sound_driver	Descrizione
ad1848	AD1848.
adlib	Adlib.
mad16	MAD16.
pas2	ProAudioSpectrum PAS16.
sb	SoundBlaster.
uart401	UART 401 (on card chip).
uart6850	UART 6850 (on card chip).
opl3	Yamaha OPL2/OPL3/OPL4 FM Synthesizer (on card chip).
opl3sa	Yamaha OPL3-SA FM Synthesizer (on card chip).
opl3sa2	Yamaha OPL3-SA2/SA3 FM Synthesizer (on card chip).

Tabella 6.6: Alcune particolari schede audio riconosciute dal driver OSS.

cdrom_driver=value

specifica i parametri di configurazione per l'unità CD-ROM non SCSI/ATAPI, secondo quanto riportato in tab. 6.8;

sound_driver	Descrizione
snd-ali5451	ALi PCI audio M5451.
snd-als4000	Avance Logic ALS4000.
snd-cmipci	C-Media CMI8338 and 8738.
snd-cs4281	Cirrus Logic CS4281.
snd-cs46xx	Cirrus Logic Sound Fusion CS46XX.
snd-emu10k1	EMU10K1 (SB Live!).
snd-ens1370	Ensoniq ES1370 AudioPCI.
snd-ens1371	Ensoniq ES1371 AudioPCI.
snd-es1938	ESS Solo-1 (ES1938, ES1946, ES1969).
snd-es1968	ESS Maestro 1/2/2E.
snd-fm801	ForteMedia FM801.
snd-intel8x0	Intel ICH (i8x0) chipsets.
snd-maestro3	ESS Maestro3/Allegro (ES1988).
snd-korg1212	Korg 1212 IO.
snd-rme32	RME Digi32, Digi32/8 and Digi32 PRO.
snd-nm256	NeoMagic 256AV and 256ZX.
snd-rme96	RME Digi96, Digi96/8 and Digi96/8 PRO/PAD/PST.
snd-rme9652	RME Digi9652 audio interface.
snd-hdsp	RME Hammerfall DSP.
snd-sonicvibes	S3 SonicVibes.
snd-trident	Trident 4DWave DX/NX & SiS SI7018.
snd-via82xx	VIA South Bridge VT82C686A/B/C, VT8233A/C, VT8235.
snd-ymfpici	Yamaha DS1/DS1E.
snd-ice1712	ICEensemble ICE1712 (Envy24).

Tabella 6.7: Alcune particolari schede audio PCI riconosciute dal driver ALSA.

cdrom_driver	Descrizione
aztcad	Aztech Interface.
cdu31a	CDU-31A and CDU-33A Sony Interface (Also Old PAS).
sonycd535	CDU-535 Sony Interface.
gscd	GoldStar Interface.
isp16	ISP16 Interface.
mcd	Mitsumi Standard Interface.
mcdx	Mitsumi XA/MultiSession Interface.
optcd	Optics Storage Interface.
cm206	Phillips CM206 Interface.
sjcd	Sanyo Interface.
sbpcd	SoundBlaster Pro Interface.

Tabella 6.8: Alcune particolari unità CD-ROM riconosciute da Linux.

icn=value

specifica le opzioni relative al driver ICN ISDN;

pcbit=value

specifica le opzioni relative al driver PCBIT ISDN;

teles=value

specifica le opzioni relative al driver Teles ISDN;

digi=value

specifica le opzioni relative al driver seriale DigiBoard;

riscom8=value

specifica le opzioni relative al driver seriale RISCom/8 Multiport;

baycom=value

specifica le opzioni relative al driver seriale Baycom Serial/Parallel Radio Modem;

Esistono altri argomenti più specifici, per i quali è consigliabile vedere la documentazione fornita con il dispositivo considerato e/o leggere il *Boot-Prompt HowTo* di P. Gortmaker.

Quando viene avviato, il kernel per prima cosa considera gli argomenti `root=`, `ro`, `rw` e `debug`. Quindi controlla un elenco di funzioni (funzioni di setup, dal nome *keyword_setup*) per le quali possono essere stati specificati dei particolari argomenti (*keyword*).

Gli argomenti non riconosciuti dal kernel vengono considerati come assegnamenti di variabili di ambiente o come parametri da passare al primo processo avviato dal kernel (*init*).

L'elenco di argomenti passati al boot è contenuto nel file `/proc/cmdline`.

6.2 I moduli

Il kernel di GNU/Linux è generalmente composto da un file `vmlinuz` o simile, e, come accennato in sez. 6.1.1 le funzionalità non di base possono essere modularizzate, ovvero il relativo codice può essere inserito in file diversi dal kernel, detti **moduli** o LKM (*Loadable Kernel Module*).⁸ Un modulo è un file in formato binario ELF (*Executable and Linkable Format*) che ha generalmente estensione `.o` ed è possibile caricarlo in memoria quando si presenta la necessità di utilizzare le sue funzionalità. L'operazione di caricamento di un modulo in memoria, consiste appunto nel caricamento del codice contenuto nel file relativo al modulo desiderato in memoria centrale in kernel space. I moduli, quindi, costituiscono a tutti gli effetti una parte integrante del kernel di GNU/Linux, sono delle vere e proprie estensioni del kernel di base e come tali possono estenderne funzionalità ma anche rimpiazzarle, modificando il comportamento del kernel di base stesso.

moduli

L'utilizzo di moduli è vantaggioso rispetto a modificare il codice del kernel, in quanto

- non si rischia di apportare delle modifiche al kernel, che potrebbero anche non permetterne più il riavvio;
- non si è obbligati a ricompilare i file sorgenti del kernel ogni volta che si vogliono aggiungere delle nuove funzionalità, o modificarne alcune già presenti;
- si ha una maggiore facilità di manutenzione e di debugging delle funzionalità del kernel;
- si può risparmiare memoria: i moduli possono essere caricati in memoria soltanto quando ce n'è bisogno;

I moduli sono generalmente utilizzati per realizzare

device driver software che gestisce l'interazione a basso livello con un particolare dispositivo;

filesystem driver software che interpreta le informazioni memorizzate su un media come file e directory;

system call i programmi che sono eseguiti in user space devono utilizzare delle chiamate ad opportune routine (system call) per avere delle funzionalità messe a disposizione dal kernel. Tali routine sono generalmente contenute nel kernel di base, ma si possono creare dei moduli che gestiscono nuove system call o soppiantano quelle originali;

network driver software che interpreta uno specifico protocollo di rete;

executable interpreter software che interpreta gli script secondo una determinata sintassi propria del linguaggio di programmazione considerato;

⁸i moduli sono comparsi per la prima volta nel 1995, con la versione 1.2 del kernel di GNU/Linux.

I moduli distribuiti assieme al kernel di base vengono generati con il comando

```
# make modules
```

come illustrato in sez. 6.1.2, che produce un insieme di file *.o. Il comando

```
# make modules_install
```

copia i file binari nella opportuna directory all'interno di `/lib/modules/kernel_version`, dove `kernel_version` è la versione del kernel considerato⁹.

Altri moduli, non distribuiti con il kernel di GNU/Linux, possono avere una propria procedura di creazione.

Alcuni moduli possono utilizzare dati che risentono della “storia” da un caricamento del modulo all'altro. Tali dati sono detti **dati persistenti** (*persistent data*) e vengono salvati su un file quando il modulo viene scaricato dalla memoria e riletto dal file quando il modulo viene nuovamente caricato in memoria.

dati persistenti

I moduli sono gestiti per mezzo dei comandi contenuti nel pacchetto `modutils`, che fanno riferimento al file di configurazione `/etc/modules.conf`.

6.2.1 Caricamento di moduli in memoria

I moduli possono essere caricati in memoria con il comando `insmod` (man page `insmod(8)`).

Comando: `insmod`

Path: `/sbin/insmod`

SINTASSI

```
# insmod [option] modulename [parms]
```

DESCRIZIONE

option indica la modalità di funzionamento di `insmod`. Può assumere i seguenti valori:

```
-e persist_name | --persist=persist_name
```

specifica se i dati persistenti del modulo devono essere acquisiti al suo caricamento e memorizzati quando il modulo viene scaricato dalla memoria. L'opzione `-e ""` è interpretata come il valore di `persistdir` indicato nel file di configurazione (per default `/etc/modules.conf`) seguito dal nome del modulo specificato da *modulename*;

```
-f | --force
```

tenta di caricare in memoria il modulo *modulename* anche se la versione del kernel per la quale è stato compilato non coincide con quella del kernel corrente (tale opzione ha effetto soltanto sul controllo della versione del kernel, non sui nomi dei simboli);

```
-h | --help
```

visualizza un aiuto sommario di `insmod`;

```
-k | --autoclean
```

imposta il flag di auto-clean per il modulo *modulename*, utilizzato da `kerneld` per lo scaricamento dalla memoria dei moduli che non sono stati utilizzati da più di un determinato periodo di tempo (per default 1 minuto);

```
-L | --lock
```

indica di utilizzare `floc` (man page `floc(2)`) per prevenire carichi simultanei dello stesso modulo;

```
-m | --map
```

visualizza un'elenco del caricamento dei moduli, per facilitarne il debug nel caso di kernel panic;

⁹la versione del kernel in esecuzione può essere visualizzata con il comando `uname --release`.

- n | **--noload**
indica di effettuare tutte le operazioni, tranne il caricamento vero e proprio del modulo in memoria;
- o *module_name* | **--name=module_name**
specifica esplicitamente il nome del modulo come indicato da *module_name* piuttosto che derivarlo da quello del relativo file;
- O *blob_name* | **--blob=blob_name**
indica di salvare il modulo in un blob il cui nome è specificato da *blob_name*. Si tratta di un file che contiene esattamente le istruzioni caricate in memoria dopo la manipolazione e la rilocazione;
- p | **--probe**
indica di testare se il modulo può essere correttamente caricato in memoria (non viene effettuato un controllo sulla rilocazione del codice);
- P *prefix* | **--prefix prefix**
indica un eventuale prefisso da aggiungere ai nomi dei simboli esportati, per i moduli creati con le versioni dei simboli che devono essere caricati in memoria con un kernel che non è stato compilato senza le versioni dei simboli;
- q | **--quiet**
indica di non visualizzare l'eventuale elenco dei simboli non risolti e/o l'eventuale differenza della versione di kernel;
- r | **--root**
indica di caricare in memoria un modulo il cui proprietario non è il superuser. Poiché i moduli vengono caricati ed eseguiti in kernel space, essi acquisiranno i privilegi di superuser e pertanto è opportuno che i relativi file siano di proprietà del superuser (il caricamento in memoria di moduli può compromettere il funzionamento del sistema). Per default, **insmod** permette il caricamento in memoria soltanto dei moduli i cui relativi files sono di proprietà del superuser;
- s | **--syslog**
indica di redirigere tutto l'output sul system log anziché sul terminale;
- S | **--kallsyms**
forza il caricamento dei moduli con i loro simboli per il debug, anche se il kernel è caricato senza simboli di debug;
- v | **--verbose**
indica di visualizzare una maggiore quantità di informazioni relativa all'operazione di caricamento del modulo;
- V | **--version**
visualizza la versione di **insmod**;
- X | **--export**
indica di esportare tutti i simboli definiti dal modulo (comportamento di default);
- x | **--noexport**
indica di non esportare tutti i simboli definiti dal modulo;
- Y | **--ksymoops**
indica di aggiungere i simboli di debug a *ksyms* (comportamento di default);
- y | **--noksymoops**
indica di non aggiungere i simboli di debug a *ksyms*;
- N | **--numeric-only**
indica di effettuare un controllo soltanto sulla parte numerica della versione del modulo con quella del kernel;

modulename indica il modulo da caricare in memoria. Questo può essere specificato con il nome del file che rappresenta il modulo eventualmente completo di path oppure senza estensione del nome del file. **insmod** ricerca il modulo nel file di configurazione */etc/modules.conf* (o quello specificato dalla variabile di ambiente **MODULECONF**), oppure, nel caso in cui tale file non esista, nelle directory elencate dalla variabile di ambiente **MODPATH** o, in sua assenza, nella directory di default;

parms sono eventuali parametri, specifici per ogni modulo, che possono essere specificati al caricamento del modulo in questione.

Nel caso di valori interi, questi possono essere specificati in forma decimale, ottale o esadecimale, nello stile del linguaggio C (es. $17 = 17_{10}$ può essere scritto anche come $021 = 21_8$ o $0x11 = 11_H$).

I parametri che non iniziano con un carattere numerico sono considerati delle stringhe di caratteri. I parametri che iniziano con il carattere `''` (doppio apice) sono interpretate come le stringhe del linguaggio C (comprese le sequenze di escape).

I vettori possono essere specificati da elenchi di elementi separati da virgole.

Ad esempio, per caricare in memoria il modulo `serial.o` (il driver della porta seriale), si può impartire il comando

```
# insmod serial.o
```

Quando un modulo viene caricato in memoria, esso diviene parte integrante del kernel in esecuzione. Spesso, un modulo è un device driver e quando esso viene caricato in memoria, ricerca un dispositivo del tipo che sa gestire, quindi, in caso positivo, si registra come il driver con un determinato *major number* e come gestore dell'interrupt utilizzato dal dispositivo. In questo modo si può gestire il dispositivo per mezzo di un file di dispositivo che fa riferimento al major number relativo al driver in questione.

L'elenco dei dispositivi associati ai major number è contenuto nel file `/proc/devices` ed un esempio è riportato di seguito

Character devices:

```
1 mem
2 pty
3 tty
4 ttyS
5 cua
6 lp
7 vcs
10 misc
13 input
14 sound
29 fb
36 netlink
128 ptm
129 ptm
130 ptm
131 ptm
132 ptm
133 ptm
134 ptm
135 ptm
136 pts
137 pts
138 pts
139 pts
140 pts
141 pts
142 pts
143 pts
162 raw
180 usb
254 pcmcia
```

Block devices:

```
1 ramdisk
2 fd
```

```

3 ide0
9 md
12 unnamed
14 unnamed
22 ide1
38 unnamed
39 unnamed

```

L'elenco degli interrupt associati ai dispositivi è contenuto nel file `/proc/interrupts` ed un esempio è riportato di seguito

```

          CPU0
0:      346058      XT-PIC  timer
1:      14930      XT-PIC  keyboard
2:         0      XT-PIC  cascade
3:         5      XT-PIC  pcnet_cs
8:         1      XT-PIC  rtc
10:        35      XT-PIC  ALi Audio Accelerator
11:        36      XT-PIC  usb-ohci, 02 Micro, Inc. 0Z6812 Cardbus Controller
12:     21575      XT-PIC  PS/2 Mouse
14:     13829      XT-PIC  ide0
15:     24121      XT-PIC  ide1
NMI:         0
ERR:         0

```

Nel caso in cui il modulo caricato sia un network driver, esso si registra come gestore del relativo dispositivo di rete, riferendosi al suo nome. L'elenco dei nomi delle interfacce di rete correntemente registrate è contenuto nel file `/proc/net/dev` ed un esempio è riportato di seguito

Inter-	Receive								Transmit	
face	bytes	packets	errs	drop	fifo	frame	compressed	multicast	bytes	packe
lo:	1889232	27641	0	0	0	0	0	0	1889232	276
eth0:	0	0	0	0	0	0	0	0	240	

Un modulo che viene caricato in memoria, generalmente, aggiorna il system log¹⁰ con messaggi che tracciano il suo avvenuto caricamento.

Il caricamento di moduli in memoria può anche non andare a buon fine e `insmod` lo segnala essenzialmente con i seguenti messaggi di errore

`unresolved symbol symbol_name`

questo messaggio segnala che il tentativo di caricare il modulo in memoria non è andato a buon fine poiché il modulo contiene dei riferimenti al simbolo *symbol_name* che non è correntemente definito dal kernel di base o da altri moduli già caricati in memoria. Il kernel ed i moduli possono infatti definire dei simboli (*symbol*) ai quali possono far riferimento altri moduli. In questo modo si vengono a definire delle dipendenze tra moduli: è possibile che un modulo necessiti della presenza di altri moduli per poter essere caricato in memoria. Quindi è importante l'ordine cronologico di caricamento dei moduli in memoria;

`couldn't find the kernel version this module was compiled for`

questo messaggio segnala che il tentativo di caricare il modulo in memoria non è andato a buon fine poiché il file indicato contiene riferimenti ad una versione di kernel che non coincide con quella in esecuzione (oppure il file indicato non è affatto un modulo);

???

¹⁰v. sez. 5.7.

Caricamento intelligente

???

Comando: `modprobe`
 Path: `/sbin/modprobe`
 SINTASSI
 # `modprobe [option]`

DESCRIZIONE

option indica la modalità di funzionamento di `modprobe`. Può assumere i seguenti valori:

`-??? ???;`

???

6.2.2 Scarico di moduli dalla memoria

???

Comando: `rmmod`
 Path: `/sbin/rmmod`
 SINTASSI
 # `rmmod [option] [modulename [...]]`

DESCRIZIONE

option indica la modalità di funzionamento di `rmmod`. Può assumere i seguenti valori:

`-a | --all`

scarica dalla memoria i moduli che hanno il flag auto-clean impostato ed imposta tale flag per gli altri moduli;

`-e | --persist`

indica di salvare le informazioni persistenti dei moduli indicati senza scaricarli dalla memoria. Se non è specificato nessun modulo, salva le informazioni persistenti di tutti moduli;

`-h | --help`

visualizza un aiuto sommario di `rmmod`;

`-r | --stacks`

indica di scaricare lo stack del modulo specificato;

`-s | --syslog`

indica di redirigere l'output nel system log anziché sul terminale;

`-v | --verbose`

visualizza più informazioni relative all'operazione di scaricamento del modulo dalla memoria;

`-V | --version`

visualizza la versione di `rmmod`;

modulename specifica il nome del modulo (o dei moduli) da scaricare dalla memoria (i moduli vengono scaricati nell'ordine in cui compaiono sulla riga di comando);

???

6.2.3 Carico/scarico automatico dei moduli

???

6.2.4 Elenco dei moduli caricati in memoria

???

Comando: **ksyms**
 Path: **/sbin/ksyms**
 SINTASSI
ksyms [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di **ksyms**. Può assumere i seguenti valori:

-a | **--all**
 visualizza tutti i simboli (per default quelli definiti dal kernel di base non sono visualizzati);
-h | **--noheader**
 indica di non visualizzare le intestazioni delle colonne;
-H | **--help**
 visualizza un aiuto sommario di **ksyms**;
-m | **--info**
 visualizza le informazioni relative ai moduli caricati in memoria;
-V | **--version**
 visualizza la versione di **ksyms**;

???

6.2.5 Il file di configurazione

???

???

6.3 I processi

Un **processo** (o *task*) è l'istanza dell'esecuzione di un determinato programma: quando si lancia un file eseguibile viene creato un processo in memoria che esegue le istruzioni indicate dal programma. In genere l'esecuzione di un programma genera un processo, ma programmi più complessi (per esempio quelli che gestiscono comunicazioni client-server) possono generare più processi. *processo*

Quando viene lanciato in esecuzione un programma, il kernel crea un processo in memoria assegnandogli un numero intero che lo identifica univocamente all'interno della macchina: il **PID** (Process IDentifier). In genere il PID è sequenziale, ovvero ad ogni processo viene assegnato il PID del processo precedente incrementato di 1 ed al primo processo lanciato dal sistema (*init*) viene assegnato il PID uguale ad 1. *PID*

La struttura dati relativa ad un processo è riportata di seguito

```
struct task_struct {
/* these are hardcoded - don't touch */
volatile long      state;           /* -1 unrunnable, 0 runnable, >0 stopped */
long               counter;
long               priority;
unsigned           long signal;
unsigned           long blocked;    /* bitmap of masked signals */
unsigned           long flags;     /* per process flags, defined below */
int errno;
long               debugreg[8];    /* Hardware debugging registers */
struct exec_domain *exec_domain;
/* various fields */
struct linux_binfmt *binfmt;
```

```

struct task_struct  *next_task, *prev_task;
struct task_struct  *next_run,  *prev_run;
unsigned long       saved_kernel_stack;
unsigned long       kernel_stack_page;
int                 exit_code, exit_signal;
/* ??? */
unsigned long       personality;
int                 dumpable:1;
int                 did_exec:1;
int                 pid;
int                 pgrp;
int                 tty_old_pgrp;
int                 session;
/* boolean value for session group leader */
int                 leader;
int                 groups[NGROUPS];
/*
 * pointers to (original) parent process, youngest child, younger sibling,
 * older sibling, respectively. (p->father can be replaced with
 * p->p_pptr->pid)
 */
struct task_struct  *p_opptr, *p_pptr, *p_cptra,
                    *p_ysptr, *p_osptra;
struct wait_queue   *wait_chldexit;
unsigned short       uid,euid,suid,fsuid;
unsigned short       gid,egid,sgid,fsuid;
unsigned long        timeout, policy, rt_priority;
unsigned long        it_real_value, it_prof_value, it_virt_value;
unsigned long        it_real_incr, it_prof_incr, it_virt_incr;
struct timer_list    real_timer;
long                 utime, stime, cutime, cstime, start_time;
/* mm fault and swap info: this can arguably be seen as either
   mm-specific or thread-specific */
unsigned long        min_flt, maj_flt, nswap, cmin_flt, cmaj_flt, cnsnap;
int swappable:1;
unsigned long        swap_address;
unsigned long        old_maj_flt;    /* old value of maj_flt */
unsigned long        dec_flt;        /* page fault count of the last time */
unsigned long        swap_cnt;       /* number of pages to swap on next pass */
/* limits */
struct rlimit        rlim[RLIM_NLIMITS];
unsigned short       used_math;
char                 comm[16];
/* file system info */
int                 link_count;
struct tty_struct     *tty;          /* NULL if no tty */
/* ipc stuff */
struct sem_undo       *semundo;
struct sem_queue      *semsleeping;
/* ldt for this task - used by Wine. If NULL, default_ldt is used */
struct desc_struct    *ldt;
/* tss for this task */
struct thread_struct  tss;
/* filesystem information */
struct fs_struct       *fs;
/* open file information */
struct files_struct    *files;
/* memory management info */
struct mm_struct       *mm;
/* signal handlers */
struct signal_struct   *sig;
#ifdef __SMP__

```

```

int                processor;
int                last_processor;
int                lock_depth;    /* Lock depth.
                                   We can context switch in and out
                                   of holding a syscall kernel lock... */
#endif
};

```

Il PID di un processo può essere ottenuto per mezzo del comando `pidof` (man page `pidof(8)`).

Comando: `pidof`
 Path: `/sbin/pidof`

SINTASSI
`# pidof [option] process [...]`

DESCRIZIONE

option specifica la modalità di funzionamento di `pidof`. Può assumere i seguenti valori

- `-s` (single shot) indica di visualizzare un solo PID;
- `-x` (scripts too) indica di visualizzare anche il PID delle shell dalla quale è stato avviato il processo *process*;
- `-o omit_PID`
 indica di escludere il PID *omit_PID* dall'output di `pidof`;

In genere `pidof` è un symbolic link al file eseguibile `/sbin/killall5`;

Inoltre il kernel riserva uno specifico spazio di memoria per ogni processo, ovvero due processi che girano “contemporaneamente” sul sistema non si accorgono l'uno dell'altro e il malfunzionamento dell'uno non causa problemi all'altro. Un processo non può accedere alla memoria assegnata ad un altro processo se non attraverso l'utilizzo di particolari meccanismi gestiti e controllati dal kernel.

Il kernel crea il processo `init` lanciando in esecuzione il comando `/sbin/init` (man page `init(8)`)¹¹ e questo è l'unico processo creato dal kernel. Qualunque altro processo viene sempre creato a partire da una richiesta effettuata da un processo già esistente. In questo modo si viene a creare una gerarchia di processi (**process tree** o *albero dei processi*) alla cui radice c'è `init`. Un processo *A* che crea un processo *B* è detto **processo genitore** (o *processo padre*) del processo *B* ed il processo *B* è detto **processo figlio** del processo *A*. Poiché ogni processo è identificato dal PID, è una convenzione diffusa riferirsi al PID del processo genitore con il termine **PPID** (Parent PID).

process tree

processo genitore

processo figlio

PPID

L'albero dei processi può essere visualizzato per mezzo del comando `pstree` (man page `pstree(1)`).

Comando: `pstree`
 Path: `/usr/bin/pstree`

SINTASSI
`$ pstree [option]`

DESCRIZIONE

option specifica la modalità di funzionamento di `pstree`. Può assumere i seguenti valori:

- `-a` visualizza anche gli argomenti specificati sulla riga di comando;
- `-c` disabilita la compattazione dei sottoalberi identici;
- `-G` utilizza i caratteri del terminale VT100 per la visualizzazione dei segmenti dei sottoalberi;

¹¹v. cap. 2.

- h evidenzia il processo corrente ed i suoi genitori;
- H *PID* evidenzia il processo specificato da *PID* ed i suoi genitori;
- l visualizza le linee lunghe anziché troncarle a 132 caratteri;
- n ordina i processi con lo stesso genitore in base al loro PID, anziché in base al loro nome;
- p visualizza anche i PID dei processi;
- u visualizza le eventuali variazioni di UID tra un processo ed il suo genitore;
- U utilizza i caratteri UTF-8 (Unicode) per la visualizzazione dei segmenti dei sottoalberi;
- V visualizza la versione di **pstree**;

Se il processo ha subito uno swap out, il nome relativo viene visualizzato tra parentesi.

L'elenco dei processi esistenti sul sistema si può ottenere per mezzo del comando **ps** (**process status** – man page **ps(1)**) o con il comando **top** (man page **top(1)**).

Comando: **ps**
 Path: **/bin/ps**
 SINTASSI
\$ ps [option]

DESCRIZIONE

option indica la modalità di funzionamento di **sync**. Può assumere i seguenti valori

- A seleziona tutti i processi;
- N | **--deselect** nega una selezione;
- a seleziona tutti i processi collegati ad un terminale, tranne i leader di sessione ???;
- d seleziona tutti i processi, tranne i leader di sessione ???;
- e seleziona tutti i processi;
- T seleziona tutti i processi collegati al terminale corrente;
- a seleziona tutti i processi collegati ad un terminale, inclusi quelli appartenenti ad altri utenti;
- g seleziona tutti i processi ed i relativi group leader;
- r seleziona soltanto i processi in stato running;
- x seleziona i processi che non sono collegati ad un terminale ???;
- Group** seleziona i processi in base al real groupname o GID;
- User** seleziona i processi in base al real username o UID;
- group** seleziona i processi in base all'effective groupname o GID;
- user** seleziona i processi in base all'effective username o UID;
- pid** | 123 seleziona i processi in base al PID;
- sid** | -123 seleziona i processi in base al SID;
- tty** seleziona i processi in base al terminale (TTY);
- 0 | 0 ???;
- c ???;
- f richiede un elenco completo;
- j | j richiede di utilizzare il formato dei job;
- l | 1 richiede il formato di visualizzazione esteso;
- o | o | **--format** utilizza un formato di output definito dall'utente;

```

-y      non visualizza i flag e visualizza rss al posto dell'indirizzo ???;
X       utilizza il formato dei registri del 386 ???;
s       utilizza il formato di visualizzazione dei segnali ???;
u       utilizza il formato di visualizzazione orientato agli utenti ???;
v       utilizza il formato di visualizzazione della memoria virtuale ???;
--help  visualizza un aiuto sommario di ps;
--version
        visualizza la versione di ps;

```

???

A differenza di **ps** che visualizza la situazione attuale dei processi, **top** visualizza ciclicamente lo stato dei processi ogni 5 secondi e mette a disposizione dei comandi interattivi per la gestione dei processi stessi.

```

Comando: top
Path: /usr/bin/top

SINTASSI
$ top [option]

```

DESCRIZIONE

???

???

Ogni processo è caratterizzato da alcune proprietà ed in particolare gli vengono assegnati determinati privilegi sul sistema, in funzione (generalmente) dell'utente che lo ha lanciato. Tra le proprietà di un processo ci sono le seguenti:

- l'identificatore del processo (PID);
- lo stato del processo;
- la priorità di esecuzione del processo;
- il time-slice assegnato al processo per l'utilizzo della CPU;
- il nome del file eseguibile lanciato;
- eventuali argomenti passati all'eseguibile al momento dell'avvio attraverso la riga di comando;
- l'identificatore del processo genitore (PPID);
- il nome del dispositivo di comunicazione tramite il quale è stato lanciato il processo (se controllato da un terminale);
- il **real UID**, ovvero l'UID dell'utente che ha lanciato il processo; *real UID*
- il **real GID**, ovvero il GID dell'utente che ha lanciato il processo; *real GID*
- l'**effective UID**, ovvero l'UID dell'utente proprietario del file che ha generato il processo; *effective UID*
- l'**effective GID**, ovvero il GID dell'utente proprietario del file che ha generato il processo; *effective GID*
- il **saved UID**, ovvero la copia del valore dell'*effective UID* al momento dell'avvio del processo; *saved UID*
- il **saved GID**, ovvero la copia del valore dell'*effective GID* al momento dell'avvio del processo; *saved GID*
- il **filesystem UID**, ovvero una copia del valore dell'*effective UID*; *filesystem UID*

- il **filesystem GID**, ovvero una copia del valore dell'*effective GID*;

filesystem GID

In genere il *real UID* coincide con l'*effective UID* (ed analogamente il *real GID* coincide con l'*effective GID*), ma se un file ha impostato il bit SUID (v. cap. 3), le proprietà di tipo *real* si riferiscono all'utente che ha lanciato l'esecuzione del file, mentre quelle di tipo *effective* si riferiscono all'utente proprietario del file stesso.

I sistemi Unix-like che si attengono alle specifiche POSIX, utilizzano anche le proprietà *saved UID* e *saved GID* che sono una copia dei valori delle relative proprietà di tipo *effective*.

Le proprietà di tipo *effective* caratterizzano il processo, ovvero determinano i privilegi che esso ha sul sistema.

Le proprietà di tipo *filesystem* sono utilizzate per l'accesso ai file, ovvero determinano i permessi del processo sul filesystem: il processo apre i file come se fosse l'utente identificato dall'*effective UID* (appartenente al gruppo *effective GID*). In particolare, quando un file viene creato, il proprietario è l'utente identificato dall'*effective UID*. Esse sono state introdotte da GNU/Linux per rendere l'accesso al NFS¹² più sicuro. In genere tali valori coincidono con quelle di tipo *effective* poiché una qualunque variazione di esse si riflette su quest'ultime, ma c'è un caso in cui le proprietà di tipo *filesystem* sono diverse dalle relative di tipo *effective*: quando il server NFS deve impostare i privilegi di accesso al NFS (in questo modo il server NFS concede agli utenti gli opportuni privilegi soltanto per l'accesso al NFS, ma non per la comunicazione con il server NFS stesso).

Il kernel di GNU/Linux permette l'accesso ai dati contenuti nella struttura dei processi per mezzo di un file system virtuale montato nella directory `/proc`. Tali dati sono utilizzati dalla maggior parte dei programmi per la gestione dei processi.

thread

Ogni percorso di esecuzione di un processo è detto **thread**. Un processo è costituito da almeno un thread che coincide con il processo stesso. Come per i processi, un thread ne può creare un altro. Per ogni thread relativo ad un processo non vengono create tutte le strutture dati necessarie alla creazione di un processo, ma un thread risulta molto più "leggero" (nel senso che occupa meno memoria) di un processo, poiché tutti i thread relativi ad un processo utilizzano le stesse strutture dati che caratterizzano il processo di cui essi fanno parte.

Per il fatto che il sistema sia multitasking e multithreading nasce l'esigenza della sincronizzazione delle operazioni effettuate dai vari task e thread. L'argomento è piuttosto complesso ed è trattato in modo approfondito in [2]. A tale scopo l'accesso dei thread o processi alle risorse è regolato da meccanismi di mutua esclusione (semafori, mutex, ...). Inoltre, poiché l'esecuzione di un thread può essere interrotta in qualunque momento (ad esempio perché termina il suo time slice di utilizzo della CPU), si introduce il concetto di **operazione atomica**, ovvero un'insieme di operazioni che un thread può eseguire con la garanzia da parte del sistema che non sarà interrotto durante l'esecuzione (tutto il gruppo di operazioni viene eseguito nello stesso time slice). Dal punto di vista di programmazione il controllo di più thread è più complesso rispetto al controllo di più processi.

operazione atomica

6.3.1 Creazione

Come è già stato accennato, la nascita di un processo può avvenire solo tramite una richiesta da parte di un altro processo già esistente, utilizzando la chiamata di sistema `fork` (man page `fork(2)`) (per esempio, quando si avvia un programma attraverso il terminale, è la shell che genera il processo relativo al programma lanciato). Quando viene creato un processo, viene creata in memoria centrale la struttura dati `task_struct` definita in `include/linux/sched.h`. I processi sono raggiungibili dal sistema attraverso due meccanismi diversi:

- una tabella `hash13 pidhash[]` (la funzione di hash è basata sul PID del processo) definita nel file `include/linux/sched.h`;

¹²il NFS verrà trattato nel cap. 16.

¹³una tabella `hash` è un insieme di elementi indicizzato sulla base di una chiave in maniera tale che la correlazione tra l'indice e la chiave stessa (realizzata per mezzo di una routine detta *funzione hash*)

- una lista ciclica doppiamente concatenata¹⁴ (questo permette di accedere facilmente ai vari processi uno dopo l'altro).

Il numero massimo di processi contemporanei che possono esistere su un sistema GNU/Linux è limitato soltanto dalla quantità di memoria fisica presente sulla macchina. Per esempio in un PC con 256 MiB di RAM, possono coesistere 32.768 processi¹⁵. Inoltre tale limite può essere modificato durante il funzionamento del sistema per mezzo dell'opportuna chiamata alla funzione di libreria `sysctl` (man page `sysctl(2)`) o modificando il valore contenuto nel file virtuale `/proc/sys/kernel/threads-max`.

Nella release del kernel 2.2 i processi erano raggiungibili dal sistema operativo attraverso il *task vector*, un vettore di puntatori in cui ogni elemento puntava ad una struttura `task_struct`. Questo limitava il numero di processi contemporanei presenti su una macchina alla dimensione massima di tale vettore, ovvero a 512 elementi.

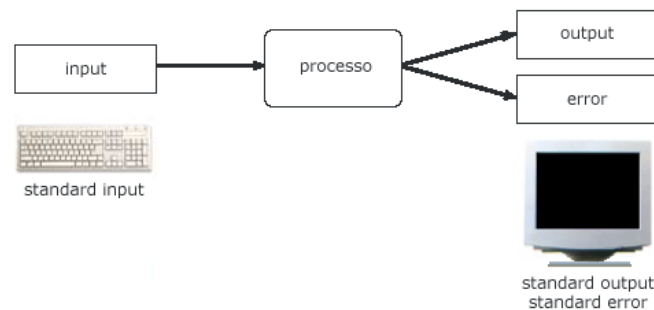


Figura 6.1: I canali di *input*, *output* ed *error* di un processo.

Ad ogni processo, come illustrato in fig. 6.1, vengono assegnati automaticamente dal sistema un canale (file) per l'**input**, ovvero il canale attraverso il quale il processo potrà ricevere gli eventuali dati da elaborare, uno per l'**output**, cioè il canale attraverso il quale il processo potrà fornire gli eventuali risultati, ed uno per l'**error**, che, come indica il nome, sarà utilizzato dal processo per fornire eventuali messaggi relativi ad errori riscontrati durante l'esecuzione. Lo *standard input* è associato al buffer di tastiera ed è individuato dal *file descriptor* 0, lo *standard output* è associato al buffer dello schermo ed è individuato dal *file descriptor* 1 e lo *standard error* è anch'esso associato al buffer dello schermo ed è individuato dal *file descriptor* 2.¹⁶

Ad ogni processo viene assegnato anche un **ambiente** (*environment*), ovvero un apposito spazio di memoria che contiene le impostazioni di variabili, dette appunto variabili di ambiente (v. ???), alle quali il processo può fare riferimento. Ad esempio ...

???

È opportuno sottolineare il fatto che lo spazio di memoria riservato all'ambiente è limitato, quindi l'utilizzo di un numero molto elevato di variabili di ambiente oppure l'utilizzo di variabili di ambiente che fanno uso di molto spazio di memoria, può facilmente fuoriuscire da tale spazio di memoria, causando problemi.

sia minima. Ogni elemento della tabella occupa una posizione (all'interno della tabella stessa) stabilita dalla funzione hash sulla base di una chiave.

¹⁴una *lista* è un insieme di elementi concatenati l'uno con l'altro in sequenza (in genere tramite l'utilizzo di puntatori) in modo tale che iniziando dal primo è possibile raggiungere sequenzialmente tutti gli elementi. Una lista è detta *ciclica* quando l'ultimo elemento è collegato col primo e *doppiamente concatenata* significa che gli elementi possono essere raggiunti in modo sequenziale uno dopo l'altro in entrambi i sensi: dal primo elemento all'ultimo o viceversa.

¹⁵poiché la variabile che contiene tale valore è di tipo `int` (sintassi C), ha la dimensione di 2 byte (sui sistemi a 32 bit) e può memorizzare valori con segno, questa va da -32.768 a 32.767.

¹⁶i *file descriptor* saranno trattati nel cap. 3.

6.3.2 Gestione della memoria

spazio di indirizzi

memoria virtuale

paging

swap

page fault

Un sistema Unix-like, utilizza il meccanismo della *memoria virtuale* che consiste nell'assegnare ad ogni processo uno **spazio di indirizzi** di memoria (indirizzi che vanno da 0 ad un valore massimo – in genere 4 GiB). Tale spazio di indirizzi è *virtuale*, nel senso che tali indirizzi non corrispondono agli indirizzi fisici delle celle di memoria centrale. Si parla pertanto di **memoria virtuale** e questa è quella che “vede” un processo.

Come già accennato precedentemente, il kernel utilizza il meccanismo di paginazione della memoria (**paging**), ovvero suddivide la memoria in pagine di dimensione fissa (in genere 4 KiB). Ogni pagina di memoria virtuale viene quindi assegnata ad una pagina di memoria fisica, ma poiché la memoria centrale potrebbe non essere sufficiente a mappare la memoria virtuale assegnata ad ogni processo (in genere la memoria fisica è molto minore di quella virtuale), alcune pagine di memoria virtuale vengono mappate sulla memoria di massa. È necessario un meccanismo che carichi le pagine mappate sulla memoria di massa in memoria centrale e viceversa: lo **swap**. Quando un processo vuole accedere ad una pagina di memoria virtuale che fa riferimento ad una pagina che non risiede (in quel momento) in memoria centrale, si verifica un **page fault**, ovvero l'hardware di gestione della memoria (il controller) genera un'interrupt che fa passare il controllo del sistema al kernel, che sospende il processo corrente ed esegue le operazioni necessarie a caricare la relativa pagina in memoria centrale. Quindi il kernel fa ripartire l'esecuzione del processo dal punto in cui era stato interrotto. Dal punto di vista del processo, il meccanismo della gestione della memoria è completamente trasparente, ovvero per il processo stesso tutto avviene come se la memoria fisica fosse effettivamente quella a lui assegnata: la memoria virtuale. La differenza che si avverte è quella relativa ai tempi di esecuzione del processo che saranno nettamente superiori a quelli che si avrebbero avendo a disposizione la quantità di memoria fisica pari a quella della memoria virtuale assegnata al processo stesso¹⁷.

La stessa quantità di memoria centrale (o di massa) può fare da supporto a più pagine di memoria virtuale (appartenenti a processi diversi) in modo da minimizzare la quantità di memoria fisica utilizzata. Questo è possibile se le informazioni contenute in tali pagine non sono modificabili da nessun processo (come il codice da eseguire): per esempio il codice della routine per la visualizzazione di una stringa sullo schermo, utilizzato dai programmi scritti in C con la chiamata alla funzione **printf**, è contenuto in una zona della memoria fisica che farà da supporto a tutte le pagine di memoria virtuale dei vari processi che ne fanno uso.

Esiste anche la possibilità di bloccare il meccanismo di swap, mantenendo le pagine in memoria centrale per il supporto delle opportune pagine virtuali, senza scaricarle sulla memoria di massa. Questo può essere necessario per processi particolari che necessitano che la velocità di esecuzione non subisca delle variazioni poco controllabili.

segmenti

La memoria virtuale assegnata ad un processo è suddivisa in **segmenti**, ovvero in blocchi di indirizzi contigui. In genere un processo viene diviso nei seguenti segmenti:

code segment (o *text segment*) - Contiene il codice che il processo deve eseguire ed in genere viene condiviso fra tutti i vari processi che eseguono lo stesso codice (lo stesso programma o la stessa parte di routine di libreria). Questo segmento viene reso accessibile in sola lettura per evitare sovrascritture accidentali (o meno).

data segment - Contiene i dati del processo (costituiti da quelle che i linguaggi di programmazione definiscono le variabili globali e le variabili statiche).

heap segment - Lo si può considerare un'estensione del *data segment*, poiché è adiacente a quest'ultimo. La memoria allocata dinamicamente dal processo viene gestita in questo segmento. La dimensione di tale segmento varia in funzione dalla necessità contingente di memoria allocata dinamicamente.

stack segment - Contiene lo stack del processo. Lo stack (pila) è una parte della memoria gestita in maniera sequenziale, cioè come una pila: l'ultima informazione

¹⁷l'incremento di tali tempi è dovuto principalmente alla velocità di accesso alla memoria di massa

(in senso cronologico) memorizzata in questa area di memoria è la prima ad essere estratta da questa. Lo stack viene utilizzato per il passaggio di parametri quando il processo chiama una funzione e per gestire le variabili locali (automatiche) della funzione chiamata. La dimensione del segmento aumenta man mano che lo stack del processo cresce, ma non viene ridimensionata se lo stack diminuisce.

Sebbene l'intervallo degli indirizzi relativi alla memoria virtuale assegnata ad un processo sia molto ampio, solo una parte di tali indirizzi è effettivamente assegnato alla memoria centrale (allocato) e quindi utilizzabile dal processo stesso. Se un processo tenta di accedere ad un indirizzo di memoria non allocato, si verifica un errore di *violazione di segmento* o **segmentation fault**.

segmentation fault

La quantità di memoria (centrale + swap) libera e occupata sul sistema è visualizzabile con il comando **free** (man page **free(1)**), che si basa sul contenuto del file **/proc/meminfo**.

Comando: **free**
 Path: **/usr/bin/free**
 SINTASSI
\$ free [option]

DESCRIZIONE

option indica la modalità di funzionamento di **free**. Può assumere i seguenti valori

- b indica di visualizzare i valori in byte;
- k indica di visualizzare i valori in KB (default);
- m indica di visualizzare i valori in MB;
- t indica di visualizzare una riga contenente i totali;
- o indica di non visualizzare la riga relativa ai buffer;
- s *delay*
 indica di visualizzare ciclicamente il report ogni *delay* secondi;
- V indica di visualizzare la versione di **free**;

Le statistiche sulla gestione della memoria virtuale relativamente ai processi, al paging, all'attività della CPU, sono ottenibili con il comando **vmstat** (man page **vmstat(8)**).
 ??? man page **vmstat** (**/usr/bin/vmstat**) ???

6.3.3 Terminazione

Quando un processo termina, esso ritorna un valore di uscita, detto **exit status**, cioè un valore numerico di un byte. Questo in genere viene utilizzato per stabilire se il suo compito è stato svolto correttamente o meno e si utilizza la convenzione di considerare terminato con successo (*successful*) un processo che ritorna un *exit status* uguale a 0, mentre si considera terminato con un errore un processo che ritorna un *exit status* diverso da 0.

exit status

Alla sua terminazione, un processo ritorna sempre un valore al processo genitore, che può ignorare o meno, ma che comunque dovrebbe attendersi, tramite la chiamata alla funzione di libreria **wait()**. In caso contrario il processo figlio non ha la conferma che il padre abbia ricevuto il suo valore di ritorno e pertanto il kernel non lo distrugge (ovvero non rimuove dalla memoria la struttura dati ad esso relativa), ma si limita a considerarlo uno zombie. È compito del processo genitore eliminare le tracce del processo figlio. Inoltre, un processo che termina potrebbe aver generato dei processi figli. Tali processi vengono "adottati" da **init**.

Può darsi che la terminazione (anomala) di un processo provochi il **core dump**¹⁸ (scarico della memoria). In pratica si ottiene la creazione di un file nella working directory, contenente l'*immagine* del processo interrotto. Questi file servono soltanto a documentare un incidente di funzionamento ed a permetterne l'analisi attraverso strumenti diagnostici opportuni (*debugger*).

core dump

¹⁸core si riferisce al fatto che nei primi elaboratori la memoria era realizzata per mezzo di nuclei di ferrite.

???

6.4 Lo scheduler

Poiché GNU/Linux è un sistema multitasking, deve gestire la presenza contemporanea di più processi che girano “contemporaneamente” sulla macchina. Questo ovviamente non è fisicamente possibile poiché generalmente le macchine hanno una sola CPU ed una CPU può eseguire soltanto un’operazione per volta¹⁹. Quindi viene adottata la tecnica di assegnazione di intervalli di tempo (*time slice*) da dedicare ad ogni singolo processo. In questo modo un processo viene eseguito fintantoché il *time slice* ad esso riservato non è terminato, dopodiché viene sospeso. Il sistema poi determina quale sia il prossimo processo da attivare ed il *time slice* da assegnargli. E così via.

scheduler

politica di scheduling

Lo **scheduler** è la parte del kernel che si preoccupa di gestire l’attivazione e la sospensione dei processi secondo una determinata logica detta anche **politica di scheduling**. La scelta del meccanismo in grado di distribuire l’opportuno *time slice* ai vari processi non è cosa banale ed è importante notare il fatto che non esiste un sistema ottimale per la temporizzazione dei processi, ma dipende in maniera essenziale dall’utilizzo che si intende fare del sistema.

preemptive multitasking

Una delle caratteristiche principali di un sistema multitasking è quella di gestire un **preemptive multitasking**, ovvero il sistema operativo assegna il tempo di CPU ai processi della durata massima del *time slice* relativo ed il processo può rilasciare la CPU anche prima che sia scaduto il suo *time slice*, ma nel caso in cui il *time slice* scada, il sistema operativo stesso (lo *scheduler*) sospende il processo e riprende il controllo del sistema, “calcolando” a quale processo cedere la CPU per il successivo *time slice*. Nei sistemi **cooperative multitasking** invece il rilascio della CPU è demandato ai singoli processi: se un processo si appropria della CPU senza mai rilasciarla agli altri, il sistema può diventare inutilizzabile, poiché lo stesso utente non riesce più ad interagire con il sistema stesso.

cooperative multitasking

In un sistema GNU/Linux un processo può trovarsi in uno dei seguenti stati (tra parentesi è riportato la lettera corrispondente visualizzata dal comando **ps**):

Runnable (R) il processo è pronto ad essere eseguito (o addirittura in esecuzione);

Interruptible sleep (S) il processo è in attesa di una risorsa del sistema (memoria, disco, ...) ma può essere interrotto da un segnale;

Uninterruptible sleep (D) il processo è in attesa di una risorsa del sistema (memoria, disco, ...) e non può essere interrotto;

Stopped (T) il processo è fermo poiché ha ricevuto un’opportuno segnale;

Zombie (Z) il processo è terminato ma il suo stato di ritorno non è stato ricevuto dal processo che l’ha generato (o lo ha adottato).

È importante tenere presente il fatto che il tempo di CPU assegnato ad un processo è soltanto quello destinato allo sfruttamento di una risorsa del sistema, ma non è detto che tale risorsa sia quella più importante per il processo (molti processi dipendono in maniera più pesante dall’I/O). Pertanto dare ad un processo una priorità elevata rispetto agli altri può, in certi casi, non portare ad un significativo incremento delle prestazioni dello stesso.

priorità dinamica

Il meccanismo di scheduling tradizionale dei sistemi Unix-like si basa sul concetto di **priorità dinamica**: un processo che ottiene il diritto di utilizzare la CPU per la durata del *time slice*, vede abbassarsi di priorità in modo tale che più o meno tutti i processi (anche quelli con priorità molto bassa) possano utilizzare la CPU per il proprio *time slice*. La priorità (dinamica) di un processo viene indicata con un numero intero decrescente: ad una priorità più elevata corrisponde un valore più basso.

priorità assoluta

Lo standard POSIX ha introdotto anche il concetto di **priorità assoluta** (o *statica*),

¹⁹esistono tecniche per il pipelining delle operazioni, che per come è strutturata internamente la CPU permettono a questa di eseguire parti di istruzioni diverse quasi nello stesso momento.

per tener conto dei sistemi **real-time**, in cui è indispensabile che determinati processi (critici) non debbano attendere l'esecuzione di altri (non critici)²⁰. In tal senso, tra due processi che si contendono l'esecuzione (nello stato *runnable*) “vince” quello che ha la stessa priorità assoluta maggiore. La priorità assoluta viene indicata con un numero intero crescente: ad una priorità assoluta più elevata corrisponde un valore più alto. *real-time*

In genere nei sistemi GNU/Linux a tutti i processi viene assegnata la stessa priorità assoluta pari a 0 e quindi lo scheduling si basa soltanto sul meccanismo della priorità dinamica. In particolare, normalmente anche il valore di priorità dinamica è impostato a 0 per tutti i processi.

Poiché l'accesso alle risorse non è istantaneo, è possibile che un processo non esaurisca l'accesso alla risorsa nel time slice ad esso assegnato. Quindi, in qualche modo il kernel si annota il fatto che un determinato processo sta accedendo ad una specifica risorsa, segnandola come occupata, in modo tale che un altro processo non vada a modificare i dati del primo (si pensi ad esempio ad una stampante e a cosa accadrebbe se un due processi stampassero sulla stessa stampante senza nessun meccanismo di gestione di tale risorsa: si mischierebbero inevitabilmente gli output dei due processi). Dunque un ulteriore processo che desidera accedere alla stessa risorsa, anche se il suo time slice non è scaduto, viene sospeso.

L'algoritmo di scheduling, che decide a quale dei processi assegnare il prossimo time slice di utilizzo della CPU, in genere si basa sui seguenti parametri:

- la priorità assegnata al processo;
- da quanto tempo il processo non ha utilizzato la CPU;

È possibile modificare la priorità (dinamica) di un processo con il comando **nice** (man page **nice(1)**).

Comando: **nice**
Path: **/bin/nice**

SINTASSI
\$ nice [option] [command [args]]

DESCRIZIONE

option specifica il modo di funzionamento del comando. Può assumere i valori seguenti:

- n adjust** in tal caso **adjust** rappresenta il valore da sommare algebricamente a quello relativo alla priorità (dinamica) del processo creato dall'esecuzione di **command** e va da -20 (priorità massima) a 19 (priorità minima). Se non è specificato si assume il valore 10 (solo il superuser può specificare dei valori negativi);
- adjustment=adjust** identico al caso **-n**;
- help** visualizza la documentazione minima di utilizzo di **nice**;
- version** visualizza la versione di **nice**.

command è il comando da lanciare con la priorità specificata da **adjust**;

args sono gli eventuali argomenti passati a **command** sulla riga di comando.

Il nome del comando deriva dal fatto che in genere questo viene utilizzato per ridurre la priorità di un processo, come “cortesía” nei confronti degli altri.

Esiste anche la possibilità di modificare il valore della priorità (dinamica) di un processo mentre questo è in esecuzione. Questo può essere fatto con il comando **renice** (man page **renice(8)**).

²⁰un sistema è detto *real-time* quando è in grado di eseguire operazioni in tempi determinabili priori. Tali sistemi si suddividono in *hard real-time* e *soft real-time* (in cui qualche sfioramento sporadico alla regola è ammesso).

Comando: **renice**

Path: **/usr/bin/renice**

SINTASSI

\$ renice priority **[[-p] pid ...]** **[[-g] pgrp ...]** **[[-u] user ...]**

DESCRIZIONE

priority è il valore da sommare algebricamente a quello relativo alla priorità (dinamica) e va da -20 (priorità massima) a 19 (priorità minima) (solo il superuser può specificare dei valori negativi);

pid è l'eventuale PID del processo di cui si desidera modificare la priorità;

pgrp è l'eventuale PGID del gruppo di processi di cui si desidera modificare la priorità;

user è l'eventuale username dell'utente dei quali processi si desidera modificare la priorità;

???

6.5 La comunicazione tra processi

IPC

I processi possono comunicare tra loro e con il kernel per coordinare le loro attività. Questo avviene per mezzo di opportuni meccanismi che vanno sotto la sigla **IPC** (Inter-Process Communication), che in GNU/Linux sono costituiti essenzialmente da i segnali, pipe e socket, sebbene siano supportati anche i meccanismi di comunicazione propri di System V.

6.5.1 I segnali

I segnali sono uno dei meccanismi di comunicazione più vecchio utilizzato sui sistemi Unix-like, per segnalare in maniera asincrona l'occorrenza di un evento ad un processo. I segnali sono sostanzialmente degli interrupt software.

Un segnale può essere inviato (generato) tramite la tastiera o direttamente dal sistema, come nel caso in cui un processo tenti di accedere ad una locazione di memoria al di fuori della sua memoria virtuale.

È bene chiarire il fatto che i segnali sono generati o inviati da un processo che sarà indicato come mittente e sono notificati, consegnati (*delivered*) ad un processo che sarà indicato come destinatario. In particolare un segnale è detto consegnato (*delivered*) al processo, quando questo viene effettivamente preso in considerazione dal processo stesso, mentre per tutto il tempo in cui il segnale è inviato (o generato) ma non ancora consegnato, si parla di segnale pendente (*pending*). Questa è soltanto un'astrazione. Non vi è nessun messaggio che viaggia da un processo all'altro, ma il processo destinatario si accorge soltanto che gli è stato notificato un segnale senza poter neanche sapere chi glielo ha inviato.

L'elenco dei segnali che possono essere inviati ad un processo può essere ottenuto per mezzo del comando **kill** (man page **kill(1)**), che è il comando utilizzato per inviare i segnali ai processi.

Comando: **kill**

Path: **/bin/kill**

SINTASSI

\$ kill [option] [process ...]

DESCRIZIONE

option è l'indicazione dell'opzione di funzionamento di **kill**. Può assumere uno dei seguenti valori

-s *signal*
 specifica il segnale (*signal*) da inviare, espresso con il suo nome o identificativo numerico;

-a
 non restringe la ricerca del PID ai comandi con lo stesso UID di **kill**;

-p
 indica che **kill** deve soltanto visualizzare il PID del processo indicato col nome, ma non deve inviargli nessun segnale;

-l [*signal*]
 visualizza l'elenco dei segnali (l'elenco si trova anche nel file `/usr/include/linux/signal.h`);

process è il processo (o l'elenco dei processi) a cui deve essere inviato il segnale *signal*. Il processo può essere specificato con una delle sintassi seguenti:

num
 dove *num* è un numero positivo diverso da 0 che rappresenta il PID del processo;

0
 indica tutti i processi appartenenti al PGID corrente (sessione corrente);

-1
 indica tutti i processi con PID > 1;

-num
 con *num* > 1, indica tutti i processi con PGID = *num*;

commandname
 indica tutti i processi lanciati dal comando *commandname*;

L'elenco dei segnali su di un elaboratore X386 è riportato in tab. 6.9

Il numero massimo dei segnali supportati è vincolato dalla dimensione del data bus del sistema: le CPU con 32 bit di data bus, possono avere fino a 32 segnali, mentre CPU a 64 bit (come gli Alpha) possono gestire fino a 64 segnali diversi.

Per ogni segnale è associata dal kernel un'azione predefinita da intraprendere, ma i processi possono specificarne delle proprie. Dunque, per ogni segnale, un processo può attuare una delle seguenti politiche

- accettare l'azione predefinita (dal kernel) per la gestione del segnale;
- ignorare il segnale;
- gestire autonomamente la notifica del segnale;

dipendentemente da com'è scritto il programma relativo.

In particolare, i segnali **SIGSTOP** e **SIGKILL** non possono essere né ignorati né gestiti autonomamente da un processo, ma la loro gestione è effettuata direttamente dal kernel, ovvero viene intrapresa l'azione predefinita: in corrispondenza della consegna di un segnale **SIGSTOP**, il processo destinatario viene sospeso, mentre alla consegna del segnale **SIGTERM** il processo destinatario viene immediatamente terminato.

I processi accettano implicitamente l'azione predefinita del kernel per la gestione dei segnali, ma nella scrittura del programma può essere specificata una politica diversa. Questo avviene definendo nel programma un **signal handler**, ovvero una specifica routine di gestione del segnale. In tal caso si parla di *intercettazione* del segnale. *signal handler*

In genere per i segnali che rappresentano eventi relativi ad errori di un processo (divisione per zero o violazioni di accesso, come ad esempio per il segnale **SIGFPE**) è definita un'azione predefinita che implica la scrittura di un file **core** (*core dump*) che annota lo stato del processo (ed in particolare della memoria ad esso relativa) prima di far terminare il processo stesso. Tale file può essere esaminato in seguito con un *debugger* (un apposito programma per l'analisi del funzionamenò di altri programmi) per investigare sulla causa dell'errore.

L'ordine di consegna dei segnali non è determinato dall'ordine in cui sono stati generati, ma può darsi che sia consegnato prima quello che è stato generato dopo.

Non tutti i processi hanno la possibilità di inviare segnali ad altri processi, ma i segnali possono essere inviati a processi che hanno gli stessi effective UID e GID del processo mittente oppure il suo stesso PGID. In particolare i processi che girano con priorità di superuser (UID = 0) possono inviare segnali a qualunque processo.

N.	Nome	Descrizione	Azione
1	SIGHUP	Hang up - terminazione del processo di controllo	A
2	SIGINT	Interrupt - interruzione del processo (^C)	A
3	SIGQUIT	Quit - interruzione del processo (^Y)	A
4	SIGILL	Illegal instruction - istruzione non permessa	C
5	SIGTRAP	Notifica raggiungimento di breakpoint	C
6	SIGABRT	Abort - generato dalla funzione <code>abort()</code>	C
7	SIGBUS	Bus error - Errore sul bus (bad memory access)	C
8	SIGFPE	Floating point exception - errore aritmetico	C
9	SIGKILL	Kill - terminazione immediata del processo	AX
10	SIGUSR1	User signal 1	A
11	SIGSEGV	Segment violation - errore di accesso alla memoria	C
12	SIGUSR2	User signal 2	A
13	SIGPIPE	Pipe interrotta (tentativo di apertura in scrittura di una pipe non ancora aperta in lettura)	A
14	SIGALRM	Alarm - Timer scaduto (funzione <code>alarm()</code>)	A
15	SIGTERM	Terminate - terminazione del processo (^X)	???
17	SIGCHLD	Child - Processo figlio terminato o sospeso	I
18	SIGCONT	Continue - riprende l'esecuzione del processo se sospeso	
19	SIGSTOP	Stop (sospende il processo)	SX
20	SIGTSTP	Interactive stop (^Z)	S
21	SIGTTIN	Tentativo di lettura dello standard input da parte di un processo in background	S
22	SIGTTOU	Tentativo di scrittura sullo standard output da parte di un processo in background	S
23	SIGURG	Urgent - Notifica di una <i>urgent condition</i> su un socket	I
24	SIGXCPU	Ecceduto il limite del time slice di CPU assegnato al processo	C
25	SIGXFSZ	Ecceduto il limite massimo della dimensione del file	C
26	SIGVTALRM	Virtual alarm - Allarme virtuale	A
27	SIGPROF	Profiling - Timer del profiling scaduto	A
28	SIGWINCH	Finestra ridimensionata (interfaccia grafica)	I
29	SIGIO	Input/Output - Notifica che si è pronti ad un'operazione di input/output	A
30	SIGPWR	Mancanza di alimentazione elettrica	A
31	SIGSYS	System call error - Errore nella chiamata ad una subroutine	C
32	SIGRTMIN	Real time minimum signal	???
63	SIGRTMAX	Real time maximum signal	???

Tabella 6.9: Elenco dei segnali inviabili ai processi.

Gestione

GNU/Linux implementa i segnali sfruttando le informazioni memorizzate nella struttura dati `task_struct` del processo. I segnali pendenti per un determinato processo, vengono tenuti nel campo `signal` della struttura `task_struct` segnati come bloccati (ad eccezione dei segnali `SIGSTOP` e `SIGKILL` che non possono essere bloccati).

Il sistema tiene traccia di come ogni processo gestisce i vari segnali nel vettore `sigaction` puntato dalla struttura `task_struct`, accessibile dal processo mediante opportune system call.

I segnali sono recapitati impostando il relativo bit nel campo `signal` della struttura `task_struct` del processo di destinazione.

Se il processo di destinazione non ha bloccato il segnale ed è nello stato *interruptible*, allora viene attivato cambiando il suo stato in *running* in modo tale che lo scheduler possa considerarlo uno dei candidati per l'assegnamento del prossimo time slice di CPU.

I segnali non sono presentati al processo destinatario nel momento che vengono generati, ma devono attendere che il processo sia nello stato *running*.

Ogni volta che un processo esce da una system call, i suoi campi `signal` e `blocked` sono controllati e quelli non bloccati gli vengono segnalati, ovvero viene intrapresa l'azione specificata in corrispondenza di tali eventi, che può essere quella specificata dal

Azione	Descrizione
A	termina il processo
I	ignora il segnale
C	termina il processo e scrivi un file core
S	sospendi (blocca) il processo
X	il segnale non può essere ignorato o intercettato dal processo

Tabella 6.10: Legenda delle azioni predefinite dei processi.

processo oppure, in sua assenza, quella di default del kernel. Potrebbe sembrare un meccanismo inaffidabile, ma i processi generalmente chiamano le system call, anche soltanto per scrivere dei caratteri sullo schermo.

In realtà è possibile che i segnali vengano controllati anche all'inizio di alcune system call, dipendentemente dal fatto se queste sono considerate lente o veloci dal sistema. Quelle veloci (fast) sono eseguite in maniera praticamente immediata, mentre quelle lente (slow) sono quelle per cui l'esecuzione potrebbe richiedere un tempo indefinito (lettura/scrittura di pipe, file di dispositivo, socket, ...).

Nel caso in cui il processo voglia gestire un particolare segnale, esso comunica al kernel l'indirizzo di memoria a cui ha inizio la routine di gestione del segnale. Quindi, all'arrivo di un segnale, o meglio alla fine di una chiamata ad una system call, il kernel si accorge che a tale processo è stato inviato un segnale. Poiché il processo in esecuzione in questo momento sta girando in kernel space, il kernel non può lanciare direttamente l'esecuzione della routine di gestione del segnale, poiché si trova ad un indirizzo di memoria nello spazio di indirizzi del processo. Il kernel si limita a cambiare l'indirizzo della prossima istruzione di codice eseguita dal processo, in modo tale che al successivo time slice di CPU assegnato al processo, il processo inizi ad eseguire il codice relativo alla gestione del segnale arrivato.

Mentre un processo sta eseguendo la routine di gestione di un segnale, il kernel impedisce la gestione di altri segnali, bloccandoli, che comunque rimangono in coda. Quando il processo termina la routine di gestione di un segnale, il kernel provvede a sbloccare la notifica dei segnali arrivati, riabilitando il processo a gestire eventuali segnali arrivati nel frattempo.

Poiché i segnali sono asincroni è possibile incorrere in eventuali *race condition*²¹.

6.5.2 Le pipe

Un comando può essere rediretto utilizzando le **pipe**²², specificando sulla riga di comando il carattere '|' (detto appunto *pipe*), come nell'esempio seguente:

```
$ ls | pr | lpr
```

In questo modo l'output del comando **ls**, cioè l'elenco dei file contenuti nella working directory, viene rediretto nell'input del comando **pr** che si occupa di paginarla. Quindi, l'output del comando **pr** viene rediretto nell'input del comando **lpr** che provvede ad inviare la stampa alla stampante di default. I singoli processi non si accorgono della redirezione del loro input/output, ed eseguono tutto come se non fosse richiesta alcuna redirezione. È la shell che si occupa di gestire la redirezione, creando le opportune pipe.

In GNU/Linux, una pipe è implementata utilizzando, per ogni processo, due strutture file (*file descriptor*) che puntano entrambe allo stessa zona di memoria (buffer) del kernel. In particolare uno dei due *file descriptor* accede al buffer del kernel (pipe) in lettura e l'altro in scrittura. Quando viene generato un processo figlio, questo condivide

²¹comportamenti imprevedibili dei programmi che non trattano correttamente la gestione asincrona degli eventi.

²²la redirezione dei comandi verrà trattata nel cap. 7.

i file descriptor del padre, quindi se uno dei due processi scrive su una pipe, l'altro può leggervi. Il kernel deve comunque garantire che vi sia una sincronizzazione tra le operazioni di lettura e scrittura su di una pipe e lo fa utilizzando meccanismi di lock, code di attesa e segnali.

Un processo accede in lettura e scrittura ad una pipe per mezzo delle routine di lettura e scrittura sui file, ma in realtà non viene occupato spazio sul filesystem.

Per utilizzare tale meccanismo di IPC è necessario quindi che i processi possano condividere i file descriptor associati alla pipe, pertanto essi devono comunque derivare dallo stesso processo padre in cui è avvenuta la creazione della pipe (in tal caso i processi sono detti *sibling*) o essere uno il figlio dell'altro.

6.5.3 Le FIFO

GNU/Linux supporta anche le pipe con nome (*named pipe*) dette anche FIFO, poiché esse operano come delle code, ovvero sono gestite in maniera tale che il primo dato inserito in esse è anche quello che viene letto per primo (First In, First Out). Una FIFO è analoga ad una pipe semplice, con la differenza che ad essa è associato un inode sul filesystem. Questo permette ai processi di poterle utilizzare anche senza avere un processo progenitore comune, che invece è necessario per l'utilizzo delle pipe semplici. Le informazioni scambiate tra i processi che utilizzano una FIFO, passano attraverso un buffer interno al kernel, senza transitare sul filesystem: l'inode associato sul filesystem serve soltanto come riferimento ai processi per l'accesso alla FIFO.

Per ogni FIFO aperta, il kernel mantiene una pipe. Una FIFO deve essere aperta sia in scrittura che in lettura affinché i dati possano transitare attraverso di essa. In genere l'apertura di una FIFO (in modalità di lettura o scrittura) rimane bloccata fintantoché essa non viene aperta nella modalità complementare (scrittura o lettura). Un processo può comunque aprire una FIFO in modalità non bloccante e nel caso la FIFO non sia già aperta nella modalità complementare a quella desiderata dal processo, il processo stesso riceve un segnale `SIGPIPE`.

6.5.4 I socket

Un meccanismo di comunicazione tra processi analogo alle pipe ma che non presenta il problema della unidirezionalità del flusso dei dati è rappresentato dai socket. Un socket inoltre consente la comunicazione tra processi in esecuzione anche su sistemi diversi, collegati in rete, infatti esso costituisce la principale interfaccia per la programmazione di rete utilizzata su tutti i sistemi operativi. Pertanto l'argomento sarà trattato più approfonditamente nel cap. 16.

Un socket è caratterizzato da un *dominio* o *protocol family*, da un *tipo* o *stile* e da uno specifico *protocollo* di comunicazione. Il dominio identifica un'insieme di protocolli (o stack di protocolli), come il TCP/IP (v. ???), il tipo indica il tipo di comportamento nella comunicazione (sfruttamento del canale, affidabilità, ...) ed il protocollo è il formato dei pacchetti da inviare e le relative regole per la comunicazione.

Va sottolineato comunque il fatto che questo tipo di comunicazione permette di avere un messaggio di lunghezza variabile e di inserire nel messaggio stesso tutta una serie di informazioni aggiuntive relative al mittente ed al destinatario del messaggio.

6.5.5 I meccanismi di IPC di System V

Visto il limite intrinseco ai meccanismi di IPC rappresentati dalle pipe e dalle FIFO, ovvero il fatto che queste forniscono dei meccanismi di comunicazione sequenziali, le rende poco flessibili per le situazioni in cui un processo vuole comunicare qualcosa a molti altri processi. Per questo sono nati altri meccanismi di IPC che fecero la loro prima comparsa sui sistemi Unix System V (1983): le *code di messaggi* (*message queues*), i *semafori* e la *memoria condivisa* (*shared memory*).

Tali meccanismi si basano su oggetti permanenti che risiedono nel kernel, i quali, a differenza di quanto avviene per i file descriptor, non mantengono un contatore dei

riferimenti che i processi fanno ad essi (un numero che indica quanti sono i riferimenti fatti a questi da parte dei vari processi), e non vengono cancellati automaticamente dal sistema una volta che non sono più utilizzati. Questo evidenzia immediatamente le seguenti problematiche:

- tali oggetti devono essere rimossi esplicitamente dalla memoria da chi ne fa uso;
- tali oggetti possono essere rimossi dalla memoria anche se esistono dei processi che li stanno utilizzando;

Tali meccanismi condividono un metodo di autenticazione comune: i processi possono accedere a tali oggetti soltanto specificando al kernel, attraverso opportune system call, un identificatore univoco. Questo è un numero progressivo assegnato dal kernel man mano che questi oggetti vengono creati (tipo il PID). Poiché tale valore viene ritornato al processo che richiede la creazione di un oggetto IPC di System V, un secondo processo può utilizzare lo stesso oggetto IPC per comunicare col primo attraverso un altro meccanismo. Il kernel associa a ciascun oggetto IPC di System V una struttura `ipc_perm` nella quale vengono memorizzati i seguenti dati:

- l'UID ed il GID del creatore dell'oggetto (`cuid` e `cgid`);
- l'UID ed il GID del proprietario dell'oggetto (`uid` e `gid`);
- la chiave dell'oggetto (`key`);
- i diritti di accesso all'oggetto (`mode`);

La chiave, specificata in fase di creazione dell'oggetto IPC, serve per poterne successivamente ricavare l'identificatore da parte dei processi.

I diritti di accesso, definiti dal proprietario dell'oggetto sono costituiti da un valore numerico che ha lo stesso significato di quello utilizzato per i file, anche se per gli oggetti IPC di System V il permesso di esecuzione non esiste e se specificato viene ignorato.

Le code di messaggi

???

I semafori

???

La memoria condivisa

???

6.6 I file di lock

???

6.7 Riferimenti

- D. A. Rusling, *The Linux Kernel Hacker's Guide*
<http://www.tldp.org/LDP/tlk/tlk.html>
- P. Gortmaker, *Boot-Prompt HowTo*
<http://www.ibiblio.org/mdw/HOWTO/BootPrompt-HOWTO.html>
- B. Henderson, *Linux Loadable Kernel Module HOWTO*
???
- ACPI - Advanced Configuration & Power Interface
<http://www.acpi.info/>

Capitolo 7

La shell ed i job

“Sobrietà e bellezza non vanno mai insieme.”
– J. W. Goethe

In questo capitolo viene trattata la shell, ovvero il meccanismo di base che permette all'utente di interagire al sistema operativo. Poiché la shell dei sistemi Unix-like è un argomento molto vasto che da solo richiederebbe la stesura di un libro, di seguito saranno trattati soltanto gli argomenti principali. L'unico modo per conoscerla a fondo è quello di utilizzarla.

7.1 La shell

Se il kernel è il nucleo del sistema, la *shell* è il guscio, ovvero il meccanismo di accesso al sistema per chi sta all'esterno. Per **shell** si intende l'interfaccia (a caratteri) tramite la quale l'utente può operare sul sistema. La shell è un programma che gestisce la comunicazione fra l'utente ed il sistema operativo, interpretando ed eseguendo i comandi dell'utente (la shell viene chiamata anche **interprete dei comandi** o *command interpreter*). Dunque, in sostanza la *shell* è un programma che attende i comandi digitati dall'utente e li interpreta secondo la propria sintassi (mette in atto un meccanismo di *parsing* dei comandi). Tale programma occupa tutto lo schermo suddividendolo generalmente in 25 righe per 80 colonne (su ogni riga possono essere visualizzati un massimo di 80 caratteri). Un esempio di shell è riportato in fig. 7.1

Per far capire all'utente che si attende l'inserimento di un comando, la shell presenta un **prompt** (invito) che tipicamente è costituito dal carattere dollaro '\$' per gli utenti che non hanno diritti amministrativi sul sistema e dal carattere cancelletto '#' per il *superuser*. Man mano che i comandi vengono impartiti ed eseguiti, la shell ripropone il prompt sulla riga seguente, rimanendo in attesa di un altro comando. Quando le righe dello schermo sono già state tutte “utilizzate” (man mano che vengono digitati i comandi, questi vengono lasciati scritti sullo schermo, come anche l'output dei vari programmi) lo schermo viene virtualmente spostato verso il basso effettuando uno **scroll**, ovvero tutte le righe vengono spostate di una posizione verso l'alto in modo da liberare l'ultima riga (chiaramente si perde la visualizzazione della prima riga, la meno recente tra quelle visualizzate sullo schermo). In realtà tale meccanismo non elimina fisicamente la prima riga, ma esiste uno **scrollback buffer** (o *buffer di shell*), ovvero una parte della memoria in cui viene tenuta traccia delle ultime righe di testo visualizzate sullo schermo. In genere il numero di righe di cui il sistema tiene traccia è superiore a quello che è visualizzato dallo schermo (generalmente il sistema tiene traccia dell'ultimo centinaio di righe di testo visualizzate). Per poter visualizzare le righe ormai “scrollate”, ovvero uscite dallo schermo, ma ancora mentenute nello scrollback buffer, è sufficiente utilizzare le combinazioni di tasti `Shift PgUp` e `Shift PgDn` che permettono la visualizzazione a pagine delle righe contenute nello scrollback buffer.

Subito dopo il prompt è visualizzato un carattere *underscore* '_' che lampeggia: il **cursore**¹. Esso indica sempre la posizione dello schermo in cui apparirà il prossimo

shell

interprete dei comandi

prompt

scroll

scrollback buffer

cursore

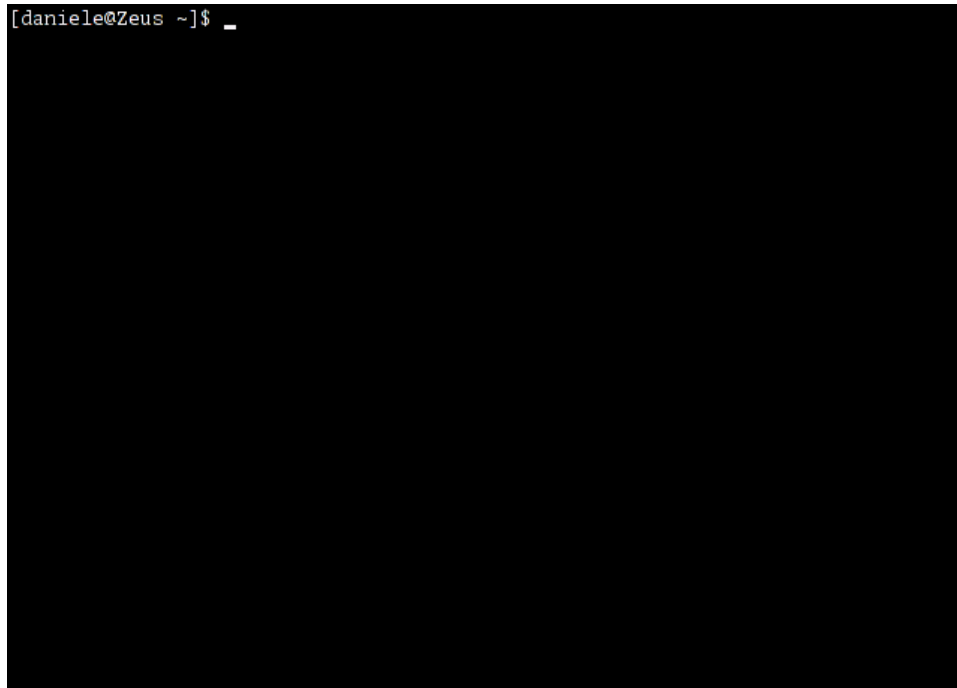


Figura 7.1: Esempio di shell.

carattere visualizzato (sia esso inserito da tastiera, che facente parte dell'output di un programma). Infatti, man mano che i caratteri vengono visualizzati sullo schermo, il cursore si sposta verso destra (ovviamente quando questo arriva al margine destro del monitor, viene riposizionato al margine sinistro della riga successiva).

scripting language
script

Ogni shell ha anche un proprio linguaggio di programmazione (**scripting language**) più o meno evoluto che permette di scrivere semplici programmi (**script**) interpretabili dalla stessa shell, molto utili per automatizzare alcune operazioni sui file.

command history

La shell in genere è anche dotata di uno storico dei comandi (**command history**), ovvero permette di richiamare velocemente i comandi più recenti digitati dall'utente.

ambiente
environment

È compito della shell gestire quello che si definisce **ambiente** o **environment**, ovvero un insieme di impostazioni generali che un programma utilizza per eseguire i propri compiti. L'*ambiente* è definito per mezzo di variabili, dette appunto **variabili d'ambiente** (*environment variable*) che la shell deve essere in grado di creare, leggere e modificare.

variabili d'ambiente

token

La shell inoltre si deve preoccupare anche della suddivisione in **token** degli argomenti passati ad un comando, ovvero deve effettuare quella che viene detta *tokenizzazione* degli argomenti. Può capitare infatti che un'applicazione possa richiedere dei parametri sulla linea di comando. L'applicazione, una volta lanciata, si troverà ad avere a che fare con gli argomenti passati sulla linea di comando: l'applicazione non dovrà gestire la suddivisione degli argomenti passati in *token* poiché questa operazione è svolta automaticamente dalla shell².

La struttura dei comandi impartiti dalla shell è la seguente:

\$ *command* [*argument* [...]]

dove

\$ rappresenta il *prompt* della shell, ovvero l'indicazione del fatto che la shell sta atten-

¹anche denominato *caret* per distinguerlo dal cursore o puntatore del mouse. Nel testo, salvo esplicito avviso contrario, sarà chiamato *cursore*.

²questa caratteristica viene apprezzata quando si ha a che fare con la scrittura dei programmi.

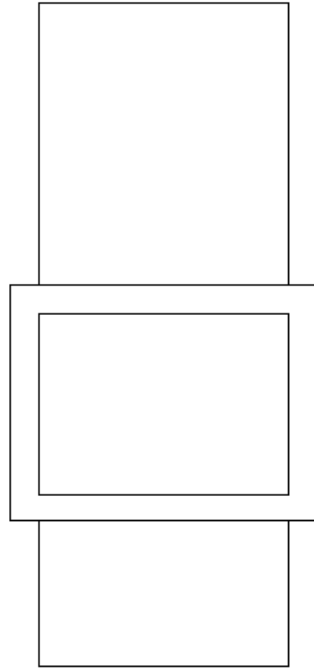


Figura 7.2: Schema della gestione dello *scrollback buffer*.

dendo la digitazione di un comando da parte dell'utente. Generalmente il prompt sui sistemi GNU/Linux è costituito da

```
[username@hostname working_directory]$
```

dove

username è il nome dell'utente che utilizza la shell;

hostname è il nome della macchina alla quale ha accesso l'utente;

working_directory è la working directory attuale;

È da notare il fatto che il simbolo utilizzato come carattere finale del *prompt* è convenzionalmente '\$' per gli utenti comuni e '#' per il superuser;

La variabile di ambiente³ nella quale è memorizzata la stringa utilizzata come prompt è PS1;

command è il comando da impartire. In genere è costituito dal nome di un file eseguibile che la shell provvede a lanciare in esecuzione, ma può essere anche il nome di un comando interno della shell utilizzata;

argument è l'eventuale elenco di argomenti da passare come input del comando *command*. Può trattarsi anche di indicazioni speciali per la redirectione dell'input o dell'output del comando⁴, ed in tal caso la shell provvede a gestire la redirectione.

Nei sistemi Unix-like è una convenzione diffusa utilizzare come argomenti che specificano delle opzioni di funzionamento di un comando, dei caratteri preceduti dal simbolo '-' (es. -a, -x, ...). Tali argomenti sono detti appunto **opzioni** e si è

opzioni

³v. sez. 7.3.3

⁴la redirectione dei comandi è trattata in sez. 7.3.11.

soliti riferirsi alla relativa notazione con il termine *short form* (forma breve). Le opzioni secondo la notazione short form possono essere indicate anche specificando l'elenco dei caratteri che le identificano di seguito al simbolo '-' (es. l'insieme delle opzioni -a e -b può essere specificato anche come -ab). Esiste anche un'altra forma convenzionale, di più recente introduzione, per l'espressione delle *opzioni* che consiste in stringhe precedute dalla sequenza di caratteri "--" (es. --name). Tale notazione è detta *long form* (forma estesa). Le opzioni specificate con la long form non possono essere raggruppate come invece può essere fatto con la short form.

Ad esempio, il comando

```
$ date -u
```

in cui è specificata l'opzione -u (short form) è identico al comando

```
$ date --utc
```

in cui l'opzione --utc è specificata secondo la sintassi più recente (long form). Il comando

```
$ date -R -u
```

indica l'utilizzo delle opzioni -R e -u. Questo può essere ottenuto anche specificando l'elenco delle lettere che indicano le opzioni, cioè "Ru", di seguito al simbolo '-', ovvero

```
$ date -Ru
```

È opportuno tenere presente che il carattere spazio ' ' (o *blank*) assume un significato particolare per la shell, infatti esso è interpretato come carattere di separazione tra i comandi e gli argomenti e tra l'elenco degli argomenti stessi da passare come input ai comandi.

Anche se l'interfaccia della shell dei sistemi Unix-like può sembrare piuttosto scarna, essa costituisce comunque un potente strumento attraverso il quale un utente può gestire tutte le funzionalità del sistema.

7.2 Le varie shell

Su GNU/Linux esistono varie shell, ognuna delle quali presenta caratteristiche e peculiarità proprie.

Bourne Shell È disponibile su qualsiasi ambiente Unix-like, quindi è la più utilizzata per creare script shell compatibili e cross-platform. Viene lanciata con il comando `sh`;

C Shell Prende il nome dal linguaggio di programmazione, ovviamente le funzionalità di tale shell derivano in modo diretto dal C. Viene lanciata con il comando `csh`;

Bourne Again Shell È una delle ultime nate ed offre le stesse capacità della *C Shell*, con l'aggiunta di alcune funzionalità come la *command history* e la *name completion*. Viene lanciata con il comando `bash`;

Korn Shell Largamente diffusa è compatibile con la *Bourne Shell* per la parte di scripting ed ha tutte le funzionalità di interazione della *C Shell* (v. <http://www.kornshell.com/>). Viene lanciata con il comando `ksh`;

Enhanced C Shell È un'evoluzione della *C Shell*, con la quale mantiene piena compatibilità e introduce feature come command line editing e name completion. Viene lanciata con il comando `tcsh`;

*Restricted Bourne Shell**Bourne Shell con Job control**Desktop Korn Shell*

Restricted Korn Shell Sono variazioni e derivazioni delle shell principali. Vengono lanciate rispettivamente dai comandi `rsh`, `jsh`, `dtksh`, e `rksh`.

7.3 Bash - Bourne Again Shell

La shell di default utilizzata dalla maggior parte delle distribuzioni GNU/Linux è la *Bourne Again Shell* (detta anche *Bash*) che può essere lanciata tramite il comando `bash` (man page `bash(1)`).

Comando: `bash`

Path: `/bin/bash`

SINTASSI

`$ bash [option] [file [argument]]`

DESCRIZIONE

option specifica delle opzioni di funzionamento della shell. Le più comuni sono le seguenti:

- `-i` lancia una shell interattiva;
- `--init-file file` | `--rcfile file`
Esegue il file *file* invece del file `~/ .bashrc` se la shell è interattiva;
- `--login` lancia una shell di login;
- `--noprofile`
indica a *bash* di non considerare il file di inizializzazione standard (per default *bash* esegue il contenuto di tali file quando è lanciata come shell di login);
- `--norc` indica a *bash* di non considerare il file di inizializzazione personale `~/ .bashrc` se è lanciata come shell interattiva.
- `-r` | `--restricted`
lancia una *restricted shell*, ovvero una shell in cui alcune operazioni non sono consentite;

file è l'eventuale file da eseguire (il file non deve necessariamente essere eseguibile);

argument è l'eventuale insieme di argomenti di input per il file *file*;

La shell è detta **shell interattiva** quando interagisce con l'utente e di conseguenza mostra il prompt come richiesta ad inserire dei comandi. Una shell è detta **shell di login** se lanciata con il parametro `--login` e questo indica a *bash* di eseguire le istruzioni contenute in particolari file al suo avvio e in altri file subito prima della sua terminazione.

La shell lanciata subito dopo la procedura di accesso al sistema è generalmente una *shell interattiva di login*.

Quando *bash* è lanciato come *shell di login*, subito dopo il suo avvio esso esegue, nell'ordine indicato, i seguenti file (se essi esistono e sono eseguibili):

1. `/etc/profile`
2. `~/ .bash_profile`
3. `~/ .bash_login`
4. `~/ .profile`

a meno che non sia stata specificata l'opzione `--noprofile`. Il file `/etc/profile` è un file di inizializzazione a livello globale (modificabile solo dal superuser), mentre i file `~/.bash_profile`, `~/.bash_login`, e `~/.profile` sono a livello di utente, ovvero l'utente in questione può personalizzarli come crede (si noti la presenza del carattere `~` che indica la home directory dell'utente);

Quando una *shell di login* sta per terminare la propria esecuzione, essa esegue il file `~/.bash_logout` (se esiste ed è eseguibile) e quindi termina.

Se *bash* è lanciato come *shell interattiva* (non di login), subito dopo il suo avvio essa esegue il file `~/.bashrc` (se esiste ed è eseguibile), a meno che non sia stata specificata l'opzione `--norc` o `--rcfile` (in quest'ultimo caso può essere specificato un altro file da eseguire).

Nel caso in cui *bash* sia lanciato tramite il comando `sh`, esso si comporta come la shell tradizionale *Bourne Shell*, ovvero subito dopo il suo avvio esegue i seguenti file nell'ordine indicato (se essi esistono e sono eseguibili):

1. `/etc/profile`
2. `~/.profile`

a meno che non sia stata specificata l'opzione `--noprofile`.

È vivamente consigliato di leggere la man page `bash(1)` per una spiegazione più completa e dettagliata del funzionamento di *Bash*.

Alcune caratteristiche della *Bourne Again Shell* (presenti anche in altre shell) sono riportate nelle sezioni seguenti.

7.3.1 Il prompt

Quando la shell funziona in modo interattivo, può mostrare due tipi di prompt:

Primario è la stringa visualizzata quando la shell è pronta a ricevere un comando (la stringa visualizzata è quella contenuta nella variabile d'ambiente `PS1`).

Secondario è la stringa visualizzata quando la shell necessita di maggiori informazioni per completare un comando (la stringa visualizzata è quella contenuta nella variabile d'ambiente `PS2`);

La stringa che costituisce il prompt può contenere delle sequenze di caratteri che vengono interpretati dalla shell come riportato in tab. 7.1.

Il *prompt primario* di default di *Bash* è

```
shell-version$
```

ovvero la variabile di ambiente `PS1` contiene la stringa `"\s-\v\$ "`, ma nei sistemi GNU/Linux, tramite il file di configurazione `/etc/profile` è impostato come

```
[username@hostname working_directory]$
```

cioè la variabile di ambiente `PS1` contiene la stringa `"[\u@\h \w]\$ "`.

Il *prompt secondario* di default di *Bash* è

```
>
```

ovvero la variabile di ambiente `PS2` contiene la stringa `"> "`.

7.3.2 Command line editing

Bash fornisce un sistema di gestione della tastiera molto complesso, con un gran numero di funzioni. Teoricamente è possibile ridefinire ogni tasto speciale e ogni combinazione

Sequenza	Significato
\a	segnale acustico (carattere ASCII BEL)
\d	data attuale (nel formato <i>DDD MMM dd</i> , dove <i>DDD</i> sono i primi tre caratteri del nome del giorno della settimana, <i>MMM</i> i primi tre caratteri del nome del mese e <i>dd</i> il numero del giorno del mese)
\D	specifica il formato (<i>format</i>) della visualizzazione della data/ora nella sintassi del comando <code>strftime</code> (man page <code>strftime(3)</code>)
\e	carattere ASCII ESC
\h	hostname
\H	hostname completo, ovvero il FQDN (Fully Qualified Domain Name) ⁵
\j	numero dei job gestiti attualmente dalla shell; \l nome del terminale sul quale è lanciata la shell
\n	interruzione di riga (<i>newline</i>), corrispondente al carattere ASCII LF
\r	interruzione di riga (<i>carriage return</i>), corrispondente al carattere ASCII CR
\s	nome della shell
\t	orario attuale (nel formato <i>hh:mm:ss</i> - 24 ore)
\T	orario attuale (nel formato <i>hh:mm:ss</i> - 12 ore)
\@	orario attuale (nel formato AM/PM - 12 ore)
\A	orario attuale (nel formato <i>hh:mm</i> - 24 ore)
\u	username dell'utente attuale
\v	versione di bash
\V	release di bash (versione + buildnumber)
\w	working directory
\W	percorso precedente alla working directory (<i>basename</i>)
\!	numero del comando attuale nella command history
\#	numero del comando attuale
\\$	# se l'utente corrente è il superuser e \$ altrimenti
\nnn	carattere corrispondente alla codifica ASCII del numero ottale indicato da <i>nnn</i>
\\	un <i>backslash</i> '\'
\[inizio di una sequenza di caratteri non stampabili (sequenza di controllo)
\]	fine di una sequenza di caratteri non stampabili (sequenza di controllo)

Tabella 7.1: Sequenze di caratteri particolari interpretabili da **bash** nella visualizzazione del prompt.

di tasti a seconda delle proprie preferenze, anche se non è consigliabile dal momento che tutto questo serve solo per gestire la riga di comando.

La shell dà la possibilità di modificare la linea di comando (**command line editing**), *command line editing* ovvero è possibile scrivere e modificare un comando, senza necessariamente doverlo scrivere di nuovo, prima di impartirlo. Questa funzionalità si rivela particolarmente utile nel caso di comandi molto lunghi.

In tab. 7.2 è riportato un elenco delle combinazioni di tasti più utilizzate in *Bash*.

Per le impostazioni del funzionamento del *command line editing*, all'avvio *Bash* legge i file `/etc/inputrc` e `~/inputrc`.

Esistono anche dei tasti che le tastiere possono non riportare, dipendentemente dal loro layout (il layout italiano riporta i simboli è, é, à, ... che non sono riportati da quello US english, il quale però ne riporta altri al loro posto). In particolare possono essere utilizzate le combinazioni di tasti riportate in tab. 7.3 per rappresentare alcuni simboli che possono non essere riportati sui tasti della tastiera stessa.

Quando vengono premuti un tasto o una combinazione di tasti non riconosciuti, si ottiene in genere una segnalazione di errore tramite l'emissione di un segnale acustico.

7.3.3 L'ambiente

Si definisce **ambiente** (o *environment*) un insieme di variabili dette appunto *variabili di ambiente* *ambiente* che possono essere accedute dai processi. Tali variabili non sono passate in input ai processi, ma sono gestite dalla shell. Le variabili di ambiente più comuni sono riportate in tab. 7.4, ma si consiglia di guardare la man page `environ(5)` per avere una lista più completa.

Combinazione di tasti	Significato
→ Ctrl F	sposta il cursore di una posizione verso destra (avanti)
← Ctrl B	sposta il cursore di una posizione verso sinistra (indietro)
Backspace	cancella il carattere alla sinistra del cursore, spostando il cursore di una opzione verso sinistra
Canc Ctrl D	cancella il carattere alla posizione del cursore, spostando tutto ciò che segue di una posizione verso sinistra
Home Ctrl A	sposta il cursore all'inizio della riga
End Ctrl E	sposta il cursore all'inizio della riga
Alt F	sposta il cursore di una parola verso destra (avanti)
Alt B	sposta il cursore di una parola verso sinistra (indietro)
Ctrl L	ripulisce lo schermo
Ins	passa alternativamente dalla modalità <i>insert</i> (default), nella quale i caratteri digitati vengono inseriti tra il carattere precedente al cursore e quello che si trova nella posizione del cursore stesso, a quella <i>overwrite</i> , nella quale i caratteri digitati sovrascrivono quello che si trova nella posizione del cursore stesso

Tabella 7.2: Particolari combinazioni di tasti della shell *Bash*.

Carattere	Combinazione di tasti
‘	AltGr ‘
~	AltGr 0
{	AltGr 8
}	AltGr 9
í	AltGr ì
ú	AltGr ù

Tabella 7.3: Caratteri particolari ottenibili con la tastiera italiana.

Molte delle variabili di ambiente sono utilizzate dalla maggior parte dei programmi ed in genere per queste si usa la convenzione di esprimerle in caratteri maiuscoli. Il kernel non usa mai queste variabili, ma esse costituiscono un modo molto comodo per definire comportamenti dei programmi a livello generale senza dover passare loro in input un elenco di argomenti troppo complesso. La shell, ad esempio, ne utilizza molte per il suo funzionamento.

La visualizzazione delle variabili di ambiente definite è ottenibile tramite il comando interno **set** che serve anche a definire molte delle impostazioni di *Bash*. La sua sintassi è mostrata in sez. 7.3.10.

Una variabile di ambiente è definita per mezzo della seguente sintassi

```
$ varname=[value]
```

dove

varname è il nome della variabile di ambiente da definire;

value è l'eventuale valore da assegnare alla variabile *varname* (se non è specificato, a *varname* viene assegnata la stringa nulla);

Ad esempio, il comando

```
$ PIPPO=ciao
```

definisce la variabile di ambiente **PIPP0** e la imposta col valore **ciao**;

Il valore contenuto in una variabile di ambiente può essere visualizzato per mezzo del comando interno **echo** che serve per visualizzare delle stringhe.

Variabile	Significato
USER	Username
LOGNAME	Username con cui è stato effettuato il login
HOME	Home directory dell'utente corrente
LANG	Localizzazione
PATH	Elenco delle directory, separate da ':', contenenti i file eseguibili
PWD	Working directory
SHELL	Tipo di shell
TERM	Tipo di terminale
PAGER	Text viewer preferito
EDITOR	Text editor preferito
BROWSER	Browser preferito
TMPDIR	Directory contenente i file temporanei

Tabella 7.4: Le variabili di ambiente.

Comando (bash): **echo**

SINTASSI

\$ echo [*option*] [*arg* ...]

DESCRIZIONE

option specifica il comportamento del comando. I possibili valori sono:

- e abilita l'interpretazione di particolari sequenze di caratteri, come riportato in tab. 7.5

Sequenza	Significato
\a	segnale acustico (carattere ASCII BEL)
\b	<i>backspace</i> (cancella il carattere precedente)
\c	sopprime i caratteri <i>newline</i> in fondo alla riga
\E	carattere ASCII ESC
\f	salto pagina (carattere ASCII FF)
\n	<i>newline</i> (carattere ASCII LF ??)
\r	<i>newline</i> (carattere ASCII CR)
\t	carattere di tabulazione (carattere ASCII HT)
\v	tabulazione verticale (carattere ASCII VT)
\\	<i>backslash</i>
\num	carattere ASCII il cui codice corrisponde al valore ottale <i>num</i>

Tabella 7.5: Sequenze di caratteri particolari interpretabili da **echo**.

- E disabilita esplicitamente l'interpretazione delle sequenze di caratteri particolari (v. tab. 7.5);
- n annulla l'effetto di "ritorno a capo" del cursore dopo aver visualizzato la stringa *arg*;

arg è la stringa da visualizzare;

Esiste anche un comando di sistema **echo** (man page **echo(1)**), corrispondente al file eseguibile **/bin/echo**, che ha una sintassi analoga al comando interno di *Bash*. Per quanto detto precedentemente, scrivendo **echo** sulla linea di comando, *bash* esegue il comando interno **echo**. Se si vuole eseguire il comando **echo** di sistema, si deve digitare sulla linea di comando **/bin/echo**.

Dunque, il comando

```
$ echo $PATH
```

visualizza il contenuto della variabile d'ambiente `PATH`.

Secondo la sintassi di *Bash*, le variabili di ambiente sono sempre riferite antepo-
nendo ad esse il simbolo '\$', tranne nel caso in cui esse vengono definite. Ovvero
per definire la variabile `PIPP0` si utilizza il suo nome così com'è scritto, mentre
quando si fa riferimento ad essa si deve utilizzare l'identificatore `$PIPP0`.

Le variabili di ambiente hanno visibilità soltanto all'interno di un determinato con-
testo che è l'ambito della shell stessa, per cui soltanto il processo attuale, ovvero la shell
stessa (con i suoi comandi interni) è al corrente delle variazioni. Affinché anche i pro-
cessi esterni (figli) siano in grado di “vedere” le impostazioni delle variabili di ambiente
è necessario utilizzare il comando interno `export`.

Comando (bash): `export`

SINTASSI

`$ export [option] [varname[=value] ...]`

DESCRIZIONE

option specifica la modalità di funzionamento del comando. Può assumere i se-
guenti valori:

- `-f` indica a `export` di considerare *varname* il nome di una funzione,
anziché quello di una variabile di ambiente;
- `-n` indica di annullare la visibilità ai processi figli delle variabili di am-
biente specificate da *varname*;
- `-p` indica a `export` di visualizzare tutte le variabili di ambiente visibili
dai processi figli;
- `--` indica a `export` di non considerare le opzioni eventualmente specifi-
cate dopo questa;

varname è l'elenco dei nomi delle variabili di ambiente (o funzioni) da rendere
visibili ai processi figli;

value è il valore da assegnare alla variabile di ambiente;

Il comando `set` visualizza l'elenco delle variabili di ambiente visibili al processo
(shell) corrente, ma se si vuole visualizzare l'elenco delle variabili d'ambiente visibili ai
processi figli è necessario utilizzare il comando `export`.

Ad esempio, il comando

```
$ PIPP0=ciao
```

definisce la variabile di ambiente `PIPP0` con il contenuto “ciao” ed il comando seguente

```
$ echo $PIPP0
```

visualizza il contenuto della stessa variabile. Ma se a questo punto si lancia un processo
figlio (qualunque comando impartito dalla linea di comando, a meno che non sia un co-
mando interno di *Bash*, implica la generazione di un processo figlio), come, per esempio,
invocando l'esecuzione di un'altro processo *bash* con il comando

```
$ bash
```

e si visualizza il contenuto della variabile di ambiente `PIPP0` tramite il comando

```
$ echo $PIPP0
```

```
$ _
```

si ottiene una risposta nulla, come se tale variabile non fosse stata definita. Se si chiude il processo *bash* appena lanciato con il comando

```
$ exit
```

e si rende visibile ai processi figli tale variabile con il comando

```
$ export $PIPP0
```

rilanciando un processo figlio, questo sarà in grado di “vedere” tale variabile. Infatti, analogamente a quanto fatto precedentemente

```
$ bash
$ echo $PIPP0
ciao
$ _
```

ma in questo caso viene visualizzato il contenuto della variabile di ambiente PIPP0 impostata a “ciao”.

Quindi, i seguenti comandi

```
$ PIPP0=ciao
$ export $PIPP0
sono equivalenti a
$ export PIPP0=ciao
```

Per eliminare una variabile di ambiente è necessario utilizzare il comando interno *unset*.

Comando (bash): *unset*

SINTASSI

```
$ unset [option] [varname ...]
```

DESCRIZIONE

option indica la modalità di funzionamento di *unset*. Può assumere i seguenti valori:

- f indica a *unset* di considerare *varname* un nome di funzione;
- v indica a *unset* di considerare *varname* un nome di variabile di ambiente;

varname è l'elenco delle variabili di ambiente (o funzioni) da rimuovere;

Alcune variabili di ambiente non possono essere rimosse.

Lo spazio di memoria dedicato alle variabili d'ambiente, ovvero l'*environment*, è piuttosto limitato. Creare molte variabili d'ambiente o tali che utilizzino molta memoria, può facilmente andare oltre il limite riservato allo scopo. Ad esempio

```
$ eval "'seq 10000 | sed -e 's/./export var&=ZZZZZZZZZZZZZ/''"
```

crea una variabile d'ambiente troppo ??? e quindi la visualizzazione dell'ambiente con il comando *du* ??? genera il seguente output

```
$ du
bash: /usr/bin/du: Argument list too long
```

```
???
???
```

7.3.4 Name completion

name completion

La shell offre l'opportunità di completare automaticamente il nome, cioè è possibile scrivere parzialmente il nome di un comando o file e premere il tasto `Tab` in modo tale che la shell possa suggerire il nome per intero; Un esempio di **name completion** è quello riportato di seguito:

```
$ pdf1 Tab
$ pdfTablatex _
```

Nel caso in cui siano possibili più nomi con la stessa radice, viene visualizzato l'elenco dei nomi che possono completare il comando (in tal caso è necessario premere due volte il tasto `Tab`, poiché dopo la prima pressione il sistema emette generalmente un segnale acustico per indicare il fatto che non riesce a completare automaticamente il nome), senza che il comando venga completato. Un esempio è riportato di seguito:

```
$ he Tab Tab
head      help      hesinfo  hexdump
$ he_
```

7.3.5 Alias

alias

La shell permette di definire dei comandi come **alias** (sinonimi) di altri. Il comando utilizzato per visualizzare/definire un alias è **alias** (man page **bash(1)**).

Comando (bash): **alias**

SINTASSI

```
$ alias [option] [name[=value] ...]
```

DESCRIZIONE

option specifica il funzionamento del comando. I suoi valori possibili sono i seguenti:

-p visualizza l'elenco degli alias definiti;

name è il nome dell'alias da visualizzare/definire (se viene fornito anche *value* l'alias viene definito altrimenti viene soltanto visualizzato – se è stato precedentemente definito);

value è il valore da assegnare all'alias;

Un utilizzo molto comune di alias è rappresentato dalla definizione del comando **ll** che permette di visualizzare l'elenco degli oggetti del filesystem contenuti in una determinata directory. La definizione di tale alias è la seguente:

```
$ alias ll='ls -l --color=tty'
```

In questo modo, digitando il comando

```
$ ll
```

la shell eseguirà il comando `ls -l --color=tty`.

È possibile anche eliminare la definizione di un alias con il comando **unalias** (man page **bash(1)**).

Comando (bash): **unalias**

SINTASSI

```
$ unalias [option] [name ...]
```

DESCRIZIONE

option indica il funzionamento del comando. Questo può assumere i seguenti valori possibili:


-a elimina la definizione di tutti gli alias precedentemente definiti;


name specifica il nome dell'alias da eliminare;


7.3.6 Command history

La shell memorizza i comandi che man mano vengono impartiti dall'utente e ne tiene uno storico (**command history**) nel file `~/.bash_history`. Inoltre permette all'utente di richiamare facilmente i comandi precedentemente impartiti, per mezzo dei seguenti tasti:

command history

 visualizza il comando digitato precedentemente;

 visualizza il comando digitato successivamente;

 visualizza il comando digitato meno recente;

 visualizza il comando digitato più recente;

In particolare *Bash* ha la caratteristica di avere una *command history* praticamente infinita, cioè mantiene la storia di tutti i comandi impartiti (per ogni utente).

7.3.7 Comandi interni ed esterni

Bash riesce ad interpretare un insieme di **comandi interni** (*internal command* o *builtin command*), implementati all'interno di *Bash*. L'elenco di tali comandi è ottenibile tramite il comando interno **help**.

comandi interni

Comando (bash): **help**

SINTASSI

\$ help [cmd]

DESCRIZIONE

cmd è il comando del quale si desidera avere i dettagli.

Se nessun comando è specificato, **help** mostra l'elenco dei comandi interni, come quello riportato di seguito:

```
GNU bash, version 2.05b.0(1)-release (i686-pc-linux-gnu)
These shell commands are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.
```

A star (*) next to a name means that the command is disabled.

```
%[DIGITS | WORD] [&]      (( expression ))
. filename                :
[ arg... ]                [[ expression ]]
alias [-p] [name=value] ... ] bg [job_spec]
bind [-lpvsPVS] [-m keymap] [-f fi break [n]
builtin [shell-builtin [arg ...]] case WORD in [PATTERN [| PATTERN].
cd [-L|-P] [dir]          command [-pVv] command [arg ...]
compgen [-abcdefgjkusv] [-o option complete [-abcdefgjkusv] [-pr] [-o
continue [n]              declare [-afFirtx] [-p] name[=valu
dirs [-clpv] [+N] [-N]    disown [-h] [-ar] [jobspec ...]
echo [-neE] [arg ...]     enable [-pnds] [-a] [-f filename]
eval [arg ...]            exec [-cl] [-a name] file [redirec
```

```

exit [n]
false
fg [job_spec]
for (( exp1; exp2; exp3 )); do COM
getopts optstring name [arg]
help [-s] [pattern ...]
if COMMANDS; then COMMANDS; [ elif jobs [-lnprs] [jobspec ...] or job
kill [-s sigspec | -n signum | -si
local name[=value] ...
popd [+N | -N] [-n]
pushd [dir | +N | -N] [-n]
read [-ers] [-u fd] [-t timeout] [
return [n]
set [--abefhkmnptuvxBCHP] [-o opti
shopt [-pqsu] [-o long-option] opt
suspend [-f]
time [-p] PIPELINE
trap [arg] [signal_spec ...] or tr
type [-afptP] name [name ...]
ulimit [-SHacdflmnpstuv] [limit]
unalias [-a] [name ...]
until COMMANDS; do COMMANDS; done
wait [n]
{ COMMANDS ; }

export [-nf] [name[=value] ...] or
fc [-e ename] [-nlr] [first] [last
for NAME [in WORDS ... ;] do COMMA
function NAME { COMMANDS ; } or NA
hash [-lr] [-p pathname] [-dt] [na
history [-c] [-d offset] [n] or hi
let arg [arg ...]
logout
printf format [arguments]
pwd [-PL]
readonly [-anf] [name[=value] ...]
select NAME [in WORDS ... ;] do CO
shift [n]
source filename
test [expr]
times
true
typeset [-afFrtx] [-p] name[=valu
umask [-p] [-S] [mode]
unset [-f] [-v] [name ...]
variables - Some variable names an
while COMMANDS; do COMMANDS; done

```

Tali comandi, assieme ai nomi dei file eseguibili ed ai simboli che rappresentano gli operatori aritmetico-logici, formano lo *scripting language* interpretato da *Bash*, cioè è possibile scrivere dei file di testo, detti **script**, contenenti comandi che possono essere interpretati ed eseguiti da *Bash* stessa (v. sez. 7.3.9).

script

Quando viene impartito un comando, la shell tenta di eseguirlo effettuando, nell'ordine, i seguenti passi:

1. Tenta di eseguire il comando interno con il nome specificato come *command*;
2. Ricerca il file specificato come *command*. Se tale file è specificato completo di path (sia esso assoluto o relativo) il file è ricercato soltanto nel path specificato. Se non è specificato nessun path, il file viene ricercato nell'elenco delle directory contenuto nella variabile d'ambiente `PATH`⁶. Se il file esiste, va al passo successivo;

È opportuno notare il fatto che nell'elenco delle directory contenute nella variabile d'ambiente `PATH`, per motivi di sicurezza, non è compresa la working directory (`.`). Infatti, includendo tale directory nell'elenco (in modo particolare all'inizio dell'elenco), un utente malizioso potrebbe aver chiamato un file presente nella sua directory con lo stesso nome di quello di un comando di sistema di uso frequente, ad esempio `ls`. In questo modo, se il superuser ha, in un certo momento, la working directory impostata su quella dell'utente in questione e digita il comando `ls` (o un alias che si riferisce ad esso), lancia in realtà il comando che si trova nella working directory, ovvero lancia in esecuzione tale file con i privilegi di superuser. Il processo da questo lanciato potrebbe fare qualunque cosa sul filesystem!

Tale directory può comunque essere aggiunta in coda all'elenco delle directory (strategia più sicura)⁷ con il comando

```
$ export PATH=$PATH:.
```

o in testa (strategia più rischiosa) con il comando

```
$ export PATH=.:$PATH
```

⁶le directory elencate nella variabile di ambiente `PATH` sono separate dal carattere `:`.

⁷in realtà un po' di rischio c'è anche in questo caso poiché l'utente malizioso potrebbe aver dato il nome ad un file contenuto in una sua directory, simile a quello di un comando di sistema molto utilizzato, ad esempio `la` (i tasti `'a'` ed `'s'` sono adiacenti sulla tastiera) in modo tale che se il superuser sbaglia accidentalmente nella digitazione del comando `ls`, possa lanciare in esecuzione il file dell'utente.

3. Tenta di eseguire il file specificato come *command*. Per poter essere eseguito il file deve esistere ed avere il permesso di esecuzione per l'utente che ha impartito il comando. In tal caso la shell crea un processo figlio per l'esecuzione delle istruzioni contenute nel file, secondo quanto riportato di seguito
 - Se si tratta di un file eseguibile in formato binario (ELF) la shell crea un processo figlio che esegue i comandi contenuti nel file (che è comprensibile dalla CPU);
 - Se si tratta di un file non binario, la shell lo considera uno *script* (v. sez. 7.3.9).

??? figura forking processi ???

La shell può essere terminata per mezzo del comando interno `exit`.

7.3.8 Gestione dei metacaratteri

La shell attribuisce ad alcuni caratteri dei significati speciali che sono utilizzati per fare riferimento facilmente a gruppi di file o directory. Tali caratteri sono detti **metacaratteri** metacaratteri o *caratteri jolly* (*wildcard*)⁸. In questo modo la shell stessa si accolla l'onere di espandere la riga di comando impartita da tastiera interpretando opportunamente tali caratteri speciali, in modo da passare al comando stesso l'elenco dei parametri ottenuto appunto dall'espansione dei *metacaratteri*.

I metacaratteri utilizzati da *Bash* son quelli di seguito riportati:

- *** ha il significato di “qualunque stringa” (compresa la stringa nulla);
- ?** ha il significato di “qualunque carattere”;
- [set]** indica uno dei caratteri compresi nell'insieme specificato da *set*. Un insieme di caratteri può essere specificato nei seguenti modi:

Intervallo è rappresentato da due caratteri separati dal simbolo ‘-’ (es. `a-d` rappresenta l'insieme dei caratteri formati da `a`, `b`, `c` e `d`);

Classe è rappresentata da un nome di classe compreso tra una coppia di ‘.’. I nomi delle classi seguono lo standard POSIX.2 e sono: `alnum`, `alpha`, `ascii`, `blank`, `cntrl`, `digit`, `graph`, `lower`, `print`, `punct`, `space`, `upper`, `word`, `xdigit` (es. `:alpha:` rappresenta l'insieme dei caratteri alfabetici);

Simbolo è rappresentato da un simbolo compreso tra una coppia di ‘.’. Tale formato è molto utile per la rappresentazione di simboli dipendenti dalla localizzazione (es. `.ss.` rappresenta il simbolo ß).

Se il primo carattere che segue ‘[’ è ‘!’ o ‘^’ allora l'insieme considerato è il complementare di quello indicato (es. `!a-d` è l'insieme dei caratteri diversi da `a`, `b`, `c` e `d`).

- {set}** tutti i caratteri compresi nell'elenco specificato da *set*. Tale elenco utilizza come separatore il simbolo ‘,’ (es. `{a,b,c}` indica la sequenza dei caratteri `a`, `b` e `c` nell'ordine specificato).

Poiché la shell può attribuire dei significati particolari ad alcuni caratteri, essa prevede anche un meccanismo (*quoting*) che permette di utilizzare tali caratteri col loro significato letterale, senza che vengano interpretati in modo particolare dalla shell stessa. Ciò, in genere, avviene per mezzo di due tecniche possibili:

Carattere di escape (*escape character*) si tratta di un carattere che può essere utilizzato subito prima del metacarattere che si desidera non venga interpretato come tale, ma in maniera letterale.

⁸talvolta si fa riferimento al concetto di sostituzione di caratteri particolari col termine *globbing*.

Bash utilizza il simbolo ‘\’ come carattere di escape. Esso annulla il significato speciale del carattere che lo segue ad eccezione del carattere ASCII CR, nel qual caso vien considerato come carattere che indica la continuazione del comando sulla riga successiva (ovvero tale sequenza di caratteri viene completamente ignorata).

Delimitatori si tratta di un simboli utilizzati a coppie per delimitare un intervallo della riga di comando all’interno del quale non devono venire interpretati i *metacaratteri* col loro particolare significato, ma tutti i caratteri devono essere interpretati in maniera letterale, così come sono scritti.

Bash utilizza due tipi di delimitatori: ‘ ’ (*single quotes*) e “ ” (*double quotes*).

Una sequenza di caratteri racchiusa tra *single quotes* ha il significato letterale dei caratteri in essa contenuti. Si tenga presente comunque che il carattere ‘ ’ non può essere contenuto all’interno di *single quotes* (neanche se preceduto dal carattere dal carattere di escape ‘\’).

Una sequenza di caratteri racchiusa tra *double quotes* ha il significato letterale dei caratteri in essa contenuti fatta eccezione per i caratteri ‘\$’, ‘ ’ e ‘\’. Il simbolo ‘\’ preserva il suo particolare significato se precede i caratteri ‘\$’, ‘ ’, “ ”, ‘\’ o CR.

Il carattere “ ” può essere visualizzato all’interno di una sequenza racchiusa tra *double quotes* con la sequenza ‘\”’;

Dal momento che venono utilizzati dei simboli per rappresentare i delimitatori ed il carattere di escape, è necessario utilizzare anche un altro simbolo per annullare il loro effetto, in modo da poter utilizzare tali simboli (delimitatori ed il carattere di escape) col loro valore letterale all’interno della riga di comando.

La shell interpreta in modo particolare anche altri simboli che assumono il significato riportato di seguito:

! viene utilizzato per richiamare un comando dallo storico;

& se viene utilizzato come ultimo carattere della linea di comando per indicare alla shell di lanciare il comando che lo precede in background⁹. In questo modo, la shell non attende la terminazione del comando, ma visualizza subito il prompt, pronta ad accettare un altro comando dall’utente;

; è il carattere utilizzato come separatore tra comandi. Sulla linea di comando possono essere impartiti più comandi contemporaneamente, separati con questo carattere. Essi saranno eseguiti equenzialmente nell’ordine specificato;

% è il carattere utilizzato per i riferimenti ai job¹⁰.

(,) ???;

<, >, | utilizzati per la redirectione dell’input/output (v. sez. 7.3.11);

7.3.9 Scripting language

scripting language

Lo **scripting language** è il linguaggio di programmazione interpretato dalla shell. Esso si compone di comandi interni (*built-in*), propri di *Bash*, file eseguibili ed operatori aritmetico-logici.

script

È possibile pertanto realizzare degli **script**, ovvero files di testo che contengono istruzioni comprensibili dalla shell e da questa interpretate ed eseguite. Tali istruzioni possono essere composte dai *comandi interni* della shell e dai comandi del sistema (nomi di file eseguibili) con la sintassi che accetta la linea di comando. Scrivere uno script di shell è come scrivere una sequenza di comandi da poter far eseguire in blocco alla shell indicando semplicemente il nome del file che li contiene (lo script)¹¹.

⁹v. sez. 7.3.12

¹⁰v. sez. 7.3.12

¹¹per questo tali file sono anche detti *batch*.

Ad esempio, se si vuol fare eseguire in sequenza i comandi

```
$ echo Sto creando il file 'pippo'
$ ls > pippo
```

si può creare un file `prova` (ad esempio con il comando `vi prova`) ed inserire al suo interno le righe seguenti

```
echo Sto creando il file 'pippo'
ls > pippo
```

poi fare in modo che tale file sia eseguibile, con il comando

```
$ chmod u+x prova
```

A questo punto sarà sufficiente digitare il comando

```
$ ./prova
```

per far eseguire i comandi in esso contenuti.

In realtà *Bash* mette a disposizione un potente scripting language, con la valutazione di espressioni condizionali, cicli, inclusioni di file, che permette di automatizzare in maniera piuttosto semplice qualunque operazione di sistema.

Quando viene lanciato in esecuzione un comando, esso può richiedere degli argomenti sulla linea di comando. Questi vengono passati come parametri al processo che viene posto in esecuzione dalla shell che si occupa di “tokenizzarli” (separarli l’uno dall’altro, riconoscendo come simbolo separatore, il carattere spazio ‘ ’).

Allo stesso modo, anche gli script *Bash* possono ricevere degli argomenti dalla linea di comando. Gli argomenti passati agli script sono memorizzati nei parametri `$0`, `$1`, `$2`, ... dello script stesso.

Come trattato nel cap. 6, quando un processo termina, esso ritorna un valore di uscita, detto *exit status*, cioè un valore numerico di un byte. L’*exit status* dell’ultimo processo eseguito viene memorizzato nel parametro `$?` .

Quando viene imputato un comando, rappresentato da un file eseguibile, la shell crea un nuovo processo per la sua esecuzione secondo quanto di seguito riportato:

- Se si tratta di un file eseguibile in formato binario (ELF) la shell crea un processo figlio che esegue i comandi contenuti nel file (che è comprensibile dalla CPU);
- Se si tratta di un file non binario, la shell lo considera uno *script*, ovvero un file contenente una sequenza di comandi che un interprete può eseguire. Quindi analizza i primi caratteri del file e se li trova uguali a “`#!`” legge i caratteri successivi fino al primo carattere di fine riga (LF) ed interpreta tale stringa come il file eseguibile in formato binario da lanciare per interpretare i comandi contenuti nello script. Quindi crea un processo figlio che esegue l’interprete indicato, al quale passa in input il file specificato dall’utente (lo script). Infatti, generalmente la prima riga di uno script *Bash* è la seguente:

```
#!/bin/bash
```

in modo tale che, se tale file viene specificato come comando (ed il file ha il permesso di esecuzione per l’utente considerato), la shell lanci in esecuzione il file `/bin/bash`, ovvero l’interprete *Bash*, al quale fare interpretare il file stesso.

Si supponga infatti di avere il file di testo `~/pippo` con il seguente contenuto:

```
#!/bin/bash
echo Ciao a tutti !
```

e di avere almeno i diritti di lettura e di esecuzione su quel file. Lanciando il comando

```
$ ~/pippo
```

la shell si accorge che il file **pippo** presente nella home directory dell'utente corrente '~' non è un file binario, quindi lo tratterà come se fosse uno script, estraendo dalla prima riga il nome dell'interprete da lanciare per eseguire tale script, ovvero **/bin/bash**, che provvederà a lanciare in esecuzione come processo figlio, passandogli come input lo script.

Va notato il fatto che le linee che iniziano con il carattere '#' sono considerate dei commenti dallo scripting language di *Bash*, pertanto tale riga viene ignorata dall'interprete stesso quando esegue il file (in genere la maggior parte degli interpreti utilizza il carattere '#' come inizio di una riga di commento).

Se il file **~/pippo** avesse contenuto le istruzioni seguenti

```
#!/usr/bin/perl
print "Ciao a tutti !"
```

la shell avrebbe lanciato in esecuzione il file **/usr/bin/perl**, ovvero l'interprete Perl, passandogli come input lo script (in questo caso si tratta di uno script Perl).

- Se il file non contiene nessuna intestazione particolare, *Bash* tenta di eseguirlo interpretando le istruzioni in esso contenute (come se fosse uno script *Bash*).

La sintassi del linguaggio di scripting di *Bash* è piuttosto articolata, e di seguito saranno riportate alcune caratteristiche essenziali per la comprensione e realizzazione di semplici script. Per una descrizione più dettagliata della programmazione con *Bash* v. [10].

I comandi vengono scritti in sequenza su righe diverse. Alcuni caratteri assumono un significato particolare, di seguito illustrato:

- # commento. Tutto il contenuto della riga che segue questo carattere è ignorato. Ad esempio considerando la riga

```
echo salve # mondo
```

mondo è considerato un commento;

- ; separatore di comandi. In questo modo possono essere espressi più comandi sulla stessa riga. Ad esempio la riga

```
echo salve; echo mondo
```

viene considerata come

```
echo salve
echo mondo
```

;; terminatore di rami **case**. Ogni ramo relativo ad un costrutto **case** deve essere terminato con questa sequenza di caratteri. Esempio

```
case "$variable" in
abc) echo "$variable = abc" ;;
xyz) echo "$variable = xyz" ;;
esac
```

. Assume vari significati:

- v. il comando interno **source**;
- l'occorrenza di un singolo carattere (v. espressioni regolari);
- parte del nome di un file o di un path;

" partial quoting – il testo contenuto tra ‘”’ viene considerato così com'è (a parte qualche carattere particolare che viene comunque interpretato);

' full quoting – il testo contenuto tra ‘’’’ viene considerato così com'è.

, separatore di espressioni aritmetiche. più espressioni possono essere specificate sulla stessa riga ed il valore di ritorno è quello dell'ultima espressione. Ad esempio

```
let "b = ((a = 1, 21 / 7))"
```

imposta **a** col valore 1 e **b** col valore 3 (= 21/7);

\ carattere di escape. Questo carattere viene utilizzato per indicare all'interprete di considerare il carattere che segue così com'è, senza considerare il particolare significato ad esso attribuito. Ad esempio

```
echo salve \# mondo
```

in questo caso il carattere **#** viene considerato come tale, non assume il significato di inizio di un commento.

/ assume vari significati:

- operazione di divisione;
- parte del nome di un path;

` command substitution – il testo contenuto tra ‘`’ (*backtick*) viene eseguito come un comando ed il risultato è reso disponibile per poterlo assegnare ad una variabile;

: null command – è l'equivalente dell'operazione assembly NOP (no operation), ovvero un'istruzione che non fa niente ed il suo valore di ritorno è **true** (0);

! assume vari significati:

- fornisce il complementare booleano del valore di ritorno o di una condizione;
- fa riferimento indiretto alle variabili;

* assume vari significati:

- moltiplicazione aritmetica;

- wildcard nei nomi di file (v. anche le espressioni regolari);
- **** elevamento a potenza;
- ?** assume vari significati:
- analogo all'operatore ternario del linguaggio C (?:);
 - esegue un test sull'impostazione di una variabile;
 - wildcard nei nomi di file (v. anche le espressioni regolari);
- \$** assume vari significati:
- \$var** variable substitution – indica il valore contenuto nella variabile **var**;
- \$** end-of-line – nelle espressioni regolari indica la fine di una riga;
- \${}** parameter substitution;
- \$num, \$@** positional parameters – parametri di ingresso, specificati sulla riga di comando: **\$1** è il primo parametro, **\$2** è il secondo, ... e **\$@** è l'elenco di tutti i parametri passati allo script sulla riga di comando;
- \$?** exit status – valore di ritorno dell'ultimo comando eseguito;
- \$\$** PID del processo che esegue le istruzioni contenute nello script;
- (...)** assume vari significati:
- command group – i comandi racchiusi tra parentesi vengono eseguiti in una subshell (un processo figlio), quindi le variabili utilizzate all'interno non saranno visibili dal resto dello script;
 - array initialization – tra parentesi vengono specificati i valori da assegnare ai vari elementi del vettore. Ad esempio
- ```
MyArray=(value1 value2 value3)
```
- {...}** assume vari significati:
- brace expansion – indica un insieme di valori specificato dall'elenco separato dal carattere **' , '** contenuto tra parentesi graffe. Ad esempio
- ```
grep Linux myfile*.{txt,htm*}
```
- trova tutte le istanze della parola "Linux" nei file che iniziano per **myfile** ed hanno estensione **txt** o che inizia per **htm**;
- raggruppamento di un blocco di comandi. Più comandi possono essere raggruppati tra **{** e **}** per poter redirigire tutto il loro input/output da o verso un file (il blocco di comandi non viene eseguito in una subshell).
- [...]** assume vari significati:
- elemento di un vettore. Ad esempio
- ```
echo ${MyArray[1]};
```
- visualizza il valore contenuto nel secondo elemento (1) el vettore **MyArray**;
- v. comando interno **test** (tra **[** e **]** viene specificata la condizione da testare). Ad esempio
- ```
[ -n "$var" ]
```
- ritorna vero (0) se la variabile **var** non è stata inizializzata;

- specifica un intervallo di valori (v. espressioni regolari);

`[[...]]`

esegue un test con una sintassi della condizione più simile a quella utilizzata dagli altri linguaggi di programmazione (possono essere utilizzati i simboli `&&`, `||`, `< e >` che non possono essere utilizzati con `[...]`). Ad esempio

```
[[ $var1 > $var2 ]]
```

ritorna vero se il contenuto della variabile `var1` è maggiore di quello della variabile `var2`.

> assume vari significati:

- redirezione dell'output. Ad esempio

```
echo "ciao" >~/myfile
```

redirige l'output del comando `echo "ciao"` nel file `~/myfile`, sovrascrivendolo se il file esiste già. Il comando

```
echo "ciao" >&2
```

redirige l'output del comando `echo "ciao"` nel file descriptor 2, ovvero l'error associato al processo.

- simbolo di confronto: maggiore;

&> redirezione dell'output e dell'error. Ad esempio

```
echo "ciao" &>~/myfile
```

redirige l'output e l'error del comando `echo "ciao"` nel file `~/myfile`, sovrascrivendolo se il file esiste già.

>> redirezione dell'output. Ad esempio

```
echo "ciao" >>~/myfile
```

redirige l'output del comando `echo "ciao"` nel file `~/myfile`, aggiungendolo a quanto già contenuto nel file. Se il file non esiste viene creato.

< assume vari significati:

- redirezione dell'input. Ad esempio

```
tail <~/myfile
```

redirige l'input del comando `tail` sul file `~/myfile`. In questo modo `tail` gestisce il contenuto del file `~/myfile` come se fosse digitato dalla tastiera (standard input);

- simbolo di confronto: minore;

\< inizio di una parola (v. espressioni regolari);

\> fine di una parola (v. espressioni regolari);

- | redirezione dell'output del comando che precede nell'input del comando che segue ???;
- >| forza la redirezione ???;
- || operazione di OR logico;
- & lancia un comando in background;
- && operazione di AND logico;
- assume vari significati:
 - redirezione dell'input (quando il comando si aspetta il nome di un file da cui prelevare i dati di ingresso) sullo standard input o del'output (quando il comando si aspetta il nome di un file in cui scrivere i dati di uscita) sullo standard output. Ad esempio


```
$ file
Usage:  file [-bciknvzL] [-f namefile] [-m magicfiles] file...
```

mentre

```
$ file -
abc
standard input:          ASCII text
```
 - simbolo di sottrazione matematica;
 - previous working directory;
- = operatore di assegnamento;
- + assume vari significati:
 - operatore di addizione;
 - v. espressioni regolari;
- % assume vari significati:
 - operatore di modulo;
 - operatore di confronto tra stringhe;
- ~ home directory;
- ~+ current working directory;
- ~- previous working directory;
- ^ beginning-of-line (v. espressioni regolari);

7.3.10 Impostazioni

Il funzionamento di *Bash* può essere definito tramite un elevato numero di opzioni che possono essere specificate sulla linea di comando che lancia *bash*, am possono essere anche specificate per mezzo del comando interno *set*.

Comando (bash): **set**

SINTASSI

\$ set [*option*] [*arg* ...]

DESCRIZIONE

option indica la modalità di esecuzione del comando. Esso può assumere uno dei seguenti valori:

- a segna per l'esportazione le variabili impostate;
- b notifica immediatamente la terminazione di un job;
- e termina immediatamente se un comando restituisce un valore diverso da 0;
- f disabilita l'espansione dei metacaratteri (globbing);
- h memorizza la posizione dei comandi;
- k tutti gli argomenti dell'assegnamento specificato sono inseriti nell'ambiente ???;
- m abilita il job control;
- n legge i comandi ma non li esegue;
- o *optname*
specifica le opzioni in altro modo. *optname* può assumere uno dei seguenti valori:
 - allexport**
identico a *option -a*;
 - braceexpand**
identico a *option -B*;
 - emacs** utilizza un'interfaccia stile emacs per il command line editing;
 - errexit**
identico a *option -e*;
 - hashall**
identico a *option -h*;
 - histexpand**
identico a *option -H*;
 - history**
abilita la command history;
 - ignoreeof**
la shell non termina dopo aver letto il carattere EOF;
 - interactive-comments**
permette di inserire commenti nei comandi interattivi;
 - keyword**
identico a *option -k*;
 - monitor**
identico a *option -m*;
 - noclobber**
identico a *option -C*;
 - noexec** identico a *option -n*;
 - noglob** identico a *option -f*;
 - nolog** accettato ma ignorato;
 - notify** identico a *option -b*;
 - nounset**
identico a *option -u*;
 - onecmd** identico a *option -t*;
 - physical**
identico a *option -P*;
 - posix** imposta il funzionamento di *bash* secondo lo standard POSIX 1003.2;
 - privileged**
identico a *option -p*;
 - verbose**
identico a *option -v*;
 - vi** utilizza un'interfaccia stile vi per il command line editing;
 - xtrace** identico a *option -x*;
- p disabilita l'elaborazione del file indicato dalla variabile di ambiente ENV (è impostato quando il real e l'effective UID sono diversi);
- t termina dopo aver letto ed eseguito un comando;

- u considera le variabili non definite come un errore;
- v visualizza le linee di uno script man mano che vengono lette dalla shell;
- x visualizza i comandi ed i loro argomenti man mano che questi vengono eseguiti;
- B abilita l'espansione delle parentesi graffe {...};
- C disabilita la sovrascrittura dei file con la redirectione dell'output;
- H abilita la sostituzione della command history con il carattere '!' (abilitata per default);
- P disabilita il seguimiento dei link simbolici nell'esecuzione di comandi che permettono il cambiamento della working directory (es. `cd`);

Specificando il simbolo '+' al posto del simbolo '-' si ottiene l'effetto opposto a quello dell'opzione in questione.

Tali impostazioni possono essere specificate anche sulla riga di comando che lancia la shell. Le opzioni correntemente impostate sono contenute nel parametro `$-` (per default sono `himBH`).

arg specifica i valori da assegnare in sequenza ai parametri `$1`, `$2`, ... ; Se *arg* non è specificato, vengono visualizzate tutte le variabili di ambiente.

7.3.11 La redirectione dei comandi

Come descritto nel cap. 6, ad ogni processo viene assegnato automaticamente dal sistema un file di input, uno di output ed uno di error.

In genere i dati di input vengono specificati sulla riga di comando della shell ed il programma li riceve come dati di ingresso, ovvero la shell li inserisce nel suo file di input.

pipeline

Il flusso dei dati di input e di output (e di error) viene in genere definito **pipeline**. Per ogni comando lanciato esiste una *pipeline* che riguarda i dati da esso coinvolti.

redirezione dei comandi

È possibile, tramite la shell, specificare un file di input, output o error diversi da quelli standard, ovvero effettuare quella che viene detta una **redirezione dei comandi**. Questo si rivela molto utile quando si devono combinare più comandi insieme. Si potrebbe infatti desiderare dare in input ad un comando, l'output di un altro, o ancora fare in modo che l'output di un comando venga effettuato su un file che non sia quello relativo allo schermo. Tutto ciò può essere fatto per mezzo di opportuni caratteri:

- > redirezione dell'output. L'output del comando che precede il simbolo '>' viene rediretto nel file specificato subito dopo. Ad esempio il comando

```
$ ls > pippo
```

creerà il file **pippo** nella working directory e ci scriverà tutto il suo output (che normalmente sarebbe finito sullo schermo). Nel caso in cui tale file esista già, questo viene svuotato completamente e quindi riempito con l'output del comando.

- >> redirezione dell'output in *append*. L'output del comando che precede la sequenza ">>" viene rediretto nel file specificato subito dopo. Ad esempio il comando

```
$ ls >> pippo
```

scriverà tutto il suo output nel file **pippo** (presente nella working directory). Tale operazione non sarà distruttiva del precedente contenuto del file **pippo**, ma la scrittura dell'output di `ls` andrà in aggiunta (*append*) al precedente contenuto del file. Nel caso in cui tale file non esista, questo viene creato e quindi riempito con l'output del comando.

- < redirezione dell'input. Il comando che precede il simbolo '<' considera il suo input il file specificato subito dopo. Ad esempio il comando

```
$ ls < pippo
```

indicherà al comando `ls` di utilizzare come input il file `pippo`;

2> redirezione dell'error. Analogo alla redirezione dell'output, ma redirige l'error, anziché l'output.

&> redirezione sia dell'output che dell'error. Redirige sia l'output che l'error del comando che sta a sinistra della sequenza “&>” nel file specificato subito dopo.

| redirezione dell'output di un comando nell'input di un altro. Redirige l'output del comando che precede il simbolo `|` nell'input del file specificato subito dopo. In questo modo si possono concatenare più comandi in cascata, in modo che l'output dell'uno costituisca l'input del successivo.

7.3.12 Job management

Sulla linea di comando della shell è possibile indicare l'esecuzione di più comandi, come mostrato in sez. 7.3.11. In genere, si fa riferimento all'insieme dei comandi impartiti tramite una linea di comando della shell col termine **job**. Dunque, un *job* rappresenta l'indicazione dell'esecuzione di una linea di comando. Ad esempio

```
$ ls
```

lancerà in esecuzione un job che contiene il solo processo `ls`. Mentre indicando di eseguire

```
$ ls | less
```

la shell lancerà in esecuzione un job che contiene i processi `ls` e `less`, in modo tale che l'output del primo processo (`ls`) costituisca l'input del secondo (`less`).

La shell permette di gestire i job (in genere ci si riferisce a tale funzionalità col termine *job control*). Infatti per mezzo di comandi opportuni è possibile sospendere, riattivare un job e portarlo in *background* o in *foreground*.

Con il comando `jobs` si può visualizzare l'elenco dei job attualmente in esecuzione. In tale elenco, il **job corrente**, ovvero quello che è stato in foreground più di recente (in genere il più recente) è contrassegnato dal simbolo `+` e quello immediatamente precedente dal simbolo `-`.

Un job che viene lanciato dalla riga di comando, in genere viene eseguito dalla shell in **foreground**, ovvero in modalità interattiva con l'utente. In tal caso la shell attende la sua terminazione prima di visualizzare un altro prompt ed essere quindi pronta a ricevere un altro comando dall'utente. Tecnicamente i processi sono considerati in foreground quando il loro PGID corrisponde al PGID del terminale. Tali processi sono in grado di ricevere tutti i segnali inviati dalla tastiera e sono i soli a cui è permesso interagire con la tastiera e lo schermo.

È possibile indicare esplicitamente alla shell di eseguire un job in **background**, ovvero in modalità non interattiva con l'utente. In tal caso la shell non attende la sua terminazione e visualizza subito un prompt per indicare che è pronta a ricevere un altro comando dall'utente. In realtà, comparirà sullo schermo anche un'indicazione con la seguente sintassi

```
[job] lastPID
```

dove

job è l'identificativo del job appena lanciato;

lastPID

è il PID dell'ultimo processo che fa parte del job;

Ad esempio l'indicazione

```
[1] 25647
```

indica l'avvio del job numero 1 ed il PID dell'ultimo processo che ne fa parte è il 25647.

Si tenga presente che a tali processi (in *background*) non è permesso di interagire con la tastiera e lo schermo e qualunque tentativo in proposito genera un segnale **SIGTTIN** o **SIGTTOU** che, per default, sospende il processo.

Un processo può essere sospeso tramite l'invio di un segnale di sonsensione riproducibile con la tastiera tramite la combinazione di tasti `Ctrl Z` (`^Z`)¹². Si può sospendere un processo anche per mezzo della combinazione di tasti `Ctrl Y` (`^Y`) ed in questo modo il processo verrà sospeso non appena tenderà di leggere dalla tastiera.

Ad un job può essere fatto riferimento tramite una sequenza di caratteri che inizia col simbolo '%'. In particolare

<code>%n</code>	indica il job identificato dal numero <i>n</i> ;
<code>%string</code>	indica il job la cui relativa linea di comando inizia con la stringa <i>string</i> (se esiste più di un job la cui linea di comando inizia con la stringa <i>string</i> , la shell dà un errore);
<code>%%?string</code>	indica il job la cui relativa linea di comando contiene la stringa <i>string</i> (se esiste più di un job la cui linea di comando contiene la stringa <i>string</i> , la shell dà un errore);
<code>%%</code> <code>%%+</code>	indica il job corrente, ovvero l'ultimo job sospeso mentre era in esecuzione in foreground.
<code>%-</code>	indica il job precedente a quello corrente;

I comandi interni `fg` e `bg` sono utilizzati per poter riprendere l'esecuzione di un job rispettivamente in *foreground* e in *background*, ed hanno la seguente sintassi

```
$ fg job
$ bg job
```

dove *job* è l'indicazione del job da portare in esecuzione in foreground o background, come riportato precedentemente (se *job* non è specificato, viene fatto riferimento al job corrente).

Un job può essere portato in foreground o background anche semplicemente specificando sulla linea di comando il suo riferimento. Ad esempio

```
$ %3
```

porterà in esecuzione in foreground il job identificato dal numero 3, mentre

```
$ %3 &
```

lo porterà in esecuzione in background.

```
???
```

Impostazione caratteri (localizzazione, layout ...)

```
???
```

7.4 Riferimenti

```
???
```

¹²Le combinazioni di tasti con `Ctrl` sono in genere rappresentate con il prefisso '^' (*caret notation*).

Capitolo 8

La stampa

???

8.1 Riferimenti

???

Capitolo 9

Il tempo

“Se solo l’avessi saputo, avrei fatto l’orologiaio.”

– A. Einstein

In questo capitolo viene trattato il modo con cui il sistema tiene traccia dello scorrere del tempo e di come i job possono essere lanciati automaticamente in determinati istanti di tempo (determinate ore del giorno, giorni del mese, ...).

9.1 L’orologio di sistema

Il sistema è in grado di gestire l’ora corrente della zona del mondo in cui si trova il computer sul quale esso è installato. A tal fine, tutti i computer sono dotati di un orologio interno detto anche **RTC** (Real Time Clock) che tiene la data e l’ora del sistema. Tale dispositivo è generalmente integrato sulla motherboard ed è alimentato da una batteria in modo tale da continuare a funzionare anche quando il PC è spento. RTC

La gestione del RTC può essere effettuata secondo due modalità differenti:

- Con un’impostazione del tempo relativa al GMT (Greenwich Mean Time)¹, ovvero l’ora relativa al meridiano di Greenwich, e l’indicazione della zona terrestre in cui è situato il computer, in modo tale che il sistema possa gestire, grazie alle indicazioni presenti nel file `/etc/localtime` (prelevate da un database), il fuso orario della zona e le regole del paese in cui ci si trova, come ad esempio la regolamentazione dell’ora legale o *DTS* (Daylight savings time)
- Con un’impostazione del tempo assoluta, senza alcun riferimento alla zona del fuso orario o al paese in cui ci si trova.

Nella gestione dell’orario di un PC sono usuali le indicazioni CET (Central Europe Time) e CEST (Central Europe Summer Time).

Il sistema si aggiorna all’ora mantenuta dal RTC soltanto al suo avvio e mantiene un proprio orologio interno separato dal RTC. Quindi si distingue tra un **hardware clock**, ovvero il RTC (denominato talvolta impropriamente *orologio del BIOS* o *orologio CMOS*) ed un **software clock** (detto anche *orologio di sistema* o *system clock*), quello gestito internamente dal sistema operativo, aggiornato al suo avvio con la data e l’ora mantenuta dal RTC (il *software clock* smette di funzionare quando il PC viene spento). hardware clock
software clock

Una definizione importante quando si ha a che fare con le date, è quella di **epoch**, ovvero l’anno dal quale si intende iniziare il conteggio degli anni con soli due cifre. Per stabilire infatti a che anno si riferisce l’indicazione a due cifre “37” (1937 o 2037 ?) il sistema si basa sull’*epoch*: se le ultime due cifre di *epoch* sono inferiori all’indicazione dell’anno a due cifre (in questo caso 37), allora la data si riferisce al secolo 1900, altrimenti si riferisce al 2000. Dunque, ad esempio, se *epoch* è impostato a 1952, 37 si riferirà al 2037, mentre se *epoch* è impostato 1923, 37 si riferirà al 1937. epoch

¹tale orario è detto anche *UT* (Universal Time) o *UTC* (Universal Time Coordinate).

??? man page setclock (/usr/sbin/setclock) ??? ??? man page timeconfig (/usr/sbin/timeconfig) ???

(controllare che siano valide anche per altre distribuzioni)

9.1.1 Impostazione della data/ora del sistema

L'impostazione della data/ora del sistema è un'operazione delicata che dovrebbe essere effettuata soltanto dall'amministratore del sistema, poiché ad essa potrebbe essere collegato l'avvio di alcuni processi tramite il meccanismo di schedulazione (v. sez. 9.2). A tal proposito si possono attuare tre strategie diverse:

- il riavvio della macchina, la modifica dell'*hardware clock* per mezzo del programma di gestione del BIOS (accessibile generalmente premendo il tasto o subito dopo l'accensione della macchina) e quindi il riavvio del sistema stesso;
- l'impostazione della nuova data/ora direttamente sull'*hardware clock* per mezzo del comando `hwclock`, seguito da un riavvio del sistema. In questo modo il *software clock* si allinea automaticamente alla data/ora impostata sull'*hardware clock* al riavvio del sistema senza subire nessuno squilibrio.
- l'impostazione della nuova data/ora direttamente sul *software clock* per mezzo del comando `date`. Così facendo però si possono provocare degli squilibri che potrebbero portare all'instabilità dell'intero sistema, per cui si renderebbe necessario un riavvio dello stesso. È opportuno ricordare che prima di riavviare la macchina è necessario aggiornare anche l'*hardware clock* per mezzo del comando `hwclock` poiché altrimenti al successivo riavvio del sistema il *software clock* verrebbe aggiornato con il vecchio *hardware clock* che non è stato modificato affatto. Si tenga presente, comunque, che generalmente la modifica della data/ora del *software clock* non porta all'instabilità del sistema, ma questa evenienza non può essere esclusa a priori.

Il comando che permette di visualizzare e/o modificare il *software clock* è `date` (man page `date(1)`).

Comando: `date`

Path: `???/date`

SINTASSI

`# date [option] [+format] [datetime]`

DESCRIZIONE

option specifica delle opzioni di funzionamento di `date`. Può assumere i seguenti valori:

-d *string*

visualizza la data/ora relativa all'indicazione fornita con *string*. Ad esempio il comando

```
# date -d '2 months 3 days'
```

visualizza la data/ora relativa al giorno che verrà tra 2 mesi e 3 giorni rispetto alla data/ora del *software clock*, mentre il comando

```
# date -d '2 days 5 hours ago'
```

visualizza la data/ora relativa al giorno che antecedente di 2 giorni e 5 ore rispetto alla data/ora del *software clock*;

-s *string*

imposta la data/ora del *software clock* con quella indicata da *string*;

format imposta la visualizzazione della data/ora del *software clock*. I valori utilizzati più frequentemente sono:

%a	giorno della settimana abbreviato a 3 lettere;
%A	giorno della settimana per esteso;
%b	nome del mese abbreviato a 3 lettere;
%B	nome del mese per esteso;
%d	giorno del mese (00..31);
%D	data secondo il formato mm/dd/yy (mm/gg/aa);
%H	ora (00..23);
%I	ora (01..12);
%j	giorno dell'anno (001..366);
%k	ora (0..23);
%l	ora (1..12);
%m	mese (01..12);
%M	minuti (00..59);
%p	indicazione AM o PM;
%s	numero di secondi trascorsi dalle 00:00:00 del 1 Gennaio del 1970;
%S	secondi (00..59);
%y	anno, ultime due cifre (00..99);
%Y	anno per esteso;

datetime indica la data/ora con la quale si vuole impostare quella del *software clock*. Il formato di tale parametro è il seguente:

MMDDhhmm[[*CC*]*YY*][*.ss*]

dove

MMDDhhmm

è l'indicazione del mese (*MM*), del giorno (*DD*), dell'ora (*hh*) e dei minuti (*mm*);

CC è l'indicazione del secolo (century), ovvero le due cifre più significative del valore dell'anno;

YY è l'indicazione dell'anno (year), ovvero le due cifre meno significative del valore dell'anno;

ss è l'indicazione dei secondi;

Il comando **date** può essere lanciato anche dagli utenti che non hanno permessi amministrativi sul sistema, con la limitazione però che questi non possono apportare nessuna modifica alla data/ora del *software clock*.

Il comando che permette di visualizzare e/o modificare l'*hardware clock* è **hwclock** (man page **hwclock(8)**), la cui sintassi è la seguente

hwclock [*option*]

dove

option specifica il modo di funzionamento del comando. I valori più utilizzati sono:

- adjust ricalibra l'*hardware clock* ovvero lo aggiorna togliendogli l'errore stimato (contenuto nel file */etc/adjtime*);
- r | --show visualizza la data/ora corrente dell'*hardware clock*;
- set --date=*newdate*
 imposta la data/ora dell'*hardware clock* con quella specificata da *newdate* (il formato è quello utilizzato dal comando **date**);
- s | --hctosys
 imposta la data/ora del *software clock* con quella dell'*hardware clock*.

`-w | --systohc`

imposta la data/ora dell'*hardware clock* con quella del *software clock*.

Il comando `hwclock` utilizza il file `/etc/adjtime` (se esiste) per tenere conto dell'errore sistematico dell'*hardware clock*. Tale file contiene due righe di testo con la seguente sintassi

```
{+|-}daily_drift last_set 0
last_cal
```

dove

`{+|-}daily_drift` è lo scostamento giornaliero dell'orologio (in secondi e decimi) rispetto ad un orologio ritenuto affidabile;

`last_set` è la data/ora dell'ultima volta che l'orologio è stato impostato o calibrato (in numero di secondi dal 1969 UTC);

`last_cal` è la data/ora dell'ultima volta che l'orologio è stato calibrato (in numero di secondi dal 1969 UTC);

Lo 0 è mantenuto per compatibilità con il comando obsoleto `clock`.

Tutte le volte che l'*hardware clock* viene impostato (tramite le opzioni `--set` o `--systohc`), `hwclock` ricalcola lo scostamento rispetto alla data/ora in esso memorizzata (`daily_drift`), basandosi sul tempo che è passato dall'ultima impostazione (`last_set`).

Pertanto è buona norma impartire il comando `hwclock --adjust` subito prima di un `hwclock --hctosys` all'avvio del sistema o anche periodicamente durante l'utilizzo del sistema per mezzo di `cron` (v. sez. ??).

L'ora del sistema diviene particolarmente critica in ambienti in cui più macchine devono essere sincronizzate alla stessa data/ora. Il kernel ha una modalità di funzionamento in cui ogni 11 minuti l'*hardware clock* viene aggiornato automaticamente con il *software clock*. Tale modalità viene attivata dall'avvio del daemon `ntpd` (tale daemon permette di utilizzare il protocollo NTP² con il quale è possibile sincronizzare macchine diverse) e qualunque altra operazione di impostazione manuale dell'*hardware clock* la disattiverà nuovamente.

9.1.2 Il calendario

Nei sistemi Unix-like è possibile avere a disposizione un calendario perpetuo che permette di visualizzare i giorni di qualunque mese di un qualunque anno. Questo può essere fatto con il comando `cal` (man page `cal(1)`) che ha la seguente sintassi:

```
$ cal [option] [month] [year]
```

dove

option sono le opzioni di visualizzazione del calendario. Queste possono assumere i seguenti valori:

- `-m` considera il Lunedì come primo giorno della settimana (anziché la Domenica);
- `-j` visualizza le date in formato giuliano (numero del giorno a partire da 1 dal 1 Gennaio);
- `-y` visualizza il calendario per l'intero anno;

month indica il numero del mese di cui si vuole visualizzare il calendario.

year indica l'anno (a 4 cifre) di cui si vuole visualizzare il calendario.

²v. cap. 19.

Nel caso in cui non si specifica nessun parametro viene visualizzato il calendario del mese in corso.

È assunto che la riforma gregoriana sia avvenuta il 3 settembre 1752. Essa ha eliminato dal calendario i dieci giorni successivi a tale data, per cui il calendario relativo a tale mese risulta un po' insolito.

9.2 La schedulazione dei job

I job possono essere fatti eseguire al sistema impartendo i comandi da tastiera attraverso una shell, ma possono essere anche lanciati in automatico, in corrispondenza di una certa data e ora. Quest'ultima modalità di lanciare i job è detta anche *pianificazione* o **schedulazione dei job** ovvero si dice che i job vengono schedulati o pianificati.

schedulazione dei job

Esistono quindi dei meccanismi attraverso i quali è possibile avvisare il sistema di avviare determinati job in corrispondenza di determinate ore del giorno, determinati giorni dell'anno, ... Tali meccanismi si basano sull'utilizzo di specifici daemon come **crond** e **anacron** e comandi come **at** e **batch**.

9.2.1 cron

Il daemon **crond** (man page **crond(8)**) viene generalmente lanciato da **/etc/rc** durante la procedura di avvio del sistema. Esso ha il controllo dell'eventuale schedulazione dei job presente sul sistema e si basa sul contenuto del file di sistema **/etc/crontab** e dei file contenuti nella directory **/var/spool/cron**.

Per ogni utente del sistema è associato un file **/var/spool/cron/username** atto a contenere le impostazioni relative ai job che l'utente in questione desidera schedulare. In genere ci si riferisce a tale file chiamandolo impropriamente file *crontab* dell'utente *username*. Qualunque job schedulato all'interno del file *crontab* di un utente viene lanciato dal sistema con i privilegi dell'utente possessore del file *crontab* stesso.

Il comando **crontab** (man page **crontab(1)**) è utilizzato per creare e modificare i file *crontab* (man page **crontab(5)**) per la schedulazione dei job. La sintassi è la seguente:

```
$ crontab [opzioni]
```

dove

opzioni indica il funzionamento del comando. Può assumere una dei seguenti valori:

- [-u *username*] *file*** sostituisce il file *crontab* relativo all'utente *username* con il file *file*;
- l [*username*]** visualizza il file *crontab* dell'utente *username*;
- e [*username*]** modifica ed eventualmente crea il file *crontab* dell'utente *username*;
- r [*username*]** cancella il file *crontab* dell'utente *username*;

Se *username* non viene specificato, viene considerato l'utente corrente (quello che ha digitato il comando).

Il file **/var/spool/cron/username** è formato da righe di testo con la seguente sintassi:

```
minute hour day_of_month month day_of_week command
```

dove

minute specifica il minuto in corrispondenza del quale deve essere lanciato il comando *command* (tale valore deve essere compreso tra 0 e 59);

hour specifica l'ora in corrispondenza della quale deve essere lanciato il comando *command* (tale valore deve essere compreso tra 0 e 23);

day_of_month specifica il giorno del mese in corrispondenza del quale deve essere lanciato il comando *command* (tale valore deve essere compreso tra 0 e 31);

month specifica il mese in corrispondenza del quale deve essere lanciato il comando *command* (tale valore deve essere compreso tra 0 e 12 e può essere anche i primi 3 caratteri del nome di un mese);

day_of_week specifica il giorno della settimana in corrispondenza del quale deve essere lanciato il comando *command* (tale valore deve essere compreso tra 0 e 7 – 0 corrisponde a Domenica (come anche 7), 1 corrisponde a Lunedì, ... Possono essere usati anche i primi 3 caratteri del nome del giorno);

command specifica il comando (job) da lanciare;

I comandi (job) sono eseguiti da *cron* quando il *software clock* raggiunge il minuto *minute*, l'ora *hour*, il mese *month* ed almeno uno dei campi *day_of_month* e *day_of_week* coincide con il giorno del mese o della settimana attuale.

Ogni campo che specifica la data/ora di avvio del comando *command*, può essere costituito da un singolo valore numerico o:

Elenco un insieme di numeri (o intervalli) separati da virgole (es. se il campo *hour* è “1,5,7-9”, viene considerato valido per le ore 1, 5, 7, 8 e 9);

Intervallo due numeri separati da un trattino ‘-’. Gli estremi sono sempre compresi (es. se il campo *hour* è “8-11”, viene considerato valido per le ore 8, 9, 10 e 11). L'intervallo può essere anche espresso con un asterisco ‘*’, nel qual caso equivale all'intervallo massimo, cioè assume il significato di “sempre” (la condizione di confronto con il valore del *software clock* è sempre verificata per quel campo).

Per un intervallo può essere specificato anche un *passo*, ovvero l'incremento con il quale devono essere scanditi gli elementi dell'intervallo stesso. Per specificare il passo è sufficiente far seguire l'intervallo da */number*, dove *number* è l'incremento del passo (es. se il campo *hour* è “0-7/2”, viene considerato valido per le ore 0, 2, 4 e 6);

Esiste anche la possibilità di specificare, all'interno dei file *crontab*, assegnamenti di variabili di ambiente. Ciò può essere fatto inserendo delle righe con la seguente sintassi:

```
name = value
```

dove *name* è il nome della variabile di ambiente da impostare e *value* è il valore con cui impostarla.

Il file */etc/crontab* è formato da righe di testo con sintassi leggermente diversa da quella dei file *crontab* degli utenti, infatti ogni riga è composta da:

```
minute hour day_of_month month day_of_week username command
```

in quanto essendo un file di schedulazione generale, è necessario specificare quale sia l'utente (*username*) che virtualmente lancia il comando *command*.

Il processo *cron* controlla ogni minuto i files sopra indicati e lancia i job in essi specificati con la pianificazione impostata. L'output del job viene inviato via posta elettronica all'utente associato al file *crontab*.

```
# Esegue 5 minuti dopo la mezzanotte di ogni giorno.
5 0 * * * $HOME/bin/prova
```

```
# Esegue alle ore 14:15 del primo giorno di ogni mese.
# L'output viene inviato tramite posta elettronica all'utente tizio.
15 14 1 * * $HOME/bin/mensile
```

```
# Esegue alle 22 di ogni giorno lavorativo (da lunedì al venerdì).
# In particolare viene inviato un messaggio di posta elettronica a caio.
0 22 * * 1-5 mail -s Sono le 22 caio%Caio,%%è ora di andare a letto!%

# Esegue 23 minuti dopo mezzanotte, dopo le due, dopo le quattro,...,
# ogni giorno.
23 0-23/2 * * * echo Ciao

# Esegue alle ore 04:05 di ogni domenica.
5 4 * * 0 echo Sono le 04:05 di Domenica
```

Un esempio del file `/etc/crontab` è riportato di seguito (il contenuto di tale file è comune a molte distribuzioni GNU/Linux):

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
02 4 * * 0 root run-parts /etc/cron.weekly
02 4 1 * * root run-parts /etc/cron.monthly
```

Il comando `run-parts` (che in genere è uno script collocato nella directory `/usr/bin/`) viene impostato ad essere eseguito a cadenza oraria, giornaliera, settimanale e mensile. Tale comando avvia tutti i files eseguibili contenuti nella directory indicata come argomento. Pertanto `run-parts /etc/cron.hourly` avvia tutti i files eseguibili contenuti nella directory `/etc/cron.hourly`.

Alcune distribuzioni GNU/Linux hanno introdotto una variante di `crond` di Paul Vixie, estendendo il `crontab` di sistema ai file contenuti nella directory `/etc/cron.d`. Questi file hanno la stessa sintassi di `/etc/crontab` e vengono scanditi assieme a quello principale. In questo modo i vari pacchetti che hanno la necessità di schedulare dei propri job sono svincolati dal dover gestire un unico file (`/etc/crontab`) ma ogni pacchetto ha il proprio file nella directory `/etc/cron.d` che può gestire come crede.

9.2.2 anacron

Il daemon `cron` gestisce la pianificazione dei job sui sistemi che stanno accesi ininterrottamente per molto tempo. Infatti esso lancia i comandi (job) quando la data/ora di sistema soddisfa le indicazioni presenti nei file `crontab` ma se la macchina in tali orari è spenta, i job pianificati non vengono lanciati.

Per dotare anche i PC che vengono accesi soltanto per un periodo di tempo limitato all'interno della giornata, di un meccanismo analogo alla pianificazione gestita da `cron`, si può utilizzare il daemon `anacron` (man page `anacron(8)`). Tale daemon è generalmente lanciato dalla procedura di avvio del sistema (`anacron -s`) e permette di schedulare l'avvio di job a cadenza giornaliera o multipli di giorni. Il processo `anacron` gestisce la pianificazione dei job da lanciare per mezzo del file `/etc/anacrontab` (man page `anacrontab(5)`) (analogo al file `/etc/crontab`) che è costituito da righe con la sintassi seguente:

period delay job-identifier command

dove

period specifica il periodo di tempo espresso in giorni;

delay specifica il ritardo in minuti;

job-identifier è l'identificatore univoco del job da lanciare (generalmente ha lo stesso nome del comando principale del job);

command specifica il comando (job) da lanciare;

Analogamente a quanto visto per i files *crontab*, esiste anche la possibilità di specificare, all'interno del file *anacrontab*, assegnamenti di variabili di ambiente. Ciò può essere fatto inserendo delle righe con la seguente sintassi:

name = *value*

dove *name* è il nome della variabile di ambiente da impostare e *value* è il valore con cui impostarla.

Quando il processo **anacron** viene avviato, esso controlla il file */etc/anacrontab* che contiene la lista dei job da avviare. Per ogni job **anacron** controlla se è stato avviato negli ultimi *period* giorni (per tenere memoria dei comandi che sono stati effettivamente lanciati secondo le indicazioni del file */etc/anacrontab*, **anacron** utilizza dei file contenuti nella directory */var/spool/anacron*). Se non è così **anacron** lancia il comando *command* dopo *delay* minuti e quando il job relativo al comando *command* è terminato, ne registra l'avvenuta esecuzione in appositi file nella directory */var/spool/anacron*. Quando non ci sono più comandi (job) da lanciare **anacron** termina.

Un esempio del file */etc/anacrontab* è riportato di seguito:

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

1      65      cron.daily      run-parts /etc/cron.daily
7      70      cron.weekly    run-parts /etc/cron.weekly
30     75      cron.monthly   run-parts /etc/cron.monthly
```

9.2.3 at

???

9.2.4 batch

???

9.2.5 watch

???

9.3 Riferimenti

???

Capitolo 10

Il suono

???

10.1 Concetti di base

Le variazioni di fenomeni naturali di qualunque tipo (temperatura, umidità, pressione atmosferica, ...) avvengono nel tempo con continuità (*Naturae non fecit saltum*), ovvero si passa da uno stato ad un altro, attraversando tutti gli stati intermedi tra i due: ad esempio, nel passaggio da 24°C a 28°C, in un determinato istante di tempo la temperatura sarà 27°C.

Il suono è il fenomeno prodotto dalla variazione della densità di un mezzo (aria). Esso viene avvertito per mezzo del sistema uditivo degli esseri viventi. Quello che si propaga è appunto la variazione di densità dell'aria provocata dalla sorgente emettrice del suono.

I dispositivi elettronici possono gestire soltanto segnali elettrici legati alla variazione di tensione e corrente nei circuiti. Per questo le variazioni di parametri naturali devono essere convertite in un segnale elettrico per poterlo quindi gestire con un sistema elettronico. I convertitori, che fanno da ponte tra il mondo naturale e quello elettronico sfruttando delle proprietà chimico-fisiche della materia, sono detti **trasduttori**. Sono esempi di trasduttori i microfoni (convertono il suono in un segnale elettrico), gli speaker delle casse che riproducono il suono (convertono i segnali elettrici in suoni), i sensori di temperatura, quelli di umidità, pressione, ...

trasduttori

I segnali elettrici convertiti con i trasduttori sono una rappresentazione più o meno fedele del fenomeno naturale e quindi anch'essi sono segnali che variano nel tempo con continuità. Tale tipo di segnali sono detti **analogici**.

analogici

Un segnale analogico ha un certo andamento nel tempo, che può essere rappresentato graficamente¹ con una curva, detta generalmente forma d'onda. Ogni forma d'onda può essere scomposta, secondo la serie di Fourier, in una somma di sinusoidi con ampiezza, fase e frequenza diverse tra loro. Una sinusoide è un'onda che è descritta dalla relazione

$$y = A \sin(\omega t + \phi)$$

dove t è il tempo (variabile indipendente), ϕ è la **fase** (iniziale) dell'onda, ω è il numero di radianti² al secondo (cioè la velocità) con cui varia il fenomeno ondulatorio, A è l'ampiezza massima dell'onda e y è la variabile dipendente. Una grandezza che varia la sua intensità nel tempo secondo questa legge ha un andamento sinusoidale con **frequenza** $\frac{\omega}{2\pi}$ Hz (Hertz), ovvero $\frac{\omega}{2\pi}$ è il numero di cicli completi che la variazione subisce in 1 secondo.

fase

frequenza

¹nello spazio cartesiano con il tempo sulle ascisse e la grandezza fisica sulle ordinate.

²un radiante è un angolo giro diviso

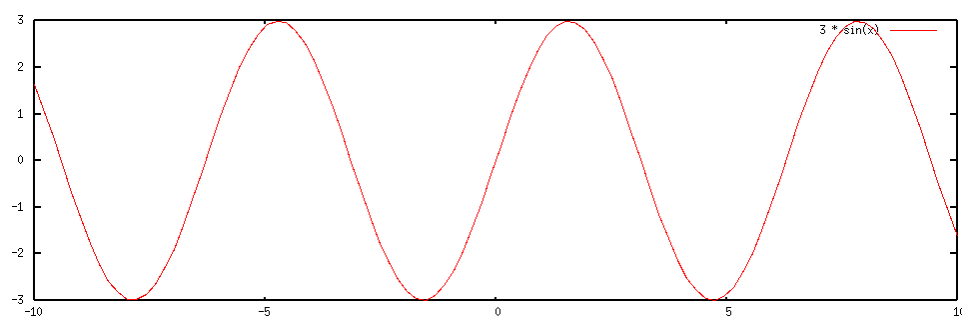


Figura 10.1: Esempio di una sinusoide.

digitali

Poiché i computer utilizzano la logica binaria, i segnali analogici devono essere convertiti in segnali **digitali** per poter essere gestiti da un computer. Questo avviene all'interno delle schede audio, dove è presente un apposito circuito, l'ADC (*Analog to Digital Converter* o *A/D converter*) cioè il convertitore di segnali analogici in digitali. Sulla scheda audio è presente anche il circuito che effettua l'operazione inversa, il DAC (*Digital to Analog Converter* o *D/A converter*) cioè il convertitore di segnali digitali in analogici per ricostruire il segnale da inviare agli speaker delle casse che riprodurranno il suono.

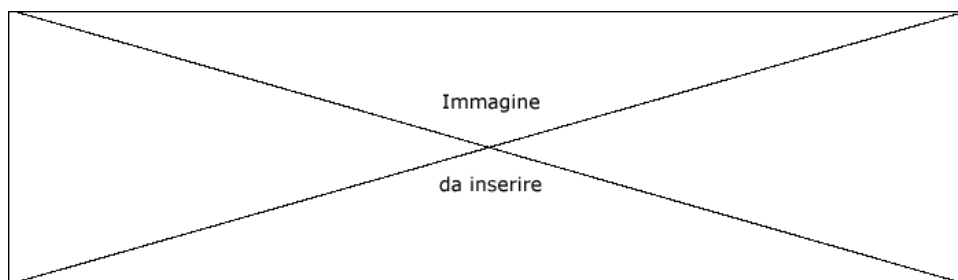


Figura 10.2: Il processo di conversione dei suoni.

campionamento

L'operazione di conversione dal segnale analogico a quello digitale è detta **campionamento** e si basa sulla teoria dell'informazione di Shannon. Il processo infatti consiste nella suddivisione dell'ampiezza del segnale analogico in un numero finito di intervalli, approssimando il segnale stesso con una serie di valori discreti (non continui) prelevati dal segnale stesso ad intervalli di tempo regolari. In questo modo, quanto più piccolo è il numero di intervalli in cui si suddivide l'ampiezza del segnale e quanto più lentamente si prelevano i campioni del segnale stesso, tanto più si perde la forma del segnale analogico originale. Un segnale convertito in digitale in seguito all'operazione di campionamento è detto segnale **digitalizzato**.

digitalizzato

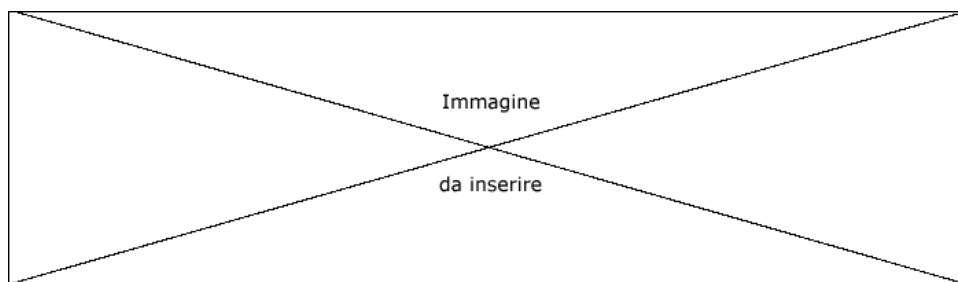


Figura 10.3: Esempio di campionamento di un segnale.

Per essere sicuri di non perdere informazioni nel campionamento del segnale, la teoria di Shannon afferma che il campionamento stesso deve avvenire con una frequenza (numero di campioni prelevati in 1 secondo) almeno doppia rispetto a quella massima presente nel segnale. Altrimenti, nella successiva ricostruzione del segnale si ha il fenomeno dell'**aliasing**: ovvero si ricostruisce un segnale analogico che ha una forma d'onda non corrispondente a quella originale poiché sono andate perdute le informazioni relative alle alte frequenze.

aliasing

Le schede audio utilizzano generalmente una suddivisione dell'ampiezza dei segnali in 256 ($= 2^8$) o 65536 ($= 2^{16}$) intervalli (spesso il numero di intervalli è indicato col numero di bit necessari alla loro rappresentazione, in questo caso 8 o 16) e permettono frequenze di campionamento da 4000 Hz a 44 kHz. I campioni possono essere relativi ad un solo canale (mono) o a due (stereo).

10.2 La riproduzione del suono

La riproduzione del suono è la tecnica per mezzo della quale i suoni vengono emessi da un computer. Si parla di **sintetizzazione** del suono quando la generazione dello stesso è effettuata da un dispositivo elettronico.

sintetizzazione

???

10.2.1 La sintesi FM

Una delle tecniche più vecchie per produrre un suono è quella basata sulla sintesi del segnale da riprodurre, partendo da alcune semplici forme d'onde di base messe a disposizione dalla scheda audio. Le forme d'onda di base (sinusoidali, triangolari e quadre) possono essere amplificate e modulate in frequenza (FM - Frequency Modulation)³ a piacere, per ottenere così il suono da riprodurre.

Attualmente le schede audio in commercio permettono la sintesi FM per compatibilità con il passato e mettono a disposizione più canali di generazione di forme d'onda, cioè più voci.

10.2.2 La sintesi con wavetable

I suoni generati con la sintesi FM non sono dei suoni che somigliano a quelli prodotti da strumenti musicali, né tantomeno a quelli della voce umana. Si ottengono infatti dei suoni "elettronici" piuttosto innaturali.

Quindi si è pensato di registrare i suoni emessi da strumenti musicali e campionarli per poterli memorizzare sulle schede audio. I suoni così memorizzati costituiscono quella che viene detta **wavetable** (tabelle delle onde) per la riproduzione dei suoni. In questo modo, quando un suono deve essere riprodotto, si può scegliere la forma d'onda dalla wavetable (suoni emessi da strumenti musicali reali) e quindi amplificarla opportunamente e modularla in frequenza per ottenere il suono desiderato.

wavetable

10.2.3 Il mixing

Una scheda audio gestisce generalmente più canali di ingresso/uscita del suono. Sulle stesse è quindi presente un apposito circuito, il **mixer**, che permette di selezionare i segnali provenienti dagli ingressi desiderati e di mixarli (mescolarli) per poi inoltrarli sull'uscita desiderata. Ogni segnale può anche essere opportunamente amplificato.

mixer

10.2.4 MIDI

???

³la modulazione di frequenza è essenzialmente la variazione della frequenza della forma d'onda di base.

MIDI message

MIDI (Musical Instrument Digital Interface) è un protocollo standard (hardware e software) per la comunicazione tra strumenti musicali elettronici. Si basa su quelli che sono detti **MIDI message** che costituiscono la quantità minima di informazione scambiabile con questo protocollo.

???

La sequenza di segnali MIDI inviati tra un dispositivo e l'altro può essere anche memorizzata su file in un formato opportuno (formato MIDI) per permettere una successiva riproduzione dei segnali stessi.

Il flusso di dati MIDI è un flusso asincrono con velocità di 31,25 kbit/s, nel quale ogni byte viene trasmesso su 10 bit (1 start bit, 8 bit di dati ed 1 stop bit). Generalmente un flusso MIDI viene generato da un *controller MIDI* (ad esempio una tastiera elettronica) o un *sequencer* e viene inviato ad un generatore di suoni MIDI (*sound source* o *sound module*), che genera appunto i suoni richiesti dal flusso. Un generatore di suoni MIDI può riprodurre più tipi di suoni contemporaneamente, ed in tal caso è detto *multitimbral*.

Molte delle tastiere elettroniche presenti in commercio includono sia la funzionalità di controller MIDI che di generatore di suoni. In questo modo, attraverso un collegamento interno, tali dispositivi possono riprodurre autonomamente, senza l'ausilio di un modulo esterno, i suoni corrispondenti ai tasti premuti sulla tastiera.

Il canale attraverso il quale passa il flusso MIDI viene suddiviso in 16 canali logici.

I MIDI message si dividono in Channel message, che sono riferiti ad uno specifico canale, e System message, non riferiti ad un particolare canale. I Channel message sono suddivisi a loro volta in Voice Message, che contengono le informazioni relative ai suoni da riprodurre, e Mode Message che impostano la modalità di funzionamento del generatore di suoni MIDI (abilitazione a rispondere a tutti i messaggi su tutti i canali o meno (Omni mode), abilitazione a suonare le note in maniera polifonica o meno (Poly mode), ...).

Ogni evento è associato ad un MIDI message. Alla pressione di un tasto sulla tastiera viene inviato il MIDI message Note On, contenente l'indicazione del tasto premuto e la velocity, cioè la rapidità con la quale il tasto è stato premuto. In questo modo il generatore di suoni riproduce il suono corrispondente al tasto premuto con lo strumento impostato sul canale indicato. L'indicazione di una velocity più elevata fa sì che la riproduzione del suono venga amplificata un po' di più del normale. Al rilascio del tasto viene inviato il MIDI Message Note Off, contenente l'indicazione del tasto rilasciato e la velocity (che in genere viene ignorata). Il controller MIDI può generare anche MIDI Message Aftertouch che informa il generatore di suoni MIDI del fatto che il tasto sulla tastiera è stato premuto ulteriormente oltre alla sua pressione normale. In genere il generatore di suoni MIDI produce un effetto sul suono generato (es. vibrato).

10.3 Configurazione

???

10.4 Riferimenti

- J. Tranter, *The Linux Sound HOWTO*, 2001
<http://www.tldp.org/HOWTO/Sound-HOWTO>
- MIDI <http://www.midi.org>
- Linux Audio Users Guide
<http://www.djcj.org/LAU/guide/index.php>

Capitolo 11

L'interfaccia grafica

“L'unico modo per liberarsi da una tentazione è cedervi.”

– O. Wilde

L'interfaccia a caratteri, la shell, sebbene molto potente, è un'interfaccia piuttosto spartana, che poco facilita l'avvicinamento di nuovi utenti al sistema GNU/Linux. Ecco quindi che, sopra la struttura consolidata già da anni, è stata inserita la gestione di un'interfaccia di tipo grafico, che meglio si adegua agli utenti ormai abituati ad utilizzare un sistema di puntamento come il mouse o la penna ottica, piuttosto che lunghe sequenze di comandi.

In questo capitolo verrà illustrata la gestione dell'interfaccia grafica (*X Window System*) e verrà fatta anche una rapida carrellata delle varie interfacce grafiche sviluppate per tale ambiente.

???

11.1 Concetti di base

Parlando di grafica, è opportuno definire alcuni concetti di base.

La luce visibile dall'occhio umano è un'onda elettromagnetica con una frequenza che va da circa 400 THz a 750 THz, ovvero una lunghezza d'onda che va da 750 nm a 400 nm.

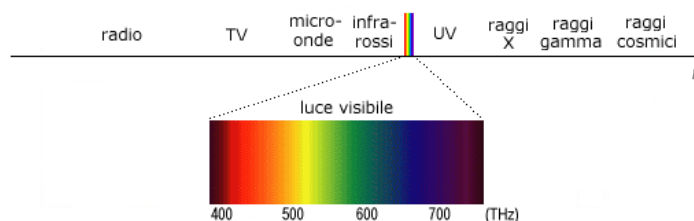


Figura 11.1: Lo spettro elettromagnetico.

I vari tipi di materiale possono essere trasparenti o opachi alla luce visibile, dipendentemente dal fatto se essi si lasciano attraversare da questa o meno. Alcuni materiali possono anche riflettere la luce, ovvero possono riflettere tutte le onde elettromagnetiche nell'intervallo visibile o rifletterne soltanto alcune, assorbendo le altre. La percezione del colore di un oggetto dipende sia dalla luce che investe l'oggetto che dalla caratteristica di riflessione della stessa da parte dell'oggetto (a meno che l'oggetto stesso non sia una sorgente luminosa). Si supponga di avere un oggetto investito da una luce bianca, ovvero composta da tutte le frequenze visibili. L'oggetto apparirà di un certo colore se questo riflette le onde elettromagnetiche che compongono tale colore ed assorbe tutte le altre. Ad esempio, un oggetto che appare essere verde è un effetto del fatto che l'oggetto

stesso riflette soltanto le frequenze delle onde elettromagnetiche visibili relative a quella gradazione di colore, mentre assorbe tutte le altre. Tale oggetto potrebbe risultare di un altro colore se investito da luce non bianca: ad esempio, risulterà nero se è investito soltanto da radiazioni di luce rossa, poiché questa non viene riflessa dall'oggetto. Un oggetto che assorbe tutte le radiazioni elettromagnetiche visibili apparirà nero qualunque sia il colore della luce che lo investe, mentre un oggetto che le riflette tutte apparirà del colore della luce che lo investe.

C'è anche da tenere conto del fatto che l'occhio umano non è ugualmente sensibile a tutte le onde elettromagnetiche che cadono nella banda visibile, ma la sensibilità ha un andamento gaussiano con il picco di massima sensibilità nelle frequenze relative alla luce verde, come riportato in fig. ??

??? figura ??? (risposta dell'occhio umano alla luce visibile)

Un monitor di un computer ha la capacità di visualizzare sullo schermo un determinato numero di punti, come se fosse una griglia: un certo numero di punti orizzontali ed un certo numero di punti verticali, il cui prodotto dà il numero di punti totali visualizzabili sull'intera superficie del monitor. Tali punti (*dot*) sono indicati dalla caratteristica **dpi** (*dot per inch*) del monitor stesso: più elevato è tale valore, più denso di punti sarà lo schermo del monitor, ovvero più punti saranno contenuti in un pollice. Molto spesso questa caratteristica è espressa tramite il suo reciproco, ovvero quello che viene chiamato **dot pitch** che è la dimensione in mm dei punti dello schermo. Tipicamente il valore va da 0,28 a 0,21, corrispondenti rispettivamente a 3,57 punti per mm (85 dpi) ed a 4,76 punti per mm (115 dpi). C'è un'ulteriore considerazione da fare: il dot pitch è generalmente riferito alla diagonale dello schermo e può essere riportato sull'orizzontale dello schermo tenendo conto del fatto che il rapporto d'aspetto dello schermo è 4/3 (rapporto tra la larghezza e l'altezza dello schermo). Quindi, ad esempio ad un dot pitch di 0,28 corrisponde un horizontal dot pitch di circa 0,22, mentre ad un dot pitch di 0,21 corrisponde ad un horizontal dot pitch di circa 0,17.

Ogni punto è formato fisicamente da tre sorgenti luminose di colore diverso: una rossa, una verde ed una blu (RGB – *Red Green Blue*). Ognuno di tali sorgenti luminose si illumina con una determinata intensità quando viene opportunamente stimolato. La realizzazione di tali sorgenti luminose viene effettuata per mezzo di fosfori, nei monitor a tubo catodico (CRT – *Cathode Ray Tube*) o di opportuni cristalli nei monitor a LCD (*Liquid Crystal Display*). Il tempo di decadenza dell'intensità luminosa emessa delle sorgenti è legato in maniera inversamente proporzionale a quella che viene definita **peristenza** dello schermo: più le sorgenti luminose hanno un'elevata persistenza più lentamente l'intensità luminosa emessa si attenuerà nel tempo se la sorgente non viene più stimolata. Quindi, uno schermo ad elevata persistenza può essere utile per immagini fisse, in quanto rende più stabile l'immagine, ma per le immagini in movimento è un disastro: le sorgenti luminose più intensamente stimulate dall'immagine precedentemente visualizzata continueranno ad emettere una forte intensità luminosa per un breve periodo di tempo ancora, quando l'immagine da visualizzare è quella successiva. Si ha quello che viene definito effetto *trailing*. Generalmente gli schermi dei comuni monitor hanno una bassissima persistenza, in maniera tale da essere adatti anche a visualizzare immagini in movimento.

Un'altra caratteristica importante dei monitor è la **banda passante**. Essa specifica la frequenza massima visualizzata sulle righe dello schermo. Infatti, per come sono realizzati i monitor, l'immagine rappresentata sullo schermo viene disegnata per righe successive, a partire dalla prima riga in alto (le righe sono disegnate da sinistra verso destra). Durante il disegno di una singola riga, la variazione luminosa massima consentita dal monitor è indicata appunto dalla banda passante. Ad esempio, un'immagine a strisce verticali bianco-nera può non essere correttamente visualizzata da un monitor, specialmente sui bordi, cioè nel passaggio tra le strisce bianche e quelle nere. Più la banda passante è elevata, più fedele sarà la rappresentazione di questo tipo di immagine sullo schermo del monitor. La banda passante di un monitor si aggira intorno ai 70 MHz (un televisore ha una banda passante generalmente inferiore a 10 MHz e per questo i contorni delle immagini visualizzate risultano meno netti rispetto a quelle visualizzate da un monitor).

L'immagine sullo schermo, almeno nei monitor CRT, necessita di essere rinfrescata, in quanto i fosfori devono essere periodicamente stimolati, altrimenti la luminosità emessa diminuisce nel tempo, molto rapidamente. Quindi, per ottenere un'immagine stabile sullo schermo, il disegno di tutte le righe deve essere effettuato abbastanza rapidamente per poter ricominciare a ridisegnare l'immagine completamente da capo prima che i fosfori abbiano diminuito visibilmente la loro intensità luminosa. Questo si traduce in una **frequenza di refresh verticale** (o frequenza di quadro) dell'ordine di 70-100 Hz. Indicativamente, più tale valore è elevato, più stabile apparirà l'immagine visualizzata sullo schermo e meno affaticante risulterà per la vista. Nei monitor LCD la cosa è un po' diversa, poiché l'emissione luminosa è legata all'orientazione dei cristalli liquidi, in modo da polarizzare opportunamente la luce visibile. L'orientazione dei cristalli liquidi è dovuta alla tensione applicata agli stessi, che è mantenuta costante (non si ha un refresh periodico).

frequenza di refresh
verticale

11.1.1 Le immagini

Un'immagine digitale, cioè la rappresentazione di un'immagine sul computer, si compone di **pixel** che costituiscono le unità minime di informazione luminosa. Ogni pixel è rappresentato da un determinato numero di bit, dipendentemente dal numero di colori che questo può assumere. In particolare ogni pixel è rappresentato da un insieme di bit che indicano nell'ordine: l'intensità di rosso, quella di verde e quella di blu (RGB) che deve essere associata al relativo punto dell'immagine per poter essere rappresentato correttamente sullo schermo. Ad esempio supponendo di rappresentare ogni pixel con 12 bit, si avranno 4 bit per il rosso, 4 per il verde e 4 per il blu, quindi un pixel associato al colore 101100001111_2 sarà rappresentato con un colore con componente rossa di intensità 1011_2 , nessuna componente verde (0000_2) e componente blu massima (1111_2).

pixel

??? figura ??? [RRRRRRRR GGGGGGGG BBBBBBBB]

Il numero di bit con il quale viene rappresentato ogni singolo pixel è anche detto **profondità di colore** dell'immagine, per cui il massimo numero di colori rappresentabili in tale immagine è dato da

profondità di colore

$$n_{max_colori} = 2^{n_{bit_pixel}}$$

dove n_{bit_pixel} è il numero di bit utilizzati per la rappresentazione di ogni pixel dell'immagine. In questo modo si hanno immagini a 8 bit per pixel che possono contenere al massimo $2^8 = 256$ colori, immagini con profondità di colore a 12 bit che possono contenere al massimo $2^{12} = 4.096$ colori, immagini con profondità di colore a 16 bit che possono contenere al massimo $2^{16} = 65.536$ colori (*highcolor*) ed immagini con profondità di colore a 24 bit che possono contenere al massimo $2^{24} = 16.777.216$ colori (*truecolor*).

I colori principali, secondo la codifica RGB, si ottengono per mezzo delle combinazioni espresse in tab. 11.1.

Colore	R	G	B
rosso	●	0	0
giallo	●	●	0
verde	0	●	0
ciano	0	●	●
blu	0	0	●
viola	●	0	●

Tabella 11.1: Combinazioni dei colori principali nella rappresentazione RGB.

La rappresentazione dei colori nelle componenti RGB è soltanto una tra le rappresentazioni possibili. Un'altra rappresentazione utilizzata è quella HSV (Hue, Saturation, Value) ovvero tinta, saturazione e valore. La *tinta* è un valore che indica il colore principale come l'angolo in gradi (da 0 a 360) nella ruota dei colori (v. fig. ??), la *saturazione* indica la quantità di bianco presente nel colore (da 0 ad 1) ed il *valore* indica l'intensità di colore.

??? figura ???

La palette

Molte volte capitano situazioni in cui le immagini utilizzano soltanto una parte dei colori possibili relativamente alla profondità di colore con cui vengono realizzate. Ad esempio, lavorando con una profondità di colore a 24 bit è difficile, specialmente in un'immagine di dimensioni ridotte, che vengano utilizzati tutti i possibili colori all'interno dell'immagine. In tal caso è possibile creare una palette (tavolozza) dei colori che sono effettivamente utilizzati nell'immagine e riferirsi a tale palette per la decodifica dei colori. Ad esempio, una volta realizzata l'immagine, scegliendo i colori da tutti quelli possibili, si supponga che risulti che il primo pixel in alto a sinistra sia realizzato con il colore $3E01BF_H$, il secondo con il colore $442E0B_H$ ed il resto dei pixel con il colore 000000_H (nero). Si può pensare di memorizzare nel file relativo all'immagine, una palette composta soltanto dai colori utilizzati ($3E01BF_H$, $442E0B_H$ e 000000_H) e di memorizzare per ogni pixel soltanto 2 bit (anziché 24 bit) che indicano la posizione del colore contenuto nella palette associato al pixel considerato (2 bit sono sufficienti a distinguere fino a 4 colori diversi). Dunque, il primo pixel, associato al colore $3E01BF_H$, si riferirà al primo colore presente nella palette (00_2), il secondo al secondo colore della palette (01_2) e gli altri al terzo colore (10_2). Così facendo si risparmiano un bel po' di byte nella memorizzazione dell'immagine (22 bit per pixel) ma, oltre all'immagine si deve memorizzare anche la palette (i colori utilizzati). Comunque per immagini di medie e grandi dimensioni che fanno uso di pochi colori, il risparmio di byte è notevole.

???

11.1.2 La risoluzione dello schermo

risoluzione

Le interfacce grafiche permettono di impostare quella che viene detta **risoluzione** dello schermo. Questa consiste nell'impostazione del numero di pixel da visualizzare sullo schermo del monitor, espressa dal numero di pixel per riga e da quello per colonna. Ad esempio, impostando una risoluzione dello schermo di 640×480 pixel si ha un adattamento del monitor a rappresentare orizzontalmente un massimo di 640 pixel e verticalmente 480 pixel. Questo fa sì che l'interfaccia grafica gestisca i punti dello schermo per rappresentare opportunamente quanto richiesto, per cui un pixel sarà formato da un numero di punti adeguato (ad esempio su un monitor a 15" un pixel sarà rappresentato da circa 4 punti: 2 orizzontali \times 2 verticali). Scegliendo una risoluzione di 1024×768 si ha un adattamento del monitor a rappresentare orizzontalmente un massimo di 1024 pixel e verticalmente 768 pixel. Questo fa sì che l'interfaccia grafica gestisca i punti dello schermo per rappresentare opportunamente quanto richiesto, per cui un pixel sarà formato da un numero di punti adeguato (ad esempio su un monitor a 15" un pixel sarà rappresentato generalmente da 1 punto). Aumentando la risoluzione è possibile che non tutti i pixel vengano visualizzati sullo schermo, poiché lo schermo stesso non ha un numero sufficiente di punti per poter rappresentare tutti i pixel richiesti.

11.1.3 Le schede video

Le schede video sono i circuiti che il sistema utilizza per rappresentare i messaggi e le immagini da visualizzare sullo schermo. Esistono essenzialmente due grandi filoni di schede video: quelle standard e quelle accelerate. La differenza consiste nella velocità di visualizzazione delle immagini. Questo incide notevolmente sulle animazioni. Infatti un'animazione è essenzialmente composta da una sequenza di immagini che devono essere visualizzate sullo schermo ad una velocità piuttosto rapida in maniera tale che l'occhio umano percepisca una continuità di movimento piuttosto che immagini separate tra loro. Questo si ottiene visualizzando le immagini in successione con una velocità di almeno 15 fps (*frames per second*), ovvero 15 immagini (o quadri) al secondo.

Le schede standard sono essenzialmente dei circuiti di gestione delle immagini piuttosto semplici, di vecchia generazione, mentre le schede accelerate montano dei veri e

propri microprocessori in grado di compiere elaborazioni complesse sulle immagini, in maniera da svincolare la CPU dai calcoli per l'elaborazione delle immagini da visualizzare. In questo modo le schede accelerate permettono di ottenere rappresentazioni sullo schermo di sequenze di immagini a velocità impressionanti: si raggiungono anche velocità dell'ordine di 50-100 fps.

11.2 X Window System

La gestione dell'interfaccia grafica di GNU/Linux si basa sul concetto di applicazione client-server (v. cap. 16): esiste cioè un'applicazione detta *server grafico* che mette a disposizione delle routine le quali possono essere chiamate da altri programmi, i *client*, per poter compiere delle operazioni di tipo grafico. In questo modo non è necessario che il client ed il server vengano eseguiti sullo stesso sistema, ma possono essere eseguiti anche su sistemi fisicamente diversi.

Il server grafico va sotto il nome di *X Window System* o *X Window* o più semplicemente *X* ed è il programma che gestisce la visualizzazione dell'interfaccia grafica.

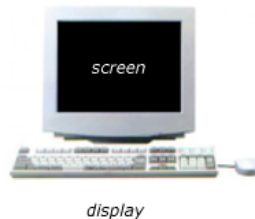


Figura 11.2: Una stazione grafica.

Il server grafico (anche detto *X server*) si basa su quello che viene chiamato **display**, *display* cioè una stazione grafica, costituita da una tastiera, un dispositivo di puntamento (mouse) e da uno o più **screen** (schermi) (v. fig. 11.2). Ogni schermo (*screen*) di ogni stazione *screen* grafica (*display*) è contraddistinto da un identificatore univoco che ha la seguente sintassi

`[hostname]:display[.screen]`

dove

hostname

è il nome della macchina sulla quale è in esecuzione il server grafico. Se non indicato viene considerata la macchina locale;

display è il numero che indica il display iniziando da 0;

screen è il numero che indica lo schermo (iniziando da 0) relativo al display identificato da *display*. Se non indicato viene considerato il primo schermo (quello identificato dal numero 0);

Le applicazioni che utilizzano l'interfaccia grafica (i client grafici) generalmente si riferiscono ad uno schermo di una stazione grafica che, se non indicati, vengono considerati quelli relativi all'identificatore `:0.0`, cioè il primo schermo relativo alla prima stazione grafica della macchina locale. La comunicazione tra il server grafico ed il client grafico avviene per mezzo di un opportuno protocollo, detto *X protocol*.

I dispositivi di puntamento sono dotati di un certo numero di tasti (ad esempio i mouse ne hanno due, tre o più). Il server grafico riconosce fino a 5 tasti che, in particolare, per i mouse corrispondono il primo al tasto sinistro, il secondo al tasto centrale (se esiste) ed il terzo a quello destro.

Poiché la gestione dell'interfaccia grafica si basa sull'utilizzo di sistemi di puntamento con almeno tre tasti, è prevista la possibilità di emulare la funzionalità dei tasti non presenti sul dispositivo di puntamento. Ad esempio, il tasto centrale, assente in un mouse a 2 soli tasti, può essere emulato con la pressione contemporanea degli altri due.

Anche se generalmente ne viene avviato soltanto uno, in un sistema GNU/Linux possono essere in esecuzione anche più server grafici contemporaneamente, ognuno dei quali gestirà i propri display con i relativi screen (v. fig. 11.3). Il server grafico non interagisce direttamente con l'utente, ma mette a disposizione un servizio che viene utilizzato dalle applicazioni (*client grafici*) per interagire con l'utente attraverso la loro interfaccia grafica.

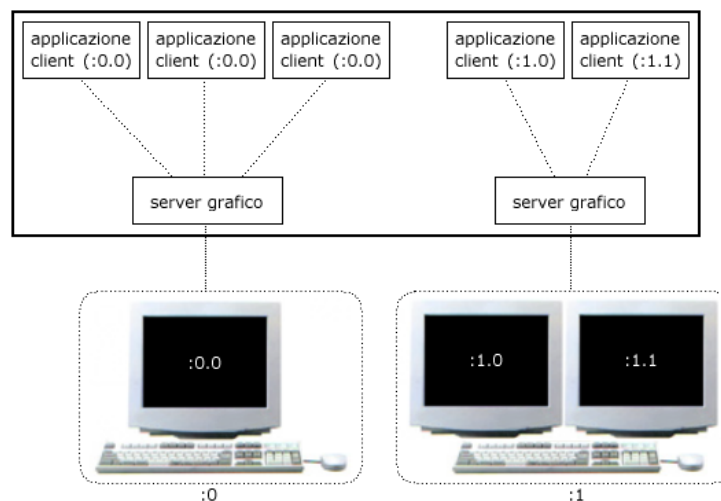


Figura 11.3: Gestione dell'interfaccia grafica.

Ad esempio, il comando **gedit** visualizza una finestra sullo schermo in cui l'utente può creare, visualizzare e/o modificare file di testo (v. fig. 11.4). L'applicazione lanciata da **gedit** è un client grafico che, colloquiando con il server grafico per mezzo dell'X protocol, accede alle routine per il disegno della propria interfaccia grafica, cioè una finestra con una *toolbar* contenente dei pulsanti grafici (*button*), dei *tab folder* (v. quello di fig. 11.4 che contiene la scritta "Untitled 1") ed un'area in cui può essere visualizzato/modificato del testo. Le routine che disegnano tutto ciò sono messe a disposizione da apposite librerie grafiche, dette *Xlib*, le quali sono accedute per mezzo del server grafico. Il client grafico specifica al server grafico soltanto quello che vuole che sia disegnato sullo schermo: come farlo è compito del server grafico e delle librerie grafiche.

Il server grafico in esecuzione rimane in ascolto sulla porta¹ 6000 + *display*. Quindi i client grafici (applicazioni che utilizzano l'interfaccia grafica) dovranno connettersi al server grafico attraverso la porta 6000 per comunicazioni relative al display 0, alla porta 6001 per comunicazioni relative al display 1, ...

11.2.1 XFree86

Esistono varie implementazioni di server grafici, fra le quali una molto interessante e free software che è *XFree86*². La versione di *XFree86* alla quale viene fatto riferimento

¹v. cap. 18.

²il nome deriva dal fatto che il server è stato sviluppato per piattaforme basate sul chip *Intel X386* (i386) e la pronuncia di "three" è simile a quella di "free" che ben ricorda il carattere libero del progetto.

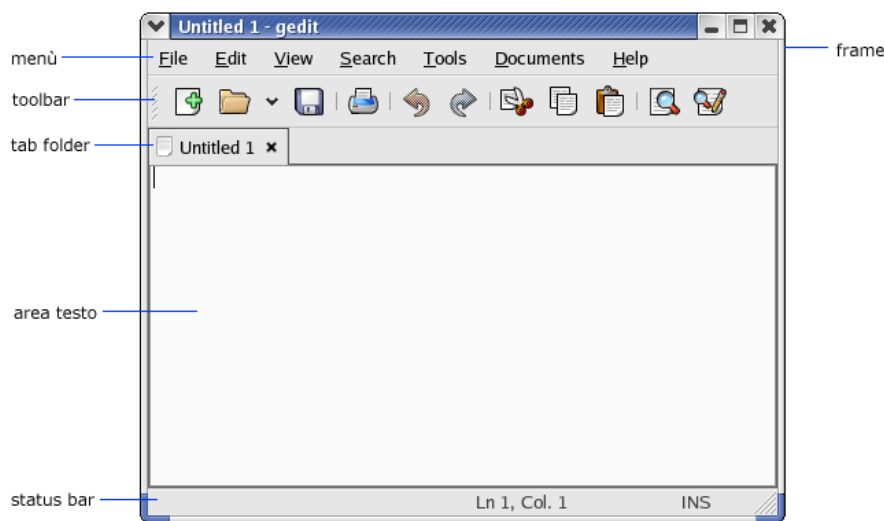


Figura 11.4: L'interfaccia grafica di gedit.

nel presente testo è la 4.³

Tale server grafico viene lanciato con il comando **XFree86** (man page **XFree86(1)**) o attraverso il symbolic link **/usr/X11R6/bin/X**. Una volta avviato, il server grafico presenta soltanto un'immagine di sfondo sulla quale viene fatto muovere un puntatore secondo gli input ricevuti dal sistema di puntamento (mouse). Per poter interagire con esso è necessario lanciare qualche programma che permetta di farlo, ovvero un client grafico.

Comando: **XFree86**
 Path: **/usr/X11R6/bin/XFree86**
 SINTASSI
XFree86 [*:display*] [*option*]

DESCRIZIONE

display indica il display (stazione grafica) alla quale è associato.

option indica la modalità di funzionamento di **XFree86**. Può assumere i seguenti valori:

- vtxx** specifica il numero del terminale virtuale utilizzato dal server grafico per la visualizzazione (sulle piattaforme sulle quali sono disponibili i terminali virtuali). Se non viene specificato, **XFree86** utilizza il primo terminale virtuale disponibile;
- allowMouseOpenFail** indica al server grafico di avviarsi anche nel caso in cui il dispositivo di puntamento non sia correttamente inizializzato;
- allowNonLocalModInDev** indica al server grafico di permettere cambiamenti sulle configurazioni della tastiera e del dispositivo di puntamento anche ai client remoti;
- allowNonLocalModInDev** indica al server grafico di permettere cambiamenti sulle configurazioni della tastiera e del dispositivo di puntamento anche ai client remoti;

³la versione 3 di XFree86 era sostanzialmente una collezione di server grafici ognuno specifico per determinate schede video. Con la versione 4 è fornito un solo server grafico migliorato, che si adatta alle varie schede video (anche se su alcune schede video ormai obsolete non si hanno le ottimizzazioni presenti nel server grafico fornito con la versione 3).

- allowNonLocalXvidtune**
indica al server grafico di permettere la connessione tramite **xvidtune** da sistemi remoti;
- bgamma *value***
specifica la correzione di gamma del blu, secondo quanto specificato da *value* (il valore va da 0,1 a 10 – il default è 1);
- configure**
indica al server grafico di caricare tutti i moduli e di effettuare un test sui dispositivi disponibili e di scrivere un file di configurazione XF86Config con i risultati ottenuti dal test (disponibile solo per il superuser);
- crt *dev***
funziona in maniera analoga all'opzione **vt**. Riservato per la compatibilità con il server grafico nativo di SCO;
- depth *n***
specifica la profondità di colore di default (i valori possibili sono 1, 4, 8, 15, 16 e 24);
- disableModInDev**
indica al server grafico di disabilitare il cambiamento delle impostazioni relative ai dispositivi;
- disableVidMode**
indica al server grafico di disabilitare le estensioni VidMode utilizzate per cambiare la modalità video;
- fbbpp *n***
specifica il numero di bit di framebuffer per pixel secondo quanto indicato da *n* (i valori possibili sono 1, 8, 16, 24 e 32). In genere tale valore viene desunto dall'opzione **-depth**;
- flipPixels**
indica al server grafico di scambiare i valori dei pixel bianchi con quelli neri;
- gamma *value***
specifica la correzione di gamma (per il rosso, il verde ed il blu) secondo quanto indicato da *value* (il valore va da 0,1 a 10 – il default è 1);
- ggamma *value***
specifica la correzione di gamma del verde, secondo quanto specificato da *value* (il valore va da 0,1 a 10 – il default è 1);
- ignoreABI**
indica al server grafico di ignorare l'indicazione della revisione ABI al caricamento dei moduli. In questo modo moduli con revisioni ABI più recenti rispetto a quella del server grafico possono essere caricati, ma potrebbero utilizzare interfacce che il server grafico potrebbe non avere;
- keeptty**
indica al server grafico di rimanere agganciato al terminale virtuale nel quale è stato avviato (utile per il debug);
- keyboard *keyb_name***
specifica il dispositivo da considerare come tastiera, tra quelli elencati nel file di configurazione (nella sezione **InputDevice**), secondo quanto indicato da *keyb_name*;
- layout *layout_name***
specifica la sezione **Layout** da considerare tra quelle presenti nel file di configurazione, secondo quanto indicato da *layout_name* (per default viene utilizzata la prima sezione **Layout** contenuta nel file di configurazione);
- logfile *filename***
specifica il file in cui il server grafico può scrivere i messaggi di log, secondo quanto indicato da *filename* (per default il file di log è **/var/log/XFree86.*n*.log** dove *n* è il display associato al server – questa opzione è utilizzabile solo dal superuser);
- logverbose [*n*]**
specifica il livello di verbosità per i messaggi di log, secondo quanto indicato da *n* (più elevato è il valore *n* più verboso sarà il log – il

valore di default è 3). Se *n* non è specificato, ogni occorrenza di questa opzione aumenta il livello di verbosità del log;

-modulepath *path*
specifica il percorso di ricerca dei moduli secondo quanto specificato da *path*, che è costituito da un elenco di directory separate dal carattere ':' (questa opzione è utilizzabile solo dal superuser);

-nosilk
indica di disabilitare il supporto per i mouse Silken;

-pixmap24
imposta la profondità di colore interna al server a 24 bit per pixel (il default è 32);

-pixmap32
imposta la profondità di colore interna al server a 32 bit per pixel;

-pointer *pointer_name*
specifica il dispositivo di puntamento da considerare, secondo quanto indicato da *pointer_name*, tra quelli elencati nella sezione **InputDevice** del file di configurazione. Se non specificato viene considerato il primo dispositivo di puntamento indicato nella sezione **InputDevice** del file di configurazione (questa opzione è ignorata se è specificato un dispositivo di puntamento nella sezione **Layout** del file di configurazione);

-probeonly
indica al server di eseguire soltanto un test sui dispositivi, con le impostazioni contenute nel file di configurazione (le impostazioni mancanti vengono rilevate automaticamente dal server);

-quit indica di non visualizzare messaggi informativi. Il livello di verbosità del log viene impostato a 0;

-rgamma *value*
specifica la correzione di gamma del rosso, secondo quanto specificato da *value* (il valore va da 0,1 a 10 – il default è 1);

-scanpci
indica al server grafico di eseguire una scansione dei dispositivi presenti sul bus PCI e di visualizzare dei messaggi informativi su di essi;

-screen *screen_name*
specifica la sezione **Screen** da considerare tra quelle presenti nel file di configurazione, secondo quanto indicato da *screen_name* (per default viene considerata la prima sezione **Screen** contenuta nel file di configurazione);

-showconfig
v. l'opzione **-version**;

-weight *n*
specifica il peso dei colori con profondità di colore a 16 bit (il valore di default di *n* è 565);

-verbose [*n*]
specifica il livello di verbosità delle informazioni visualizzate sullo standard error, secondo quanto specificato da *n*. Se *n* non viene specificato, ogni occorrenza dell'opzione **-verbose** incrementa il livello di verbosità degli errori (il valore di default di *n* è 0);

-version
indica a **XFree86** di visualizzare la versione ed altre indicazioni;

-xf86config *file*
specifica il file di configurazione secondo quanto indicato da *file*;

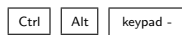
???

Ctrl Alt Backspace

termina l'esecuzione del server grafico. L'effetto di tale combinazione di tasti può essere disabilitato con la direttiva **DontZap** nel file di configurazione;

Ctrl Alt keypad +

cambia la risoluzione dello schermo in quella successiva specificata nel file di configurazione. L'effetto di tale combinazione di tasti può essere disabilitato con la direttiva **DontZoom** nel file di configurazione;



cambia la risoluzione dello schermo in quella successiva specificata nel file di configurazione. L'effetto di tale combinazione di tasti può essere disabilitato con la direttiva `DontZoom` nel file di configurazione;



cambia il terminale virtuale visualizzato sullo schermo;

Quando il server grafico viene avviato da un utente non superuser, esso ricerca un file di configurazione secondo l'ordine dei percorsi di seguito riportati

```
/etc/X11/cmdline
/usr/X11R6/etc/X11/cmdline
/etc/X11/$XF86CONFIG
/usr/X11R6/etc/X11/$XF86CONFIG
/etc/X11/XF86Config-4
/etc/X11/XF86Config
/etc/XF86Config
/usr/X11R6/etc/X11/XF86Config.hostname
/usr/X11R6/etc/X11/XF86Config-4
/usr/X11R6/etc/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config.hostname
/usr/X11R6/lib/X11/XF86Config-4
/usr/X11R6/lib/X11/XF86Config
```

dove *cmdline* è il path relativo (senza `..`) specificato con l'opzione `-xf86config` sulla riga di comando all'avvio del server grafico, *\$XF86CONFIG* è il path relativo (senza `..`) indicato dalla variabile d'ambiente *XF86CONFIG* e *hostname* è il nome della macchina (v. *hostname*).

Se il server grafico viene avviato dal superuser, l'ordine dei percorsi di ricerca del file di configurazione è quello seguente

```
cmdline
/etc/X11/cmdline
/usr/X11R6/etc/X11/cmdline
$XF86CONFIG
/etc/X11/$XF86CONFIG
/usr/X11R6/etc/X11/$XF86CONFIG
$HOME/XF86Config
/etc/X11/XF86Config-4
/etc/X11/XF86Config
/etc/XF86Config
/usr/X11R6/etc/X11/XF86Config.hostname
/usr/X11R6/etc/X11/XF86Config-4
/usr/X11R6/etc/X11/XF86Config
/usr/X11R6/lib/X11/XF86Config.hostname
```

dove *cmdline* è il path (assoluto o relativo) specificato con l'opzione `-xf86config` sulla riga di comando all'avvio del server grafico, *\$XF86CONFIG* è il path (assoluto o relativo) indicato dalla variabile d'ambiente *XF86CONFIG*, *\$HOME* è il path indicato dalla variabile d'ambiente *HOME* e *hostname* è il nome della macchina (v. *hostname*).

In genere, il file di configurazione considerato all'avvio del server grafico sulla maggior parte dei sistemi GNU/Linux è `/etc/X11/XF86Config` (man page `XF86Config(5)`).

Il file di configurazione contiene indicazioni sui dispositivi ed è composto da sezioni con la seguente struttura

```
Section "SectionName"
    SectionEntry
```

...
EndSection

Il contenuto del file è case-insensitive ed il carattere ‘_’ viene ignorato.
I nomi delle possibili sezioni sono elencati in tab. 11.2.

Section Name	Significato
Files	Percorsi relativi ai file.
ServerFlags	Impostazioni di flag del server.
Module	Caricamento dinamico di moduli.
InputDevice	Dispositivi di input (tastiera, mouse, ...).
Device	Schede grafiche.
VideoAdaptor	Adattatori video.
Monitor	Monitor.
Modes	Risoluzioni dello schermo.
Screen	Schermo.
ServerLayout	Layout.
DRI	DRI.
Vendor	Vendor.
Keyboard	Tastiera (obsoleto – utilizzare InputDevice).
Pointer	Dispositivo di puntamento (obsoleto – utilizzare InputDevice).

Tabella 11.2: Nomi delle sezioni ne file di configurazione di XFree86.

Le sezioni a più alto livello sono quelle indicate come **ServerLayout**. Queste collegano le impostazioni relative ai dispositivi di input (le cui impostazioni sono definite nella sezione **InputDevice**) e di output (definiti in varie sezione e raccolti nella sezione **Screen**) che verranno utilizzati. La sezione **Screen** collega una scheda grafica ad un monitor, mentre le schede grafiche ed i monitor sono definiti rispettivamente nelle sezioni **Device** e **Monitor**.

All’interno di una sezione, ogni riga contiene una parola chiave (keyword) ed eventuali argomenti che specificano l’impostazione.

La sezione **Files** è utilizzata per specificare i percorsi di ricerca dei file. Le direttive che possono essere contenute in tale sezione sono le seguenti

FontPath *path*

Imposta il percorso di ricerca per i file relativi ai font. L’argomento *path* è un elenco di percorsi separato dal carattere ‘,’. I percorsi di ricerca possono essere dei path assoluti o identificatori di server di font. Gli indicatori dei server di font hanno la sintassi seguente

trans/hostname:port_number

dove

trans indica il tipo di transport layer (v. cap. ??) da utilizzare per il collegamento al font server (può essere **unix** per i socket Unix-domain e **tcp** per i socket TCP/IP);

hostname

indica il nome della macchina sulla quale è in esecuzione il font server;

port_number

indica il numero della porta sulla quale è in ascolto il font server (in genere è 7100);

Se tale direttiva non è specificata, viene considerato il seguente elenco di percorsi

```
/usr/X11R6/lib/X11/fonts/misc/
/usr/X11R6/lib/X11/fonts/Speedo/
/usr/X11R6/lib/X11/fonts/Type1/
/usr/X11R6/lib/X11/fonts/CID/
```

```
/usr/X11R6/lib/X11/fonts/75dpi/
/usr/X11R6/lib/X11/fonts/100dpi/
```

È consigliato utilizzare il seguente elenco di percorsi

```
/usr/X11R6/lib/X11/fonts/local/
/usr/X11R6/lib/X11/fonts/misc/
/usr/X11R6/lib/X11/fonts/75dpi:unscaled
/usr/X11R6/lib/X11/fonts/100dpi:unscaled
/usr/X11R6/lib/X11/fonts/Type1/
/usr/X11R6/lib/X11/fonts/CID/
/usr/X11R6/lib/X11/fonts/Speedo/
/usr/X11R6/lib/X11/fonts/75dpi/
/usr/X11R6/lib/X11/fonts/100dpi/
```

I percorsi non validi non vengono considerati.

RGBPath "*path*"

Imposta il percorso del database dei colori RGB. Se tale direttiva non è specificata, viene considerato il percorso `/usr/X11R6/lib/X11/rgb`.

ModulePath "*path*"




Imposta il percorso di ricerca per il caricamento dei moduli relativi al server grafico. L'argomento *path* è un elenco di directory separato dal carattere `'`.

La sezione **ServerFlags** è utilizzata per specificare le opzioni globali del server grafico. Le direttive che possono essere contenute in tale sezione sono le seguenti







Option "NoTrapSignals" "*value*"

Indica al server grafico, dipendentemente dal valore di *value*, di raccogliere (**Off**) o meno (**On**) segnali "fatali" (utile in caso di debug);

Option "DontZap" "*value*"

Indica al server grafico, dipendentemente dal valore di *value*, di ignorare (**On**) o meno (**Off**) la combinazione di tasti    utilizzata per terminare il server grafico (default: **Off**);

Option "DontZoom" "*value*"

Indica al server grafico, dipendentemente dal valore di *value*, di ignorare (**On**) o meno (**Off**) le combinazioni di tasti    e    utilizzate per cambiare la risoluzione dello schermo (default: **Off**);

Option "DisableVidModeExtension" "*value*"

Indica al server grafico, dipendentemente dal valore di *value*, di disabilitare (**On**) o meno (**Off**) le estensioni VidMode utilizzate dal client **xvidtune** per cambiare la risoluzione dello schermo (default: **Off**);

Option "AllowNonLocalXvidtune" "*value*"








Indica al server grafico, dipendentemente dal valore di *value*, di abilitare (**On**) o meno (**Off**) i client **xvidtune** (o quelli che utilizzano le estensioni VidMode) a potersi connettere da un'altra macchina (default: **Off**);

Option "DisableModInDev" "*value*"

Indica al server grafico, dipendentemente dal valore di *value*, di disabilitare (**On**) o meno (**Off**) l'utilizzo delle estensioni XFree86-Misc per la modifica dei dispositivi di input (default: **Off**);

Option "AllowNonLocalModInDev" "*value*"

Indica al server grafico, dipendentemente dal valore di *value*, di abilitare (**On**) o meno (**Off**) la possibilità di modificare le impostazioni dei dispositivi di input (mouse e tastiera) da remoto (default: **Off**);

- Option "AllowMouseOpenFail" "value"**
Indica al server grafico, dipendentemente dal valore di *value*, di permettere (**On**) o meno (**Off**) l'avvio nel caso in cui il dispositivo di puntamento non sia stato correttamente configurato (default: **Off**);
- Option "VTInit" "command"**
Indica al server grafico, di eseguire il comando *command* subito dopo l'apertura del terminale virtuale. Tale comando viene eseguito come `/bin/sh -c command`;
- Option "VTSysReq" "value"**
Indica al server grafico, dipendentemente dal valore di *value*, di abilitare (**On**) o meno (**Off**) la modalità di cambiamento del terminale virtuale visualizzato sullo schermo secondo lo standard *SysV*, che utilizza la sequenza di tasti . (default: **Off**);
- Option "BlankTime" "value"**
Imposta il timeout dopo il quale lo schermo deve essere annerito, secondo quanto specificato da *value*. Tale valore è espresso in minuti (default: 10);
- Option "StandbyTime" "value"**
Imposta il timeout dopo il quale lo schermo deve passare in stand-by (DPMS), secondo quanto specificato da *value*. Tale valore è espresso in minuti (default: 20);
- Option "SuspendTime" "value"**
Imposta il timeout dopo il quale lo schermo deve passare in suspend mode (DPMS), secondo quanto specificato da *value*. Tale valore è espresso in minuti (default: 30);
- Option "OffTime" "value"**
Imposta il timeout dopo il quale lo schermo deve passare in off mode (DPMS), secondo quanto specificato da *value*. Tale valore è espresso in minuti (default: 40);
- Option "Pixmap" "value"**
Imposta la profondità di colore, secondo quanto specificato da *value*. Tale valore è espresso in bit e può assumere i valori 24 o 32 (default: 32);
- Option "PC98" "value"**
Indica al server grafico, dipendentemente dal valore di *value*, se si tratta di una macchina giapponese PC-98 (**On**) o meno (**Off**) (default: auto detected);
- Option "NoPM" "value"**
Indica al server grafico, dipendentemente dal valore di *value*, di disabilitare (**On**) o meno (**Off**) gli eventi collegati alla gestione del consumo di energia elettrica (Power Management) (default: auto detected);
- Option "Xinerama" "value"**
Indica al server grafico, dipendentemente dal valore di *value*, di abilitare (**On**) o meno (**Off**) l'estensione XINERAMA (default: **Off**);
- Option "AllowDeactivateGrabs" "value"**
Indica al server grafico, dipendentemente dal valore di *value*, di abilitare (**On**) o meno (**Off**) l'utilizzo della sequenza di tasti    per disattivare i grab relativi a tastiera e mouse (default: **Off**);
- Option "AllowClosedownGrabs" "value"**
Indica al server grafico, dipendentemente dal valore di *value*, di abilitare (**On**) o meno (**Off**) l'utilizzo della sequenza di tasti    per terminare i client che hanno un grab relativo a tastiera o mouse (default: **Off**);

La sezione **Module** è utilizzata per specificare i moduli che devono essere caricati dinamicamente. Le direttive che possono essere contenute in tale sezione sono le seguenti

Load "modulename"

Indica al server grafico di caricare il modulo specificato da *modulename*. Il nome del modulo deve essere racchiuso tra doppi apici ed espresso in forma standard: non deve essere specificato il nome del file, ma il nome del modulo (case sensitive) senza l'inclusione del prefisso `'lib'` né dei suffissi `'a'`, `'o'` o `'so'`.

I moduli da caricare possono essere specificati anche mediante un'altro formato: creando una sezione **SubSection** che ha il nome del modulo da caricare, contenente le opzioni da passare al modulo quando questo viene caricato. Di seguito è riportato un esempio di definizione del modulo `"extmod"`

```
SubSection "extmod"
    Option "omit XFree86-DGA"
EndSubSection
```

I moduli vengono ricercati nelle directory specificate con la direttiva **ModulePath**. Inoltre vengono considerate le seguenti directory

```
/usr/X11R6/lib/modules/fonts
/usr/X11R6/lib/modules/extensions
```

La sezione **InputDevice** è utilizzata per specificare le impostazioni relative ad ogni dispositivo di input. In genere il file di configurazione del server grafico contiene almeno due sezioni **InputDevice**: una per la tastiera ed una per il dispositivo di puntamento (mouse). Una sezione **InputDevice** è considerata attiva se viene fatto riferimento ad essa in una sezione **ServerLayout** o tramite la riga di comando con le opzioni `-keyboard` e `-pointer`. Le direttive che possono essere contenute in tale sezione sono le seguenti

Identifier "name"

specifica il nome univoco che identifica il dispositivo di input;

Driver "driver_name"

specifica il nome del driver da utilizzare per il dispositivo di input. Il driver più utilizzato per la tastiera è `keyboard` e per il mouse è `mouse`;

Option "CorePointer"

indica che il dispositivo considerato è il dispositivo di puntamento primario (ce ne può essere soltanto uno così definito – se non esplicitamente definito, viene considerato primario il primo dispositivo di puntamento definito nel file di configurazione);

Option "CoreKeyboard"

indica che il dispositivo considerato è la tastiera primaria (ce ne può essere soltanto uno così definito – se non esplicitamente definito, viene considerata primaria la prima tastiera definita nel file di configurazione);

Option "AlwaysCore" "value" | Option "SendCoreEvents" "value"

indica al server grafico, dipendentemente dal valore di *value*, di abilitare (`On`) o meno (`Off`) l'invio, da parte del dispositivo, di eventi primari (default: `Off`);

Option "HistorySize" "value"

specifica la dimensione della storia dei movimenti del dispositivo (default: `0`);

Option "SendDragEvents" "value"

???

La sezione **Device** è utilizzata per specificare le impostazioni relative ad ogni scheda grafica utilizzata. Le direttive che possono essere contenute in tale sezione sono le seguenti

Identifier "name"

specifica il nome univoco che identifica la scheda grafica;

Driver "*driver.name*"

specifica il nome del driver da utilizzare per la scheda grafica;

BusID "*value*"

specifica il bus sul quale si trova la scheda grafica. Per il bus PCI/AGP, la sintassi di *value* è la seguente

"PCI:bus:device:function"

dove

bus indica il numero del bus;

device indica il numero della scheda grafica;

function

indica ???;

Tali impostazioni (facoltative per un sistema con una sola scheda grafica) possono essere ricavate lanciando il server grafico con l'opzione `-scanpci`;

Screen *value*

specifica lo schermo sul quale deve essere inviato l'output della scheda grafica. Questa opzione è necessaria in caso di schede grafiche multiheaded (cioè con più uscite) che possono pilotare più monitor (*value* inizia da 0);

Chipset "*value*"

specifica il chipset presente sulla scheda grafica. In genere questa direttiva non viene in genere utilizzata poiché i driver riconoscono automaticamente il chipset;

Ramdac "*value*"

specifica il tipo di RAMDAC (*Random Access Memory Digital Analog Converter*) presente sulla scheda grafica. Tale direttiva non viene in genere utilizzata poiché i driver riconoscono automaticamente la RAMDAC;

DacSpeed *value* [...]

specifica la velocità della RAMDAC in MHz. Se viene fornito un solo valore *value*, esso viene applicato a tutti i framebuffer, altrimenti il primo valore viene associato al framebuffer relativo alla profondità di colore di 8 bit, il secondo a quello a 16 bit, il terzo a 24 bit ed il quarto a 32 bit. Tale direttiva non viene in genere utilizzata poiché i driver riconoscono automaticamente la velocità della RAMDAC;

Clocks *value* [...]

specifica il refresh della scheda grafica in MHz. Tale direttiva non viene in genere utilizzata poiché i driver riconoscono automaticamente il refresh ottimale della scheda grafica;

ClockChip "*value*"

specifica il tipo di clock presente sulla scheda grafica. Tale direttiva non viene in genere utilizzata poiché i driver riconoscono automaticamente il tipo di clock della scheda grafica;

VideoRam *value*

specifica la quantità di RAM (in KiB) presente sulla scheda grafica. Tale direttiva non viene in genere utilizzata poiché il server grafico riconosce automaticamente la quantità di memoria presente sulla scheda grafica;

BiosBase *value*

specifica l'indirizzo di base del BIOS della scheda grafica (VGA). Tale direttiva non viene in genere utilizzata poiché il server grafico riconosce automaticamente l'indirizzo di base del BIOS;

MemBase *value*

specifica l'indirizzo di base della memoria della scheda grafica (VGA). Tale direttiva non viene in genere utilizzata poiché il server grafico riconosce automaticamente l'indirizzo di base della memoria;

IOBase *value*

specifica l'indirizzo di base dell'I/O della scheda grafica. Tale direttiva non viene in genere utilizzata poiché il server grafico riconosce automaticamente l'indirizzo di base dell'I/O;

ChipID *value*

specifica il numero che identifica la scheda grafica nel sistema (per le schede PCI coincide generalmente con il device ID). Tale direttiva non viene in genere utilizzata poiché il server grafico riconosce automaticamente il ChipID;

ChipRev *value*

specifica il numero di revisione del chip della scheda grafica. Tale direttiva non viene in genere utilizzata poiché il server grafico riconosce automaticamente il ChipID;

TextClockFreq *value*

specifica la frequenza di refresh (in MHz) utilizzata in modalità testo;

Options "*value*"

specifica opzioni specifiche per il driver o indipendenti da esso;

La sezione **Monitor** è utilizzata per specificare le impostazioni relative al monitor. Le direttive che possono essere contenute in tale sezione sono le seguenti

Identifier "*name*"

specifica il nome univoco che identifica il monitor;

VendorName "*value*"

specifica il costruttore del monitor;

ModelName "*value*"

specifica il modello del monitor;

HorizSync "*value*"

specifica l'intervallo di frequenze di refresh orizzontale supportate dal monitor (*value* è un elenco di valori singoli – separati dal carattere ‘,’ – o intervalli – valori singoli separati dal carattere ‘-’). Se non specificato le frequenze sono intese in kHz. Se questa direttiva non viene specificata viene considerato l'intervallo 28-33 kHz;

VertRefresh "*value*"

specifica l'intervallo di frequenze di refresh verticale supportate dal monitor (*value* è un elenco di valori singoli – separati dal carattere ‘,’ – o intervalli – valori singoli separati dal carattere ‘-’). Se non specificato le frequenze sono intese in Hz. Se questa direttiva non viene specificata viene considerato l'intervallo 43-72 Hz;

DisplaySize *width height*

specifica la larghezza (*width*) e l'altezza (*height*) (in mm) dell'area visibile dello schermo. Questi valori vengono utilizzati per calcolare il valore del dpi;

Gamma *value ...*

specifica la correzione di gamma per il monitor. Può essere specificata come singolo valore o tre valori distinti che si riferiscono rispettivamente alle componenti rossa, verde e blu (i valori vanno da 0.1 a 10.0 – il default è 1.0);

UseModes "*modesection*"

indica di utilizzare le impostazioni definite nella sezione **Modes** identificata dal nome *modesection*;

Mode "name"

definisce una sottosezione in cui sono indicate alcune particolari impostazioni che generalmente non sono necessarie, poiché quelle previste dallo standard VESA sono sufficienti. La sottosezione ha la seguente sintassi

```
Mode "name"
    ...
EndMode
```

Le direttive che possono essere contenute in tale sezione sono le seguenti

DotClock value

specifica la frequenza di clock per i pixel (in MHz);

HTimings hdisp hsyncstart hsyncend htotal

specifica i tempi di gestione della scansione orizzontale dei pixel;

VTimings vdisp vsyncstart vsyncend vtotal

specifica i tempi di gestione della scansione verticale dei pixel;

Flags "value"..."

specifica flag specifici, come riportato in tab. 11.3;

Value	Descrizione
Interlace	indica che il sistema di visualizzazione è interlacciato.
DoubleScan	indica che ogni riga da visualizzare sullo schermo deve essere visualizzata da due righe di pixel.
+HSync	indica di utilizzare la polarità positiva del segnale di sincronizzazione orizzontale.
-HSync	indica di utilizzare la polarità negativa del segnale di sincronizzazione orizzontale.
+VSync	indica di utilizzare la polarità positiva del segnale di sincronizzazione verticale.
-VSync	indica di utilizzare la polarità negativa del segnale di sincronizzazione verticale.
Composite	indica che il segnale di sincronizzazione è di tipo composito.
+CSync	indica di utilizzare la polarità positiva del segnale di sincronizzazione composito.
-CSync	indica di utilizzare la polarità negativa del segnale di sincronizzazione composito.

Tabella 11.3: Possibili valori per la direttiva **Flags**.

HSkew value

specifica il numero di pixel di cui spostare l'immagine verso il margine destro dello schermo;

VScan value

specifica il numero di volte che ogni riga dello schermo deve essere ridisegnata (default: 1);

ModeLine "name" value ...

specifica le opzioni analogamente a quanto avviene con la sottosezione **Mode**, ma lo fa attraverso una sola riga. I valori specificano, nell'ordine: la frequenza di clock del pixel (in MHz), le temporizzazioni orizzontali (*hdisp*, *hsyncstart*, *hsyncend* e *htotal*), le temporizzazioni verticali (*vdisp*, *vsyncstart*, *vsyncend* e *vtotal*) ed un eventuale elenco di flag (v. tab. 11.3);

Options "value"

specifica ulteriori opzioni come **DPMS**, **SyncOnGreen**, ...;

La sezione **Modes** è utilizzata per specificare le impostazioni relative alle modalità video (risoluzioni) indipendentemente dallo schermo. La sezione **Monitor** può fare riferimento ad una sezione **Modes** per mezzo della direttiva **UseModes**. Le direttive che possono essere contenute in tale sezione sono le seguenti

Identifier "name"
 specifica il nome univoco della sezione;

Mode "name"
 v. Mode della sezione **Monitor**;

ModeLine "name" value ...
 v. ModeLine della sezione **Monitor**;

La sezione **Screen** è utilizzata per specificare le impostazioni relative allo schermo, ovvero quello che collega la scheda grafica (sezione **Device**) al monitor (sezione **Monitor**). Una sezione **Screen** è considerata attiva se è fatto riferimento ad essa nella sezione **ServerLayout** o con l'opzione **-screen** sulla riga di comando. Se nessuno dei riferimenti è stato indicato, viene considerata attiva la prima sezione **Screen** presente nel file di configurazione. Le direttive che possono essere contenute in tale sezione sono le seguenti

Identifier "name"
 specifica il nome univoco della sezione;

Device "name"
 specifica il nome della sezione **Device** contenente le impostazioni della scheda grafica da considerare;

Monitor "name"
 specifica il nome della sezione **Monitor** contenente le impostazioni del monitor da considerare;

VideoAdaptator "value"
 specifica una descrizione relativa all'adattatore video;

DefaultDepth value
 specifica la profondità di colore di default (in bit). Il default dipende dal driver, ma in molti casi è 8;

DefaultFbBpp value
 specifica il layout del framebuffer da utilizzare per default (in genere questa direttiva viene utilizzata soltanto con profondità di colore di 24 bit – 24 bit packed o 32 sparse bit);

Options "value"
 indica eventuali opzioni specifiche;

Option "Accel"
 abilita la XAA (*X Acceleration Architecture*) ovvero un meccanismo che permette al server grafico di sfruttare l'accelerazione hardware 2D presente sulla scheda grafica (per default questa direttiva è attivata);

Option "NoMTRR"
 disabilita il supporto per il MTRR (*Memory Type Range Register*), una caratteristica dei microprocessori moderni che permette di migliorare le performance video di un fattore 2,5;

Option "XaaNoCPUToScreenColorExpandFill"
 disabilita l'accelerazione video per l'espansione rettangolare dei pattern in memoria;

Option "XaaNoColor8x8PatternFillRect"
 disabilita l'accelerazione video per il riempimento colorato delle regioni rettangolari;

Option "XaaNoColor8x8PatternFillTrap"
 disabilita l'accelerazione video per il riempimento colorato delle regioni trapezoidali;

Option "XaaNoDashedBresenhamLine"
 disabilita l'accelerazione video per la visualizzazione delle linee tratteggiate con il metodo Bresenham;

- Option "XaaNoDashedTwoPointLine"**
disabilita l'accelerazione video per la visualizzazione delle linee tratteggiate tra due punti arbitrari;
- Option "XaaNoImageWriteRect"**
disabilita l'accelerazione video per il trasferimento di pattern rettangolari dalla memoria di sistema alla memoria video;
- Option "XaaNoMono8x8PatternFillRect"**
disabilita l'accelerazione video per il riempimento monocromatico delle regioni rettangolari;
- Option "XaaNoMono8x8PatternFillTrap"**
disabilita l'accelerazione video per il riempimento monocromatico delle regioni trapezoidali;
- Option "XaaNoOffscreenPixmap"**
disabilita l'accelerazione video per la visualizzazione delle pixmap memorizzate nella memoria video in offscreen;
- Option "XaaNoPixmapCache"**
disabilita il caching per la visualizzazione dei pattern memorizzati nella memoria video in offscreen;
- Option "XaaNoScanlineCPUToScreenColorExpandFill"**
disabilita l'accelerazione video per l'espansione dei pattern rettangolari a partire da quelli memorizzati nella memoria di sistema (una riga per volta);
- Option "XaaNoScanlineImageWriteRect"**
disabilita l'accelerazione video per il trasferimento di pattern colorati rettangolari dalla memoria di sistema alla memoria video (una riga per volta);
- Option "XaaNoScreenToScreenColorExpandFill"**
disabilita l'accelerazione video per l'espansione rettangolare dei pattern memorizzati nella memoria video in offscreen;
- Option "XaaNoScreenToScreenCopy"**
disabilita l'accelerazione video per la copia delle regioni rettangolari da una zona all'altra della memoria video;
- Option "XaaNoSolidBresenhamLine"**
disabilita l'accelerazione video per la visualizzazione delle linee continue con il metodo Bresenham;
- Option "XaaNoSolidFillRect"**
disabilita l'accelerazione video per la visualizzazione di rettangoli colorati;
- Option "XaaNoSolidFillTrap"**
disabilita l'accelerazione video per la visualizzazione di aree colorate trapezoidali;
- Option "XaaNoSolidHorVertLine"**
disabilita l'accelerazione video per la visualizzazione di linee continue orizzontali e verticali;
- Option "XaaNoSolidTwoPointLine"**
disabilita l'accelerazione video per la visualizzazione di linee continue tra due punti arbitrari;

La sezione **Screen** può contenere una o più sottosezioni **Display** con la seguente sintassi

```
SubSection "Display"
...
EndSubSection
```

Le direttive che possono essere contenute in tale sottosezione sono le seguenti

- Depth** *value*
indica la profondità di colore (quanti bit sono necessari per rappresentare un pixel);
- FbBpp** *value*
indica il formato del framebuffer (ovvero quanti bit sono utilizzati per rappresentare un pixel);
- Weight** *red_value green_value blue_value*
specifica il “peso” da assegnare ad ogni colore di base;
- Virtual** *x_dim y_dim*
specifica la risoluzione virtuale dello schermo (la dimensione lungo l'asse *x* deve essere in genere un multiplo di 8 o 16). Le risoluzioni video troppo grandi rispetto a quella qui specificata vengono scartate. Se questa direttiva non viene indicata, la risoluzione virtuale viene calcolata dalle direttive specificate nella direttiva **Modes**;
- Viewport** *x0 y0*
specifica la posizione del vertice in alto a sinistra del display. Questa direttiva viene utilizzata quando la risoluzione dello schermo è diversa da quella virtuale;
- Modes** "*value*" "*...*"
specifica l'elenco di risoluzioni che possono essere utilizzate. È possibile passare da una risoluzione all'altra con le combinazioni di tasti Ctrl Alt keypad + e Ctrl Alt keypad -. Se questa direttiva non è specificata, viene utilizzata la risoluzione più grande tra quelle specificate nella sezione **Monitor**;
- Visual** "*value*"
specifica la visualizzazione di default (v. tab. 11.4). Se non specificato viene utilizzato **StaticGray** per profondità di colore a 1 bit, **StaticColor** per profondità di colore a 4 bit, **PseudoColor** per profondità di colore a 8 bit, **TrueColor** per profondità di colore a 15, 16 e 24 bit;

Value	Depth	Descrizione
StaticGray	1, 4, 8	???
GrayScale	4, 8	???
StaticColor	4, 8	???
PseudoColor	4, 8	???
TrueColor	8, 15, 16, 24	???
DirectColor	8, 15, 16, 24	???

Tabella 11.4: Possibili valori per la direttiva **Visual**.

- Black** *red green blue*
specifica l'intensità di rosso, verde e blu per rappresentare il colore nero;
- White** *red green blue*
specifica l'intensità di rosso, verde e blu per rappresentare il colore bianco;
- Options** "*value*"
indica ulteriori opzioni (v. **Screen**);

La sezione **ServerLayout** è utilizzata per collegare le impostazioni relative ai vari dispositivi: essa collega una o più sezioni **Screen** ed una o più sezioni **InputDevice**. Le direttive che possono essere contenute in tale sezione sono le seguenti

- Identifier** "*name*"
specifica il nome univoco della sezione;
- Screen** [*number*] "*name*" [*position*]
specifica i riferimenti dello schermo. Il valore *number* è il valore di screen da considerare quando è specificata una scheda video multiheaded (se non è specificato, gli screen sono numerati partendo da 0 dal primo presente nel file di

configurazione). Il valore *name* specifica il nome della sezione **Screen** a cui si fa riferimento. Il valore *position* indica come vanno gestiti gli screen che compongono la stazione grafica. Questi possono essere specificati con una delle seguenti sintassi

Absolute *x y*
specifica le coordinate (*x,y*) del vertice in alto a sinistra dello schermo (se non vengono specificato è sott'inteso (0,0));

RightOf "*screen_name*"
indica che lo screen considerato è il prolungamento verso destra dello screen *screen_name*;

LeftOf "*screen_name*"
indica che lo screen considerato è il prolungamento verso sinistra dello screen *screen_name*;

Above "*screen_name*"
indica che lo screen considerato è il prolungamento verso l'alto dello screen *screen_name*;

Below "*screen_name*"
indica che lo screen considerato è il prolungamento verso il basso dello screen *screen_name*;

Relative "*screen_name*"*x y*
indica la posizione del vertice in alto a sinistra dello screen considerato relativamente allo screen *screen_name*;

InputDevice "*name*" [*"option"*] [...]
specifica i riferimenti ai dispositivi di input. Il valore *name* specifica il nome della sezione **InputDevice** a cui si fa riferimento. Il parametro *option* può assumere i valori specificati nella sezione **InputDevice** (quelli più utilizzati sono **CorePointer**, **CoreKeyboard** e **SendCoreEvents**);

Option "*value*"
indica opzioni specifiche (v. sezione **ServerFlags**);

Di seguito è riportato un esempio di una sezione **ServerLayout**

```
Section "ServerLayout"
    Identifier "Layout 1"
    Screen "MGA 1"
    Screen "MGA 2" RightOf "MGA 1"
    InputDevice "Keyboard 1" "CoreKeyboard"
    InputDevice "Mouse 1" "CorePointer"
    InputDevice "Mouse 2" "SendCoreEvents"
    Option "BlankTime" "5"
EndSection
```

La sezione **DRI** è utilizzata per specificare le impostazioni relative alla DRI (*Direct Rendering Infrastructure*) (v. <http://www.xfree86.org/current/DRI.html>).

La sezione **Vendor** è utilizzata per specificare le impostazioni relative al fornitore.
???

Vista la complessità di **XFree86**, sono stati introdotti altri comandi che ne facilitano l'utilizzo, come **xinit** (*X Window Initializer* – man page **xinit(1)**) e **startx** (man page **startx(1)**).

xinit viene generalmente utilizzato per lanciare in esecuzione il server grafico ed un client grafico che lo utilizza. Quando il client grafico termina la sua esecuzione, **xinit** invia un segnale di interruzione al server grafico, che a sua volta termina la propria esecuzione.

Comando: **xinit**
Path: ???/**xinit**

SINTASSI

```
# xinit [[client] client_option] [-- [server] [display] server_option]
```

DESCRIZIONE

client indica il client grafico da avviare. L'indicazione del nome del client grafico deve iniziare con il carattere '/' o '.'. Se non viene specificato nessun client grafico, **xinit** lancia in esecuzione, se esiste, lo script `~/xinitrc`. È opportuno sottolineare che il comando che lancia il server grafico, contenuto nello script, deve essere lanciato in background, in modo tale da permettere l'esecuzione di altri comandi. Il comando invece che si riferisce solitamente al window manager (v. sez. ??) non deve essere lanciato in background, altrimenti lo script termina e quindi **xinit** termina anche l'esecuzione del server grafico. Se il file `~/xinitrc` non esiste, **xinit** lancia **xterm** come di seguito riportato

```
xterm -geometry +1+1 -n login -display :0
```

Un esempio di file `~/xinitrc` che avvia un orologio, vari terminali e lascia in esecuzione il window manager **twm** è di seguito riportato

```
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
xclock -g 50x50-0+0 -bw 0 &
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
twm
```

client_option indica le eventuali opzioni da passare al client grafico;

server indica il server grafico da avviare. L'indicazione del nome del server grafico deve iniziare con il carattere '/' o '.'. Se non viene specificato nessun server grafico, **xinit** lancia in esecuzione, se esiste, lo script `~/xserverrc`. Se tale file non esiste, **xinit** lancia **X** come di seguito riportato

```
X :0
```

display è l'indicazione del display al quale associare il server grafico, con la notazione *host:display*;

server_option indica le eventuali opzioni da passare al server grafico;

???

??? man page startx ???

11.3 X Display Manager

L'accesso al sistema GNU/Linux, da parte di un utente, può essere effettuato anche tramite l'interfaccia grafica. Quando il sistema viene avviato direttamente in modalità grafica (runlevel 5), viene lanciata da **init** l'applicazione che permette di effettuare il login (accesso al sistema) da interfaccia grafica: **xdm** (**X display manager** – man page **xdm(?)**). Il processo **xdm** si prenderà cura di visualizzare una finestra in cui l'utente può digitare il proprio username e successivamente la propria password per effettuare il login.

xdm inoltre comunica con gli X server, tramite il protocollo XDMCP (X Display Manager Control Protocol).

Comando: **xdm**

Path: `/etc/X11/xdm`

SINTASSI

xm option

DESCRIZIONE

option indica la modalità di funzionamento di *xm*. Può assumere i seguenti valori:

```

-config config.file
    specifica il file di configurazione, secondo quanto indicato da con-
fig.file (il default è /usr/X11R6/lib/X11/xm/xm-config);
-nodaemon
    indica di non lanciare in esecuzione xm come daemon;
-debug level
    specifica il livello di verbosità di debug, secondo quanto indicato da
level (più tale valore è elevato, più messaggi vengono visualizzati);
-error error.logfile
    specifica il file in cui scrivere eventuali messaggi di errore, secondo
    quanto specificato da error.logfile;
-resources resource.file
    specifica il file in cui sono impostati i parametri di configurazio-
    ne per la finestra di autenticazione, secondo quanto specificato da
resource.file;
-server server.entry
    specifica ???;
--udpPort port.number
    specifica il numero della porta (port.number) sulla quale xm deve
    rimanere in ascolto per la gestione dei messaggi XDMCP (per default
    XDMCP utilizza la porta UDP 177 – well-known port number);
--session session.program
    specifica il programma da far eseguire in sessione subito dopo che
    l'utente ha effettuato l'accesso;
-xrm resource
    specifica ???;

? ;

```

xm viene generalmente lanciato in esecuzione tramite un'apposita direttiva contenuta nel file */etc/inittab*, come quella seguente

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Il file */etc/X11/prefdm* è uno script che contiene le istruzioni per l'avvio del display manager desiderato. Ogni desktop environment in genere ha un proprio display manager che rimpiazza *xm*. Ad esempio KDE e Gnome forniscono i relativi display manager denominati rispettivamente *kdm* (KDE display manager) e *gdm* (Gnome display manager).

Il comportamento di *xm* risente delle direttive contenute nei file di configurazione relativi, di seguito riportati:

```

/usr/X11R6/lib/X11/xm/xm-config
    file di configurazione di default;

~/.Xauthority
    file in cui xm memorizza le chiavi per i client;

/usr/X11R6/lib/X11/xm/chooser
    file ??? di default;

/usr/X11R6/lib/X11/xrdb
    resource database di default;

Xaccess
;

```

```
Xservers
    ;
Xresources
    ;
    ???
```

11.4 Window manager

window
window manager
frame

Il server grafico non permette agli utenti di gestire il posizionamento, lo spostamento, il ridimensionamento della parte di schermo relativa alle varie applicazioni (**window** o finestra). A tale scopo è destinato un client grafico particolare: il **window manager**. Uno *window manager* aggiunge infatti alle *window* una cornice o **frame** che presenta in genere anche una zona in alto nella quale viene riportato il nome dell'applicazione.

Con l'uso del mouse, tramite operazioni piuttosto intuitive, è possibile interagire con i *frame* per spostare, ridimensionare e chiudere la finestra relativa, nella quale viene eseguita l'applicazione.

Gli window manager sono svariati, ed ognuno presenta delle proprie caratteristiche sia in relazione all'interazione dell'utente (gli shortcut da tastiera, il significato attribuito ai pulsanti del mouse, ...) che con quella del sistema (memoria occupata, integrazione con il *desktop manager*, ...). Questo permette agli utenti di poter scegliere il window manager che meglio si adatta alle proprie esigenze.

11.4.1 FVWM

11.4.2 IceWM

11.4.3 Window Maker

11.4.4 After Step

11.4.5 Enlightenment

11.4.6 Metacity

11.4.7 Sawfish

11.5 Desktop environment

11.5.1 GNOME

*Gnome*⁴ (GNU Network Object Model Environment) è un desktop manager, piuttosto che un desktop environment completo, che significa che per funzionare ha bisogno anche di uno window manager.

???

11.5.2 KDE

*KDE*⁵ (K Desktop Environment)⁶ è un desktop environment, ovvero un'ambiente grafico completo, molto user friendly.

???

temi dei desktop
konqueror e nautilus.

???

⁴v. <http://www.gnome.org>.

⁵v. <http://www.kde.org>.

⁶il nome deriva da un precedente ambiente grafico *CDE* (Command Desktop Environment).

11.6 Gestione dei font

In modalità grafica è possibile utilizzare più stili per la scrittura del testo, ovvero è possibile scegliere quello che in gergo viene chiamato il **font** (fonte) dei caratteri. Un *font* è appunto uno stile di scrittura: ad esempio ... (v. fig. 11.5). Inoltre è possibile scegliere la dimensione del font da visualizzare sullo schermo.

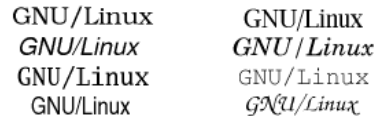


Figura 11.5: Alcuni esempi di font.

Ogni font definisce il simbolo grafico che deve essere visualizzato in corrispondenza di ogni tasto, o combinazione di tasti, premuto sulla tastiera. Tale simbolo viene detto **glifo** (*glyph*).

Esistono inoltre varie tipologie di font, tra le quali, quelle realizzate con tecnologia *TrueType* (???) permettono la migliore resa grafica. Tali file, che generalmente terminano con estensione **.ttf**, in sostanza, non contengono i disegni dei vari glifi, ma le istruzioni per disegnarli. In questo modo le font avranno la stessa definizione con qualunque dimensione viene selezionata.

La gestione dei font in GNU/Linux non è di semplice utilizzo, anche perché la stessa si è evoluta nel tempo, con le varie versioni del server grafico (X window). Esistono quindi applicativi che fanno riferimento a versioni diverse di gestione dei font.

11.6.1 Fontconfig

Il sistema di gestione dei font più recente, contenuto in XFree86 vers. 4.2 è denominato *Fontconfig* ed è destinato a divenire quello di riferimento. Questo è il sistema utilizzato dalle applicazioni sviluppate con i toolkit grafici messi a disposizione dalle librerie Qt vers. 3 e GTK+ vers. 2.

Tale sistema permette di accedere direttamente ai font presenti sul sistema ed utilizzare librerie come Xft per ottenere particolari effetti di visualizzazione. Un esempio è il rendering dei font con **anti-aliasing**, che consiste nella visualizzazione di una sfumatura (*smoothing*) dei contorni dei glifi in maniera da dare l'idea all'utente che la rappresentazione del carattere non abbia scalettature, anche se necessariamente le avrà (v. fig. 11.6).

Fontconfig non funziona con OpenOffice.org ed altre applicazioni che usano una propria tecnologia di rendering dei font.

Il file di configurazione di *Fontconfig* è **/etc/fonts/fonts.conf**.

I file relativi ai font di sistema sono contenuti nella directory **/usr/share/fonts/local**, mentre quelli relativi ai font personali degli utenti sono contenuti nella directory **~/.fonts**.

Nel caso in cui si vogliano aggiungere dei font, è necessario copiare i relativi file nell'opportuna directory e quindi aggiornare la cache delle informazioni relative ai font con il comando **fc-cache**. Ad esempio, se i file sono stati copiati nella directory **~/.fonts**, si deve impartire il comando

```
$ fc-cache ~/.fonts
```

I font relativi agli utenti possono essere anche installati da interfaccia grafica, copiando i relativi file all'URI **fonts:///** di Nautilus.

???

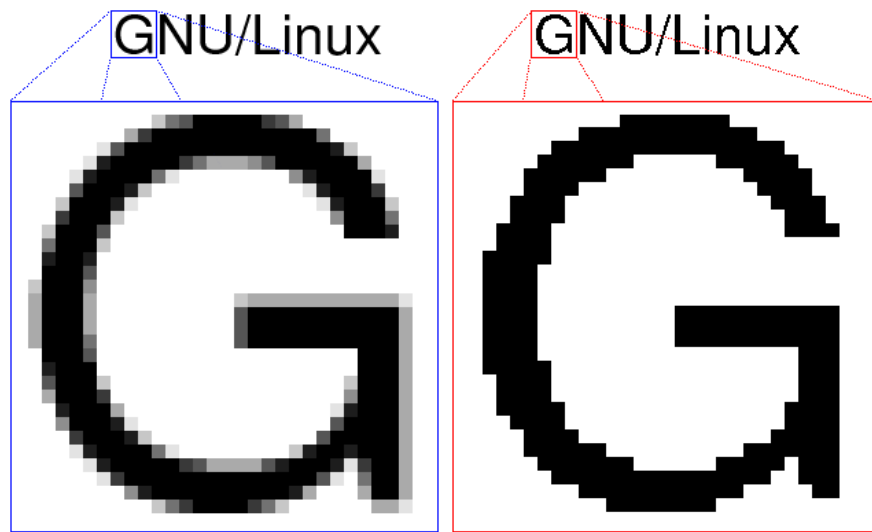


Figura 11.6: Visualizzazione dei font con filtro anti-aliasing (testo a sinistra) e senza (testo a destra).

11.6.2 X Font server

Questo è il sistema di gestione dei font ideato più di 15 anni fa e gestito da *xfs*. Tale sistema è in genere fornito con le varie distribuzioni per compatibilità con le applicazioni che ancora lo utilizzano.

L'X server controlla nella sezione *Files* del file di configurazione */etc/X11/XF86Config*, le directory che contengono i file per i font, specificate dalle direttive *FontPath*. Quindi si connette al server *xfs* (su una specifica porta) per la gestione dei font. Quindi per far avviare l'interfaccia grafica è necessario che sia avviato *xfs*, attraverso lo script */etc/rc.d/init.d/xfs* che legge le impostazioni contenute nel file di configurazione */etc/X11/fs/config*.

```
Section "Files"
    FontPath      "/usr/local/share/fonts/ttfonts"
    FontPath      "/usr/X11R6/lib/X11/fonts/misc:unscaled"
    FontPath      "/usr/X11R6/lib/X11/fonts/100dpi:unscaled"
    FontPath      "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
    FontPath      "/usr/X11R6/lib/X11/fonts/Type1"
    FontPath      "/usr/X11R6/lib/X11/fonts/Speedo"
    FontPath      "/usr/X11R6/lib/X11/fonts/misc"
    FontPath      "/usr/X11R6/lib/X11/fonts/100dpi"
    FontPath      "/usr/X11R6/lib/X11/fonts/75dpi"
End Section
```

Per aggiungere un font a *xfs* è necessario copiare i relativi file in una directory (ad es. */usr/share/fonts/local/myfonts*). Quindi bisogna aggiungere tale directory al percorso di ricerca dei file relativi ai font, con il comando *chkfontpath*, come riportato di seguito

```
$ chkfontpath --add /usr/share/fonts/local/myfonts
```

Quindi devono essere aggiornate le informazioni dei font con il comando *ttmkfdir*, come riportato di seguito

```
% ttmkfdir -d /usr/share/fonts/local/myfonts -o /usr/share/fonts/local/myfonts/fonts.scale
```

Quindi, una volta riavviato *xfs* con il seguente comando

```
% service xfs reload
```

sarà possibile utilizzare i nuovi font.

???

???

11.7 Riferimenti

- *X Consortium*
<http://www.x.org>
- K. Taylor, *XDM and X Terminal mini-HOWTO*
<http://www.tldp.org/HOWTO/XDM-Xterm/index.html>
- VA Linux Systems, Inc. *DRI User Guide*
<http://www.xfree86.org/current/DRI.html>

???

Capitolo 12

Cenni sui database

“Di solito, si ottiene con tutta sicurezza ed assai presto ciò che non si ha fretta di ottenere.”

– J. J. Rousseau

???

12.1 Introduzione

La gestione di consistenti quantità di informazioni, come l’archiviazione ed il recupero, può essere effettuata per mezzo di opportuni applicativi che utilizzano il filesystem per l’archiviazione dei dati. In particolare sono state sviluppate delle applicazioni in grado di effettuare un trattamento molto efficace delle informazioni che si desidera gestire: i **database**, o più semplicemente DB. Questi applicativi sono in grado di gestire enormi quantità di informazioni mettendo a disposizione dell’utente un’interfaccia agevole e più o meno standard.

database

Sarà fatta, inoltre, una panoramica sull’utilizzo di due dei database generalmente forniti insieme con le distribuzioni di GNU/Linux, quali *PostgreSQL* e *MySQL*.

???

12.2 Tipi di database

I database sono sistemi per la gestione di grosse quantità di informazioni che offrono prestazioni elevate nella ricerca delle informazioni stesse. Alla base di un database ci sta quello che viene definito il DBMS (DataBase Management System), ovvero il “motore” del database che è in grado di gestire a basso livello le operazioni da effettuare sulle informazioni (memorizzazione, aggiornamento, cancellazione e ricerca).

La forma più semplice di database è costituita da un file contenente tante righe quante sono le informazioni che devono essere memorizzate nel database. Questo tipo di archiviazione dei dati ha notevoli limitazioni: tempi di ricerca elevati, problemi di ridondanza delle informazioni memorizzate, problemi nella manutenzione delle informazioni, ... Il vantaggio è che si basa su una struttura semplicissima e relativamente facile da gestire. Questo tipo di database viene utilizzato generalmente quando la quantità di informazioni da memorizzare è molto piccola.

Da questo tipo di database si sono sviluppati i database **gerarchici**, che utilizzano più file per memorizzare le informazioni, introducendo una relazione gerarchica di tipo padre-figlio (ad albero) fra gli stessi. Un database di questo tipo è IMS (Information Management System) di *IBM* (spesso è riferito con il nome del linguaggio da questo utilizzato, DL/I – Data Language I). Oggigiorno questo tipo di database non viene più utilizzato.

gerarchici

I database **reticolari** (network) espandono il concetto di relazione gerarchica, rendendo possibile il raggiungimento di un nodo da più percorsi. Tale tipologia di database

reticolari

viene anche riferita con il nome CODASYL DBTG (Conference on Data System Languages, Data Base Task Group). Nonostante il fatto che sia possibile definire relazioni multiple tra le varie informazioni, la gestione di tali tipi di database diviene sempre più crescente all'aumentare del numero di relazioni, tant'è che oggi questo tipo di database non viene più utilizzato.

relazionale

Dal modello a rete è nato il database **relazionale** o **RDB** (Relational DataBase). In questo tipo di database, la struttura logica è indipendente da quella fisica e fornisce sia elevate prestazioni che flessibilità di gestione, sia dei dati che della struttura del database stesso.

Il database a oggetti (object-oriented) risolve alcune limitazioni del modello relazionale, in particolare la scarsa capacità di gestione dei BLOB (Binary Large Object) cioè dei tipi di dati complessi come le immagini, documenti, ... Questo è dovuto al fatto che i database sono nati per gestire dati (sequenze di 0 e di 1) con dimensioni relativamente piccole. I BLOB sono oggetti che possono avere dimensioni decisamente più grosse rispetto a quelle per le quali sono nati i database. I BLOB vengono generalmente memorizzati all'esterno del DB e sul DB viene memorizzato soltanto un riferimento ad essi (ad esempio il loro path sul filesystem). I database ad oggetti gestiscono nativamente i BLOB, ma ogni database utilizza un proprio modo di gestione dei BLOB. Le informazioni vengono memorizzate in oggetti che hanno relazioni di tipo gerarchico. Questi tipi di database non sono molto diffusi: vengono utilizzati soltanto in ambienti CAD ed engineering.

12.3 Entità e relazioni

entità

Quando si decide di utilizzare un database per la gestione di informazioni, la prima cosa da individuare sono le **entità**, ovvero gli “oggetti” che costituiscono gli elementi fondamentali dell'organizzazione delle informazioni. Ogni entità sarà caratterizzata da proprietà, dette **campi**. Possono quindi essere memorizzate, all'interno del DB, varie occorrenze di entità dello stesso tipo o di tipi diversi, ognuna con i propri valori caratteristici.

campi

Ad esempio, si supponga di voler gestire, per mezzo di un database, una biblioteca. A tale scopo si possono individuare le seguenti entità (quello riportato è solo un esempio, possono essere individuate ulteriori entità e/o campi):

libro

entità con i seguenti campi

codice

codice che identifica univocamente un libro all'interno della biblioteca;

titolo

titolo del libro;

nome autore

nome dell'autore del libro;

cognome autore

cognome dell'autore del libro;

editore

editore del libro;

data di pubblicazione

data di pubblicazione del libro;

luogo di pubblicazione

luogo di pubblicazione del libro;

numero di pagine

numero di pagine di cui è composto il libro;

codice collocazione

codice che identifica la posizione del libro negli scaffali della biblioteca;

in prestito

indica se il libro è attualmente in prestito a qualcuno;

autore

entità con i seguenti campi

nome

nome dell'autore;

cognome

cognome dell'autore;

data di nascita

data di nascita dell'autore;

luogo di nascita

luogo di nascita dell'autore;

editore

entità con i seguenti campi

nome

nome dell'editore;

indirizzo sede

indirizzo dell'editore;

data di inserimento

data di inserimento dell'editore nell'archivio;

lettore

entità con i seguenti campi

codice

codice che identifica univocamente un lettore (dal punto di vista della biblioteca);

nome

nome del lettore;

cognome

cognome del lettore;

indirizzo

indirizzo del lettore;

numero documento

numero del documento di identità del lettore;

collocazione

entità con i seguenti campi

codice

codice che identifica univocamente uno scaffale all'interno della biblioteca;

reparto

descrizione del reparto della biblioteca;

scaffale

identificazione dello scaffale all'interno del reparto considerato;

prestito

entità con i seguenti campi

codice lettore

codice che identifica univocamente un lettore;

codice libro

codice che identifica univocamente un libro;

data inizio prestito

data dalla quale inizia il prestito;

data fine prestito

data alla quale ha termine il prestito;

prestito concluso

flag che indica se il prestito è concluso (il libro è stato riportato in biblioteca);

A questo punto si è praticamente definita la struttura dell'organizzazione delle informazioni. Le singole entità con i loro valori dei campi sono di seguito riportati.

Esempi di occorrenze dell'entità *libro*

Codice: 127
Titolo: Moby Dick
Nome autore: Hermann
Cognome autore: Melville
Editore: ???
Data pubblicazione: ???
Luogo di pubblicazione: ???
Numero di pagine: ???
Codice collocazione: 134
In prestito: No

Codice: 564
Titolo: Le avventure di Huckleberry Finn
Nome autore: Mark
Cognome autore: Twain
Editore: ???
Data pubblicazione: ???
Luogo di pubblicazione: ???
Numero di pagine: ???
Codice collocazione: 237
In prestito: No

Codice: 845
Titolo: Psicopatologia della vita quotidiana
Nome autore: Sigmund
Cognome autore: Freud
Editore: ???
Data pubblicazione: ???
Luogo di pubblicazione: ???
Numero di pagine: ???
Codice collocazione: 954
In prestito: No

Esempi di occorrenze dell'entità *autore*

Nome: Hermann
Cognome: Melville
Data di nascita: ???
Luogo di nascita: ???

Nome: Mark
Cognome: Twain
Data di nascita: ???
Luogo di nascita: ???

Nome: Sigmund
Cognome: Freud
Data di nascita: ???

Luogo di nascita: ???

???

Le singole occorrenze delle entità costituiscono le informazioni contenute nel database che vengono gestite dal DBMS.

12.4 I RDBMS

I database relazionali, sono i database più diffusi su tutti i sistemi. Tali database memorizzano le informazioni in appositi contenitori detti tabelle tra le quali possono essere definite delle relazioni logiche. Si tenga comunque presente che la rappresentazione dei dati raccolti in tabelle è soltanto una rappresentazione logica, poiché i dati possono essere effettivamente memorizzati dal DBMS in maniera diversa.

???

E. F. Codd, che ha definito il modello relazionale dei database, ha espresso le caratteristiche che un RDBMS deve soddisfare

1. *Information rule*

“Le informazioni sono rappresentate in uno solo modo: come valori di colonne all’interno di righe”

Dunque, i dati sono rappresentati per mezzo di tabelle. Questa regola denota la semplicità e quindi la versatilità dei RDBMS. Questo è il requisito fondamentale del modello relazionale.

2. *Guaranteed access rule*

“Ogni valore può essere acceduto fornendo il nome della tabella, la colonna e la chiave”

Tutte le informazioni sono univocamente identificate ed accessibili tramite quest’insieme di valori.

3. *Systematic treatment of null values*

“Separazione della gestione delle informazioni mancanti da quelle non significative”

4. *Relational online catalog*

“Il *catalog* o *data dictionary* deve essere accessibile con gli strumenti messi a disposizione dal DBMS per l’accesso ai dati”

Il *catalog*, ovvero la struttura del DB, deve essere parte del DB stesso.

5. *Comprehensive data sublanguage*

“Supporto di almeno un linguaggio che include funzionalità per la definizione delle strutture dati (DDL, Data Definition Language), per la gestione delle stesse (DML, Data Manipulation Language), per la gestione della sicurezza, dei vincoli e dell’integrità delle informazioni”

Ad oggi significa: deve supportare SQL (tutte le implementazioni di DBMS oggi implementano SQL come linguaggio di gestione dello stesso).

6. *View updating rule*

“Tutte le *viste* teoricamente possibili devono essere realmente possibili”

Le informazioni possono essere presentate come combinazioni logiche di tabelle, le *viste*. Ogni vista deve supportare le stesse funzionalità delle tabelle.

7. *High-level insert, update and delete*

“Supporto di aggiornamenti a più record con una singola istruzione”

Questo implica che le operazioni di inserimento, aggiornamento e cancellazione dovrebbero essere supportate per ogni insieme di informazioni accessibile del DB.

8. *Physical data independence*

“Dal livello logico del DBMS è possibile effettuare qualunque operazione che è possibile da quello fisico”

Gli utenti ed i programmi che accedono al DB non devono essere dipendenti dalla sua struttura fisica.

9. *Logical data independence*

“Gli utenti ed i programmi sono indipendenti dalla struttura logica del DB”

La struttura logica dei dati, può essere modificata con un impatto minimo sui programmi che accedono al DB.

10. *Integrity independence*

“I vincoli di integrità dei dati devono essere memorizzati nel *catalog*, non nei programmi”

Le modifiche dei vincoli di integrità dei dati non dovrebbero riflettersi sui programmi (questo semplifica la logica dei programmi).

11. *Distribution independence*

“Le applicazioni dovrebbero funzionare anche in un database di tipo distribuito (DDB, Distributed DataBase)”

12. *Nonsubversion rule*

“Se esiste un’interfaccia per la visualizzazione/gestione di un record alla volta, la sicurezza e l’integrità del database non deve essere violata”

Non deve esistere nessuna backdoor che escluda la sicurezza imposta dal DBMS, ovvero non deve essere possibile modificare la struttura del DB se non attraverso il linguaggio messo a disposizione dal DBMS.

Esiste anche una regola zero per i RDBMS

“Un RDBMS deve essere in grado di gestire i database attraverso le proprie capacità di gestire le relazioni, senza appoggiarsi a caratteristiche aggiuntive supportate dal sistema sul quale esso gira”

???

???

12.4.1 Le tabelle

tabelle

colonne

campi

righe

record

Un database è composto essenzialmente da dei contenitori di dati, detti **tabelle**, nei quali possono essere memorizzate le informazioni. Generalmente ogni entità viene rappresentata da una tabella che è suddivisa in **colonne** o **campi** che corrispondono ai **campi** delle entità. Le singole occorrenze delle entità sono le **righe** delle relative tabelle, che in gergo sono dette anche **record**.

I campi possono contenere valori dei tipi di dati gestiti dal DBMS in questione, tra i quali

- valori numerici interi (integer, long, ...)
- valori numerici con parte decimale (float, ...)
- valori alfanumerici (char, varchar, ...)
- date (date, timestamp, ...)

sequenze

contatori

Generalmente i DBMS mettono a disposizione dei meccanismi per generare automaticamente valori numerici progressivi, ovvero quelle che vengono chiamate **sequenze** o **contatori**, utili nel caso di campi che devono assumere un valore univoco. Ad esempio, ogni volta che viene creato un nuovo record in una tabella, la sequenza genera un nuovo

valore, che sarà il successivo di quello precedentemente generato. Questo meccanismo facilita l'inserimento di un valore univoco, per ogni record, all'interno di un determinato campo.

???

Quella riportata in tab. 12.1 è un esempio del contenuto della tabella *libro*, relativa all'entità *libro*.

Codice	Titolo	Nome_autore	Cognome_autore	Editore	Data Pubbl	Luogo Pubbl	N_Pag	Collocaz	P
127	Moby Dick	Herman	Melville	???	???	???	???	134	
564	Le avventure di Huckleberry Finn	Mark	Twain	???	???	???	???	237	
845	Psicopatologia della vita quotidiana	Sigmund	Freud	???	???	???	???	954	

Tabella 12.1: Esempio del contenuto della tabella *libro*.

???

12.4.2 Le chiavi e gli indici

Le occorrenze delle singole entità sono rappresentate da record, ovvero sono ennuple di valori corrispondenti ai campi dell'entità e talvolta in letteratura prendono il nome di *tuple*. I record sono generalmente distinguibili l'uno dall'altro dai valori assunti dai vari campi. Ogni sottoinsieme dei campi di una tabella costituisce una **chiave** di ricerca dei record. Rifacendosi all'esempio relativo alla gestione di una biblioteca, tra le possibili chiavi di ricerca dei record contenuti nella tabella *libro* vi saranno quindi:

chiave

- Codice
- Titolo
- Cognome_autore
- Cognome_autore + Nome_autore
- Titolo + Cognome_autore + Nome_autore + Editore

Sarà quindi possibile effettuare una ricerca dei record per qualunque delle chiavi di ricerca.

L'insieme minimo dei campi che distinguono univocamente i record di una tabella costituisce la **chiave primaria** della tabella. La chiave primara è quindi univoca: all'interno di una tabella non possono esservi due (o più) record che hanno lo stesso valore della chiave primaria. Nell'esempio della biblioteca, la chiave primaria della tabella *Libro* può essere costituita semplicemente dal campo *Codice*, mentre la chiave primaria della tabella *Prestito* può essere costituita dall'insieme dei campi *codice_lettore* + *codice_libro* + *data_inizio_prestito*;

chiave primaria

Dunque, definendo un chiave primaria relativa ad una tabella, il DBMS controllerà, ad ogni richiesta di aggiornamento della tabella, se il record in oggetto può essere effettivamente memorizzato nella tabella, ovvero se l'inserimento del nuovo record o la modifica di uno già presente violeranno l'univocità della chiave primaria. In caso affermativo il DBMS si rifiuterà di effettuare l'operazione richiesta.

Il DBMS è in grado effettuare ricerche di record con prestazioni elevate, grazie alla definizione di *indici*. Un **indice** è una struttura dati che mantiene organizzati i record secondo un'ordine logico specificato dal creatore dell'indice. Ad esempio, se si desidera mantenere i record della tabella *libro* ordinati secondo l'ordine alfabetico del cognome e nome dell'autore, si deve creare un indice relativo ai campi *Cognome_autore* e *Nome_autore* della tabella *libro*. In questo modo, per ogni aggiornamento della tabella *libro*, verranno mantenuti ordinati i riferimenti ai record della tabella, secondo i campi scelti.

indice

L'indicizzazione di una tabella si basa su un riferimento mantenuto ordinato nella tabella considerata all'interno di un'altra struttura dati. In particolare è necessario avere

un riferimento, ovvero una chiave, da memorizzare in una struttura dati, tipicamente una tabella gestita automaticamente dal DBMS, che è composta ad esempio da un campo di ordinamento, ad esempio un campo *Indice*, che viene incrementato ad ogni inserimento di dati in questa tabella. Ogni volta che la tabella interessata dall'indice viene aggiornata (viene cambiato uno dei campi di un record sui quali si basa l'indice o viene inserito un nuovo record), viene aggiornata anche la tabella relativa all'indice, in maniera tale da mantenere i riferimenti ai record ordinati per i valori dell'indice. Questo meccanismo è oneroso in fase di aggiornamento dei dati, ma risulta utile per effettuare ricerche, poiché consente di sfruttare l'ordinamento dei record.

Esistono varie tecniche di gestione degli indici. Ad esempio si può pensare di creare un file (o tabella) indice nel quale sono memorizzati i riferimenti ai record della tabella in questione in maniera ordinata secondo la chiave desiderata. Inoltre si può pensare, qualora il file indice diventi molto grande, di suddividere il file in n file più piccoli ed indicizzare i file indici stessi. Si viene così a delineare una struttura di indici a più livelli. Si possono anche gestire indici per mezzo di tecniche associative. Ad esempio una chiave univoca può essere utilizzata per generare un codice secondo una funzione hash e tale codice può essere memorizzato in un file, assieme al riferimento al record (una funzione hash è tale per cui genera codici "abbastanza" scorrelati in relazione alla chiave considerata). In questo modo, la ricerca dei record secondo la chiave considerata diventa molto veloce...???

??? figura di esempio ???

???

12.4.3 Le relazioni e l'integrità referenziale

relazioni

Un database relazionale o RDB (Relational DataBase) permette anche di specificare le **relazioni** esistenti tra le entità. Ad esempio, nel caso della biblioteca esisterà sicuramente una relazione tra libro ed autore, in quanto l'esistenza di un libro è legata all'esistenza di un autore (se non esiste l'autore, non esiste il libro). Quindi vi sarà una relazione tra i campi *nome_autore* e *cognome_autore* della tabella *libro* ed i campi *nome* e *cognome* della tabella *autore*. In particolare l'insieme dei campi *nome_autore* e *cognome_autore* costituisce quella che viene detta una **foreign key** (chiave esterna) per la tabella *autore*.

foreign key

Inoltre è possibile che un autore possa aver scritto più libri presenti sugli scaffali, quindi la relazione tra l'entità *autore* e l'entità *libro* è del tipo 1:n, cioè ad 1 autore possono corrispondere più libri. Dunque i campi *nome_autore* e *cognome_autore* della tabella *libro*, non potranno costituire una chiave univoca per tale tabella. Tale tipo di relazione è quella più utilizzata tra le tabelle.

Un altro tipo di relazione quella n:m che indica che ad un record della tabella *A* possono corrispondere più record della tabella *B* e ad un record della tabella *B* possono corrispondere più record della tabella *A*. Questo tipo di relazione può essere implementato soltanto con l'ausilio di una terza tabella *C* in modo tale che tra la tabella *C* e la tabella *A* esista la relazione 1:n e la stessa relazione esista tra la tabella *C* e la tabella *B*. In questo caso la tabella *C* non rappresenta un'entità ma rappresenta una relazione.

integrità referenziale

Per mezzo di una relazione, il DBMS è in grado di controllare l'integrità dei riferimenti tra i record delle tabelle, ovvero quella che comunemente viene detta **integrità referenziale**. Cioè, quando viene aggiornata una tabella il DBMS è in grado di verificare se esistono i record ad essa collegati nelle altre tabelle. Se ciò non accade, il DBMS non permette l'aggiornamento della tabella: questo vincolo rafforza l'integrità referenziale del database.

Ad esempio, nel caso della gestione di una biblioteca è naturale mettere in relazione i campi *nome_autore* e *cognome_autore* della tabella *libro* con i campi *nome* e *cognome* della tabella *autore*, specificando che tra la tabella *autore* e la tabella *libro* c'è una relazione 1:n. In questo modo se si tenta di inserire un record nella tabella *libro* senza che sia presente il record corrispondente nella tabella *autore*, il record non viene inserito nella tabella *libro*.

È possibile tracciare uno schema composto dalle strutture dati che costituiscono il database, ovvero le entità, e le relazioni esistenti tra esse. Tale schema è noto come **ERD** (Entity Relation Diagram) o diagramma entità-relazioni.

ERD

???

12.4.4 La normalizzazione dei dati

Il problema dell'integrità di dati archiviati in un database è un'indice dell'affidabilità del database stesso. Un DB dovrebbe garantire la coerenza delle informazioni in esso archiviate, altrimenti si corre il rischio che le informazioni che risultano da una ricerca nel database siano errate o addirittura fuorvianti.

Per poter risolvere il problema dell'integrità e la coerenza dei dati all'interno di un database è necessario seguire i seguenti punti:

1. Minimizzare la ridondanza delle informazioni
2. Creare le relazioni necessarie a garantire il corretto aggiornamento delle tabelle

???

12.4.5 Il linguaggio SQL

Per la gestione della struttura e delle informazioni presenti in un database in genere viene utilizzato un apposito linguaggio di programmazione standardizzato dall'ANSI, l'**SQL** (*Structured Query Language*) che costituisce sia il DDL (*Data Definition Language*) che il DML (*Data Manipulation Language*), ovvero ha i comandi per creare/modificare le strutture dati e le relazioni tra esse e quelli per inserire/modificare i dati in esse contenuti. Quasi tutti i DBMS hanno comunque sviluppato delle particolari estensioni a tale linguaggio.

SQL

Il linguaggio SQL si basa sul linguaggio naturale, tant'è che i comandi derivano direttamente dal vocabolario inglese.

???

Le viste logiche

???

I cursori

???

12.4.6 Le stored procedure

I DBMS in genere permettono la memorizzazione di procedure, scritte nel linguaggio da essi interpretato, che vanno sotto il nome di **stored procedure** o *funzioni*. Tali procedure subiscono inoltre un processo di ottimizzazione di esecuzione, una sorta di compilazione. In questo modo si possono creare delle procedure automatizzate anche di elevata complessità, facendole poi successivamente eseguire indicando semplicemente al DBMS il nome della relativa procedura.

stored procedure

???

12.4.7 I trigger

???

12.4.8 Le transazioni

???

12.5 La gestione degli utenti

???

12.6 Implementazioni di database

Esistono varie implementazioni di RDBMS (Oracle 8 di *Oracle*, DB2 di *IBM*, Sybase SQL Server di *Sybase*, SQL Server 7.0 di *Microsoft*) e con GNU/Linux ne vengono fornite due: *PostgreSQL* e *MySQL*. Il primo è un RDBMS più completo dal punto di vista delle caratteristiche di DBMS, mentre l'altro è un RDBMS che punta molto sulla velocità di esecuzione dei comandi.

???

12.6.1 PostgreSQL

???

12.6.2 MySQL

???

12.7 Riferimenti

???

Capitolo 13

Applicazioni utili

“Soltanto chi parla di quello che ha sperimentato è fiducioso.”
– H. Hesse

???

13.1 Gestione del filesystem

???

13.1.1 Midnight Commander

???

13.1.2 Nautilus e Konqueror

???

13.2 Terminale grafico

???

13.3 Calcoli

???

13.4 Scrittura di testi

???

13.5 Grafica

???

13.6 Audio

???

13.7 Office automation

???

13.8 Visualizzazione file PDF

???

13.9 Web browser

???

13.10 Tool per lo sviluppo

???

13.11 Riferimenti

???

Capitolo 14

Installazione del software

“Non conta il colore del gatto, conta che acchiappi il topo.”
– Confucio

L'installazione del software in un sistema GNU/Linux è una delle cose che può disorientare i nuovi utenti, poiché non esiste un sistema univoco per installare le applicazioni, né tantomeno esiste, al momento, un sistema user-friendly.

14.1 I pacchetti

L'installazione del software in un sistema GNU/Linux avviene attraverso l'opportuno utilizzo di quelli che sono detti **pacchetti** (*package*) di installazione. I pacchetti sono file, generalmente in formato compresso, che, opportunamente trattati, installano il software considerato nel sistema. Il trattamento dei file, quindi l'installazione del software, avviene con metodi che dipendono dal particolare tipo di pacchetto utilizzato. Esistono vari tipi di pacchetti: i tarball, gli rpms ed i deb. L'utilizzo dei vari tipi di pacchetti è illustrato nelle sezioni seguenti.

In genere, i pacchetti che vengono installati su un sistema, sono quelli relativi ad applicazioni già in formato binario ed eseguibili direttamente dalla CPU. Esiste anche la possibilità, qualora l'autore del software lo renda disponibile, di procurarsi il pacchetto contenente i file sorgenti dell'applicazione in modo da poterli compilare direttamente sul proprio sistema, impostando le ottimizzazioni di compilazione desiderate (per alcuni software questo è l'unico modo per poterli installare). Di solito, un pacchetto contenente files sorgenti contiene all'interno del proprio nome la stringa `‘.src.’`, subito prima dell'estensione o è caratterizzato da un'estensione particolare che inizia per `‘.s’`. In tab. 14.1 sono riportate alcune tra le estensioni più utilizzate per i nomi dei pacchetti.

Per i sistemi Unix-like l'installazione di un software consiste nella copia dei file che lo compongono all'interno del filesystem, in modo tale che questi siano raggiungibili dal sistema operativo (ovvero il loro percorso risulti in uno di quelli in cui il sistema operativo va a ricercare i file eseguibili per poterli lanciare in esecuzione).

Le installazioni di software deve essere sempre effettuata dal superuser o comunque da un utente che ha i diritti di amministrazione del sistema, poiché in fase di installazione è necessario avere i diritti di scrivere ovunque nel filesystem.

14.1.1 Pacchetti tarball

Un **tarball** è un file contenente un insieme di altri file, in formato non compresso, che ha generalmente l'estensione `.tar`. Tale file viene gestito dal comando `tar` (man page `tar(1)`) che è un “archiviatore” di file, ovvero non fa altro che raggruppare più file all'interno di un unico file.

Estensione	Descrizione
.tar	File tarball.
.src.tar	File tarball contenente file sorgenti.
.tar.Z	File tarball compresso con compress .
.taZ	File tarball compresso con compress .
.taz	File tarball compresso con compress .
.tar.gz	File tarball compresso con gzip .
.tgz	File tarball compresso con gzip .
.tar.bz2	File tarball compresso con bzip2 .
.tar.bz	File tarball compresso con bzip2 .
.tbz2	File tarball compresso con bzip2 .
.tbz	File tarball compresso con bzip2 .
.zip	File compresso con zip .
.deb	File compresso contenente file binari per Debian.
.dsc	File di descrizione per la gestione dei file sorgenti per Debian.
.orig.tar.gz	File compresso contenente i file sorgenti per Debian.
.diff.gz	File compresso contenente le differenze dei file sorgenti delle versioni successive per Debian.
.rpm	File compresso contenente file binari per rpm .
.srpm	File compresso contenente file sorgenti per rpm .

Tabella 14.1: Alcune estensioni utilizzate nei nomi dei pacchetti utilizzati da GNU/Linux.

Comando: **tar**

Path: **/bin/tar**

SINTASSI

\$ tar [option] file_name [file_name_2 [...]]

DESCRIZIONE

option indica le opzioni di funzionamento di **tar**. Può assumere i seguenti valori:

- A | --catenate | --concatenate**
aggiunge uno o più archivi ad un archivio;
- c | --create**
crea un nuovo archivio;
- d | --diff | --compare**
confronta il contenuto di un archivio con i file presenti su un filesystem, evidenziandone le differenze;
- delete**
rimuove uno o più file da un archivio;
- r | --append**
aggiunge uno o più file in un archivio;
- t | --list**
elenca i file contenuti in un archivio;
- u | --update**
aggiunge soltanto i file più recenti di quelli già presenti nell'archivio;
- x | --extract | --get**
estrae i file da un archivio;
- atime-preserve**
preserva la data/ora dell'ultimo accesso dei file estratti dall'archivio;
- b num | --block-size num**
imposta la dimensione dei blocchi in $num \times 512$ byte (default $num=20$);
- B | --read-full-blocks**
ridimensiona i blocchi man mano che li legge;
- C dir | --directory dir**
cambia la directory in *dir*;
- checkpoint**
visualizza i nomi delle directory durante la lettura dell'archivio;
- f [hostname:]filename | --file [hostname:]filename**
indica l'archivio rappresentato dal file *filename* sul computer *hostname* (default */mnt/rmt0*);

```

--force-local
    il file contenente l'archivio è sul sistema locale anche se è stato
    specificato il carattere ':';
-F filename | --info-script filename | --new-volume-script filename
    esegue lo script contenuto nel file filename alla fine di ogni volume
    (nastro);
-G | --incremental
    indica di considerare l'archivio di tipo incrementale (vecchia versione
    GNU);
-g filename | --listed-incremental filename
    indica di considerare l'archivio di tipo incrementale (nuova versione
    GNU);
-h | --dereference
    indica di non considerare i symlink, ma direttamente i file ai quali
    essi si riferiscono;
-i | --ignore-zeros
    indica di ignorare i blocchi di zero contenuti nell'archivio (normal-
    mente hanno il significato di EOF, ovvero fine del file);
-j | -I | --bzip
    tratta l'archivio con bzip2 (l'uso di -I è deprecato);
--ignore-failed-read
    indica di ignorare gli errori di lettura dei file;
-k | --keep-old-files
    non sovrascrive i file già presenti sul filesystem con quelli contenuti
    nell'archivio;
-K filename | --starting-file filename
    indica di iniziare le operazioni dal file filename;
-l | --one-file-system
    indica di creare l'archivio sul filesystem locale;
-L num | --tape-length num
    specifica l'ampiezza del volume (nastro) in  $num \times 1024$  byte;
-m | --modification-time
    indica di non estrarre dall'archivio la data/ora relativa all'ultima
    modifica dei file;
-M | --multi-volume
    indica di effettuare le operazioni relative ad un archivio su più volumi
    (nastri);
-N date | --after-date date | --newer date
    memorizza i file più recenti di date;
-o | --old-archive | --portability
    indica di creare un archivio nel formato V7 anziché ANSI;
-O | --to-stdout
    indica di estrarre i file sullo standard output (sul monitor del termi-
    nale);
-p | --same-permissions | --preserve-permissions
    indica di estrarre i file preservandone i permessi di accesso;
-P | --absolute-paths
    indica di non rimuovere il path dai nomi dei file presenti nell'archivio;
--preserve
    come -p -s;
-R | --record-number
    visualizza il numero del file nell'archivio in ogni messaggio;
--remove-files
    indica di cancellare i file dopo averli aggiunti all'archivio;
-s | --same-order | --preserve-order
    indica di estrarre i file nello stesso ordine di quelli contenuti nell'ar-
    chivio;
--same-owner
    indica di estrarre i file contenuti nell'archivio preservandone il pro-
    prietario;
-S | --sparse
    tratta in maniera efficiente gli sparse file;

```

```

-T filename | --files-from=filename
    legge l'elenco dei nomi dei file da aggiungere/estrarre a/da l'archivio
    dal file filename;
--null indica di leggere i nomi dei file terminati dal carattere NUL;
--totals
    visualizza il numero totale dei byte scritti con --create;
-v | --verbose
    visualizza l'elenco dei file su cui sono state effettuate le operazioni;
-V volname | --label volname
    crea l'archivio con il nome di volume volname;
--version
    visualizza la versione del comando tar;
-w | --interactive | --confirmation
    indica di richiedere una conferma per ogni operazione;
-W | --verify
    tenta di verificare l'archivio subito dopo averlo effettivamente scritto
    sul supporto magnetico;
--exclude filename
    indica di escludere il file filename dalle operazioni;
-X filename | --exclude-from filename
    indica di escludere dalle operazioni l'elenco di file contenuto nel file
    filename;
-Z | --compress | --uncompress
    tratta l'archivio con compress;
-z | --gzip | --ungzip
    tratta l'archivio con gzip;
--use-compress-program cmdname
    tratta l'archivio con cmdname (NB. cmdname deve essere compatibile
    con l'opzione -d);

```

file_name è il nome del file da aggiungere/estrarre in/da l'archivio (possono essere specificati più file o una o più directory);

Quindi, per installare il software contenuto in un pacchetto `.tar`, è sufficiente copiare il file che lo contiene all'interno di una directory ed estrarre i file in esso contenuti con il seguente comando

```
# tar -xvf package
```

dove *package* è il nome del pacchetto, ovvero quello del file con estensione `.tar`.

Per esigenze di spazio, tali pacchetti vengono generalmente compressi mediante i programmi `compress`, `gzip` o `bzip2`. Si ottengono così file con dimensioni più contenute, con le estensioni `.tar.Z`, `.tar.Z` o `.tar.gz` (tarball compressi con `compress`), `.tar.gz` o `.tgz` (tarball compressi con `gzip`) e `.tar.bz2`, `.tar.bz`, `.tar.tbz2` o `.tar.tbz` (tarball compressi con `bzip2`).

Per installare un pacchetto tarball compresso con `compress` o `gzip`, si deve prima decomprimere il file con l'opportuno decompressore, `uncompress` o `gunzip` (`gunzip` è completamente compatibile con `uncompress`), ottenendo un tarball che può essere trattato come precedentemente illustrato. Si possono anche effettuare entrambi i passi contemporaneamente, con il comando

```
# tar -zxvf package
```

dove *package* è il nome del pacchetto, ovvero quello del file con estensione `.tar.Z`, `.tar.Z`, `.tar.gz`, `.tar.gz` o `.tgz`.

Per installare un pacchetto tarball compresso con `bzip2`, si deve prima decomprimere il file con l'opportuno decompressore `gunzip2`, ottenendo un tarball che può essere trattato come precedentemente illustrato. Si possono anche effettuare entrambi i passi contemporaneamente, con il comando

```
# tar -jxvf package
```

dove *package* è il nome del pacchetto, ovvero quello del file con estensione `.tar.bz2`, `.tar.bz`, `.tbz2` o `.tbz`.

A questo punto sarà possibile utilizzare il pacchetto stesso dopo aver letto le relative istruzioni presenti, generalmente, in un file denominato **README** (o qualcosa del genere) all'interno della directory creata dalla decompressione del pacchetto stesso.

Poiché il comando **tar** esegue soltanto l'estrazione di più file da un unico file, eventualmente compresso, non si ha la possibilità di eseguire comandi specifici, inerenti all'installazione di un software su un filesystem. Pertanto è probabile che, una volta estratti i file del software considerato, si debba anche lanciare uno specifico comando per l'inizializzazione di alcuni file, che generalmente viene effettuata con uno script di shell denominato **install.sh** (o qualcosa del genere).

Nel caso in cui si voglia installare un software di cui si ha un tarball contenente i file sorgenti dello stesso (il nome del file è del tipo *pkg.name.scr.ext*, dove *pkg.name* è il nome indicativo del pacchetto e *ext* è l'estensione che denota il tipo del pacchetto), è necessario effettuare la compilazione dei sorgenti prima di poter utilizzare il software. La compilazione dei sorgenti è un'operazione complessa e delicata e poiché non esiste una procedura standard è necessario leggere la documentazione distribuita assieme ai sorgenti.¹

Tipicamente la procedura di installazione di un pacchetto di sorgenti si riassume nei seguenti passi:

```
# tar -zxvf package
# cd package_dir
# ./configure
# make
# make install
```

dove *package* è il nome del pacchetto tarball compresso (in questo caso si è considerato soltanto il caso della compressione con **compress** o **gzip**) e *package_dir* è la directory creata dalla decompressione del pacchetto.

Di seguito è presentata una breve descrizione dei comandi:

tar -zxvf package	scompatta il pacchetto <i>package</i> . Generalmente viene creata una directory (<i>package_dir</i>) in cui viene messo il contenuto del pacchetto;
cd package_dir	imposta la working directory a quella appena creata dalla scompattazione del pacchetto <i>package</i> ;
./configure	verifica le dipendenze tra i file sorgenti e crea il <i>makefile</i> (file contenente le opportune direttive per il compilatore);
make	per mezzo del <i>makefile</i> creato precedentemente compila i sorgenti, creando i file eseguibili dal sistema;
make install	copia i file necessari (eseguibili e non) nelle opportune directory in modo da rendere utilizzabile l'applicazione considerata.

14.1.2 Pacchetti deb

È il sistema utilizzato essenzialmente da *Debian*. Utilizza il comando **dpkg** (man page **dpkg(8)**). Fa differenza tra pacchetti binari (contenenti applicazioni già compilate, pronte per essere eseguite) e pacchetti sorgenti (contenenti i file sorgenti da compilare per ottenere l'applicazione eseguibile). I pacchetti binari sono costituiti da un unico

¹per poter compilare i file sorgenti devono essere installati sul sistema i pacchetti necessari per la compilazione dei file (Development).

file con estensione `.deb`, mentre i pacchetti sorgenti sono costituiti da una terna di file: un file di descrizione con estensione `.dsc`, un file contenente i sorgenti con estensione `.orig.tar.gz` ed un file contenente le differenze da applicare ai sorgenti per la realizzazione della specifica versione del software con estensione `.diff.gz`. Esiste anche la possibilità di gestire i pacchetti ad un livello più alto con i comandi `dselect` e `apt-get`.

14.1.3 Pacchetti rpm

Un pacchetto rpm è un file in formato compresso, generalmente con estensione `.rpm`, che contiene più file. Esso contiene anche delle informazioni relative alla directory del filesystem nella quale devono essere estratti i file in esso contenuti, ed altri dettagli relativi alla versione del pacchetto, ... Tali file vengono gestiti dal comando `rpm` (*Red Hat* Package Manager – man page `rpm(8)`), ed è uno dei sistemi più utilizzati dalle varie distribuzioni di GNU/Linux (Red Hat, SuSE, Mandrake).

La gestione dei pacchetti rpm si basa su di un database (un sistema di memorizzazione dati) nel quale vengono registrate tutte le applicazioni che man mano vengono installate sul sistema. In questo modo si può interrogare il sistema per sapere se un'applicazione è già stata installata ed eventualmente conoscere qual'è la versione installata, in modo da poter gestire l'aggiornamento ad una versione successiva. I pacchetti possono avere delle dipendenze, ovvero un pacchetto può dipendere da altri pacchetti, cioè può aver bisogno che altri pacchetti siano già installati sul sistema per poter funzionare. Questo perché le applicazioni contenute all'interno di un pacchetto possono aver bisogno di applicazioni, librerie o file che si trovano in altri pacchetti per poter funzionare correttamente. Quando un *pacchetto A* deve essere installato, `rpm` controlla che eventuali altri pacchetti necessari siano installati sulla macchina ed in caso negativo visualizza un messaggio che informa l'utente che il *pacchetto A* non può essere installato perché non sono soddisfatte le dipendenze, ovvero mancano dei pacchetti necessari al funzionamento delle applicazioni contenute nel *pacchetto A*. In caso positivo invece, viene controllata anche la versione di un eventuale pacchetto dello stesso tipo del *pacchetto A* già installato sulla macchina: l'aggiornamento del pacchetto viene effettuata soltanto nel caso in cui il *pacchetto A* sia identificato da una versione più recente di quello già installato sul sistema. Quindi, se si desidera installare un *pacchetto A* che dipende da un *pacchetto B*, è necessario procurarsi il *pacchetto B* ed installarlo prima di procedere con l'installazione del *pacchetto A*. Questo può anche instaurare un meccanismo di installazione di pacchetti a catena, in quanto un pacchetto può dipendere da altri pacchetti, ognuno dei quali dipende da altri pacchetti e così via.

Comando: `rpm`

Path: `/bin/rpm`

SINTASSI

`$ rpm [option] [specific_option] [pkg.filename | pkgname]`

DESCRIZIONE

option indica le opzioni di funzionamento di `rpm`. Può assumere i seguenti valori:

`-? | --help`

visualizza un aiuto sommario di `rpm`;

`--version`

visualizza la versione di `rpm`;

`--quiet`

indica di procedere in modalità silenziosa (visualizza soltanto eventuali errori);

`-v`

indica di procedere in modalità verbosa (visualizza più informazioni del normale);

`--vv` indica di procedere in modalità di massima verbosità;

`--rcfile filelist`

indica di considerare l'elenco dei file espressi da *filelist* come file di configurazione di `rpm` (*filelist* è l'elenco di file separati tra loro dal carattere ':'). Per default *filelist* è `'/usr/lib/rpm/rpmrc:/usr/lib/rpm/redhat/rpmrc:~/rpmrc'`;


```

--pipe command
    reindirige l'output di rpm nell'input di command;
--dbpath dir
    specifica la directory (dir) contenente il database di rpm (la directory
    di default è /var/lib/rpm);
--root dir
    specifica la directory (dir) da considerare come root directory per
    il controllo delle dipendenze e per l'esecuzione di eventuali script
    lanciati da rpm;
{-i | --install} [install_options] pkg_filename
    indica di installare il pacchetto contenuto nel file pck_filename;
{-U | --upgrade} [install_options] pkg_filename
    indica di installare o aggiornare il pacchetto contenuto nel file pck_filename;
{-F | --freshen} [install_options] pkg_filename
    indica di aggiornare il pacchetto contenuto nel file pck_filename sol-
    tanto se una versione più datata è già installata sul sistema;
{-e | --erase} [erase_options] pkgname
    indica di rimuovere (disinstallare) il pacchetto pkgname;
{-q | --query} [select_options] [query_options] [pkgname]
    indica di ricercare uno o più pacchetti nel database;
    ???

```

specific_option indica le opzioni per ogni specifica operazione di **rpm**. Possono assumere i seguenti valori:

install_options

sono opzioni specifiche per l'installazione/aggiornamento di pacchetti. Possono assumere i seguenti valori:

```

--aid
    aggiunge gli eventuali pacchetti suggeriti a quelli da installare;
--allfiles
    installa o aggiorna tutti i file missingok nel pacchetto (indipenden-
    temente dal fatto che esistano o meno);
--badreloc
    utilizzato assieme a --relocate indica di effettuare la rilocalazione su
    tutti i percorsi dei file, non soltanto per quelli inclusi nel pacchetto
    binario;
--excludepath path
    indica di non installare i file il cui percorso inizia con path;
--excludedocs
    indica di non installare nessun file di documentazione;
--force
    forza l'installazione del pacchetto: è equivalente all'uso combinato
    di --replacepkgs, --replacefiles e --oldpackage;
-h | --hash
    indica di visualizzare 50 caratteri '#' durante l'estrazione dei file;
--ignoresize
    indica di non verificare lo spazio disponibile sui filesystem prima di
    installare il pacchetto;
--ignorearch
    indica di procedere con l'installazione/aggiornamento dei file anche
    se l'architettura per il quale è stato creato il pacchetto non coincide
    con quella del sistema;
--ignoreos
    indica di procedere con l'installazione/aggiornamento dei file anche
    se il sistema operativo per il quale è stato creato il pacchetto non
    coincide con quello del sistema;
--includedocs
    indica di installare anche i file di documentazione (default);
--justdb
    indica di aggiornare soltanto il database ma non il filesystem;
--nodigest
    indica di non verificare il message digest2 del pacchetto;

```

²v. sez. 23.3.5.

```

--nosignature
    indica di non verificare la firma digitale3 del pacchetto;
--nodeps
    indica di non effettuare il controllo delle dipendenze prima di pro-
cedere all'installazione/aggiornamento del pacchetto;
--nosuggest
    indica di non suggerire i pacchetti che contengono la dipendenza
mancante;
--noorder
    indica di non cambiare l'ordine dei pacchetti da installare (l'ordine
dell'elenco dei pacchetti da installare viene normalmente modificato
in maniera tale da soddisfare le dipendenze);
--nopre
    indica di disabilitare l'esecuzione degli script subito prima dell'e-
strazione dei file;
--nopost
    indica di disabilitare l'esecuzione degli script subito dopo l'estrazio-
ne dei file;
--nopreun
    indica di disabilitare l'esecuzione degli script subito prima della
disinstallazione del pacchetto;
--nopostun
    indica di disabilitare l'esecuzione degli script subito dopo la disin-
stallazione del pacchetto;
--noscripts
    indica di disabilitare l'esecuzione degli script subito prima e subi-
to dopo l'installazione o disinstallazione del pacchetto: equivale a
--nopre --nopost --nopreun --nopostun;
--notriggerin
    indica di disabilitare l'esecuzione degli script relativi ai trigger di
inserimento di pacchetti nel database;
--notriggerun
    indica di disabilitare l'esecuzione degli script relativi ai trigger di
rimozione di pacchetti dal database;
--notriggerpostun
    indica di disabilitare l'esecuzione degli script relativi ai trigger di
post-rimozione di pacchetti dal database;
--notriggers
    indica di disabilitare l'esecuzione degli script relativi ai trigger di ag-
giornamento del database: equivale a --notriggerin --notriggerun
--notriggerpostun;
--oldpackage
    indica di permettere l'aggiornamento di un pacchetto già installato
con uno meno recente;
--percent
    indica di visualizzare la percentuale relativa all'operazione di estra-
zione dei file dal pacchetto;
--prefix path
    specifica il path in cui devono essere installati i file binari rilocabili
estratti dal pacchetto;
--relocate oldpath=newpath
    specifica di sostituire il path di installazione dei i file binari rilocabili
presenti nel pacchetto, il cui path inizia con oldpath, con newpath;
--repackage
    indica di ricostruire il pacchetto prima di disinstallare i file dal
sistema (per default il pacchetto viene ricreato in /var/tmp);
--replacefiles
    indica di procedere con l'installazione del pacchetto anche se l'estra-
zione dei file in esso contenuti rimpiazzerà alcuni file già presenti
nel sistema, installati da altri pacchetti;
--replacepkgs
    indica di procedere con l'installazione del pacchetto anche se esso è
già installato nel sistema;

```

³v. sez. 23.4.

--test
 indica di effettuare tutte le operazioni di installazione/aggiornamento del pacchetto, ma di non installare/aggiornare affatto il pacchetto: esegue un test, riportando eventuali potenziali problemi;

erase_options

sono opzioni specifiche per la disinstallazione di pacchetti. Possono assumere i seguenti valori:

--allmatches
 indica di rimuovere tutti i pacchetti che iniziano con il nome *pkgname* (normalmente **rpm** restituisce un errore se più di un pacchetto installato inizia con *pkgname*);

--nodeps
 indica di non controllare le dipendenze con altri pacchetti prima di rimuovere *pkgname*;

--nodeps
 indica di non controllare le dipendenze con altri pacchetti prima di rimuovere *pkgname*;

--nopreun
 indica di disabilitare l'esecuzione degli script subito prima della disinstallazione del pacchetto;

--nopostun
 indica di disabilitare l'esecuzione degli script subito dopo la disinstallazione del pacchetto;

--noscripts
 indica di disabilitare l'esecuzione degli script subito prima e subito dopo la disinstallazione del pacchetto: equivale a **--nopreun** **--nopostun**;

--notriggerun
 indica di disabilitare l'esecuzione degli script relativi ai trigger di rimozione di pacchetti dal database;

--notriggerpostun
 indica di disabilitare l'esecuzione degli script relativi ai trigger di post-rimozione di pacchetti dal database;

--notriggers
 indica di disabilitare l'esecuzione degli script relativi ai trigger di rimozione dei pacchetti dal database: equivale a **--notriggerun** **--notriggerpostun**;

--repackage
 indica di ricostruire il pacchetto prima di disinstallare i file dal sistema (per default il pacchetto viene ricreato in */var/tmp*);

--test
 indica di effettuare tutte le operazioni di disinstallazione del pacchetto, ma di non disinstallarlo affatto: esegue un test, riportando eventuali potenziali problemi;

query_options

sono opzioni specifiche per la ricerca di pacchetti nel database. Possono assumere i seguenti valori:

{--qf | --queryformat} *qryformat*
 specifica il formato da utilizzare nella visualizzazione del risultato della ricerca. Il formato è specificato da *qryformat* ed ha una sintassi analoga a quella della funzione **printf** del linguaggio C, anche se lo specificatore di tipo può essere omesso e sostituito con il nome del campo da visualizzare racchiuso tra parentesi graffe '{' e '}'. L'elenco dei campi visualizzabili da **rpm** può essere ottenuto con l'opzione **--querytags** ed è riportato in tab. 14.2, 14.3 e 14.4. Per default viene utilizzata il seguente formato di visualizzazione **--qf "%{NAME}-%{VERSION}-%{RELEASE}"**. Il formato di visualizzazione può essere specificato postponendo al campo da visualizzare uno dei seguenti specificatori di tipo:

:armor
 visualizza una chiave pubblica in ASCII;

:base64
 codifica i valori binari utilizzando valori numerici di 64 bit;

Campo	Descrizione
HEADERI18N TABLE HEADERIMAGE HEADERIMMUTABLE HEADERREGIONS HEADERSIGNATURES ICON INSTALLCOLOR INSTALLPREFIX INSTALLTID INSTALLTIME INSTPREFIXES LICENSE NAME OBSOLETEFLAGS OBSOLETENAME OBSOLETES OBSOLETEVERSION OLDFILENAMES OPTFLAGS OS PACKAGER PATCH PATCHESFLAGS PATCHESNAME PATCHESVERSION PAYLOADCOMPRESSOR PAYLOADFLAGS PAYLOADFORMAT PLATFORM POSTIN POSTINPROG POSTUN POSTUNPROG PREFIXES PREIN PREINPROG PREUN PREUNPROG PROVIDEFLAGS PROVIDENAME PROVIDES PROVIDEVERSION PUBKEYS RELEASE REMOVETID REQUIREFLAGS REQUIRENAME REQUIREVERSION RHNPLATFORM RPMVERSION RSAHEADER	nome del pacchetto

Tabella 14.3: Campi visualizzabili dei pacchetti rpm (II parte).

```

:fflags
    visualizza i flag del file;
:hex
    visualizza il valore in notazione esadecimale;
:octal
    visualizza il valore in notazione ottale;
:perms
    visualizza i permessi del file;
:shescape
    visualizza il valore ponendo dei caratteri di escape prima dei
    caratteri ‘’;
:triggertype
    visualizza il suffisso del trigger;

```

Campo	Descrizione
SERIAL	
SHA1HEADER	
SIGPGP	
SIGMD5	
SIGPGP	
SIGSIZE	
SIZE	
SOURCE	
SOURCEPACKAGE	
SOURCEPKGID	
SOURCERPM	
SUMMARY	
TRIGGERCONDS	
TRIGGERFLAGS	
TRIGGERINDEX	
TRIGGERNAME	
TRIGGERSCRIPTPROG	
TRIGGERSCRIPTS	
TRIGGERTYPE	
TRIGGERVERSION	
URL	
VENDOR	
VERIFYSRIPT	
VERIFYSRIPTPROG	
VERSION	
XPM	

Tabella 14.4: Campi visualizzabili dei pacchetti rpm (III parte).

```
-a | --all
    indica di ricercare tutti i pacchetti installati;
{-f | --file} filename
    indica di ricercare il pacchetto contenuto nel file filename;
--fileid md5
    indica di ricercare il pacchetto il cui message digest del file che lo
    conteneva è specificato da md5;
{-g | --group} groupname
    indica di ricercare tutti i pacchetti il cui gruppo proprietario è
    specificato da groupname;
--hdrid sha1
    indica di ricercare il pacchetto il cui message digest dell'header del
    file che lo conteneva è specificato da sha1;
```

???

14.2 Riferimenti

???

Capitolo 15

Sviluppo di applicazioni

“Nulla è impossibile per colui che non deve farlo.”
– Anonimo

???

15.1 Introduzione

???

15.2 I programmi

???

15.2.1 L’interprete

???

15.2.2 Il compilatore

???

15.3 I linguaggi di programmazione

???

15.3.1 C

???

15.3.2 C++

???

15.3.3 Java

???

15.3.4 Perl

???

15.3.5 Phyton

???

15.3.6 Ruby

???

15.3.7 Tcl/Tk

???

15.4 Gli strumenti

???

15.4.1 Anjuta

???

15.4.2 Quanta

???

15.4.3 Glade

???

15.4.4 KDevelop

???

15.5 Riferimenti

???

Parte II

La rete

Capitolo 16

Concetti di base

“Da quando ho imparato a camminare mi piace correre.”
– F. Nietzsche

???

16.1 Le interfacce

Se due sistemi vogliono comunicare tra loro, devono utilizzare dei meccanismi in grado di portare le informazioni dall'uno all'altro senza alcuna perdita. Due o più computer comunicano tra loro attraverso delle apposite periferiche dette **interfacce di rete**, cioè dei dispositivi che permettono di inviare e ricevere informazioni attraverso un canale di comunicazione. La comunicazione avviene generalmente per mezzo di appositi cavi a cui le interfacce sono collegate: i **cavi di rete**. Possono essere utilizzati anche altri canali di comunicazione più esoterici, come quelli wireless (che non hanno bisogno di un cavo fisico, ma la comunicazione avviene via radio). Quindi, se due sistemi comunicano tra loro è perché lo fanno le loro relative interfacce di rete poiché la comunicazione avviene sempre tramite esse.

interfacce di rete

cavi di rete

Un computer può essere dotato di una o più interfacce di rete. Nel caso in cui su un computer siano installate più di una interfaccia di rete, si parla di sistema **multihomed**. Pertanto è bene tenere a mente che per quanto concerne le reti, vengono identificate le interfacce di rete, non le macchine (i computers).

multihomed

Un'interfaccia di rete è costituita fisicamente da una **scheda di rete** o un **modem** (**modulator/demodulator**). Il funzionamento dei due tipi di dispositivi è diverso perché essi realizzano a basso livello un sistema di comunicazione diverso. I modem sono essenzialmente utilizzati per comunicare sulla linea telefonica.

scheda di rete
modem

16.2 I server ed i client

Un'applicazione in esecuzione su di un sistema potrà utilizzare un'interfaccia di rete per comunicare con un'altra applicazione che è in esecuzione, in generale, su un altro sistema¹. In particolare si distinguono due categorie di applicazioni che utilizzano la comunicazione via rete: le applicazioni di tipo *client* e quelle di tipo *server*. Un'applicazione **client** è quella che richiede un servizio ad un'altra applicazione (*server*). Un'applicazione **server** non fa altro che attendere eventuali richieste da altre applicazioni (*client*) e servirle.

client
server

Server Client

Inizializzazione Invio richiesta ad un server

Attesa richiesta da un client Attesa risposta dal server

¹la comunicazione di rete può comunque avvenire tra due applicazioni in esecuzione sullo stesso sistema.

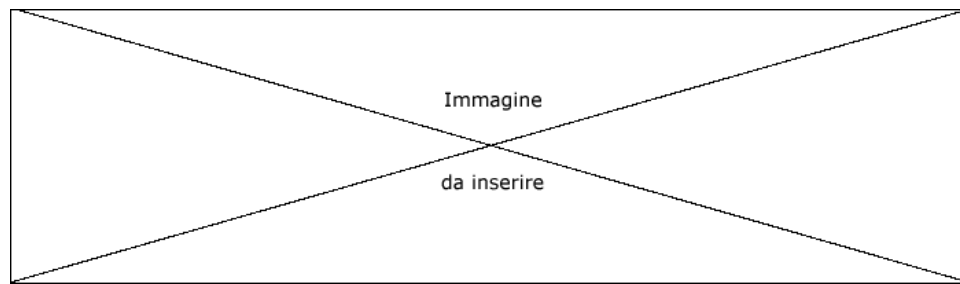


Figura 16.1: Schematizzazione applicazioni *client* e *server*.

Elaborazione richiesta

Invio risposta

La differenziazione tra *server* e *client* è estesa anche alle macchine: una macchina *server* è utilizzata per fornire uno o più servizi all'interno di una rete, quindi generalmente si tratta di macchine con hardware in grado di offrire prestazioni elevate (bassi tempi di accesso ai dischi, grande quantità di memoria centrale, ...), dipendentemente dai servizi che la stessa deve fornire. Una macchina *client* è utilizzata per lavorare (si parla infatti anche di *workstation*) e per utilizzare i servizi messi a disposizione dai server presenti sulla rete.

16.3 I protocolli

La modellizzazione della comunicazione di rete è stata effettuata supponendo di suddividere l'accesso alle funzionalità della comunicazione a vari livelli di astrazione a partire da quello più basso che è quello dei segnali coinvolti nell'effettiva comunicazione (la cui natura e tipologia dipendono dalla natura del mezzo di trasmissione) fino a quello più elevato che è quello che “vedono” le applicazioni che vogliono comunicare attraverso la rete. Un'applicazione che intende inviare delle informazioni sulla rete, utilizzerà l'interfaccia software messa a disposizione dal livello più alto, che si preoccuperà di trattare opportunamente le informazioni passandole al livello immediatamente inferiore e così via fino ad arrivare al livello più basso in cui i segnali logici saranno trasformati in segnali elettrici o elettromagnetici e quindi inviati sulla rete. Allo stesso modo, l'interfaccia di rete che riceve le informazioni le tratterà in maniera opportuna passandole man mano ad un livello sempre più elevato, fino ad arrivare all'applicazione di destinazione. Ogni livello passa le informazioni a quello immediatamente inferiore (in trasmissione) o superiore (in ricezione) tramite un insieme di funzioni che costituiscono l'interfaccia di comunicazione tra un livello e l'altro.

Per ogni livello è definito un insieme di regole di comunicazione, ovvero quello che in gergo viene chiamato **protocollo**. Un protocollo è quindi una logica che fa comunicare due interfacce di rete allo stesso livello. Se ogni livello esegue correttamente il proprio dovere, si può pensare che la comunicazione avvenga tra le due interfacce di rete al livello considerato, sebbene la comunicazione avvenga in realtà sempre tra i livelli più bassi (v. sez. 16.5).

Ogni protocollo definisce un'unità logica da utilizzare per lo scambio di informazioni, il **pacchetto**, che assume una denominazione specifica dipendentemente dal protocollo considerato. Questo in genere è suddiviso in una parte di informazioni di servizio e di controllo, l'**header** (intestazione) ed il **footer** (parte terminale), ed una parte riservata alle informazioni da comunicare, il **payload**.

L'insieme dei livelli in cui è suddivisa la gestione dell'interfaccia di comunicazione è comunemente detto **stack di protocolli**.

In genere, il pacchetto di un determinato livello dello stack di protocolli, costituisce il payload dell'unità di informazioni del livello immediatamente sottostante. Questo meccanismo è detto **incapsulazione** (*data encapsulation*) poiché concettualmente equivale

protocollo

pacchetto

header

footer

payload

stack di protocolli

incapsulazione

all'inserimento di in pacchetto all'interno di un altro, come accade per una lettera che viene inserita in una busta prima di spedirla.

???

16.4 LAN e WAN

Le reti sono categorizzate anche in base alla loro estensione, poiché da questa dipendono il tipo di collegamento, il meccanismo di accesso al canale ed i protocolli relativi ai livelli più bassi dello stack. Le reti si dividono pertanto in

LAN (Local Area Network o reti locali) Esse hanno un'estensione contenuta generalmente all'interno di uno stesso edificio.

Local Area Network

WAN (Wide Area Network o reti estese) Reti più estese delle LAN.

Wide Area Network

16.5 Lo stack OSI

Il modello che è stato la pietra miliare nella definizione degli stack di protocolli di rete, ma di cui non esiste nessuna implementazione pratica, è il modello OSI (Open System Interconnection) proposto da ISO², che si suddivide nei 7 livelli di seguito elencati (v. anche fig. 16.2)

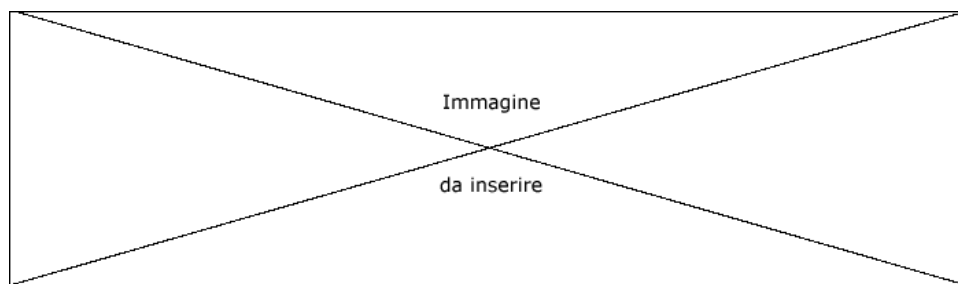


Figura 16.2: Rappresentazione dello stack OSI.

1. **Physical** Il *livello fisico* è sotto il dominio dell'ingegneria più che dell'informatica. Sulla base del mezzo di trasmissione utilizzato vengono individuate la natura e le caratteristiche del segnale da utilizzare per la trasmissione dei dati (modulazione/demodulazione, conversione digitale/analogica e viceversa, ...). Questo livello si preoccupa di trasformare i dati da inviare negli appositi segnali elettrici o elettromagnetici che vengono fisicamente inviati sulla rete e di convertire i segnali ricevuti negli equivalenti dati digitali.
2. **Data link** Il *livello di collegamento* gestisce la metodologia di accesso al mezzo trasmissivo. Stabilisce se la trasmissione può essere iniziata o meno (per esempio controlla se c'è già qualche altro sistema che sta già impegnando il mezzo di comunicazione). Inoltre si preoccupa che le informazioni arrivino a destinazione in maniera corretta, aggiungendo alle informazioni da trasmettere dei codici di controllo di basso livello.
3. **Network** Il *livello di rete* ha il compito di attuare la strategia di instradamento per far giungere i dati a destinazione. Marca i dati con l'identificativo del mittente e del destinatario. Ogni interfaccia di comunicazione di rete che riceve dei dati confronta l'identificativo del destinatario con il proprio, in modo da riconoscere

²ISO (International Organization for Standardization) è un organismo che redige standard internazionali.

se i dati sono destinati a lei. In caso affermativo passa i dati al livello superiore, altrimenti scarta il messaggio, ignorandolo.³

4. **Transport** Il *livello di trasporto* si preoccupa di scomporre lo stream di dati da inviare in pacchetti più piccoli per occupare la rete in maniera intelligente (senza renderla inaccessibile agli altri sistemi magari per qualche minuto se la quantità di dati da trasmettere è elevata). Viceversa si deve anche preoccupare di ricostruire lo stream di dati a partire dai pacchetti ricevuti nel corretto ordine con cui sono stati inviati. Quindi in ogni pacchetto saranno inserite informazioni aggiuntive per identificare l'ordine di invio di un pacchetto rispetto agli altri. Si preoccupa inoltre di gestire errori di comunicazione (inserimento e controllo di checksum) ed implementa i meccanismi necessari per richiedere quello che eventualmente è andato perduto nella comunicazione.
5. **Session** Il *livello di sessione* ha la responsabilità di gestire la comunicazione, ovvero implementa il protocollo al livello più elevato: si preoccupa di assicurarsi che l'interfaccia di comunicazione di rete a cui si vogliono spedire i dati sia pronta a riceverli (inviando un segnale tipo un ping) ed alla fine della comunicazione manda un segnale di chiusura (tipo un altro ping) in modo che il destinatario sappia che la comunicazione è terminata e possa effettuare le operazioni opportune (liberazione di risorse di memoria dedicate alla comunicazione, ...).
6. **Presentation** Il *livello di presentazione* ha la responsabilità di gestire la conversione dei dati da inviare secondo lo standard di codifica del sistema di destinazione. I sistemi che devono comunicare possono essere diversi (per esempio Windows e Unix), quindi i dati potrebbero avere codifiche diverse (es. ASCII e EBCDIC). A questo livello avviene la conversione nella codifica opportuna. In realtà non esistono implementazioni di questo livello, quindi la codifica deve essere fatta a livello di applicazione.
7. **Application** Il *livello di applicazione* è appunto il livello da cui le applicazioni “vedono” la rete, ovvero il livello più astratto. Un'applicazione deve soltanto indicare quali informazioni inviare ed a chi inviarle. L'applicazione che desidera inviare un certo stream di dati ad un'altra applicazione presente su di un altro sistema, connesso in rete tramite un'opportuna interfaccia di rete, invierà lo stream dal livello 7 al livello 6 e per lei il gioco è fatto (non si preoccupa di ciò che avviene dopo). L'applicazione “vede” soltanto l'interfaccia con il livello 6 e nient'altro. In realtà lo stream da inviare viene processato ad ogni livello dal 6 fino all'1 che si preoccupa effettivamente di inviarlo sulla rete. Dall'altra parte, l'interfaccia di rete ricevente riceve le informazioni trasmesse e queste passeranno dal livello 1 al livello 7 dopo opportuni controlli e ricostruzioni, per fornire così lo stream inviato all'applicazione ricevente.

???

16.6 Lo stack TCP/IP

Per le reti ci sono stati vari standard di protocolli di comunicazione che ovviamente hanno portato a problemi di interconnessione tra reti diverse. Quello che oggi prevale (e sarà destinato a farlo sempre di più) è il TCP/IP (che sarebbe meglio chiamare **Internet suite**), una suite di protocolli nata con *Internet*, negli anni '70, ovvero per le reti geografiche (WAN), poco affidabili rispetto alle LAN, ma che ha buone prestazioni anche su LAN (sebbene su LAN esistano dei protocolli più efficienti in termini di rapporto tra i codici di controllo e le effettive informazioni da trasmettere). Lo stack TCP/IP è composto dai 5 livelli riportati in fig. 16.3.

³un'interfaccia di rete può funzionare anche in modalità *promiscua*, il che le consente di far passare ai livelli superiori tutti i pacchetti che arrivano al suo connettore, anche se non destinati a lei (questa modalità è utilizzata da particolari applicazioni dette *sniffer*).

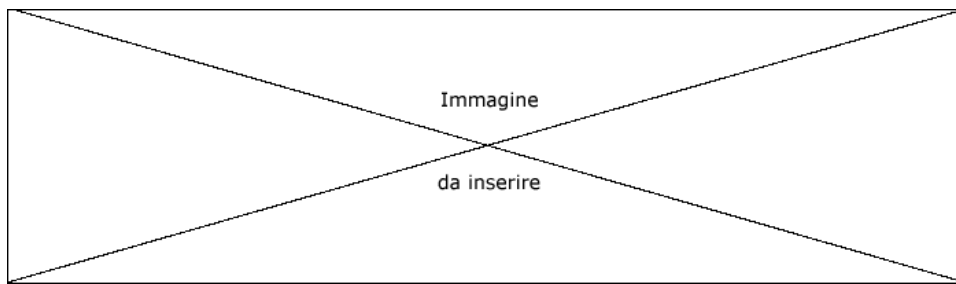


Figura 16.3: Rappresentazione dello stack TCP/IP.

- A. **Link** corrisponde ai livelli 1 (Physical) e 2 (Data link) dello stack OSI.
- B. **Network** si mappa esattamente sul livello 3 (Network) dello stack OSI.
- C. **Transport** ingloba i livelli 4 (Transport) e 5 (Session) dello stack OSI.
- D. **Application** ingloba i livelli 6 (Presentation) e 7 (Application) del modello OSI.

Gli standard che specificano i protocolli dello stack TCP/IP sono pubblicati attraverso il meccanismo delle RFC (Request For Comments). Una **RFC** è un documento che viene presentato alla **IETF** (*Internet Engineering Task Force*)⁴, che viene pubblicato per ricevere eventuali commenti. Una RFC che aspira a divenire uno standard passa attraverso 3 stadi, detti **livelli di maturità**: da *Proposed Standard* a *Draft Standard* a *Standard*. Lo stato dei protocolli è mantenuto in un apposito documento, denominato *Internet Official Protocol Standards*.

RFC

IETF

livelli di maturità

Nel seguito del presente testo, salvo esplicito avviso, sarà fatto sempre riferimento a questo stack di protocolli.

16.7 Altri stack di protocolli

Esistono altri stack di protocolli, in buona parte sviluppati prima dell'esplosione di *Internet* e quindi del TCP/IP. Ad esempio, IBM ha sviluppato SNA, Novell SPX/IPX e Xerox aveva sviluppato alla fine degli anni settanta XNS, Microsoft ha realizzato NETBIOS. Tutti questi stack hanno qualche parentela con il modello OSI, o lo hanno in parte ispirato.

16.8 Il livello fisico e di collegamento

Una **scheda di rete** è la realizzazione fisica di un'*interfaccia di rete*. A bordo della scheda vi è la circuiteria necessaria per l'implementazione del livello fisico (il cosiddetto *transceiver*) e gran parte del livello di collegamento (i livelli 1 e 2 dello stack OSI). Sebbene i protocolli del livello di collegamento non siano necessariamente collegati a quelli del livello fisico, di fatto ogni scheda di rete implementa entrambi i livelli (a meno del driver per la gestione del livello di collegamento). Ecco quindi perché si è soliti parlare di schede di rete *Ethernet* piuttosto che di schede di rete *Token ring*, cioè specifiche per la tecnologia con cui sono stati realizzati i livelli più bassi del meccanismo di comunicazione.

scheda di rete

Ogni scheda di rete è identificata univocamente da un numero detto **indirizzo fisico**, memorizzato all'interno del firmware presente sulla scheda stessa (non esistono due schede di rete al mondo con lo stesso valore). Tale valore è memorizzato su 48 bit (6 byte).

indirizzo fisico

I mezzi trasmissivi più diffusi che costituiscono il canale attraverso il quale avviene la comunicazione tra le schede di rete sono i cavi coassiali, i doppini intrecciati non

⁴v. <http://www.ietf.org>.

schermati UTP (Unshielded Twisted Pair) o schermati STP (Shielded Twisted Pair) e le fibre ottiche. Esistono comunque anche reti che si basano sulla trasmissione di onde radio o su mezzi ancora più esoterici.

??? 10BaseT ... ???

topologia fisica

topologia logica

La **topologia fisica** di una rete è la conformazione delle linee di trasmissione, ossia il percorso dei cavi tra un'interfaccia di rete e l'altra (il cablaggio). La **topologia logica** è invece il percorso che compiono i dati. In generale, non è detto che la topologia fisica e quella logica coincidano. Le topologie più diffuse sono le seguenti

Mesh Si tratta di una famiglia di topologie, più che di una topologia sola, ma in generale, ogni interfaccia di rete è connessa direttamente con più interfacce, al limite tutte le altre. Viene utilizzata per cluster di computer per il calcolo parallelo.

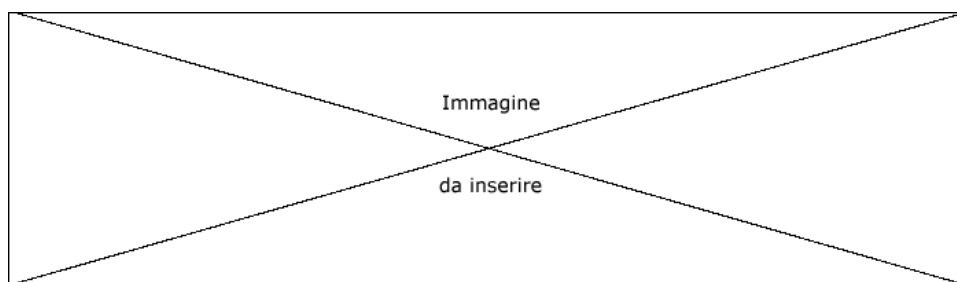


Figura 16.4: Schematizzazione di una rete di tipo *mesh*.

Bus Le interfacce sono connesse tutte ad un unico cavo principale (il bus) tramite un altro cavo che è una derivazione di quello principale.

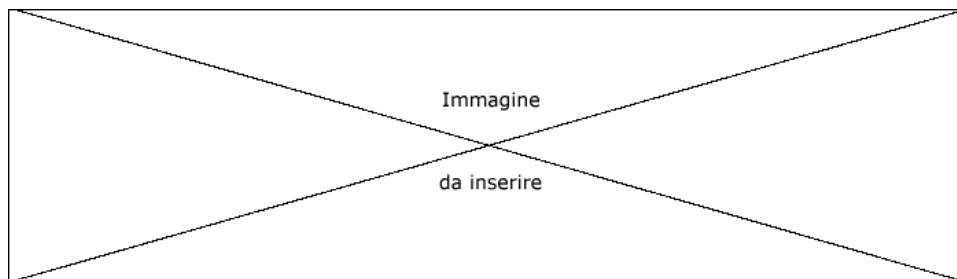


Figura 16.5: Schematizzazione di una rete di tipo *bus*.

Anello Le interfacce sono connesse tutte in maniera tale da formare fisicamente un circuito chiuso su sé stesso (anello). Ad ogni interfaccia sono collegati fisicamente due cavi (uno di ingresso e l'altro di uscita), pertanto l'interfaccia stessa costituisce l'elemento di continuità dell'anello stesso.

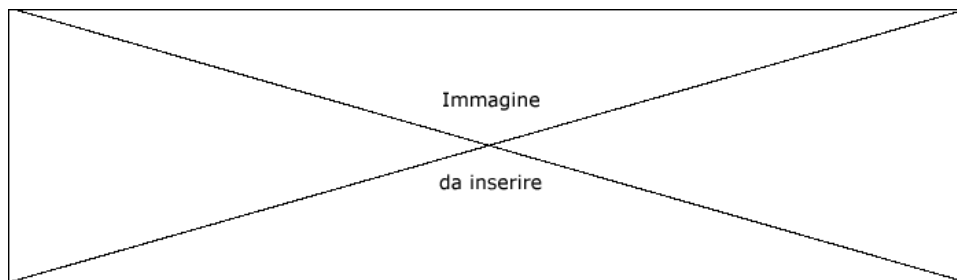


Figura 16.6: Schematizzazione di una rete di tipo *anello*.

Stella Le interfacce sono connesse tutte ad un concentratore (hub) che costituisce il *centro stella*.

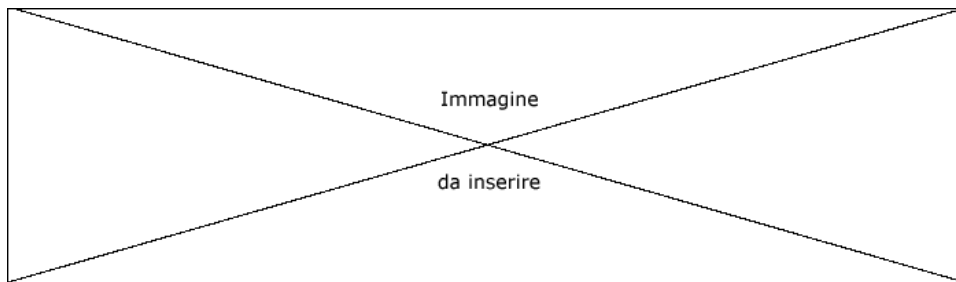


Figura 16.7: Schematizzazione di una rete di tipo *stella*.

Poiché, a parte il caso di *mesh*, i canali di comunicazione sono molto minori rispetto al numero di interfacce, è necessario stabilire come viene gestito l'accesso al canale, ossia il CAM (**Channel Access Method**). I metodi principali sono essenzialmente due: la *contesa* e il *token passing*.

Channel Access Method

Nel modello a *contesa*, le interfacce cercano di accedere al canale in maniera arbitraria, poi negoziano tra loro chi ha la precedenza. Lo svantaggio principale è che se il canale è molto trafficato la fase di contesa può essere lunghissima.

Nel modello a *token passing* (che si basa generalmente su reti con tipologia ad anello) sul canale è sempre presente un messaggio detto *token* (gettone), che può essere in due stati: libero o occupato. Se ad un'interfaccia arriva un token libero, l'interfaccia può occupare il canale inviando un messaggio oltre al token (il cui stato viene cambiato in occupato). Nel caso in cui l'interfaccia non debba comunicare niente o non possa farlo (perché trova il token occupato), passa il token e l'eventuale messaggio ricevuto all'interfaccia successiva. Quando il messaggio arriva all'interfaccia di destinazione, questa estrae il messaggio contenente le informazioni ad essa inviate ed inoltra il token all'interfaccia successiva con lo stato cambiato in libero.

Il tipo di *commutazione* di una rete indica come vengono gestite le trasmissioni tra un'interfaccia e l'altra. I due tipi principali sono i seguenti

Commutazione di circuito Tra l'interfaccia A e l'interfaccia B viene riservato un circuito, ossia un tratto di canale che le collega. Solo A e B possono utilizzarlo, finché non si conclude la chiamata. Questo è il sistema con cui funziona la rete telefonica fissa classica (PSTN). Lo svantaggio più evidente è che la linea è occupata per tutto il periodo di tempo in cui viene attivata una sessione di comunicazione tra A e B, anche se A e B non si scambiano nessuna informazione durante la sessione.

Commutazione di pacchetto L'interfaccia A condivide un canale con molte altre interfacce. Ogni interfaccia può inviare un messaggio avente una lunghezza massima fissata. Chiaramente ci deve essere una fase in cui i messaggi più lunghi vengono suddivisi in parti. Ogni interfaccia invia a turno un pacchetto contenente il messaggio da inviare o una parte di esso, realizzando una sorta di multiplexing. Lo svantaggio più grande è dato dal fatto che la banda a disposizione di un'interfaccia dipende dal traffico sulla rete e non è predicibile.

16.8.1 Ethernet

Ethernet II è lo standard di fatto utilizzato ad oggi per *Internet* (che viene privilegiato rispetto allo standard IEEE 802.3, sebbene tutte le schede Ethernet gestiscano entrambi gli standard, leggermente diversi).⁵ Si tratta della tecnologia più diffusa per il collegamento delle reti locali. Questa ha una modalità di accesso al canale a contesa, nota

⁵ Affinché un'interfaccia di rete sia compatibile con *Internet*, deve essere in grado di trasmettere pacchetti *Ethernet II* ed in grado di ricevere pacchetti *Ethernet II* e IEEE 802.3.

come CSMA/CD (Carrier Sense Multiple Access / Collision Detection). Qualunque interfaccia può impegnare il canale per iniziare una comunicazione se nessun altro sistema sta già comunicando (Multiple Access) (a tale scopo non c'è alcun impedimento fisico, ma soltanto un controllo di tipo logico sul segnale portante: Carrier Sense). Dopo aver impegnato il mezzo di comunicazione si deve essere sicuri che nessun'altra interfaccia abbia deciso di impegnarlo in contemporanea (o quasi). Il protocollo si basa sull'ascolto di eventuali collisioni (Collision Detection): se viene rilevata una collisione (due tentativi di comunicazione quasi contemporanei), la comunicazione viene interrotta per un tempo variabile (in maniera casuale) e quindi il tentativo di impegnare il canale viene effettuato nuovamente (il sistema che riprova per primo ha il diritto di comunicare). Questo approccio si porta dietro l'indeterminazione del tempo di trasmissione: non è possibile determinare a priori il tempo necessario per scambiarsi dei dati. Inoltre, a causa delle collisioni, c'è un limite al numero delle interfacce interconnesse con questo protocollo.

MAC address

Nella nomenclatura del protocollo Ethernet l'indirizzo fisico è detto **MAC address** (Media Access Control Address). Tale valore viene generalmente espresso come 6 coppie di cifre esadecimali separate dal simbolo ':'. Ad esempio 00:E0:98:36:1B:DF è la rappresentazione di un *MAC address*.

Secondo lo standard IEEE 802 infatti, il livello Data Link (Control) del modello ISO OSI, riferito anche come DLC, è suddiviso in due livelli: il Logical Link Control (LLC) ed il Media Access Control (MAC).

Per le reti che non rispettano lo standard IEEE 802 ma si riferiscono al modello ISO OSI, si parla di *DLC Address* (Data Link Control Address).

frame

Nella specifica *Ethernet II* i pacchetti, detti **frame**, hanno la struttura riportata in fig. 16.8.

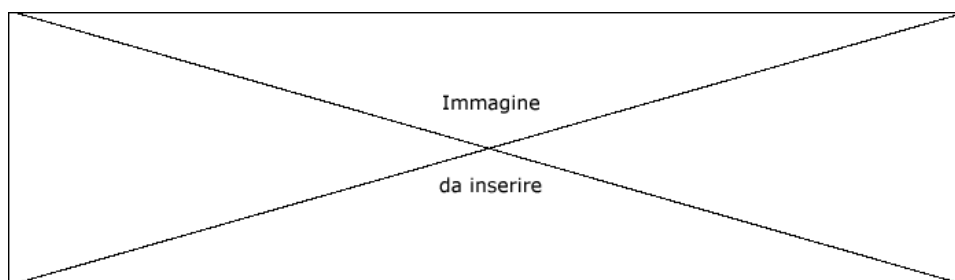


Figura 16.8: Rappresentazione di un *frame* Ethernet.

preambolo

Ogni *frame* è preceduto da un segnale, detto **preambolo** (*preamble*) o *start of frame*, una sequenza di bit che serve a sincronizzare il ricevitore. È costituito da 64 bit (8 byte) che hanno valori alternati di 0 e 1 e gli ultimi due bit sono posti ad 1.

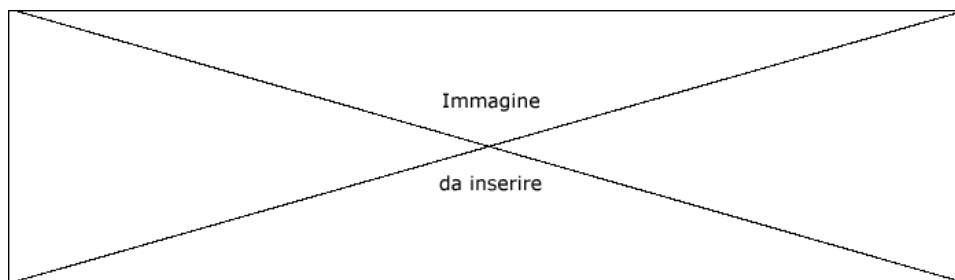


Figura 16.9: Rappresentazione di un *preambolo* Ethernet.

Dest. Address è il MAC address dell'interfaccia destinataria del frame.

Source Address è il MAC address dell'interfaccia mittente del frame.

Type indica il tipo di dati (il protocollo utilizzato) incapsulati nel campo Data. Questo serve in quanto il driver della scheda di rete dovrà passare il pacchetto al gestore del protocollo opportuno (demultiplexing). I tipi più utilizzati dallo stack TCP/IP sono 0800 (IP), 0806 (ARP) e 8035 (RARP).

Data è il payload del frame che può contenere da 46 fino a 1500 byte.

FCS è il Frame check Sequence, ovvero un valore utilizzato per il rilevamento di eventuali errori di trasmissione. Infatti tale valore viene generato con operazioni aritmetico-logiche sulle informazioni presenti nel frame. Se il ricevitore riceve un valore nel FCS che è diverso da quello che risulterebbe dai dati ricevuti, il frame ricevuto ha subito qualche modifica durante il trasporto sul canale.

La topologia fisica a bus fa sì che tutte le interfacce ricevano tutti i pacchetti. Nel caso in cui il MAC address di destinazione non coincida con il proprio, il frame viene scartato e non viene passato ai livelli superiori. Esiste inoltre un indirizzo particolare, FF:FF:FF:FF:FF:FF, che è il **broadcast** di Ethernet; nel caso che venga posto come indirizzo di destinazione, tutte le interfacce si riterranno destinatarie del frame ricevuto.

broadcast

Esiste anche una modalità di funzionamento dell'interfaccia di rete, detta **modo promiscuo** (*promiscuous mode*) in cui il comportamento della scheda è quello di passare ai livelli superiori tutti i pacchetti che essa riceve, anche quelli che non sono indirizzati ad essa.

modo promiscuo

Si noti che la minima lunghezza di un frame Ethernet è 64 byte. Un frame più corto di 64 byte è detto **runt** e non è considerato valido.

runt

Il throughput delle reti *Ethernet* può essere 10 Mbit/s, 100 Mbit/s, 1 Gbit/s o 10 Gbit/s.

16.8.2 Token ring

Il protocollo *Token ring* è stato sviluppato da IBM e si trova principalmente all'interno dei suoi sistemi. È stato poi standardizzato da IEEE nella specifica 802.5. Si basa su un modello di accesso di tipo token passing, su una topologia logica ad anello ed una fisica a stella. Soltanto il sistema che ha ricevuto il token vuoto può iniziare una comunicazione inviando il token ed i dati da trasmettere al sistema adiacente, il quale riceverà il pacchetto, controllerà se i dati sono per lui ed in caso affermativo li preleverà rispeditendo poi il token al sistema successivo. In caso negativo ripasserà il token più i dati al sistema successivo. I frame token ring sono di tre tipi diversi: il token, il frame di abort e il frame di dati/comandi.

???

Il throughput delle reti *Token ring* può essere 4 Mbit/s, 16 Mbit/s o 100 Mbit/s. La tecnologia token ring è in generale più costosa della *Ethernet*.

16.8.3 Protocolli Punto-punto

I protocolli punto-punto servono per connettere tra loro due interfacce di rete, quindi non fanno a rigore parte dei protocolli di rete. In genere vengono usati per le connessioni dial-up attraverso modem. Esistono tre protocolli di questo tipo: **SLIP** (Serial Line Internet Protocol), **CSLIP** (Compressed Serial Line Internet Protocol) e **PPP** (Point to Point Protocol). Tali protocolli sono nati per implementare il livello di collegamento quando non esiste la scheda di rete vera e propria, simulandola tramite un collegamento via modem.

SLIP
CSLIP
PPP

??? cos'è un modem e come funziona ???

Lo SLIP è un protocollo piuttosto semplice: mette un carattere di controllo in testa ed in coda al pacchetto da inviare e quindi controlla che nel pacchetto non vi siano altre occorrenze del carattere utilizzato per delimitare l'inizio e la fine del frame. Se vi sono occorrenze di tale carattere, ogni occorrenza viene fatta precedere dal carattere ESC e

quindi tutte le occorrenze del carattere ESC vengono a loro volta raddoppiate. Quindi invia il frame sulla linea seriale. Ha alcuni difetti, tra cui:

- Ognuno dei due nodi deve conoscere a priori il proprio indirizzo IP.
- Trasporta solo il protocollo IP.

Il CSLIP è analogo allo SLIP ma è più efficiente perché se non è cambiato niente dall'ultimo frame, non ritrasmette l'header.

Il PPP risolve i problemi dello SLIP. È composto da due protocolli:

LCP (Link Control Protocol) è il protocollo che stabilisce e configura la connessione. Permette di negoziare varie opzioni tra i due nodi, tra cui ad esempio lo scambio di password. In particolare sono previsti il PAP (Password Authentication Protocol) ed il CHAP (Challenge-Handshake Authentication Protocol).

NCP (Network Control Protocol) è una famiglia di protocolli, ognuno specifico per il protocollo di livello superiore considerato (c'è l'NCP per l'IP, quello per AppleTalk, ...), che specifica come viene trasportato il protocollo di livello superiore.

16.8.4 MTU

MTU

La **MTU** (Maximum Transmission Unit) di una rete è la lunghezza massima del pacchetto che vi può transitare. Questa generalmente dipende dal protocollo del livello link. Alcuni valori tipici di MTU sono riportati in tab. 16.1.

Protocollo	MTU
Token Ring 16 Mbit/s	17914
Token Ring 4 Mbit/s	4464
Ethernet	1500
PPP	< 1500

Tabella 16.1: MTU di alcuni protocolli.

I pacchetti con dimensioni maggiori della MTU vengono suddivisi in frammenti (*fragments*), ovvero in pacchetti più piccoli dai router incontrati durante il percorso ed il pacchetto viene poi ricomposto dall'interfaccia di rete di destinazione.

16.9 I dispositivi di rete

I dispositivi di rete sono tutte le apparecchiature che rendono possibile la comunicazione in rete. Di seguito è riportato un elenco dei dispositivi di rete più comuni.

16.9.1 Hub

hub

Un **hub** è un sistema che lavora a livello 1 e permette di centralizzare il cablaggio della rete. Quando la topologia fisica di una rete è a stella, il centro stella è in genere costituito da un *hub*. Dipendentemente dalla qualità dell'hub, questo può presentare anche una porta AUI che è uno standard per l'interfacciamento tra il livello 1 ed il livello 2. Pertanto a tale porta può essere collegato qualunque transceiver che permette di adattare l'hub a qualunque tipo di cablaggio di rete si desidera: connettore BNC, RJ, fibra ottica, ...

Alcuni tipi di hub per le reti Ethernet sfruttano un meccanismo, detto *autonegoziazione*, tramite il quale possono capire a che velocità può trasmettere una periferica connessa ad una loro porta e quindi gestire sia schede di rete a 10 Mb/s che a 100 Mb/s. Questi dispositivi sono detti **switching hub**.

switching hub

16.9.2 Bridge

bridge

Un **bridge** è un sistema che lavora a livello 2 ed ha il compito di connettere tra loro due segmenti di rete. Ascolta i messaggi che provengono dalla rete A, scarta quelli non validi (ad esempio i *runt*), legge l'indirizzo di destinazione e li inoltra solo se sono diretti sulla rete B. Lo stesso fa per i messaggi che provengono dalla rete A. In questa maniera il traffico nelle due reti rimane separato, per quanto possibile. Nel caso di Ethernet si dice che si hanno due *domini di collisione* separati.

I bridge Ethernet sono detti trasparenti, in quanto le due reti funzionano come una sola e le macchine non hanno bisogno di sapere che il bridge è presente. Infatti il bridge riesce a capire quali macchine sono sulla rete A e quali sulla rete B con un meccanismo chiamato *backward learning*. Quando la macchina 1 manda un messaggio, il bridge sente su quale porta arriva e quindi capisce che la macchina 1 sta sulla rete A. Dopo sufficiente tempo, ha una mappa completa della disposizione delle stazioni.

Un bridge può anche connettere due reti con tecnologie diverse (ad esempio Ethernet e Token ring). In questo caso si parla di *translational bridge*, in quanto il bridge deve anche tradurre i pacchetti nel formato adatto.

16.9.3 Switch

Uno **switch** è un sistema che lavora a livello 2, simile ad un *hub* ma è dotato di intelligenza per ottimizzare la comunicazione sulla rete: memorizza in una sua cache il MAC address delle interfacce connesse ad ogni connettore in modo da far diminuire drasticamente le collisioni, in quanto un pacchetto inviato da un'interfaccia verso un'altra interesserà soltanto il connettore del destinatario, gli altri non si accorgeranno neanche della comunicazione tra i due.

switch

16.9.4 Router

Un **router** è un sistema che svolge il compito di instradare i messaggi ricevuti, verso un'altra rete in modo da farli arrivare a destinazione. Essenzialmente si tratta di un sistema che lavora a livello 3 tra due o più reti diverse, in modo da riconoscere chi è il destinatario del pacchetto (e questo avviene proprio a livello 3, tramite l'indirizzo IP) e quindi instradarlo verso la rete opportuna, determinata in base ad una logica descritta da una *routing table* che informa il sistema di inviare i pacchetti con determinati indirizzi di destinazione su una certa rete e certi altri su di un'altra.

router

Un router ben configurato non dovrebbe "routare" il traffico broadcast (in modo da non sovraccaricare la rete) tranne quello diretto ad una specifica applicazione (porta).

16.9.5 Gateway

Un **gateway** è un sistema che lavora a livello 7 ed ha il compito di effettuare operazioni più complesse di un *router* che sono quelle relative alla conversione di protocollo di sessione/trasporto. Per esempio potrebbe capitare di inviare un pacchetto da una rete secondo il protocollo TCP/IP ad un'interfaccia che sta su una rete che utilizza un altro protocollo. A questo punto, per far comunicare le due reti è necessario che vi sia un gateway che, oltre a instradare correttamente i pacchetti, effettui anche la necessaria conversione di protocollo.

gateway

16.10 Riferimenti

- Ricerca di RFC
<http://www.ietf.org/rfc>

???

Capitolo 17

TCP/IP - Network

“Gli ostacoli sono quelle cose spaventose che vedi quando togli gli occhi dalla meta.”
– H. Ford

Come già accennato in precedenza, l'insieme dei protocolli più diffuso per la comunicazione di rete è il TCP/IP o meglio l'Internet suite, il cui protocollo a livello più basso (livello di trasporto) è l'IP (Internet Protocol). Attualmente è in uso la versione 4 (IPv4), ma esiste già una nuova versione, la 6 (IPv6 o IPng - IP **n**ext **g**eneration), che per il momento non ha un largo uso.

17.1 Network byte order

È importante sottolineare il fatto che i bit vengono trasmessi sulla rete nell'ordine in cui si trovano nei loro rispettivi pacchetti. Tale sistema di gestione dei byte è detto **network byte order**.

network byte order

Le informazioni possono essere memorizzate all'interno della memoria centrale in due modalità differenti:

big endian

le informazioni, la cui memorizzazione occupa più di un byte (word, double word) sono memorizzate in celle consecutive della memoria, in maniera tale che il byte più significativo dell'informazione viene memorizzato nella cella di memoria con indirizzo più basso, mentre quello meno significativo viene memorizzato nella cella di memoria con indirizzo più alto. Se si osserva il contenuto della memoria centrale con indirizzi crescenti, i byte appaiono memorizzati nell'ordine canonico.

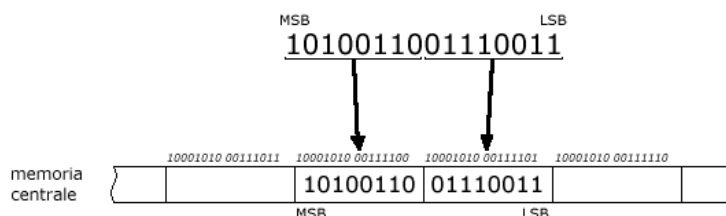


Figura 17.1: La modalità di memorizzazione *big endian*.

little endian

le informazioni, la cui memorizzazione occupa più di un byte (word, double word) sono memorizzate in celle consecutive della memoria, in maniera tale che il byte più significativo dell'informazione viene memorizzato nella cella di memoria con indirizzo più alto, mentre quello meno significativo viene memorizzato nella cella di memoria con indirizzo più basso. Se si osserva il contenuto della memoria

centrale con indirizzi crescenti, i byte appaiono memorizzati al rovescio (i bit all'interno di ogni byte sono comunque nell'ordine canonico). Questa modalità di memorizzazione delle informazioni è utilizzata su architetture basate su X386.

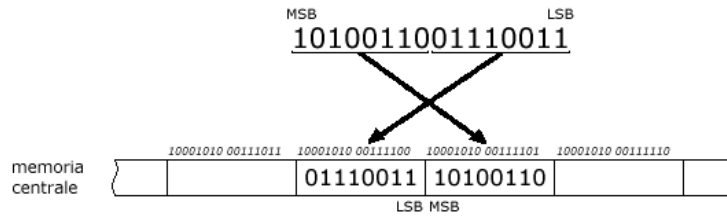


Figura 17.2: La modalità di memorizzazione *little endian*.

Sui PC (architetture *Intel* X386 e compatibili) e sulle piattaforme *Digital*, le informazioni sono memorizzate secondo la modalità *little endian*, mentre tutte le altre piattaforme (*Motorola*, *IBM*, *Sun*) utilizzano la modalità di memorizzazione delle informazioni *big endian*.

Esistono anche dei processori che permettono di passare da un tipo di memorizzazione all'altro. Comunque GNU/Linux adotta il tipo di memorizzazione (o **endianess**) tipico dell'architettura sulla quale è installato e mantiene sempre quello. È comunque possibile controllare il tipo di *endianess* utilizzata dal proprio sistema, attraverso un semplice programma, analogo a quello di cui di seguito è riportato il contenuto del file sorgente:

endianess

```
#include <stdio.h>

int endianess(void)
/* Return value:
 * 0 : big endian
 * 1 : little endian
 */
{
    short int TestValue, RebuiltValue;
    char *pValue;

    TestValue = 0xABCD;
    pValue = (char*)&TestValue;
    RebuiltValue = (pValue[1] << 8) + (0x00ff & pValue[0]);
    return (TestValue == RebuiltValue);
}

main()
{
    if (endianess() == 0)
        printf("Il sistema utilizza big endian.\n");
    else
        printf("Il sistema utilizza little endian.\n");
}
```

Una volta salvato in un file, ad esempio `testend.c`, il programma può essere compilato con il comando

```
$ gcc testend.c -o testend
```

e quindi fatto eseguire. Un esempio è riportato di seguito

```
[daniele@Zeus ~]$ ./testend
```



```
Il sistema utilizza little endian.
[daniele@Zeus ~]$ _
```

Poiché il network byte order è analogo alla memorizzazione big endian, le informazioni trasmesse dai PC basati su architettura *Intel X386* o compatibili devono essere quindi opportunamente trattate prima di poter essere inviate sulla rete, ovvero le informazioni che costituiscono la parte di controllo del messaggio (header) deve essere trasformata da little endian in network byte order.

17.2 IP - Internet Protocol (IPv4)

Il protocollo IP è specificato dalla RFC 791 e rappresenta il protocollo di base dello stack TCP/IP (*Internet suite*). Tale protocollo fornisce un servizio di consegna dei pacchetti, detti **IP datagram** (v. sez. 17.2.5), inaffidabile (*unreliable*), che esegue le operazioni necessarie al trasporto dell'informazione (*best-effort*), senza connessione (*connectionless*).

Per inaffidabile si intende che il protocollo, sebbene gestisca a dovere la trasmissione dei pacchetti (best-effort service), non gestisce alcun meccanismo che garantisca che un pacchetto IP arrivi correttamente a destinazione. Se qualcosa va storto, il protocollo IP si limita a scartare il pacchetto ricevuto e tenta di inviare un messaggio ICMP (v. sez. 17.6) al mittente per informarlo dell'accaduto. Qualsiasi tipo di meccanismo che garantisca il recapito dei pacchetti deve essere implementato ad un livello superiore (ad esempio a livello di trasporto). Il protocollo IP è inoltre senza connessione, ovvero non mantiene nessuna informazione relativa allo stato della comunicazione tra due interfacce: ogni pacchetto viene gestito indipendentemente dagli altri. Potrebbe succedere che due pacchetti (pacchetto A e pacchetto B) inviati in sequenza giungano a destinazione nella sequenza inversa (pacchetto B, pacchetto A) per il fatto che ognuno di essi può seguire percorsi diversi nella rete, dipendentemente dal meccanismo di routing utilizzato dai router e dal traffico presente sulle varie parti della rete. L'ordine dei pacchetti non viene ripristinato dal protocollo IP, ma anche questo è compito dei protocolli di livello superiore.

17.2.1 Gli indirizzi

Ogni interfaccia di rete è identificata, secondo IPv4, da un numero a 32 bit detto **indirizzo IP**, che ovviamente deve essere univoco per ogni interfaccia presente su una determinata rete.¹ La rappresentazione di un indirizzo IP avviene in genere secondo la notazione **decimal dotted notation**, che si ottiene suddividendo il numero binario corrispondente all'indirizzo IP in gruppi da 8 bit e rappresentando i corrispondenti valori decimali, separati l'uno dall'altro per mezzo di un punto '.'. Per esempio, l'indirizzo IP 12345678_H è rappresentato, secondo la *decimal dotted notation*, da 18.52.86.120, infatti 12_H = 18, 34_H = 52, 56_H = 86 e 78_H = 120.

indirizzo IP

decimal dotted notation

Ogni indirizzo IP è suddiviso in due parti logiche: la prima, che costituisce la parte più significativa dell'indirizzo, è detta **net id** (parte rete) e la seconda, quella che costituisce la parte meno significativa dell'indirizzo, è detta **host id** (parte host o parte interfaccia).

net id

host id

??? Fig. indirizzi IP (net - host) ???

È possibile gestire gli indirizzi IP relativi alle interfacce di rete presenti su un sistema GNU/Linux tramite il comando **ifconfig** (v. sez. 20.3).

¹si tenga a mente che l'*indirizzo IP* è assegnato ad una scheda di rete, non ad una macchina (una macchina può montare anche più schede di rete).

Le classi di indirizzi

Gli indirizzi IP sono suddivisi in classi. La suddivisione è nata dall'esigenza di assegnare indirizzi IP univoci per le macchine connesse permanentemente ad *Internet* (rete geografica mondiale). Tali classi definiscono una gerarchia di indirizzi IP, secondo quanto elencato di seguito.

Classe A L'indirizzo IP ha il bit 31 (MSB) posto a 0. I 24 bit meno significativi sono utilizzati per rappresentare la parte interfaccia, mentre i rimanenti 8 sono utilizzati per rappresentare la parte rete.

??? figura classe A ???

Quindi esisteranno al massimo $2^7 = 128$ reti di *classe A* (in quanto il bit più significativo è fissato a 0) ad ognuna delle quali potranno essere collegate al massimo $2^{24} = 16777216$ interfacce di rete. Tali valori saranno quindi compresi tra 0.0.0.0 e 127.255.255.255.

Classe B L'indirizzo IP ha il bit 31 (MSB) posto ad 1 ed il bit 30 posto a 0. I 16 bit meno significativi sono utilizzati per rappresentare la parte interfaccia, mentre i rimanenti 16 sono utilizzati per rappresentare la parte rete.

??? figura classe B ???

Quindi esisteranno al massimo $2^{14} = 16384$ reti di *classe B* (in quanto i due bit più significativi sono fissati a $(10)_2$) ad ognuna delle quali potranno essere collegate al massimo $2^{16} = 65536$ interfacce di rete. Tali valori saranno quindi compresi tra 128.0.0.0 e 191.255.255.255.

Classe C L'indirizzo IP ha i bit 31 (MSB) e 30 posti ad 1 ed il bit 29 posto a 0. Gli 8 bit meno significativi sono utilizzati per rappresentare la parte interfaccia, mentre i rimanenti 24 sono utilizzati per rappresentare la parte rete.

??? figura classe C ???

Quindi esisteranno al massimo $2^{21} = 2097152$ reti di *classe C* (in quanto i tre bit più significativi sono fissati a $(110)_2$) ad ognuna delle quali potranno essere collegate al massimo $2^8 = 256$ interfacce di rete. Tali valori saranno quindi compresi tra 192.0.0.0 e 223.255.255.255.

Classe D L'indirizzo IP ha i bit 31 (MSB), 30 e 29 posti ad 1 ed il bit 28 posto a 0. Non c'è suddivisione tra parte rete ed interfaccia, ma l'intero indirizzo è utilizzato per il *multicast*. Tali valori saranno quindi compresi tra 224.0.0.0 e 239.255.255.255.

??? figura classe D ???

Classe E L'indirizzo IP ha i bit 31 (MSB), 30, 29 e 28 posti ad 1 ed il bit 27 posto a 0. Tali indirizzi sono riservati per sperimentazioni varie. Tali valori saranno quindi compresi tra 240.0.0.0 e 247.255.255.255.

??? figura classe E ???

Classe	Intervallo di indirizzi
A	0.0.0.0 - 127.255.255.255
B	128.0.0.0 - 191.255.255.255
C	192.0.0.0 - 223.255.255.255
D	224.0.0.0 - 239.255.255.255
E	240.0.0.0 - 247.255.255.255

Tabella 17.1: Le classi degli indirizzi IP.

Gli indirizzi validi per *Internet* (detti anche *indirizzi pubblici*) sono forniti dall'organismo IANA (Internet Assigned Numbers Authority) che si preoccupa di garantirne l'univocità.

Gli indirizzi riservati

All'interno dell'insieme dei valori utilizzabili per gli indirizzi IP, da 0.0.0.0 a 223.255.255.255, esistono dei particolari indirizzi che sono riservati e quindi non utilizzabili come indirizzi IP per *Internet*. Tali valori sono utilizzati come indirizzi IP di interfacce di rete presenti su LAN private, che utilizzano il protocollo TCP/IP ma non sono direttamente connesse ad internet (*indirizzi privati*). Di tali indirizzi ce ne sono per ogni classe:

Indirizzi	Classe
10.X.X.X	A
172.16.X.X - 172.31.X.X	B
192.168.X.X	C

Tabella 17.2: Gli indirizzi IP privati.

Sono da considerarsi riservati anche gli indirizzi 127.X.X.X che individuano la rete di **loopback**, ovvero una rete fittizia costituita dall'interfaccia di rete stessa. Inviando le informazioni ad uno di tali indirizzi IP, queste transiteranno dal livello D al livello B dello stack TCP/IP e quindi, riconosciuto il particolare indirizzo di destinazione (*loopback*), le informazioni ritorneranno fino al livello D e saranno trattate dall'applicazione come appena ricevuti.

Inoltre, non sono considerati indirizzi IP validi quelli che hanno un *net id* od un *host id* costituiti da tutti bit a 0 o ad 1. Per esempio, non sono indirizzi IP validi i seguenti: 0.12.123.43, 15.0.0.0, 17.255.255.255, 131.10.0.0, 192.167.3.255.

Gli indirizzi IP con tutti i bit dell'*host id* impostati ad 1 sono detti **directed broadcast**, ovvero sono utilizzati come indirizzo IP di destinazione quando si deve inviare un messaggio a tutte le interfacce di una determinata rete: per esempio, un messaggio con indirizzo IP di destinazione 192.167.58.255 sarà inviato a tutte le interfacce presenti sulla rete 192.167.58.X.

Gli indirizzi IP con tutti i bit dell'*host id* a 0 sono utilizzati per identificare la rete nel suo complesso: per esempio, l'indirizzo IP 192.167.58.0 identifica la rete 192.167.58.X (viene utilizzato nelle routing tables).

???

17.2.2 CIDR

Successivamente si è pensato di svincolare gli indirizzi dal concetto di classe, per due motivi principali:

- Molti utilizzatori di *Internet* (enti o aziende) avevano bisogno di un numero di Indirizzi IP da assegnare alle interfacce di rete dei computer connessi su *Internet* superiori a quelli gestibili con una rete di classe C (254), ma generalmente inferiori a quelli gestibili con una rete di classe B (65534). Tipicamente si facevano assegnare una rete di classe B ma molti degli indirizzi possibili rimanevano inutilizzati. Ciò ha portato ad una rapida diminuzione degli indirizzi di classe B.
- Le tabelle di instradamento (routing tables) dei router delle backbones (linee principali di *Internet*) erano diventate enormi e difficili da mantenere, in quanto contenevano un numero enorme di indirizzi di rete, di cui molti erano relativi ad una stessa organizzazione o azienda, che magari si era fatta assegnare più reti di classe C.

Il **CIDR** (Classless Inter-Domain Routing) è il metodo utilizzato per indicare gli indirizzi IP, che ha soppiantato l'utilizzo delle classi. Questo si basa sui concetti di *subnet mask* o di *net prefix*, che indicano quanti sono i bit dell'indirizzo IP che identificano il *net id* e premettono di avere una gestione molto più flessibile dello stesso.

La **subnet mask** è una sequenza di 32 bit (la stessa lunghezza degli indirizzi IP) ognuno dei quali indica se considerare i corrispondenti bit dell'indirizzo IP come facenti parte del *net id* (1) oppure facenti parte dell'*host id* (0). Ad esempio ...

```

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
+-----+-----+
                        net id                        host id

```

192.167.58.147/255.255.240.0

net prefix

Allo stesso modo è possibile utilizzare il **net prefix** che è un numero che indica il numero di bit di un indirizzo IP che fanno parte del net id.

192.167.58.147/20

Gli indirizzi IP vengono quindi specificati postponendo all'indirizzo vero e proprio la relativa subnet mask od il relativo net prefix.

In questo modo è possibile, ad esempio, creare una rete con 510 indirizzi ($2^9 - 2$), semplicemente scegliendo il prefisso 23. Contemporaneamente si riducono le routing tables, perché le reti che prima erano formate da più reti di classe C, possono essere realizzate con un'unica rete con prefisso minore di 24 in modo da utilizzare un solo riferimento (un unico record) nelle routing tables.

??? esempi ???

???

17.2.3 Il routing

Il meccanismo che decide il tragitto dei pacchetti dall'interfaccia mittente a quella di destinazione è detto *meccanismo di routing* o più semplicemente **routing**.

routing

L'IP utilizza un meccanismo di routing piuttosto semplice che riesce a trasferire gli IP datagram da un'interfaccia ad un'altra per passi successivi, detti **hop** (salti). La logica è la seguente: se il destinatario si trova sulla stessa rete (subnet) del mittente, l'IP datagram viene passato al livello sottostante (livello link), che si occuperà di effettuare l'inoltro sulla rete e l'interfaccia destinataria può così riceverlo. Nel caso in cui il destinatario non si trovi sulla stessa rete del mittente, l'IP datagram viene inviato ad un'altra macchina (*router*) (nell'IP datagram viene sostituito l'indirizzo IP di destinazione con quello del router, ma il MAC address rimane quello relativo all'interfaccia di destinazione reale), sulla quale ricade l'onere di consegnare il messaggio ad un'altra interfaccia che può essere quella di destinazione od un altro router, e così via. Se i router sono ben programmati, l'IP datagram arriverà a destinazione.

hop

Per ogni macchina è definita una tabella di instradamento o **routing table**, che contiene gli indirizzi IP delle macchine a cui inviare gli IP datagram la cui interfaccia di destinazione appartiene ad una subnet diversa da quella mittente. In tale tabella saranno presenti delle righe che esprimono regole del tipo: se l'indirizzo IP di destinazione è XXX.XXX.XXX.XXX invia l'IP datagram a YYY.YYY.YYY.YYY.

routing table

Il meccanismo di routing applicato dall'IP osserva nell'ordine i seguenti passi

1. Se nella *routing table* è presente l'indirizzo IP completo dell'interfaccia di destinazione, esegue la regola relativa.
2. Se nella *routing table* è presente l'indirizzo IP della rete (subnet) di cui fa parte l'indirizzo di destinazione, esegue la regola relativa.
3. Se nella *routing table* è presente l'indirizzo della *default route*, esegue la regola relativa.

Se nessun passo funziona, l'interfaccia non è in grado di recapitare l'IP datagram.

???

La routing table di una macchina GNU/Linux viene gestita con il comando **route** (v. sez. 20.4). Un esempio di routing table è riportato di seguito.

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.1.0	*	255.255.255.0	U	0	0	0	eth0
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
default	192.168.1.1	0.0.0.0	UG	0	0	0	eth0

Si tratta di una macchina con una interfaccia di rete il cui indirizzo IP è 192.168.1.10, il cui router di default sia 192.168.1.1. Il significato dell'output di **route** è il seguente

Destination

è l'indirizzo IP di destinazione del pacchetto, che può essere un indirizzo di un host, l'indirizzo di una rete (quindi tutti i pacchetti indirizzati ad un host di questa rete soddisferanno questa regola), oppure **default**, ossia tutti i pacchetti che non soddisfano le regole precedenti;

Gateway

nel caso che ci sia bisogno di un router per instradare alcuni pacchetti, questo campo contiene l'indirizzo del router. Ovviamente tale macchina deve essere raggiungibile (sulla stessa rete);

Genmask

rappresenta la subnet mask relativa all'indirizzo visualizzato nella colonna *Destination*;

Flags in questo campo appaiono delle lettere che hanno il significato riportato in tab. 17.3;

Flag	Significato
U	il collegamento è attivo (Up)
H	la destinazione è un'interfaccia di rete (Host)
G	il destinatario è raggiungibile attraverso un router, il cui indirizzo IP è specificato nel campo Gateway
R	(reinstall route for dynamic routing)
D	(dynamically installed by daemon or redirect)
M	(modified from routing daemon or redirect)
A	(installed by addrconf)
C	(cache entry)
!	scarta gli IP datagram da inviare all'indirizzo IP specificato nel campo Destination

Tabella 17.3: Possibili flag dell'output del comando **route**.

Iface è l'interfaccia di rete attraverso la quale l'IP datagram deve essere inviato.

Le tabelle di instradamento di un router possono essere molto più complesse di quelle di una macchina che non instrada pacchetti (non funge da router). Inoltre esiste la possibilità di aggiornare automaticamente la tabella tramite dei protocolli di routing dinamico (RIP, OSPF, BGP, ...). Le problematiche legate a questi protocolli esulano dagli scopi di questo testo.

Un router è una macchina sulla quale sono installate almeno due interfacce di rete e che effettua l'instradamento dei pacchetti, ovvero sulla macchina è attivata la caratteristica di *IP forwarding*, che viene impostata inserendo il valore '1' nel file `/proc/sys/net/ipv4/ip_forward`. Questo può essere fatto con il comando

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

???

17.2.4 Unicast e multicast

In genere la trasmissione delle informazioni avviene mediante l'invio di pacchetti IP che contengono l'indirizzo del mittente e del destinatario. Quindi ogni pacchetto viene inviato da un'interfaccia ad un'altra.

Il **multicast** è una tecnica di invio di informazioni ad un gruppo di interfacce di rete in maniera simultanea (ad esempio trasmissioni video su Internet). In genere la trasmissione delle informazioni avviene da un mittente ad un destinatario e quindi per inviare la stessa informazione a più destinatari sarebbe necessario inviare la stessa informazione

multicast

tante volte quante sono le interfacce di rete destinatarie. La tecnica multicast ovvia a questo problema permettendo di inviare una sola volta (o qualcuna di più per essere sicuri di non perdere pacchetti durante il percorso a causa del rumore) le informazioni per tutti i destinatari. Gli indirizzi IP multicast sono indirizzi assegnati ad un'interfaccia di rete per un numero finito di trasmissioni. Tale tipo di trasmissione arriva fino a livello C per tutte le interfacce coinvolte: scegliendo indirizzi IP specifici, riservati a questo scopo, non si corre il rischio di far lavorare anche le interfacce di rete non coinvolte nella comunicazione.

???

17.2.5 L'IP datagram

IP datagram

Il protocollo IP prevede la trasmissione di pacchetti, detti **IP datagram**, costituiti dai seguenti campi (v. fig. ??)

??? figura IP datagram (IPv4) ???

Version

(4 bit) è un numero che indica la versione del protocollo IP utilizzata (attualmente è utilizzata al versione 4);

Header length

(4 bit) è un numero che indica il numero di double-word presenti nell'header dell'IP datagram (incluse le opzioni). Tale campo può indicare al massimo una lunghezza dell'header di $15 \times 4 = 60$ byte, Nel caso in cui non siano presenti opzioni, tale campo contiene il valore 5 (header composto da 20 byte);

TOS (Type Of Service)

(8 bit) indica la precedenza da assegnare all'IP datagram. Ognuno dei bit corrisponde ad una specifica indicazione di precedenza secondo quanto riportato in tab. 17.4, anche se tale funzionalità è ignorata dalla maggior parte delle odierne implementazioni dello stack TCP/IP;

Bit	Significato	Descrizione
0	Unused	Deve essere 0
1	Minimize monetary cost	Indica di utilizzare il percorso che minimizza i costi in termini economici.
2	Maximize reliability	Indica di utilizzare il percorso che offre maggior garanzie di recapito dell'informazione.
3	Maximize throughput	Indica di utilizzare il percorso che offre la maggior velocità di trasferimento.
4	Minimize delay	Indica di utilizzare il percorso che minimizza il ritardo di comunicazione.
5	Unused	Deve essere 0
6	Unused	Deve essere 0
7	Unused	Deve essere 0

Tabella 17.4: I bit del campo TOS.

Se il campo contiene il valore 0, indica un servizio standard. L'utilizzo dei bit del campo TOS da parte di alcune applicazioni (protocolli di livello superiore) è riportata in tab. 17.5.

Total length

(16 bit) è un numero che indica la lunghezza in byte dell'intero IP datagram (quindi al massimo un IP datagram può essere composto da 65535 byte);

Identification

(16 bit) è un numero che identifica univocamente ogni IP datagram inviato da un'interfaccia di rete. Generalmente viene incrementato di una unità ogni volta che viene inviato un nuovo IP datagram;

Flag (3 bit) ???

Applicazione	Min. delay	Max. throughput	Max. reliability	Min. monetary cost
telnet	1	0	0	0
FTP (control)	1	0	0	0
FTP (data)	0	1	0	0
SMTP (command)	1	0	0	0
SMTP (data)	0	1	0	0
NNTP	0	0	1	0
SNMP	0	0	0	1

Tabella 17.5: Le applicazioni principali ed il loro utilizzo del campo TOS.

Fragment offset

(13 bit) ???

TTL (Time To Live)

(8 bit) è un numero che indica il numero massimo di hop (salti), ovvero il numero di routers che l'IP datagram può attraversare per arrivare a destinazione (inizializzato dal mittente ad un valore di default – generalmente 32 o 64). Man mano che l'IP datagram viene inoltrato da un router ad un altro, questo campo viene decrementato di una unità. Quando il valore raggiunge lo 0, l'IP datagram viene scartato dal router che ce l'ha in gestione in quel momento e viene generato un messaggio ICMP per informare il mittente dell'accaduto. Questo meccanismo elimina la possibilità che gli IP datagram rimangano in un anello di routers (mal configurati) per un tempo indeterminato;

La RFC 1009 specifica che un router che tiene un IP datagram per più di 1 secondo, decrementi il valore del campo TTL del numero di secondi che ha tenuto l'IP datagram, ma poche implementazioni di stack TCP/IP rispettano questa direttiva.

Protocol

(8 bit) è un numero che indica il tipo di protocollo di livello superiore (ICMP, IGMP, UDP o TCP) incapsulato nell'IP datagram;

Header checksum

(16 bit) è un numero di controllo generato sulla base del contenuto dell'header dell'IP datagram;

Source IP address

(32 bit) è un numero che indica l'indirizzo IP dell'interfaccia di rete mittente;

Destination IP address

(32 bit) è un numero che indica l'indirizzo IP dell'interfaccia di rete di destinazione;

In generale le opzioni non sono presenti, quindi l'header di un IP datagram può essere considerato lungo 20 byte.

???

17.3 IPv6

???

17.4 Frammentazione

Come già evidenziato nel cap. 16, se un pacchetto ha dimensioni maggiori di quelle consentite dallo strato di più basso livello, ovvero della MTU, questo viene scomposto in pacchetti più piccoli, detti **frammenti** (*fragments*). Quindi, se un IP datagram ha *frammenti*

dimensioni maggiori della MTU relativa alla rete che deve attraversare, viene scomposto in più IP datagram, ognuno dei quali contiene una parte del payload relativo all'IP datagram da inviare.

Il livello Network prima di inviare un IP datagram richiede all'interfaccia di rete di destinazione la sua MTU per poter eventualmente effettuare la frammentazione degli IP datagram.

Poiché un IP datagram può attraversare più reti prima di arrivare a destinazione, può darsi che la frammentazione avvenga anche sui routers intermedi perché magari la MTU si abbassa passando da una rete all'altra. Dunque è possibile che frammenti siano a loro volta frammentati.

L'IP datagram viene ricostruito ricomponendo i frammenti soltanto dall'interfaccia di rete di destinazione effettiva.

17.5 ARP - Address Resolution Protocol

Come è già stato illustrato nel cap. 16, ogni scheda di rete è identificata da un valore univoco a livello mondiale, detto **indirizzo fisico**, ed è quello utilizzato dalle schede per colloquiare al livello più basso. Gli indirizzi IP sono utilizzati dal protocollo IP che però è a sua volta incapsulato nel protocollo del livello più basso (Ethernet, Token ring, ...). La relazione esistente tra gli indirizzi fisici e gli indirizzi IP viene gestita con lo scambio di pacchetti relativi al protocollo **ARP** ed un'apposita memoria cache su ogni sistema. Tale memoria, detta anche **cache ARP**, contiene le informazioni relative agli indirizzi fisici ed ai relativi indirizzi IP per una durata di di tempo predefinita (default 20 minuti).

È possibile visualizzare la cache ARP relativa allo stack TCP/IP presente su di un sistema GNU/Linux, tramite il comando `arp` (man page `arp(8)`).

Comando: `arp`
 Path: `/sbin/arp`
 SINTASSI
`$ arp [option]`

DESCRIZIONE

option indica la modalità di funzionamento di `arp`. Può assumere i seguenti valori

`-v` | `--verbose`
 indica di visualizzare più dettagli;
`-v` | `--verbose`
 indica di visualizzare più dettagli;

???

Ogni volta che il livello network deve inviare sulla rete un IP datagram, prepara l'IP datagram, con gli indirizzi relativi al mittente ed al destinatario, quindi il protocollo di più basso livello controlla se l'indirizzo IP di destinazione è contenuto nella cache ARP. Nel caso in cui lo sia, provvede a copiarlo nell'indirizzo fisico di destinazione del pacchetto del livello link, altrimenti, dopo aver considerato la routing table, richiede sulla rete l'indirizzo fisico dell'interfaccia di rete a cui il pacchetto deve essere spedito (può essere il destinatario, se il suo indirizzo appartiene alla stessa rete di quella del mittente, o un router che si preoccuperà di instradare il pacchetto) tramite un opportuno pacchetto ARP.

??? ARP request ???

Ottenuto tale indirizzo, lo memorizzerà nella cache ARP e quindi lo copierà nell'indirizzo fisico di destinazione del pacchetto. Quindi il pacchetto viene inviato sulla rete.

L'IP datagram che viene inviato ad una determinata scheda di rete, è caratterizzato dall'indirizzo IP del mittente e quello del destinatario. Inoltre esso è incapsulato in un

indirizzo fisico

ARP
cache ARP

pacchetto del livello link (frame Ethernet o altro protocollo) che identifica la scheda di rete mittente e destinataria con i relativi indirizzi fisici. C'è un'indicazione ridondante del mittente e del destinatario. Questa ridondanza viene sfruttata per indirizzare i messaggi diretti a schede di rete non direttamente raggiungibili, ad appositi router che si preoccuperanno di instradare opportunamente i pacchetti verso il destinatario. Infatti, nel caso in cui il messaggio debba essere destinato ad una scheda di rete raggiungibile direttamente dal mittente, il suo indirizzo fisico comparirà come indirizzo fisico di destinazione ed il suo indirizzo IP comparirà come indirizzo IP di destinazione. Nel caso in cui la scheda di rete di destinazione non sia direttamente raggiungibile da quella mittente, l'indirizzo IP di destinazione del pacchetto sarà quello della scheda di rete di destinazione, mentre l'indirizzo fisico di destinazione sarà l'indirizzo fisico della scheda di rete del router che si occuperà dell'instradamento del pacchetto.

In questo modo il router che riceverà il pacchetto si accorgerà del fatto che non è effettivamente diretto a lui (confrontando il suo indirizzo IP con quello del destinatario del pacchetto) e provvederà quindi ad inoltrarlo al destinatario (se direttamente raggiungibile) o ad un altro router.

17.6 ICMP - Internet Control Message Protocol

Il protocollo ICMP è spesso considerato come parte integrante del livello Network, poiché è utilizzato per comunicare errori o avvertimenti dai protocolli di tale livello (IP) o di livello superiore (TCP, UDP, ...).

Un pacchetto ICMP è definito nella RFC 792, e si compone dei seguenti campi (v. fig. 17.3):

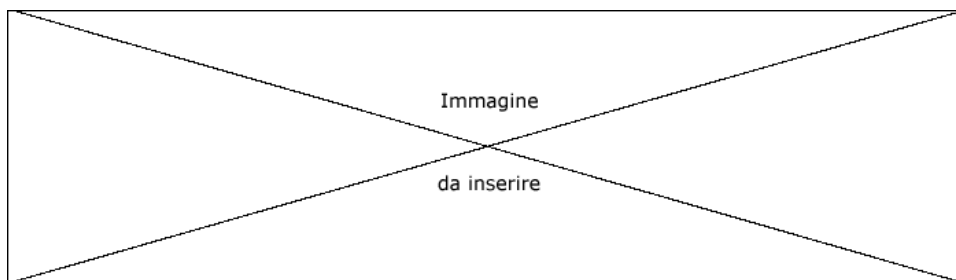


Figura 17.3: Il pacchetto ICMP.

Type (8 bit) indica il tipo del messaggio. I messaggi ICMP sono catalogati in base al loro tipo e codice come illustrato in tab. ??;

Code (8 bit) indica il codice del messaggio;

Checksum

(16 bit) contiene un codice di controllo che tiene conto delle informazioni presenti nel messaggio;

Le informazioni che seguono questi campi dipendono dal tipo del pacchetto ICMP considerato (campi *type* e *code*).

I pacchetti ICMP si dividono in *query* (richieste) ed *error* (errori) e gli errori sono generalmente trattati dallo stack TCP/IP in maniera leggermente diversa dalle richieste (ad esempio un pacchetto ICMP di errore non viene mai generato in risposta all'arrivo di un altro pacchetto ICMP di errore, che genererebbe un ciclo infinito di pacchetti ICMP di errore). Inoltre un pacchetto ICMP di errore riporta sempre l'header IP ed i primi 8 byte dell'IP datagram che ha causato l'invio di tale pacchetto. In tab. 17.6 sono riportati tutti i tipi dei pacchetti ICMP, per ognuno dei quali è evidenziato se è considerato come query o error.

Type	Code	Descrizione	Query	Error
0	0	Echo Reply	•	
3	0	Network Unreachable		•
3	1	Host Unreachable		•
3	2	Protocol Unreachable		•
3	3	Port Unreachable		•
3	4	Fragmentation needed but no frag. bit set		•
3	5	Source routing failed		•
3	6	Destination network unknown		•
3	7	Destination host unknown		•
3	8	Source host isolated (obsolete)		•
3	9	Destination network administratively prohibited		•
3	10	Destination host administratively prohibited		•
3	11	Network unreachable for TOS		•
3	12	Host unreachable for TOS		•
3	13	Communication administratively prohibited by filtering		•
3	14	Host precedence violation		•
3	15	Precedence cutoff in effect		•
4	0	Source quench		•
5	0	Redirect for network		•
5	1	Redirect for host		•
5	2	Redirect for TOS and network		•
5	3	Redirect for TOS and host		•
8	0	Echo request	•	
9	0	Router advertisement	•	
10	0	Route solicitation	•	
11	0	TTL equals 0 during transit		•
11	1	TTL equals 0 during reassembly		•
12	0	IP header bad (catchall error)		•
12	1	Required options missing		•
13	0	Timestamp request (obsolete)	•	
14	0	Timestamp reply (obsolete)	•	
15	0	Information request (obsolete)	•	
16	0	Information reply (obsolete)	•	
17	0	Address mask request	•	
18	0	Address mask reply	•	

Tabella 17.6: Tipi di pacchetti ICMP.

???

Un pacchetto ICMP, prima di essere spedito sulla rete, viene comunque incapsulato in un IP datagram.

???

17.7 Diagnosi di problemi di connettività

Per verificare la connettività delle interfacce di rete è spesso utile ricorrere ad alcuni semplici programmi messi a disposizione dal sistema, che attraverso l'invio di particolari pacchetti effettuano un test di massima sulla raggiungibilità delle interfacce di rete.

17.7.1 ping

Il comando **ping** (man page **ping(8)**, il cui nome deriva dal suono dei sonar per lo scandaglio del fondale marino presenti sulle navi, è utilizzato per verificare la comunicazione tra due interfacce di rete per mezzo dell'invio di pacchetti ICMP. Il programma invia dei pacchetti ICMP di tipo *Echo request* ad una determinata interfaccia di rete ed attende i relativi pacchetti ICMP *Echo reply* di risposta.

Comando: **ping**

Path: **/bin/ping**

SINTASSI

\$ ping [option] destination

DESCRIZIONE

option indica la modalità di funzionamento di **ping**. Può assumere i seguenti valori

- a emette un segnale acustico per ogni pacchetto trasmesso;
- A indica di essere adattativo, ovvero l'intervallo di tempo tra l'invio di un pacchetto e quello successivo dipende dal tempo di ritorno dei pacchetti stessi;
- b permette di effettuare ping in broadcast;
- B non permette di cambiare l'indirizzo del mittente nei pacchetti inviati;
- c *count*
indica di fermarsi dopo aver inviato *count* messaggi ICMP di tipo "echo request";
- f indica di visualizzare un carattere '.' per ogni messaggio ICMP di tipo "echo request" inviato ed un carattere Backspace per ogni messaggio ICMP di tipo "echo reply" ricevuto;
- F ???.
- i *interval*
specifica il tempo di attesa (in secondi) tra l'invio di un pacchetto e l'altro (il valore di default è 1 s);
- I *source_addr*
specifica l'indirizzo IP da inserire come mittente;
- l *preload*
indica di inviare inizialmente *preload* pacchetti senza attendere la relativa risposta;
- L indica di non effettuare il loopback dei pacchetti inviati in multicast;
- n indica di visualizzare gli indirizzi IP in forma numerica;
- p *value*
indica di aggiungere in coda ai pacchetti un valore di 16 bit espresso in notazione esadecimale;
- Q *tos* specifica il valore da assegnare al campo TOS (v. RFC 1349 e 2474);
- q indica di non visualizzare niente in output tranne quelle di riepilogo all'inizio ed alla terminazione dell'esecuzione;
- r indica di non considerare la routing table, ma di inviare i messaggi ICMP come se l'interfaccia di destinazione fosse raggiungibile direttamente;
- R indica di includere l'opzione "record route" nei messaggi ICMP di tipo "echo request" e di visualizzare il relativo buffer al ricevimento dei messaggi ICMP di risposta;
- s *packetsize*
specifica il numero di byte di dati da inviare (il valore di default è 56, per un totale di 64 byte, di cui 8 sono dell'header ICMP);
- S *sndbuf*
imposta il numero di pacchetti da bufferizzare (*sndbuf*) nel socket (il valore di default è 1);
- t *ttl* imposta il valore da assegnare al campo TTL dell'IP datagram;
- T *timestamp_options*
imposta le opzioni di timestamp dell'IP datagram;
- M *hint*
indica la strategia del Path MTU Discovery, secondo quanto illustrato in tab. 17.7;

hint	Descrizione
do	Proibisce la frammentazione (anche locale).
want	Attiva il PMTU discovery ed abilita la frammentazione locale.
dont	Non imposta il flag DF.

Tabella 17.7: Possibili valori di *hint* del comando **ping**.

- U visualizza il tempo totale user-to-user al posto del network round trip time (che potrebbe differire a causa di malfunzionamenti dei DNS);
- v indica di visualizzare un output più verboso;
- V visualizza la versione di `ping`;
- w *deadline*
specifica il tempo (in secondi) dopo il quale `ping` deve terminare;
- W *timeout*
specifica il tempo (in secondi) di attesa della risposta ad un pacchetto inviato;

destination ;

L'exit status di `ping` è riportato in tab. 17.8.

Valore	Descrizione
0	Tutto ok.
1	Non è stato ricevuto un numero di pacchetti di risposta uguale a quello dei pacchetti inviati.
2	Altro tipo di errore.

Tabella 17.8: Possibili valori dell'exit status di `ping`.

In genere se non si è in grado di comunicare con pacchetti ICMP, non lo si potrà fare con protocolli di più alto livello.

È comunque possibile, attraverso un meccanismo di firewalling², fare in modo che un'interfaccia di rete ignori i pacchetti ICMP ricevuti e quindi non risponda a questi ultimi, ma svolga il suo compito in maniera opportuna con pacchetti relativi a protocolli di livello più elevato (TCP, UDP, ...). In genere questo modo di funzionamento delle interfacce di rete viene utilizzato sui sistemi che danno un servizio continuo su *Internet* per evitare di sommergere altri sistemi di risposte a pacchetti ICMP non desiderate.

??? esempio ping ???

17.7.2 traceroute

Il comando `traceroute` (man page `traceroute(8)`) permette di scoprire qual'è il percorso dei pacchetti sulla rete, dal mittente alla destinazione. Per la natura dell'algoritmo di instradamento dei pacchetti, questa affermazione non è del tutto vera, ma questo è il metodo più utilizzato per scoprire il percorso dei pacchetti.

Il funzionamento di `traceroute` si basa sull'invio di IP datagram. Ogni IP datagram inviato, contiene un valore via via crescente nel campo TTL.

Il primo IP datagram inviato da `traceroute` avrà il TTL contenente il valore 1. Così facendo, la prima interfaccia di rete che riceverà l'IP datagram lo scatterà, inviando un pacchetto ICMP di risposta al mittente per informarlo dell'accaduto. In questo modo `traceroute` conoscerà la prima interfaccia di rete che ha ricevuto l'IP datagram. Se tale interfaccia non è quella di destinazione (il suo indirizzo IP non è quello del destinatario dell'IP datagram inviato), `traceroute` invia un altro IP datagram ma con il campo TTL contenente il valore 2. Questa volta la seconda interfaccia di rete scatterà l'IP datagram rispondendo al mittente con un pacchetto ICMP. E così via. Si arriva così a delineare, interfaccia di rete dopo interfaccia di rete, il percorso seguito da un generico IP datagram inviato dall'interfaccia di rete considerata a quella di destinazione.

Comando: `traceroute`
 Path: `/usr/sbin/traceroute`
 SINTASSI
 \$ `traceroute [option] destination [packetlen]`

²v. sez. 25.3.

DESCRIZIONE

option indica la modalità di funzionamento di **traceroute**. Può assumere i seguenti valori

- f *first_ttl*
imposta il valore iniziale del TTL degli IP datagram secondo quanto specificato da *first_ttl* (il valore di default è 1);
- F imposta il bit di non frammentazione;
- d abilita il debug;
- g *gateway*
specifica il gateway ???;
- i *iface*
specifica l'interfaccia di rete (*iface*) dalla quale inviare i pacchetti;
- I indica di utilizzare messaggi ICMP di tipo echo, invece degli UDP datagram;
- m *max_ttl*
imposta il valore massimo del TTL degli IP datagram secondo quanto specificato da *max_ttl* (il valore di default è 30);
- n visualizza gli indirizzi soltanto in forma numerica;
- p *port*
specifica la porta UDP di base (*port*) da utilizzare (per default è la 33434). Si suppone infatti che sulle macchine che ricevono i pacchetti UDP, non ci sia in ascolto nessun processo sull'intervallo di porte [base, base+nhops-1];
- q *nqueries*
imposta il numero di pacchetti da inviare con lo stesso TTL (per default è 3);
- r indica di non considerare la routing table, ma di inviare i pacchetti come se l'interfaccia di destinazione fosse raggiungibile direttamente;
- s *src_addr*
specifica l'indirizzo dell'interfaccia (*src_addr*) da utilizzare come mittente;
- t *tos* imposta il valore del TOS degli IP datagram secondo quanto specificato da *tos* (il valore di default è 0);
- v (verbose) indica di visualizzare tutti i messaggi ICMP di ritorno;
- w *waittime*
imposta il tempo massimo di attesa (in secondi) per la risposta ad un pacchetto inviato (per default è 5 secondi);
- x disabilita il calcolo del checksum dell'IP;
- z *pause*
imposta il tempo (in millisecondi) tra un invio di un pacchetto e quello successivo (per default è 0);

destination indica l'indirizzo IP dell'interfaccia di destinazione o il suo relativo nome;

packetlen indica la lunghezza massima dei pacchetti (in byte);

Ogni volta che viene ricevuto un messaggio ICMP come risposta ad un pacchetto inviato, **traceroute** visualizza l'indirizzo dell'interfaccia mittente relativa. Nel caso in cui non si abbia ricevuto alcuna risposta entro il tempo massimo di attesa impostato viene visualizzato un carattere '*', al posto dell'indirizzo IP dell'interfaccia mittente

Si consideri il seguente esempio

```
$ traceroute allspice.lcs.mit.edu.
traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
 1  helios.ee.lbl.gov (128.3.112.1)  0 ms  0 ms  0 ms
 2  lilac-dmc.Berkeley.EDU (128.32.216.1)  19 ms  19 ms  19 ms
 3  lilac-dmc.Berkeley.EDU (128.32.216.1)  39 ms  19 ms  19 ms
 4  ccngw-ner-cc.Berkeley.EDU (128.32.136.23)  19 ms  39 ms  39 ms
 5  ccn-nerif22.Berkeley.EDU (128.32.168.22)  20 ms  39 ms  39 ms
 6  128.32.197.4 (128.32.197.4)  59 ms  119 ms  39 ms
```

```

7  131.119.2.5 (131.119.2.5)  59 ms  59 ms  39 ms
8  129.140.70.13 (129.140.70.13)  80 ms  79 ms  99 ms
9  129.140.71.6 (129.140.71.6)  139 ms  139 ms  159 ms
10 129.140.81.7 (129.140.81.7)  199 ms  180 ms  300 ms
11 129.140.72.17 (129.140.72.17)  300 ms  239 ms  239 ms
12 * * *
13 128.121.54.72 (128.121.54.72)  259 ms  499 ms  279 ms
14 * * *
15 * * *
16 * * *
17 * * *
18 ALLSPICE.LCS.MIT.EDU (18.26.0.115)  339 ms  279 ms  279 ms

```

La prima riga visualizzata da **traceroute** conferma l'interfaccia di rete che verrà ricercata ed il numero massimo di hops (TTL). Le righe 2 e 3 riportano la stessa indicazione: ciò è dovuto ad un bug presente sul kernel del sistema lilac-dmc.Berkeley.EDU che effettua l'inoltro dei pacchetti con TTL = 0. Le righe 12, 14, 15, 16 e 17 non riportano gli indirizzi delle interfacce poiché non è stato risposto alcun messaggio ICMP "time exceeded" o è stato fatto con un valore di TTL troppo basso per raggiungere il destinatario. In corrispondenza di ogni riga, viene visualizzato anche il tempo di risposta stimato. Inoltre può essere visualizzato un carattere '!' nel caso in cui il pacchetto IP relativo al messaggio ICMP di risposta abbia un valore di TTL ≤ 1 . Altre annotazioni possibili sono quelle riportate in tab. 17.9

Sequenza	Descrizione
!H	(Host unreachable) Interfaccia irraggiungibile.
!N	(Network unreachable) Rete irraggiungibile.
!P	(Protocol unreachable) Protocollo irraggiungibile.
!S	(Source route failed) ???.
!F- <i>pmtu</i>	(Fragmentation needed) ??? (viene visualizzato il path MTU).
!X	(Communication administratively prohibited) ???.
!V	(Host precedence violation) ???.
!C	(Precedence cutoff in effect) ???.
! <i>num</i>	(ICMP unreachable code <i>num</i>) ???.

Tabella 17.9: Possibili annotazioni nell'output di **traceroute** (v. RFC 1812).

???

Il fatto è che la politica di gestione dell'instradamento dei pacchetti da parte di un router può cambiare da un momento all'altro o addirittura il percorso da far seguire ad un pacchetto può cambiare da un momento all'altro perché un indirizzo non è momentaneamente raggiungibile (è caduto il collegamento, lo stack TCP/IP non risponde, ...). Quindi non è detto che il percorso dei pacchetti rimanga lo stesso tra l'invio di un IP datagram e l'altro, ma, con molta probabilità, il percorso non cambierà.

???

17.8 Riferimenti

???

Capitolo 18

TCP/IP - Transport

“Ci vuole tutta una vita per capire che non è necessario capire tutto.”
– (Proverbio cinese)

???

18.1 Porte e socket

Per i protocolli a livello di trasporto, viene introdotto il concetto di *porta*. Una **porta** (*port*) è un valore numerico specificato su 2 byte (da 0 a 65535) che identifica un particolare canale utilizzabile per la comunicazione. In questo modo è possibile instaurare simultaneamente più comunicazioni con lo stesso protocollo, cosicché due applicazioni possono comunicare l'una con l'altra indipendente dal fatto che sulla rete stiano avvenendo altre comunicazioni che utilizzano lo stesso protocollo: è sufficiente utilizzare porte diverse.

Si viene così a delineare il **socket** (v. RFC 793) che rappresenta un punto di connessione per una comunicazione, identificato univocamente da un indirizzo IP ed un numero di porta. Un'applicazione che vuole comunicare con un'altra utilizzando un protocollo a livello di trasporto, deve richiedere un *socket* al sistema operativo, in particolare comunica al sistema di utilizzare una determinata porta in modo tale che tutti i pacchetti destinati a tale porta le vengano recapitati. Quindi invia il pacchetto ad un determinato socket (indirizzo IP di destinazione, porta di destinazione). Il pacchetto verrà ricevuto dall'interfaccia di rete il cui indirizzo IP è quello specificato nel pacchetto ed il sistema ricevente recapiterà il pacchetto ricevuto all'applicazione a cui si riferisce la porta di destinazione.

È bene sottolineare il fatto che i numeri di porta sono relativi soltanto al protocollo considerato: una determinata porta per il protocollo TCP è diversa dallo stesso numero di porta per il protocollo UDP (si tratta effettivamente di porte diverse), anche se in genere viene utilizzato lo stesso numero di porta per un servizio che gestisce entrambi i protocolli.

18.2 UDP - User Datagram Protocol

Si tratta di un protocollo molto semplice, descritto nella RFC 768, per l'invio/ricezione di pacchetti che sono poco più che degli IP datagram. Non attua nessun meccanismo che garantisca la consegna dei pacchetti.

18.2.1 L'UDP datagram

I pacchetti del protocollo UDP sono detti **UDP datagram** e la loro struttura è riportata in fig. 18.1.

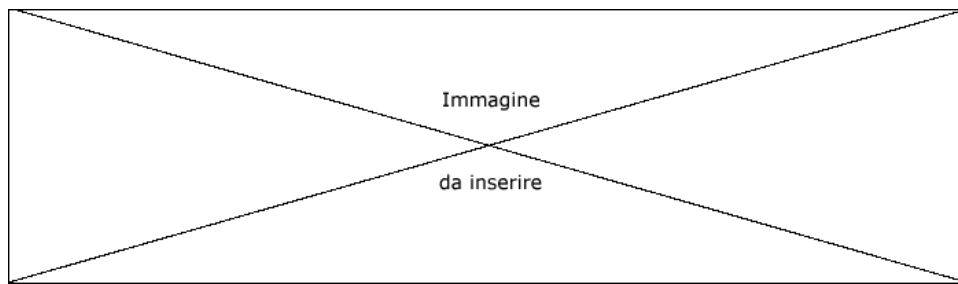


Figura 18.1: Struttura di un *UDP datagram*.

Source port number

è un numero che indica la porta del processo mittente;

Destination port number

è un numero che indica la porta del processo di destinazione;

UDP lenght

è un numero che indica la lunghezza in byte dell'intero UDP datagram (il valore minimo contenuto da tale campo è 8, che indica un UDP datagram senza alcuna informazione nel relativo payload).

UDP checksum

è un codice di controllo calcolato sull'intero contenuto dell'UDP datagram (il calcolo di tale codice di controllo è facoltativo).

18.3 TCP - Transmission Control Protocol

Ogni *socket* è identificato da una coppia $\langle \text{indirizzo IP}, \text{porta} \rangle$. La connessione (*connection*) tra due socket è univocamente identificata dall'insieme dei socket considerati, ovvero da una tupla di 4 elementi: $\langle \text{indirizzo IP mittente}, \text{porta mittente}, \text{indirizzo IP destinatario}, \text{porta destinatario} \rangle$.

Il protocollo TCP fornisce un meccanismo di comunicazione di tipo connesso (*connection-oriented*), cioè la comunicazione tra due socket *A* e *B* avviene soltanto dopo aver stabilito una connessione tra i socket stessi. La comunicazione tra i socket è di tipo full-duplex, ovvero il flusso dei pacchetti può avvenire contemporaneamente in entrambe le direzioni (da *A* a *B* e da *B* ad *A*). Si tratta inoltre di un protocollo affidabile (*reliable*), nel senso che è gestito un meccanismo che garantisce il recapito dei pacchetti dall'interfaccia mittente a quella di destinazione.

18.3.1 Il TCP segment

TCP segment

I pacchetti del protocollo TCP sono detti **TCP segment** e la loro struttura è riportata in fig. 18.2.

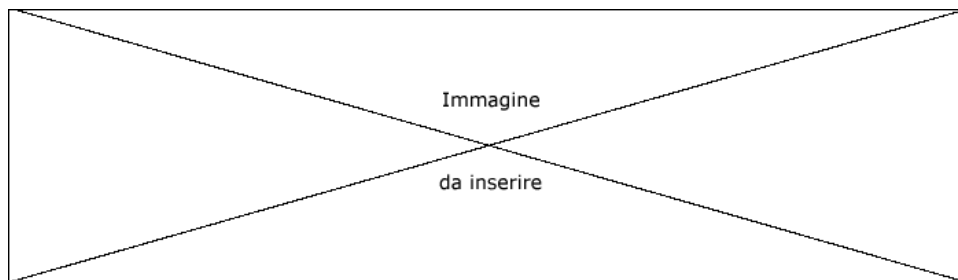


Figura 18.2: Struttura di un *TCP segment*.

Source port number

è un numero che indica la porta del processo mittente;

Destination port number

è un numero che indica la porta del processo di destinazione;

Sequence number

è un numero che identifica il pacchetto all'interno della sequenza dei pacchetti scambiati tra mittente e destinatario (contiene valori da 0 a $2^{32} - 1$ e quando arriva al valore massimo riparte da 0).

Acknowledgment number

è un numero che riporta il valore del *Sequence number* atteso nel TCP segment di risposta al TCP segment considerato (il contenuto di tale campo viene considerato soltanto nel caso in cui sia impostato il flag *ACK*).

Header length

è un numero che indica la lunghezza dell'header del TCP segment in termini di double word (può indicare una lunghezza di header fino a $(2^4 - 1) * 32/8 = 60$ byte).

URG è un flag che indica che l'*urgent point* è valido.

ACK è un flag che indica che l'*acknowledgment number*.

PSH è un flag che indica che il ricevente dovrebbe passare i dati all'applicazione al più presto.

RST è un flag che indica che la connessione deve essere resettata.

SYN è un flag che indica di sincronizzare i sequence number per iniziare una nuova connessione.

FIN è un flag che indica la terminazione della connessione.

Window size

indica il numero di byte che sono accettati in ricezione (tale valore va da 0 a $2^{16} - 1 = 65535$ byte).

TCP checksum

è un numero che esprime il codice di controllo relativo all'intero TCP segment.

Urgent pointer

è un numero che aggiunto al *sequence number* indica il sequence number dell'ultimo byte contenente i dati di tipo urgente (il contenuto di tale campo viene considerato soltanto nel caso in cui sia impostato il flag *URG*).

Options

Specifica eventuali opzioni (v. RFC 793 e 1323). Ogni opzione è identificata da un codice (campo *kind*) di 1 byte. Quelle possibili sono le seguenti

End of option list**No operation (NOP)**

questo tipo di opzione viene utilizzata essenzialmente per mantenere l'allineamento dei campi ad un multiplo di byte maggiore di 1. Infatti se ad esempio il TCP segment senza l'opzione avesse una dimensione di 127 byte e si volesse un allineamento dei campi a multipli di 2 byte, si può ottenere lo scopo indicando come opzione una NOP, in maniera tale che il segment arrivi a 128 byte.

Maximum segment size

Ognuno dei due socket in genere specifica questa opzione nel primo TCP segment scambiato, per specificare la massima dimensione del TCP segment che il mittente può trasmettere. È formata dai seguenti campi

Len indica la lunghezza totale dell'opzione (tutti i campi);
Size specifica la dimensione massima del TCP segment trasmissibile che va da 0 a $MTU - IP_header_length - TCP_header_length$.

Window scale factor

Len indica la lunghezza totale dell'opzione (tutti i campi);
Shift Count

Timestamp

Len indica la lunghezza totale dell'opzione (tutti i campi);
Timestamp
 è il valore del timestamp (data e ora) corrente;
Timestamp echo reply

???

Anche se generalmente viene impostato un solo flag per volta, è possibile che in un TCP segment siano impostati più flag.

La RFC 1025 indica che un TCP segment con tutti i flag impostati (SYN, URG, PSH, FIN) ed un byte di dati è detto *Kamikaze packet*, conosciuo anche come *nastygram*, *Christmas tree packet* e *lamp test segment*.

18.3.2 La connessione

Apertura

Si supponga che un'applicazione (socket) *A* (client) voglia comunicare con un'altra, *B* (server). Secondo il protocollo TCP, la prima operazione da effettuare è quella di stabilire una connessione. A tal fine vengono effettuate le seguenti operazioni

1. Il client invia al server il primo TCP segment (detto anche *SYN segment*) contenente
 - il flag SYN impostato, per indicare che si sta tentando di stabilire una connessione;
 - il valore del sequence number indica l'**ISN** (Initial Sequence Number) del client.
2. Il server risponde con un altro TCP segment (anche questo viene denominato *SYN segment*) contenente
 - l'ISN del server;
 - l'acknowledgment number contenente il valore successivo al sequence number del TCP segment appena ricevuto;
 - il flag ACK impostato, che indica che il server si aspetta che il successivo TCP segment che riceverà avrà il valore del sequence number uguale al valore contenuto nel campo acknowledgment number;
3. Il client risponde a sua volta con un altro TCP segment, per confermare che ha ricevuto l'ISN del server. Esso contiene
 - l'acknowledgment number contenente il valore successivo al sequence number del TCP segment appena ricevuto;
 - il flag ACK impostato, che indica che il client si aspetta che il successivo TCP segment che riceverà avrà il valore del sequence number uguale al valore contenuto nel campo acknowledgment number;

ISN

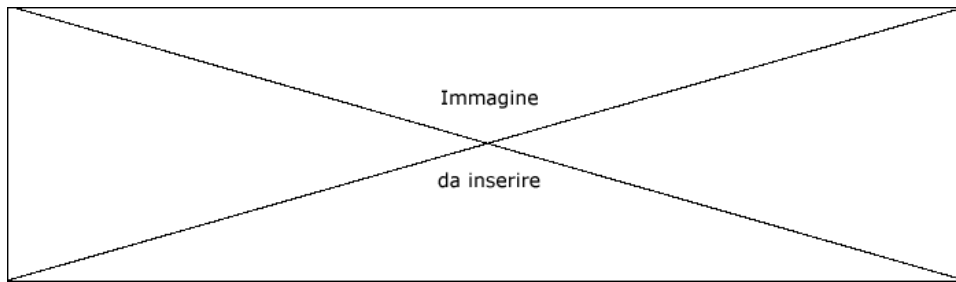


Figura 18.3: Diagramma relativo all'apertura di una connessione TCP.

Questi tre TCP segment stabiliscono una connessione. Per questo motivo tale meccanismo è anche detto *three-way handshake*.

L'applicazione *A* effettua quella che viene chiamata un'apertura attiva della connessione (*active open*), mentre l'applicazione *B* fa un'apertura passiva (*passive open*).

La RFC 793 specifica che l'ISN dovrebbe essere visto come un contatore incrementato ogni $4 \mu s$. Lo scopo di questo valore è quello di fare in modo che i pacchetti che rimangono congestionati nella rete non vengano presi successivamente in considerazione nel caso in cui siano consegnati in ritardo, rispetto al flusso dei pacchetti inviati nella connessione esistente.

Chiusura

Si supponga che un'applicazione (socket) *A* (client) voglia chiudere la connessione esistente con un'altra, *B* (server). A tal fine vengono effettuate le seguenti operazioni

1. Il client invia un TCP segment che contiene
 - il flag FIN impostato, che indica che si desidera interrompere la connessione: il client non intende inviare altri pacchetti al server;
2. Il server risponde con un altro TCP segment contenente
 - il flag ACK impostato, che indica che il server ha recepito la richiesta;
3. Il server invia un altro TCP segment per chiudere a sua volta la comunicazione nell'altra direzione. Tale TCP segment contiene
 - il flag FIN impostato, che indica che si desidera interrompere la connessione: il server non intende inviare altri pacchetti al client;
4. Il client risponde con un altro TCP segment contenente
 - il flag ACK impostato, che indica che il client ha recepito la richiesta;

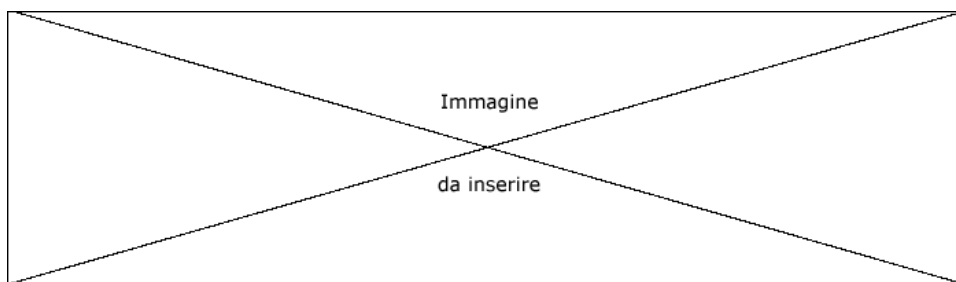


Figura 18.4: Diagramma relativo alla chiusura di una connessione TCP.

La chiusura completa della connessione coinvolge 4 TCP segment. Anche in questo caso, si parla di chiusura attiva (*active close*) da parte di chi effettua la chiusura del canale di comunicazione per primo, per distinguerla dalla chiusura passiva (*active close*) effettuata da chi riceve la richiesta di chiusura attiva.

18.4 Comunicazione tra client e server

Come illustrato in cap. 16, la comunicazione di rete avviene tra due processi detti client e server. In particolare il server attende una richiesta da parte di un client e risponde alla richiesta stessa. In questa sezione viene esaminato più in dettaglio la procedura di comunicazione tra client e server per i protocolli a livello di trasporto.

L'interfaccia software più utilizzata per la scrittura di programmi di comunicazione è quella della BSD che si rifà ai socket (l'alternativa della System V è la TLI - Transport Layer Interface). Tale interfaccia di programmazione permette la comunicazione tra due processi, sia sulla stessa macchina che su macchine diverse, utilizzando per default lo stack di protocolli TCP/IP. Dal punto di vista della programmazione, lavorare con i socket è analogo a lavorare con i file.

well known port numbers

Il server deve innanzitutto riservare un socket relativo ad un indirizzo IP ed a una porta ed indica il tipo di protocollo da utilizzare nella comunicazione. Le porte riservate per servizi standard, sono dette **well known port numbers** e sono riportate dalla RFC 1060 (ad esempio, la porta 80 è quella standard dei server HTTP, la porta 25 è quella standard dei server SMTP, ...). Il file ??? riporta un elenco delle porte associate ai vari servizi.

Per server relativi a servizi non standard è buona regola utilizzare un numero di porta che non è elencato fra quelli associati ai servizi standard.

Dunque, il server rimane in attesa di una richiesta da parte di un client, richiedendo al sistema operativo di mettergli a disposizione un buffer nel quale verranno accodate al massimo un determinato numero di richieste ricevute.

Non appena il server riceve una richiesta da parte di un client, questo procede alla sua elaborazione fino a fornire una risposta al client stesso. Mentre il server è occupato nell'elaborazione della richiesta e nella preparazione della relativa risposta, potrebbero arrivare altre richieste da parte di altri client. Tali richieste saranno memorizzate dal sistema operativo nel buffer precedentemente riservato a tale scopo, fino ad un massimo indicato dal server. Le richieste che eccederanno tale valore saranno automaticamente scartate.

Per ovviare al problema della possibile perdita delle richieste che arrivano durante l'asservimento di un'altra richiesta, si utilizza in genere la tecnica di affidare la gestione della comunicazione con un client ad un processo o thread a parte. In questo modo il processo (o thread) principale che riceve una richiesta da parte di un client crea il processo (o thread) figlio facendo gestire ad esso l'elaborazione della richiesta e la generazione della relativa risposta, ed esso ritorna subito in ascolto di altre eventuali richieste da parte dei client.

Quindi il server invia la risposta al client e ritorna in attesa di ulteriori richieste (se ci sono delle richieste pendenti, queste sono viste dal server come ulteriori richieste) da elaborare.

Il client invece deve inizialmente riservare un socket in relazione soltanto ad un indirizzo IP ed al protocollo utilizzato nella comunicazione, mentre il numero di porta non è indispensabile, ma viene generalmente lasciato decidere al sistema operativo, che ne sceglierà uno opportuno non riservato e non già utilizzato. Quindi, si connette al server specificando il relativo indirizzo IP e numero di porta (è necessario che il client conosca il numero di porta relativo al server).

Quindi, procede all'invio di una richiesta al server e si mette in attesa di una risposta da parte dello stesso.

Al termine delle operazioni può chiudere l'eventuale connessione precedentemente aperta.

???

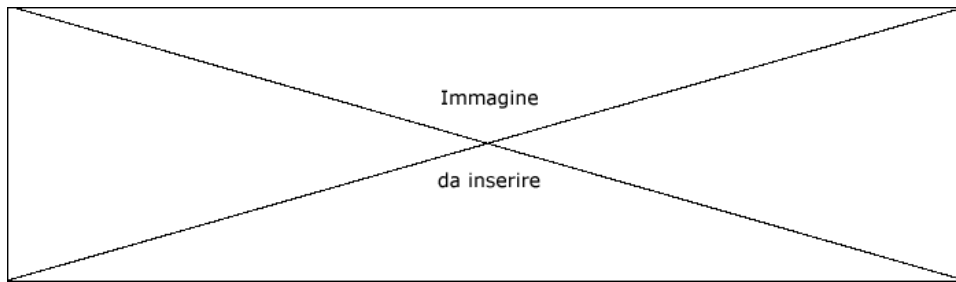


Figura 18.5: Schematizzazione di un'applicazione server.

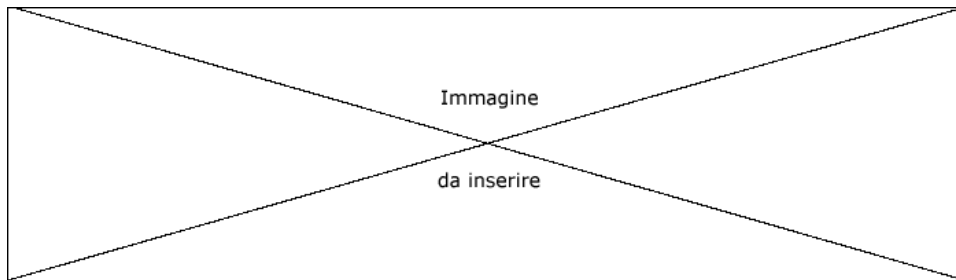


Figura 18.6: Schematizzazione di un'applicazione client.

18.5 netstat e tcpdump

Per effettuare dei test sul funzionamento della comunicazione via rete, sono utili i comandi come **netstat** (man page **netstat(8)**) e **tcpdump** (man page **tcpdump(8)**). Il primo è utilizzato per mostrare le connessioni di rete attive, le routing table, le statistiche relative ai pacchetti inviati/ricevuti dalle interfacce di rete e così via, mentre il secondo viene utilizzato per catturare i pacchetti che transitano sulle interfacce di rete e poter così effettuare un'analisi successiva delle informazioni in essi contenute.

Comando: **netstat**

Path: **??/netstat**

SINTASSI

netstat [*option*]

DESCRIZIONE

option indica la modalità di funzionamento di **netstat**. Può assumere i seguenti valori:

- r** | **--route**
visualizza le routing table;
- g** | **--groups**
visualizza le informazioni dei membri relativi ai gruppi multicast;
- i** | **--interface=iface**
visualizza l'elenco delle interfacce di rete o quella specificata (*iface*);
- M** | **--masquerade**
visualizza l'elenco delle connessioni che usano la tecnica del masquerading o NAT (v. cap. ??);
- s** | **--statistics**
visualizza le statistiche per ogni protocollo;
- h** | **--help**
visualizza un aiuto sommario di **netstat**;
- V** | **--version**
visualizza la versione di **netstat**;

Se non viene specificata nessuna opzione, viene visualizzato l'elenco dei socket aperti presenti sul sistema.

18.6 Riferimenti

???

Capitolo 19

TCP/IP - Application

“Di tutto conosciamo il prezzo, di niente il valore.”
– F. Nietzsche

???

19.1 SMTP - Simple Mail Transfer Protocol

???

19.2 POP - Post Office Protocol

???

19.3 HTTP - HyperText Transfer Protocol

???

19.3.1 Apache

???

19.3.2 HTML - HyperText Markup Language

???

19.3.3 Javascript

???

19.3.4 CGI - Common Gateway Interface

???

19.3.5 PHP

???

19.3.6 Tomcat

???

19.3.7 JSP - Java Scripting Page

???

19.4 FTP - File Transfer Protocol

???

19.5 NTP - Network Time Protocol

???

19.6 DHCP - Dynamic Host Control Protocol

???

19.7 SNMP - Simple Network Message Protocol

???

19.8 DNS - Domain Name Server

???

19.9 NFS - Network FileSystem

???

19.10 SAMBA

???

19.11 Telnet

???

19.12 rcp

???

19.13 rlogin

???

19.14 rsh

???

19.15 Riferimenti

???

Capitolo 20

Impostazioni di rete

“Si odono solo le domande alle quali si è in condizione di trovare una risposta.”
– F. Nietzsche

???

20.1 Le interfacce di rete

Le interfacce di rete sono dispositivi che permettono ai sistemi di collegarsi tra loro. Su un sistema GNU/Linux sono identificate da un nome come riportato in tab. 20.1.

i_name	Significato
lo	interfaccia di loopback (localhost)
ethn	n-esima interfaccia di rete ethernet (con $n = 0, 1, 2, \dots$)
pppn	n-esima interfaccia di rete di tipo dialup con protocollo PPP (con $n = 0, 1, 2, \dots$)
sln	n-esima interfaccia di rete di tipo dialup con protocollo SLIP (con $n = 0, 1, 2, \dots$)
irlann	n-esima interfaccia ad infrarossi (con $n = 0, 1, 2, \dots$)
plipn	n-esima interfaccia con protocollo PLIP (con $n = 0, 1, 2, \dots$)
trn	n-esima interfaccia di rete token ring (con $n = 0, 1, 2, \dots$)

Tabella 20.1: Nomi delle interfacce.

???

20.2 Impostazioni di sistema

??? descrizione delle impostazioni (es. il nome del sistema) ???

Le impostazioni di rete a livello di sistema, ovvero relative a tutte le interfacce di rete presenti, sono memorizzate nel file `/etc/sysconfig/network`. Esso contiene direttive espresse tramite righe con la seguente sintassi

keyword=value

dove

keyword

è la parola chiave che identifica una direttiva. Può assumere i seguenti valori

NETWORKING

indica se il sistema è abilitato ad utilizzare la rete o meno, dipendentemente dal valore *value*: **yes** abilita l'uso della rete, **no** lo disabilita;

HOSTNAME

indica il FQDN (Fully Qualified Domain Name) del sistema secondo quanto specificato dal valore *value* (ad esempio **hostname.example.com**).

Per compatibilità con vecchi software, il file `/etc/HOSTNAME` dovrebbe contenere lo stesso valore di *value*;

GATEWAY

indica l'indirizzo IP del gateway (in dotted decimal notation) secondo quanto specificato dal valore *value*;

GATEWAYDEV

indica l'interfaccia di rete alla quale si riferisce il gateway (ad esempio `eth0`);

NISDOMAIN

indica il nome del dominio NIS, secondo quanto specificato dal valore *value*;

Il nome del sistema viene impostato nelle strutture dati in memoria (file `/proc/sys/kernel/hostname`) con il comando `hostname` (man page `hostname(1)`)

Comando: `hostname`

Path: `/bin/hostname`

SINTASSI

`$ hostname [option] [name]`

DESCRIZIONE

option indica la modalità di funzionamento di `hostname`. Può assumere i seguenti valori

`-a` | `--alias`

visualizza i vari nomi oltre al primo (alias) della macchina, ricavandoli dal file `/etc/hosts` o dal DNS;

`-d` | `--domain`

visualizza il nome del dominio del DNS ovvero la parte dominio del FQDN (come il comando `dnsdomainname`);

`-F filename` | `--file filename`

specifica di considerare il contenuto del file *filename* (le righe del file che iniziano con il carattere '#' vengono ignorate);

`-f` | `--fqdn` | `--long`

visualizza il FQDN;

`-h` | `--help`

visualizza un aiuto sommario di `hostname`;

`-i` | `--ip-address`

visualizza gli indirizzi IP del sistema;

`-n` | `--node`

visualizza il nome del nodo DECnet o lo imposta secondo quanto specificato da *name* o dall'opzione `-F`;

`-s` | `--short`

visualizza il nome del sistema in forma abbreviata (senza il nome del dominio);

`-V` | `--version`

visualizza la versione di `hostname`;

`-v` | `--verbose`

aumenta la verbosità delle informazioni riportate da `hostname`;

`-y` | `--yp` | `--nis`

visualizza il dominio del NIS o lo imposta secondo quanto specificato da *name* o dall'opzione `-F`;

name indica il nome con cui impostare quello di sistema;

???

20.3 Impostazioni delle interfacce

Le interfacce di rete vengono impostate per mezzo del comando `ifconfig` (man page `ifconfig(8)`).

Comando: `ifconfig`
Path: `/sbin/ifconfig`

SINTASSI

`ifconfig [interface] [addr] [setting]`

DESCRIZIONE

interface indica il nome dell'interfaccia alla quale si vuole riferirsi;

addr indica l'indirizzo IP da assegnare all'interfaccia *interface*

setting specifica ulteriori dettagli di configurazione dell'interfaccia *interface*. Può assumere i seguenti valori

- `up` attiva l'interfaccia (questo è implicito se viene specificato *addr*);
- `down` disattiva l'interfaccia;
- `[-]arp` abilita/disabilita (disabilita se utilizzato il '-') l'utilizzo del protocollo ARP;
- `[-]promisc` abilita/disabilita la modalità promiscua. Se abilitata l'interfaccia riceverà tutti i pacchetti che transitano sul tratto di rete fisicamente connesso ad essa;
- `[-]allmulti` abilita/disabilita la modalità all-multicast. Se abilitata l'interfaccia riceverà tutti i pacchetti multicast che transitano sul tratto di rete fisicamente connesso ad essa;
- `metric value` specifica la metrica dell'interfaccia;
- `metric value` specifica la MTU dell'interfaccia;
- `dstaddr addr` specifica l'indirizzo IP dell'altro capo della comunicazione in un collegamento punto-punto (obsoleto, v. `pointopoint`);
- `netmask addr` specifica la subnet mask (in dotted decimal notation);
- `add addr/prefixlen` aggiunge un indirizzo IPv6 all'interfaccia;
- `del addr/prefixlen` rimuove un indirizzo IPv6 dall'interfaccia;
- `tunnel aa.bb.cc.dd` specifica un nuovo dispositivo SIT (IPv6-in-IPv4) effettuando un tunnelling alla destinazione specificata;
- `irq addr` imposta la linea di interrupt (IRQ) utilizzata dall'interfaccia;
- `io_addr addr` specifica l'indirizzo dello spazio di I/O dell'interfaccia;
- `mem_start addr` specifica l'indirizzo di inizio della memoria condivisa utilizzata dall'interfaccia;
- `media type` specifica la porta fisica o il tipo di connessione utilizzata dall'interfaccia (es. `ethernet 10base2`, `10baseT`, ...);
- `[-]broadcast [addr]` imposta l'indirizzo di broadcast con *addr* o, se *addr* non è specificato, imposta/annulla il flag `IFF_BROADCAST` per l'interfaccia;
- `[-]pointopoint [addr]` abilita/disabilita la modalità punto-punto dell'interfaccia, impostando/annullando il flag `IFF_POINTOPOINT`. Se *addr* è specificato, imposta l'indirizzo dell'altro capo della connessione con *addr*;

```

hw class address
    imposta l'indirizzo fisico dell'interfaccia, dove class è il nome della
    classe di indirizzi fisici e address è l'equivalente ASCII dell'indirizzo
    fisico;
multicast
    imposta la modalità multicast per l'interfaccia;
multicast
    imposta la modalità multicast per l'interfaccia;
txqueuelen length
    imposta la lunghezza della coda di trasmissione dell'interfaccia;

```

???

Le impostazioni con le quali vengono automaticamente configurate le interfacce di rete all'avvio del sistema sono memorizzate nei file `etc/sysconfig/network-scripts/ifcfg-i_name`, dove *i_name* è il nome dell'interfaccia considerata (v. tab. 20.1). Questi contengono delle direttive espresse tramite righe con la seguente sintassi

keyword=value

dove *keyword* e *value* dipendono dal tipo di interfaccia e sono illustrati nelle sezioni seguenti.

20.3.1 Interfacce ethernet

I file di configurazione del tipo `ifcfg-ethn` possono contenere le seguenti *keyword*

BOOTPROTO

indica il tipo di protocollo che il sistema deve utilizzare per configurare la scheda di rete, secondo quanto specificato da *value* (v. tab. 20.2);

<i>value</i>	Significato
bootp	protocollo BOOTP
dhcp	protocollo DHCP
none	nessun protocollo (configurazione statica)

Tabella 20.2: Protocolli di configurazione delle schede di rete.

BROADCAST

indica l'indirizzo IP di broadcast (in decimal dotted notation) secondo quanto specificato da *value* (l'utilizzo di questa direttiva è deprecato);

DEVICE indica il nome del dispositivo di rete secondo quanto specificato da *value*;

DNS1 indica l'indirizzo IP del DNS primario (in decimal dotted notation) secondo quanto specificato da *value*. Tale valore deve essere anche specificato nel file `/etc/resolv.conf` se la direttiva **PEERDNS** è impostata a **yes**;

DNS2 indica l'indirizzo IP del DNS secondario (in decimal dotted notation) secondo quanto specificato da *value*. Tale valore deve essere anche specificato nel file `/etc/resolv.conf` se la direttiva **PEERDNS** è impostata a **yes**;

IPADDR indica l'indirizzo IP dell'interfaccia di rete considerata (in decimal dotted notation) secondo quanto specificato da *value*;

NETMASK

indica la netmask relativa all'indirizzo IP specificato con la direttiva **IPADDR** (in decimal dotted notation), secondo quanto specificato da *value*;

NETWORK

indica l'indirizzo IP della rete alla quale appartiene la l'interfaccia considerata (in decimal dotted notation), secondo quanto specificato da *value* (l'utilizzo di questa direttiva è deprecato);

ONBOOT indica se l'interfaccia di rete considerata deve essere attivata automaticamente all'avvio del sistema o meno, secondo quanto specificato da *value* (**yes** attiva l'interfaccia di rete all'avvio, **no** non attiva l'interfaccia di rete all'avvio);

PEERDNS

indica se il file `/etc/resolv.conf` deve essere modificato con l'indirizzo del DNS fornito dal sistema remoto, secondo quanto specificato da *value* (**yes** il file `/etc/resolv.conf` deve essere modificato – impostazione di default se si usa `BOOTPROTO=dhcp` –, **no** il file `/etc/resolv.conf` non deve essere modificato);

SRCADDR

indica l'indirizzo IP da utilizzare come indirizzo mittente nei pacchetti che vengono inviati (in decimal dotted notation), secondo quanto specificato da *value*;

USERCTL

indica se gli utenti senza permessi amministrativi possono controllare l'interfaccia considerata, secondo quanto specificato da *value* (**yes** gli utenti non amministratori possono controllare l'interfaccia di rete, **no** gli utenti non amministratori non possono controllare l'interfaccia di rete);

Di seguito è riportato un esempio del contenuto del file `ifcfg-eth0` nel caso di una configurazione con un indirizzo statico.

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=192.168.1.0
NETMASK=255.255.255.0
IPADDR=192.168.1.27
USERCTL=no
```

Il caso seguente si riferisce invece ad un file `ifcfg-eth0` con indirizzi IP dinamico, negoziato con il protocollo DHCP.

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

20.3.2 Interfacce dialup

I file di configurazione del tipo `ifcfg-pppn` e `ifcfg-slipn` possono contenere, oltre alle *keyword* illustrate in sez. 20.3.1, quelle seguenti.

DEFROUTE

indica se l'interfaccia considerata è la default route o meno secondo quanto specificato da *value* (**yes** l'interfaccia è la default route, **no** l'interfaccia non è la default route);

DEMAND indica se l'interfaccia considerata deve permettere a `pppd` di stabilire una connessione al momento che un utente tenta di utilizzarla, secondo quanto specificato da *value* (**yes** tenta di stabilire una connessione, **no** la connessione deve essere stabilita manualmente);

IDLETIMEOUT

indica il numero di secondi di inattività dopo i quali l'interfaccia deve chiudere automaticamente la connessione secondo quanto specificato da *value*;

INITSTRING

indica la stringa di inizializzazione da passare al modem secondo quanto specificato da *value* (viene utilizzata essenzialmente per le interfacce di tipo SLIP);

LINESPEED

indica il baud rate dell'interfaccia secondo quanto specificato da *value* (valori tipici sono 9600, 19200, 38400 e 57600);

MODEMPORT

indica il file di dispositivo relativa alla porta alla quale è connesso il modem da utilizzare per la connessione secondo quanto specificato da *value*;

MTU

indica il valore della MTU (Maximum Transfer Unit) per l'interfaccia considerata, secondo quanto specificato da *value*;

NAME

indica il nome dell'insieme delle configurazioni relative alle connessioni di tipo dialup, secondo quanto specificato da *value*;

PAPNAME

indica lo username da utilizzare durante la comunicazione con il protocollo PAP (Password Authentication Protocol), secondo quanto specificato da *value*;

PERSIST

indica se l'interfaccia deve essere mantenuta attiva anche se la comunicazione è stata interrotta, secondo quanto specificato da *value* (**yes** l'interfaccia deve essere mantenuta sempre attiva, **no** l'interfaccia non deve necessariamente essere mantenuta sempre attiva);

REMIP

indica l'indirizzo IP del sistema remoto, secondo quanto specificato da *value* (*value* viene generalmente lasciato in bianco);

WVDIALSECT

indica il nome con il quale l'interfaccia deve essere associata alla configurazione di un dialer, secondo quanto specificato da *value* (il `/etc/wvdial.conf` contiene i numeri telefonici da comporre ed altre informazioni relative all'interfaccia);

Esistono comunque tool che permettono la gestione delle interfacce di rete in modo più user friendly. Ad esempio la Red Hat mette a disposizione il *Network Administration Tool* lanciabile con il comando `redhat-config-network`.

???

20.4 La routing table

La routing table di un sistema GNU/Linux viene gestita con il comando `route` (man page `route(8)`).

Comando: `route`
Path: `/sbin/route`

SINTASSI
`# route ???`

DESCRIZIONE

option indica la modalità di funzionamento di `route`. Può assumere i seguenti valori
`-v | --verbose`
;

???

20.5 La risoluzione dei nomi

Come illustrato precedentemente, gli indirizzi IP sono dei numeri e non sono così facili da ricordare come i nomi che magari hanno un significato che meglio può aiutare a ricordare il nome stesso. Pertanto è stato elaborato un meccanismo per associare un nome ad un indirizzo IP, in maniera tale che dal nome si possa univocamente ricondursi all'indirizzo IP ad esso associato.

La risoluzione dei nomi è il meccanismo con il quale l'indirizzo IP di un'interfaccia viene sostituito al suo corrispondente nome mnemonico. Esso si basa sulle impostazioni contenute in appositi file. Quando deve essere effettuata una risoluzione di nomi, come ad esempio quando si specifica un indirizzo relativo ad una pagina web (es. `http://www.google.it`), il sistema controlla, in sequenza, il contenuto dei seguenti file

1. `/etc/hosts`
2. `/etc/resolv.conf`

Il file `/etc/hosts` contiene l'associazione tra indirizzi IP e nomi mnemonici e costituisce quindi la tabella di lookup statica dei nomi mnemonici delle interfacce. Esso contiene righe con la seguente sintassi (v. RFC 952)

```
ip_address name_1 [name_2 [name_3 [...]]]
```

dove

ip_address

è l'indirizzo IP al quale possono essere associati uno o più nomi mnemonici;

name_i è uno dei possibili nomi mnemonici (alias) associati all'indirizzo IP specificato da *ip_address*. I nomi possono essere costituiti da qualunque carattere alfanumerico, e dai simboli '-' e '.';

Tale file conterrà ad esempio almeno la riga seguente

```
127.0.0.1    localhost.localdomain localhost
```

che specifica i nomi mnemonici di default relativi all'interfaccia di loopback.

Il file `/etc/resolv.conf` contiene gli indirizzi IP dei DNS, ai quali il sistema richiederà la risoluzione dei nomi. Esso è costituito da righe, ognuna delle quali contiene l'indirizzo IP (in decimal dotted notation) dell'interfaccia alla quale risponde un DNS.

???

20.6 Riferimenti

???

Capitolo 21

Gestione centralizzata degli utenti

???

21.1 NIS

???

21.2 LDAP - Lightweight Directory Access Protocol

???

21.3 Riferimenti

???

Capitolo 22

Cluster

???

22.1 Load balancing

???

22.2 High Availability (HA)

???

22.3 High Performance Computing (HPC)

???

22.4 Riferimenti

???

Parte III

Sicurezza

Capitolo 23

Protezione delle informazioni

“La porta meglio chiusa, è quella che si può lasciare aperta.”
– (Proverbio cinese)

In questo capitolo verranno trattati i concetti relativi la sicurezza del trasporto dell'informazione. Un'informazione è sicura quando una persona estranea alla comunicazione non può riuscire ad accedere all'informazione stessa. Questo è il campo dei sistemi di cifratura. Inoltre esiste anche un meccanismo che garantisce l'integrità dell'informazione e la sua non ripudiabilità da parte dell'autore della stessa, che va sotto il nome di firma digitale.

23.1 Concetti generali

La comunicazione tra due entità, avviene per mezzo di un flusso di informazioni che transita attraverso un canale. Tale canale può essere più o meno sicuro, dipendentemente dal fatto i chi ha la possibilità di accedervi. Nella comunicazione si identificano il *mittente*, il *destinatario* e gli altri, ovvero il *nemico*. Una canale è considerato sicuro quando il nemico non può accedervi.

Con la diffusione della comunicazione per via telematica, è sempre più sentita l'esigenza della sicurezza e riservatezza delle informazioni. Infatti, se con la posta ordinaria le informazioni riservate vengono inviate in busta chiusa che soltanto il destinatario è autorizzato ad aprire, con la posta elettronica vengono inviate informazioni su un canale insicuro, che non offre nessuna garanzia di riservatezza. È come se con la posta ordinaria si inviassero delle cartoline: chiunque può leggere il messaggio presente sul retro della cartolina stessa.

Per ottenere le garanzie relative alla sicurezza delle informazioni trasmesse, sono nati dei meccanismi per camuffare i messaggi agli occhi del *nemico* descritti nelle sezioni seguenti. Si parla in genere di tecniche che permettono di garantire alle informazioni veicolate da un mittente ad uno o più destinatari le seguenti caratteristiche

Riservatezza

le informazioni sono fruibili soltanto dal destinatario e non da altri;

Integrità

il destinatario deve essere in grado di verificare se le informazioni che gli sono arrivate hanno subito delle modifiche rispetto a quelle inviate dal mittente;

Autenticazione

il destinatario deve essere in grado di verificare se le informazioni ricevute sono state effettivamente inviate da chi afferma di essere il mittente;

Non ripudiabilità

il mittente che ha inviato le informazioni non può disconoscere di aver inviato le informazioni stesse;

23.2 Steganografia

steganografia

Il termine **steganografia** deriva dalle parole di origine greca $\sigma\tau\epsilon\gamma\alpha\nu\omicron\varsigma$ (*stèganos* = nascosto) e $\gamma\rho\alpha\phi\epsilon\iota\nu$ (*gràfein* = scrivere). È la scienza che studia i metodi per nascondere un'informazione in un'altra in modo tale che la sua presenza passi inosservata (non desti sospetto) per tutti coloro che non sono i destinatari dell'informazione stessa.

Tecniche di questo tipo venivano utilizzate già dagli antichi romani, che tra le righe di un messaggio ne scrivevano un altro utilizzando del succo di limone o del latte, che in seguito poteva essere riletto avvicinando il supporto cartaceo al calore di una fiamma. I greci, come riporta Erodoto, inviavano messaggeri ai quali era stato precedentemente tatuato il messaggio sulla testa ed arrivati a destinazione venivano loro rasati i capelli in modo da poter leggere il messaggio.

Il primo vero passo in avanti per quanto riguarda le tecniche steganografiche fu compiuto dall'abate Johann Heidenberg (1462-1516), conosciuto come Giovanni Tritemio, con i suoi trattati "Steganographia" e "Clavis Steganographiae", nei quali vengono gettate le basi per un modello di steganografia usata ancora oggi. Il sistema più semplice indicato da Tritemio è quello di considerare soltanto le prime lettere di ogni parola del testo per nascondere un messaggio. Ad esempio se si desidera comunicare il messaggio "scappa dal rifugio" con una tecnica di questo tipo, si dovrebbe scrivere un testo simile al seguente: "stiamo cercando amici per poter andare domani al lago, restando insieme forse un giorno intero ostinatamente". Leggendo infatti soltanto le prime lettere di ogni parola si ottiene il messaggio nascosto.

Tritemio elaborò una quarantina di sistemi steganografici che sfruttano varie combinazioni di acronimi e dei dischi rotanti basati sulla sostituzione alfabetica di Cesare, secondo la quale ogni lettera del messaggio deve essere sostituita dalla lettera che occupa, nell'alfabeto, la posizione della lettera considerata traslata di X posti, dove X è conosciuto sia dal mittente che dal destinatario (ad esempio se $X=3$ ogni lettera A che compare nel messaggio nascosto deve essere sostituita con la lettera D, ogni lettera B deve essere sostituita con la lettera E, ogni lettera C deve essere sostituita con la lettera F, ...). Ad esempio, il messaggio "Mio zio è andato a Zurigo non per incontrare Silvia e nemmeno le Kromaney, quindi domani si farà il solito giro nel centro storico. Dovrebbe mandarmi un kimono per sabato, e allora...", considerando la prima lettera di una parola sì e una no, nasconde il messaggio incomprensibile, "zazpsnkdffnsnmksa", che per mezzo della sostituzione di Cesare con $X=13$ (che quindi sostituisce le 'a' in 'o', le 'b' in 'p', le 'c' in 'q', ecc...) si trasforma in "nonfidartidicaio".

???

23.3 Crittografia

crittografia

La **crittografia**, dal greco $\kappa\rho\upsilon\pi\tau\omicron\varsigma$ (*kryptòs* = nascosto) e $\gamma\rho\alpha\phi\epsilon\iota\nu$ (*gràfein* = scrivere) è la scienza che studia i metodi per "cammuffare" le informazioni in maniera tale che chi eventualmente le legge non sia in grado di risalire all'informazione originale se non è a conoscenza della *chiave* di decodifica del messaggio cifrato. La **chiave** è appunto il valore di un parametro che fornito al meccanismo di cifratura considerato, è in grado di permettere la decodifica corretta dell'informazione. Questi metodi garantiscono la protezione delle informazioni trasmesse da un mittente ad un destinatario attraverso un canale insicuro, al quale chiunque può accedere, in particolare forniscono un certo livello di *riservatezza* (o *privacy*) alle informazioni trasmesse (v. fig. 23.1).

chiave

Uno dei primi metodi crittografici era utilizzato nell'antica Roma da Cesare, il quale inviava messaggi nei quali le singole lettere che componevano le parole erano sostituite con altre lettere dell'alfabeto secondo una certa regola: fissato un valore numerico X che conoscevano soltanto il mittente ed il destinatario, ogni lettera del messaggio originale veniva sostituita con la lettera che si trovava X posizioni più avanti nell'alfabeto. Ad esempio, considerando l'alfabeto inglese attuale, fissato $X = 3$, il messaggio "Ciao Giulio" diventa "Fldr Jltolr", infatti le lettere 'a' diventano 'd', le lettere 'b', diventano 'e', ecc.

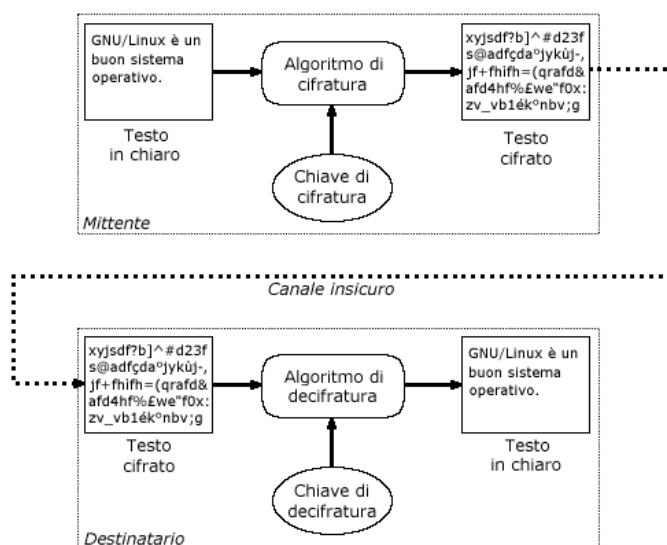


Figura 23.1: Schematizzazione dell'uso dei sistemi crittografici.

Lo studio dei metodi crittografici deriva dal calcolo matematico e si basa sull'elevata complessità delle operazioni di calcolo relativo alle funzioni inverse a quelle di cifratura, nel caso in cui non si conosca la chiave per decifrare il messaggio.

Creare un meccanismo di cifratura sicuro non è banale. Nel caso di crittografia si ipotizza che il nemico sia in grado non solo di accedere al canale attraverso il quale avviene la comunicazione, ma sia anche a conoscenza dell'algoritmo di cifratura (principio di Kerchoffs). Infatti, la robustezza dei metodi crittografici, non risiede nel meccanismo di cifratura stesso ma nella complessità della decifratura delle informazioni nel caso in cui non si conosca la chiave. Più un algoritmo di cifratura è noto, più viene testato e di conseguenza più è affidabile.

Nessun metodo di cifratura è *assolutamente* sicuro, in quanto esiste sempre la possibilità di decifrare il testo cifrato, anche senza conoscere la chiave a priori. Esistono infatti due metodi principali per tentare la decifratura delle informazioni senza conoscere la chiave: l'attacco per forza bruta o per dizionario. L'**attacco per forza bruta** è il meccanismo con il quale si tenta di decifrare l'informazione provando tutte le possibili chiavi di decifratura, fintantoché non si riesce a trovare quella giusta. È evidente che passando in rassegna tutte le possibili chiavi prima o poi si riesce a decifrare l'informazione: si tratta essenzialmente di stimare quant'è il "prima o poi". Da un punto di vista computazionale, esistono metodi di cifratura che per provare tutte le possibili chiavi di decifrazione, richiedono, con i sistemi ad oggi esistenti, un tempo massimo dell'ordine di decine di anni (da un punto di vista probabilistico è necessario considerare il fatto che mediamente la decifrazione del messaggio avviene dopo la metà dei tentativi di tutte le possibili chiavi, anche se c'è comunque la possibilità che la decifrazione dell'informazione avvenga al primo tentativo o dopo pochi tentativi). L'**attacco per dizionario** permette di decifrare l'informazione od ottenere la stessa cifratura considerando i valori più probabili della chiave, da cui il termine attacco per dizionario: un dizionario che contiene i valori più probabili utilizzati come chiave. In questo modo si possono effettuare un numero di tentativi di gran lunga inferiore a quelli relativi al totale delle chiavi possibili. Per questo è buona regola utilizzare chiavi che non riconducano facilmente a chi le ha utilizzate. Questo metodo è utilizzato generalmente per la scoperta di password (v. sez. 23.3.5).

attacco per forza bruta

attacco per dizionario

Problema delle patent... ???

23.3.1 Un po' di matematica

La teoria crittografica si basa sullo studio di funzioni matematiche non lineari, per poter così sfruttare il loro andamento non facilmente prevedibile. In particolare le funzioni utilizzate sono quelle modulari, ovvero quelle che utilizzano i resti delle divisioni intere. Si farà riferimento pertanto all'insieme dei numeri naturali $\mathbb{N} = \{1, 2, 3, 4, 5, 6, 7, \dots\}$. Sui concetti matematici di seguito esposti si basano i sistemi crittografici descritti nelle sezioni successive.

I moduli

congruenti modulo n

Dati tre numeri $a, b, n \in \mathbb{N}$ con $n > 0$, si dice che a e b sono **congruenti modulo n** e si scrive $a \cong b \pmod{n}$, se $a - b$ è divisibile per n , ovvero se $\frac{|a-b|}{n} \in \mathbb{N}$.

Alcuni esempi

$$36 \cong 0 \pmod{4}$$

$$23 \cong 3 \pmod{20}$$

$$16 \cong 1 \pmod{5}$$

È facile dimostrare le seguenti proprietà

$$a + b \cong c + d \pmod{m} \quad (23.1)$$

$$ab \cong cd \pmod{m} \quad (23.2)$$

classe di congruenza

Dati $a, n \in \mathbb{N}, n > 0$, si definisce **classe di congruenza** di a modulo n , l'insieme $C_n(a) \doteq \{\dots, a - 2n, a - n, a, a + n, a + 2n, \dots\} \subset \mathbb{N}$.

Per le relazioni 23.1 e 23.2, l'addizione e la moltiplicazione tra le classi di congruenza è ben definita

$$C_n(a) + C_n(b) = C_n(a + b)$$

$$C_n(a) \cdot C_n(b) = C_n(ab)$$

Si è soliti rappresentare $C_n(a)$ con il minimo valore \bar{a} appartenente all'insieme. Tale valore definisce l'operatore modulo mod che restituisce il resto della divisione intera tra due numeri, ovvero $a \text{ mod } n = \bar{a}$. La scrittura $a \text{ mod } n = b$ equivale a $a \cong b \pmod{n}$, ovvero significa che il resto della divisione intera a/n è b .

Di seguito sono riportati alcuni esempi

$$25 \text{ mod } 44 = 25$$

$$73 \text{ mod } 7 = 3$$

$$56 \text{ mod } 56 = 0$$

$$27 \text{ mod } 26 = 1$$

Si definisce Z_n come l'insieme delle classi di equivalenza rispetto ad un certo modulo, cioè $Z_n \doteq \{0, 1, 2, 3, 4, 5, \dots, n - 1\}$. Ad esempio $Z_5 = \{0, 1, 2, 3, 4\}$

Per la generica relazione $x \text{ mod } m = r$ valgono le seguenti proprietà

- $0 < r < m \quad \forall x \in \mathbb{N}$
- $x \text{ mod } m = x \quad \forall x < m$
- $x \text{ mod } x = 0 \quad \forall x \in \mathbb{N}$
- $(x + 1) \text{ mod } x = 1 \quad \forall x \in \mathbb{N}$
- $x \text{ mod } 1 = 0 \quad \forall x \in \mathbb{N}$
- $0 \text{ mod } m = 0 \quad \forall m \in \mathbb{N}$
- $(x + y) \text{ mod } m = (x \text{ mod } m + y \text{ mod } m) \text{ mod } m \quad \forall x, y \in \mathbb{N}$
- $xy \text{ mod } m = (x \text{ mod } m)(y \text{ mod } m) \text{ mod } m \quad \forall x, y \in \mathbb{N}$

Le ultime due relazioni indicano il fatto che trattandosi di resti, se uno dei due membri di un'equivalenza risulta maggiore di m , deve essere ulteriormente diviso per m per rispettare l'uguaglianza.

In particolare, dall'ultima proprietà deriva la proprietà del resto di un quadrato

$$x \bmod m = r \Rightarrow x^2 \bmod m = r^2 \quad \forall x \in \mathbb{N}$$

Un numero $n \in \mathbb{N}, n > 0$ è detto **primo** quando è divisibile solo per sé stesso e per 1, ovvero se $n \bmod m = 0 \Leftrightarrow m \in \{1, n\} \subset \mathbb{Z}_n$. Se n è divisibile per altri valori, è detto **composto**.

primo

composto

È facile verificare che 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 31 sono numeri primi.

Dunque, escluso il numero 2 (che è primo perché divisibile solo per 1 e per 2, cioè sé stesso), tutti i numeri primi sono dispari.

L'insieme dei numeri primi sarà indicato con il simbolo \mathcal{P} .

Verificare la primalità di un numero n significa verificare se $n \in \mathcal{P}$, e questo può essere fatto abbastanza velocemente sfruttando il *piccolo teorema di Fermat*, che afferma quanto segue

Sia $p \in \mathcal{P}$. Allora si ha

$$n^p \bmod p = n \quad \forall n \in [1, p-1] \subset \mathbb{N}$$

Ad esempio, il numero 5 è primo ed infatti

$$1^5 \bmod 5 = 1, 2^5 \bmod 5 = 2, 3^5 \bmod 5 = 3 \text{ e } 4^5 \bmod 5 = 4$$

Il teorema permette così, dato un numero, di stabilire se esso è composto. Infatti è sufficiente trovare un valore $b \in [1, p-1]$ per cui risulti $b^p \bmod p \neq b$ per poter affermare che il numero è composto. Ma se il numero soddisfa il piccolo teorema di Fermat, non è detto che sia un numero primo: il piccolo teorema di Fermat pone soltanto una condizione necessaria, ma non sufficiente per la verifica della primalità di un numero.

Infatti, per il numero 4, che non è primo, si ha

$$1^4 \bmod 4 = 1, 2^4 \bmod 4 = 0 \text{ e } 3^4 \bmod 4 = 1$$

ma anche 561 ($= 3 \times 11 \times 17$) e 1729 ($= 7 \times 13 \times 19$), ad esempio, pur essendo composti, soddisfano il piccolo teorema di Fermat.

I numeri composti che soddisfano il piccolo teorema di Fermat sono detti *numeri di Charmichael* (o pseudoprimi) e sono meno densi rispetto ai numeri primi.

Due numeri n ed m , con $n \neq m$ si dicono **primi tra loro** se non hanno nessun divisore comune, eccetto il numero 1.

primi tra loro

Le funzioni modulari

Le **funzioni modulari** sono funzioni $y = f(x) : \mathbb{N} \rightarrow \mathbb{N}$ che hanno la forma seguente

funzioni modulari

$$y = Ex \bmod m$$

dove E ed m sono dei valori costanti che definiscono la specifica funzione modulare considerata.

Poiché y è un resto, esso sarà un valore compreso tra 0 e $m-1$. Ad esempio, $y = 4x \bmod 7$ è una funzione modulare, della quale alcuni valori di x ed y sono riportati in tab. 23.1.

x	0	1	2	3	4	5	6	7	8	9
y	0	4	1	5	2	6	3	0	4	1

Tabella 23.1: Esempi dei valori assunti da una funzione modulare.

Si noti l'andamento "disordinato" dei valori assunti da y in corrispondenza di quelli di x .

Le funzioni modulari hanno varie proprietà, tra cui

$$x \bmod m = z \bmod k \Rightarrow kx \bmod m = kz \bmod m \quad \forall k \in \mathbb{N} \quad (23.3)$$

$$x \bmod m = z \bmod k \Rightarrow (k+x) \bmod m = (k+z) \bmod m \quad \forall k \in \mathbb{N} \quad (23.4)$$

La relazione 23.3 è evidente dal fatto che

$$kx \bmod m = (k \bmod m)(x \bmod m) = (k \bmod m)(z \bmod m) = kz \bmod m$$

mentre la 23.4 si ottiene da

$$(k+x) \bmod m = (k \bmod m) + (x \bmod m) = (k \bmod m) + (z \bmod m) = (k+z) \bmod m$$

Nel caso in cui E ed m siano primi tra loro, una funzione modulare $y = Ex \bmod m$ è invertibile, ovvero esiste una funzione inversa $x = Dy \bmod m$. Quindi

$$y = Ex \bmod m \Rightarrow Dy \bmod m = DEx \bmod m = x(DE \bmod m) \bmod m$$

cioè è sufficiente trovare un valore $D \in \mathbb{N} : DE \bmod m = 1$ affinché la relazione precedente diventi

$$Dy \bmod m = x \bmod m$$

Per determinare il valore D che dà la funzione modulare inversa, si può procedere con un algoritmo iterativo. Infatti, indicando con Q il quoziente della divisione intera tra DE ed m si può scrivere $DE = Qm + 1$, pertanto si può procedere facendo partire Q da 1 e calcolando il valore $D = \frac{Qm+1}{E}$ incrementando di volta in volta il valore di Q . Il processo può essere arrestato non appena D sarà un valore intero.

Ad esempio, la funzione $y = 11x \bmod 10800$ è invertibile poiché 11 e 10800 sono primi tra loro e la funzione inversa è $x = 5891y \bmod 10800$ (ottenuta con $Q = 6$).

funzione di Eulero

La **funzione di Eulero** $N(m) : \mathbb{N} \rightarrow \mathbb{N}$ dato un valore $m \in \mathbb{N}$ fornisce il numero di valori in $[1, m-1]$ che non hanno fattori in comune con m .

Ad esempio $N(9) = 6$, infatti sono 1, 2, 4, 5, 7, ed 8, ovvero 6 valori nell'intervallo $[1, 8]$ che non hanno fattori in comune con 9. Ed ancora $N(12) = 4$, infatti sono 1, 5, 7, ed 11, ovvero 4 valori nell'intervallo $[1, 11]$ che non hanno fattori in comune con 12.

Tale funzione gode delle seguenti proprietà

- $N(m) = m - 1 \quad \forall m \in \mathcal{P}$
- $N(m) = \prod_{i=1}^n (p_i - 1) \quad \forall m = \prod_{i=1}^n p_i \quad \text{con } p_i \in \mathcal{P}$

Ad esempio $10 = 2 \times 5 \Rightarrow N(10) = (2-1)(5-1) = 1 \times 4 = 4$, infatti i numeri che non hanno divisori in comune con 10 nell'intervallo $[1, 9] \subset \mathbb{N}$ sono 4 (1, 3, 7 e 9).

funzioni modulari
esponenziali

Le **funzioni modulari esponenziali** sono funzioni $y = f(x) : \mathbb{N} \rightarrow \mathbb{N}$ che hanno la forma seguente

$$y = x^E \bmod m$$

dove E ed m sono dei valori costanti che definiscono la specifica funzione modulare esponenziale considerata.

Anche in questo caso, trattandosi di un resto, y assumerà valori compresi tra 0 e $m-1$.

Le funzioni modulari esponenziali sono invertibili soltanto se E ed $N(m)$ sono primi fra loro.

Dalla 23.3 si ha che

$$x \bmod m = r \Rightarrow x^k \bmod m = r^k \bmod m$$

e tenendo conto di un teorema della teoria dei numeri si può scrivere che

$$y = x^E \bmod m = x^{E \bmod N(m)} \bmod m$$

quindi

$$y^D \bmod m = x^{ED \bmod N(m)} \bmod m$$

Per trovare la funzione inversa è sufficiente trovare un valore $D \in \mathbb{N}$ tale che $DE \bmod N(m) = 1$, ovvero $DE = QN(m) + 1$ (dove Q è il quoziente della divisione intera tra DE ed $N(m)$). Quindi, analogamente a quanto descritto per le funzioni modulari semplici, si può procedere con moltiplicazioni successive, calcolando $D = \frac{QN(m)+1}{E}$ con Q che vale inizialmente 1 ed incrementandolo ogni volta di una unità, fintantoché D non risulta essere un numero intero.

Ad esempio, la funzione modulare esponenziale inversa di $y = x^{1183} \bmod 2867$ è $x = y^7 \bmod 2867$.

Le curve ellittiche

Esistono sistemi di cifratura che si basano sui problemi connessi con le curve ellittiche, proposti per la prima volta da V. Miller e N. Koblitz, verso la metà degli anni '80. Come per gli algoritmi che si basano sulle funzioni modulari, gli algoritmi basati sulle curve ellittiche basano la loro robustezza sulla complessità operativa nella ricerca di una funzione matematica inversa. In particolare, dati due punti G e Y su una curva ellittica tale che $Y = kG$ non è banale trovare il valore intero k . Tale problema è noto come il problema del logaritmo discreto delle curve ellittiche.

Al momento attuale i metodi per il calcolo dei logaritmi discreti delle curve ellittiche sono molto meno efficienti rispetto a quelli per la fattorizzazione dei numeri o per il calcolo dei logaritmi discreti standard. Quindi per avere la stessa sicurezza degli algoritmi basati sul calcolo modulare, sono insufficienti chiavi di lunghezza inferiore.

???

23.3.2 Cifratura a chiave simmetrica

La cifratura a chiave simmetrica utilizza una sola chiave sia per la fase di cifratura che per quella di decifratura dell'informazione. Quindi solo il mittente ed il destinatario devono conoscere entrambi la stessa chiave per potersi scambiare informazioni cifrate, per questo tale metodo è detto anche *cifratura a chiave segreta*.

Il messaggio viene cifrato dal mittente per mezzo di una chiave ed inviato al destinatario, il quale è in grado di decifrarlo soltanto se è a conoscenza della chiave utilizzata dal mittente (v. fig. 23.2).

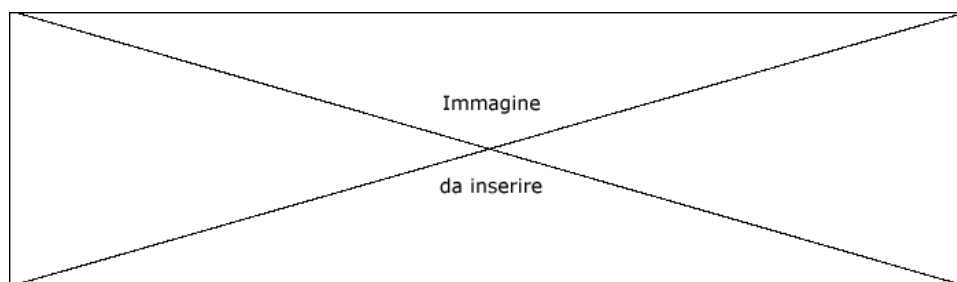


Figura 23.2: Schematizzazione della comunicazione con cifratura a chiave simmetrica.

Il problema relativo a questo tipo di cifratura, consiste essenzialmente nella modalità di scambio della chiave tra il mittente ed il destinatario. Infatti essa deve necessariamente essere scambiata attraverso un canale sicuro, altrimenti anche il nemico conoscerebbe la chiave potendo così decifrare agevolmente le informazioni cifrate trasmesse dal mittente.

Il più semplice esempio di sistema a cifratura simmetrica è quello a lettere permutate: ogni lettera dell'alfabeto del messaggio in chiaro viene fatta corrispondere ad un'altra lettera in modo biunivoco. La chiave è la relazione tra le lettere in chiaro e quelle permutate. Il sistema può sembrare inespugnabile: la totalità delle chiavi (tutte le permutazioni possibili di ogni lettera) è $26!$ ovvero circa 4×10^{26} . Se il nemico impiegasse anche soli $1 \mu s$ per provarne una, provarle tutte vorrebbe dire avere a disposizione un

tempo dell'ordine di 13×10^{12} anni. In realtà il sistema non è robusto quanto sembra. Infatti, considerando la regolarità statistica delle lettere della lingua, ad esempio, inglese, si hanno le frequenze di impiego riportate in tab. 23.2.

Lettera	Frequenza
A	0,114
E	0,111
I	0,104
O	0,099
T	0,088

Tabella 23.2: Frequenza statistica di alcune lettere nella lingua inglese.

Quindi, calcolando l'utilizzo delle lettere nel messaggio cifrato si può sostituire, tenendo conto della frequenza statistica, ad ognuna di esse la lettera che probabilmente sarà quella in chiaro. In questo modo si riesce agevolmente a ricostruire il messaggio originale e pertanto tale meccanismo di cifratura non è affatto sicuro.

La cifratura simmetrica si distingue anche in base all'applicazione dell'algoritmo sul testo in chiaro. Si parla di cifratura a flusso¹ (*stream cipher* o *state cipher*) quando l'algoritmo di cifratura viene applicato ad un bit o ad un byte alla volta. Se l'algoritmo viene applicato a gruppi di bit o di byte sempre della stessa dimensione, si parla di cifratura a blocchi (*block cipher*). La cifratura più utilizzata è quella a blocchi.

Gli algoritmi di cifratura a blocchi si suddividono ulteriormente nelle seguenti categorie

ECB - Electronic CodeBook

è la categoria che raccoglie la cifratura a blocchi più semplice ma anche meno affidabile: ogni blocco viene cifrato sempre con la stessa chiave in successione. A questa categoria appartengono lo stragrande maggioranza degli algoritmi di cifratura utilizzati ad oggi;

CBC - Cipher Block Chaining

i metodi di cifratura che appartengono a questa categoria collegano i blocchi cifrati ai precedenti nel seguente modo: il blocco cifrato corrente viene ottenuto dalla cifratura del blocco di testo in chiaro considerato in XOR con il blocco cifrato precedente;

CFB - Cipher FeedBack

anche gli algoritmi appartenenti a questa categoria collegano i blocchi con i precedenti ma lo fanno nel seguente modo: il blocco cifrato corrente viene ottenuto dallo XOR di parte del blocco in chiaro considerato con il blocco cifrato precedente (l'idea è quella di elaborare i dati non appena sono disponibili anziché aspettare che sia completata l'elaborazione di un intero blocco);

OFB - Output FeedBack

a questa categoria appartengono le codifiche a blocchi più veloci: i blocchi sono sempre collegati con i precedenti, ma il collegamento avviene tra l'output del passo precedente ed il blocco considerato. Viene utilizzato nelle comunicazioni ad alta velocità (come quelle dei satelliti);

Algoritmi di cifratura

Di seguito sono riportati alcuni tra gli algoritmi più utilizzati per la cifratura a chiave simmetrica.

DES / DEA

(Data Encryption Standard / Data Encryption Algorithm) creato dall'*IBM* nel 1977 e modificato dalla NSA (National Security Agency) per essere adottato

¹la cifratura a flusso viene generalmente realizzata in hardware tramite un LFSR (Linear Feedback Shift Register).

come standard federale negli Stati Uniti (FIPS – Federal Information Processing Standard – 46-3). Il DEA² è una versione migliorata del DES, tant'è che spesso si fa riferimento a quest'ultimo come DES. È l'algoritmo a chiave simmetrica più studiato, conosciuto ed utilizzato in tutto il mondo.

Si basa su una chiave di 64 bit, suddivisa in blocchi da 8 bit (l'ultimo bit di ogni blocco è considerato come codice di controllo ed ignorato durante la fase di cifratura/decifratura).

La cifratura del messaggio viene effettuata suddividendo lo stesso a blocchi di 64 bit e suddividendo ulteriormente ogni blocco in 8 sottoblocchi da 8 bit. Se il blocco da cifrare non raggiunge i 64 bit, viene effettuato un padding, considerando i bit mancanti come tutti a 0, oppure ci sono versioni che integrano il blocco di bit con il valore opposto a quello dell'ultima parte del blocco, e versioni che riempiono i bit mancanti con valori casuali, inserendo nell'ultimo byte il numero di bit casuali aggiunti.

Un blocco da cifrare viene innanzitutto trasposto, ovvero la sua posizione nel file viene cambiata con quella di un altro, quindi suddiviso in due sottoblocchi di 32 bit. Dopodiché viene applicata una serie di 16 passi che operano ricorsivamente trasposizioni e sostituzioni dei bit di ogni sottoblocco in base a delle sottochiavi derivate dalla chiave di 64 bit, diverse per ogni passaggio. In ogni passo, l'output del sottoblocco sinistro diviene l'input del sottoblocco destro e viceversa.

Offre una protezione relativamente facile da eludere a causa del fatto che la chiave è piuttosto piccola (v. “Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design”, O'Reilly).

3DES

(Triple-DES / TDEA) è l'applicazione ripetuta per tre volte del DES. Lo standard ANSI X9.52 definisce la cifratura Triple-DES con le chiavi k_1 , k_2 e k_3 come $C = E_{k_3}(D_{k_2}(E_{k_1}(M)))$, dove E_{k_i} e D_{k_i} sono rispettivamente la funzione di cifratura e di decifratura DES operata con la chiave k_i , M è il messaggio in chiaro e C è quello cifrato. Questo standard è spesso riferito con il nome DES-EDE (esiste anche la variante DES-EEE che opera tre cifrature consecutive con il DES).

Per le chiavi, lo standard ANSI X9.52 prevede tre opzioni possibili

- k_1 , k_2 e k_3 (chiavi indipendenti);
- $k_1 = k_3$ e k_2 (due chiavi indipendenti);
- $k_1 = k_2 = k_3$ (una sola chiave);

Considerando l'ultima opzione elencata, la cifratura 3DES è compatibile con la DES.

Lo studio sul 3DES, indica che la cifratura DES ripetuta più volte non fornisce comunque un sostanziale aumento della sicurezza delle informazioni cifrate.

RC6 è un algoritmo di cifratura a blocchi creato nel 1998 da R. Rivest, M. Robshaw, R. Sidney e Y. Yin, come miglioramento di RC5.

Utilizzando una chiave a 128 bit, si basa, come RC5, sulle rotazioni dei dati, che lo rende un algoritmo di cifratura piuttosto veloce.

IDEA

(International Data Encryption Algorithm) conosciuto anche con il nome IPES (Improved Proposed Encryption Standard) è un algoritmo creato da X. Lai e J. Massey nel 1991. Si tratta di un sistema di cifratura a blocchi di 64 bit come il DES, con chiave di 128 bit, che ne aumenta la robustezza.

²standard ANSI X3.92.

La cifratura avviene, analogamente a quella del DES, suddividendo il messaggio da cifrare in blocchi di 64 bit e suddividendolo successivamente in 4 sottoblocchi di 16 bit. Ogni sottoblocco viene sottoposto ad 8 passi nei quali sono coinvolte 52 sottochiavi diverse ottenute dalla chiave a 128 bit.

IDEA è al momento sistema di cifratura a chiave simmetrica più utilizzato dai software commerciali di crittografia per la sua velocità di codifica/decodifica e l'elevata sicurezza offerta.

Blowfish

creato nel 1993 da B. Schneier, è un algoritmo di cifratura a blocchi di 64 bit, nato nell'intento di fornire un'alternativa libera e veloce ai sistemi di cifratura già esistenti. Supporta chiavi di lunghezza da 32 fino a 448 bit.

Si tratta di un software più veloce di DES ed IDEA, libero e gratuitamente disponibile per chiunque.

AES (Advanced Encryption Standard) è il sistema di cifratura standard FIPS 197 successore del DES. Nel 2001 è stato scelto l'algoritmo di cifratura Rijndael tra quelli proposti (MARS di IBM, RC6 di *RSA Laboratories*, Rijndael di J. Daemen e V. Rijmen, Serpent di R. Anderson, EBiham e L. Knudsen, Twofish di B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall e N. Ferguson) in seguito all'annuncio dell'iniziativa del 1997 e lo standard è stato pubblicato nel 2002. L'intento del NIST è quello di avere un sistema di cifratura che rimarrà sicuro a lungo in questo secolo.

L'AES supporta chiavi di lunghezza 128, 192 e 256 bit.

Gli algoritmi di cifratura a chiave simmetrica hanno la caratteristica di essere molto veloci nelle fasi di cifratura e decifratura.

???

23.3.3 Cifratura a chiave asimmetrica

La cifratura a chiave asimmetrica utilizza due chiavi diverse: una per la cifratura dell'informazione e l'altra per la decifratura. Le informazioni cifrate con una delle due chiavi possono essere decifrate solo con l'altra. Le due chiavi prendono generalmente il nome di **chiave pubblica** e **chiave privata** dal fatto che una delle due chiavi deve essere fornita a tutti quelli dai quali si vogliono ricevere messaggi cifrati, mentre l'altra deve essere tenuta segretamente nascosta dal proprietario della coppia di chiavi (è indipendente quale delle due chiavi si scelga come pubblica e quale come privata, ma una volta fatta la scelta è per sempre). Tale metodo, per contrapposizione al precedente è anche detto *cifratura a chiave pubblica*.

Il messaggio viene cifrato dal mittente per mezzo della chiave pubblica del destinatario ed inviato al destinatario stesso, il quale è in grado di decifrarlo poiché è a conoscenza della sua chiave privata (v. fig. 23.3).

chiave pubblica
chiave privata

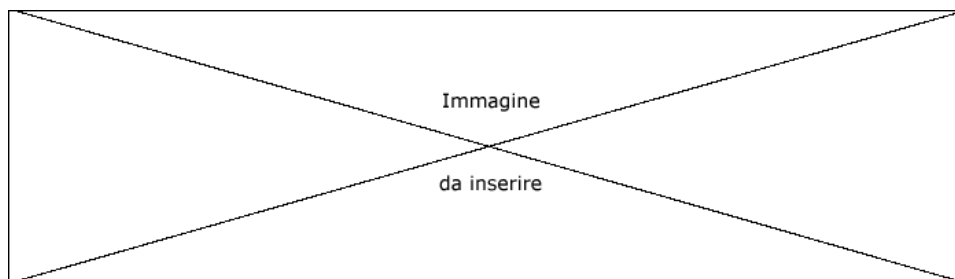


Figura 23.3: Schematizzazione della comunicazione con cifratura a chiave asimmetrica.

In questo caso lo scambio della chiave pubblica può avvenire anche attraverso un canale insicuro, poiché chi reperisce la chiave pubblica relativa ad un'entità può soltanto

inviare informazioni cifrate a tale entità, infatti soltanto chi conosce la chiave privata può decifrare le informazioni cifrate con la chiave pubblica.

La sicurezza del meccanismo di cifratura deriva dalla difficoltà di derivare una delle due chiavi conoscendo l'altra. Gli algoritmi di cifratura si basano sul fatto che al momento non esiste alcun metodo in grado di effettuare in tempi brevi la fattorizzazione in numeri primi. In sostanza, considerando numeri primi molto grandi e facendone il prodotto, trovare i due numeri primi nei quali è scomponibile il risultato ha un'elevata complessità computazionale.

Algoritmi di cifratura

Di seguito sono riportati alcuni tra gli algoritmi più utilizzati per la cifratura a chiave simmetrica.

RSA l'algoritmo creato nel 1977, prende il suo nome dalle iniziali dei suoi inventori (tre docenti del MIT): R. Rivest, A. Shamir e L. Adelman.

La generazione delle chiavi viene effettuata con i seguenti passi

1. Scegliere due numeri primi p e q abbastanza grandi e se ne calcoli il prodotto $n = p \times q$, detto modulo;
2. Scegliere un numero d tale che $d > 1$ e che non abbia fattori a comune con $(p-1)(q-1)$, cioè d e $(p-1)(q-1)$ siano primi tra loro;
3. Calcolare e per il quale risulti $ed \bmod (p-1)(q-1) = 1$, ovvero $(ed-1)$ sia divisibile per $(p-1)(q-1)$;

La chiave pubblica è rappresentata da (n, e) e quella privata da (n, d) .

La cifratura dei messaggi avviene seguendo i passi di seguito riportati

1. Codificare il messaggio M , in maniera nota e reversibile, in un numero $m : m < n$;
2. Calcolare $c = m^e \bmod n$, dove (n, e) è la chiave pubblica del destinatario;

Il messaggio cifrato è c .

La decifratura di un messaggio c avviene mediante i seguenti passi

1. Calcolare $m = c^d \bmod n = m^{ed} \bmod n = m \bmod n = m$ ($m < n$), dove (n, d) è la chiave privata del destinatario di c ;

La sicurezza del metodo di cifratura RSA deriva dal fatto che attualmente non è stato trovato un metodo per la fattorizzazione di numeri primi molto grandi che sia computazionalmente non molto complesso.

La lunghezza della chiave si riferisce generalmente al numero di bit necessari alla rappresentazione del valore n (il modulo). I due numeri primi p e q il cui prodotto è n , dovrebbero essere dello stesso ordine di grandezza, ma non troppo "vicini" tra loro (la loro differenza non deve essere un numero "piccolo"), in modo da rendere più difficile la fattorizzazione di n .

La lunghezza della chiave del metodo di cifratura RSA, dipende dalla sicurezza che si desidera ottenere con questo meccanismo: quanto sono importanti le informazioni trasmesse e per quanto tempo devono rimanere "segrete" agli altri. Più lunga è la chiave, più sicuro è il metodo, ma più tempo occorre per cifrare e decifrare i messaggi.

Un test di tentativo per la rottura del sistema RSA è stato effettuato nel 1999 con una lunghezza della chiave di 512 bit. La chiave è stata fattorizzata con successo in 7 mesi. Questo indica che le chiavi di 512 bit non si possono ritenere ad oggi molto sicure. In genere si utilizzano chiavi di 1024 bit per la generazione di chiavi di cifratura per la protezione di informazioni di importanza a livello aziendale e

chiavi di lunghezza superiore, ad esempio 2048 bit, per la generazione di chiavi con importanza più elevata (ad es. la coppia di chiavi a livello di amministratore utilizzato come autorità di certificazione).

ElGamal

è un metodo che prende il nome da quello del suo autore ed è basato sul problema dei logaritmi discreti.

Per generare la coppia di chiavi si sceglie a caso un numero primo p grande, per cui il problema dei logaritmi discreti risulti intrattabile in Z_p , quindi si sceglie g una radice modulo p (quindi g genera Z_p). Si sceglie un valore k tale che $1 < k < p - 1$. Quindi si calcola $h = g^k \bmod p$. La chiave pubblica è rappresentata dalla terna (p, g, h) e quella privata dalla terna (p, g, k) .

Un messaggio M viene codificato, in maniera nota e reversibile, in un numero $m : 1 \leq m \leq p - 1$ e quindi, scelto un valore $s : 1 < s < p - 1$, cifrato per mezzo di una chiave pubblica (p, g, h) calcolando $c_1 = g^s \bmod p$ e $c_2 = mh^s \bmod p$. Il messaggio cifrato è rappresentato da (c_1, c_2) .

Un messaggio cifrato (c_1, c_2) viene decifrato per mezzo della relativa chiave privata (p, g, k) calcolando $c_1^{-k} c_2 \bmod p = (g^s)^{-k} mh^s \bmod p = (g^k)^{-s} mh^s \bmod p = h^{-s} mh^s \bmod p = m \bmod p$.

DSA (Digital Signature Algorithm) è un algoritmo creato da Schnorr e ElGamal nel 1994 e si basa sul problema dei logaritmi discreti. Spesso è riferito anche come DSS (Digital Signature Standard) del NIST (FIPS 186). È nato appositamente come algoritmo per la generazione di firme digitali ed utilizza chiavi con lunghezza da 512 a 1024 bit.

L'operazione di cifratura è più veloce di quella per la verifica della cifratura stessa (mentre con RSA la verifica della cifratura è molto più veloce rispetto alla generazione della cifratura stessa) se gli esponenti pubblico e privato sono scelti in maniera opportuna (come avviene usualmente). Non è vantaggioso poiché sono più le volte che di un messaggio si deve verificarne la firma rispetto al numero di volte che questo deve essere firmato (una soltanto).

Il metodo di cifratura è analogo a quello dell'algoritmo *ElGamal*.

La generazione delle chiavi è effettuata con il seguente metodo

1. Scegliere un numero primo p di L bit con $512 \leq L \leq 1024$ e $L \bmod 64 = 0$;
2. Scegliere un numero primo q di 160 bit, tale che $p - 1 = qz$ con $z \in \mathbb{N}$;
3. Scegliere $h \in \mathbb{N}$ con $1 < h < p - 1$ tale che $g = h^z \bmod p < 1$;
4. Scegliere x a caso, con $0 < x < q$;
5. Calcolare $y = g^x \bmod p$;

La chiave pubblica è rappresentata da (p, q, g, y) e quella privata da x .

La firma di un messaggio avviene secondo i seguenti passi

1. Scegliere un valore a caso s (detto *nonce*³), con $1 < s < q$;
2. Calcolare $s_1 = (g^s \bmod p) \bmod q$;
3. Calcolare $s_2 = (H(m) - s_1 x) s^{-1} \bmod q$, dove $H(m)$ è la funzione hash SHA-1 applicata al messaggio m .

La firma è rappresentata da (s_1, s_2) .

La verifica della firma avviene seguendo i seguenti passi

1. Calcolare $w = (s_2)^{-1} \bmod q$;
2. Calcolare $u_1 = H(m)w \bmod q$;

³nel gergo relativo alla sicurezza un *nonce* sta a significare "number used once".

3. Calcolare $u_2 = s_1 w \bmod q$;
4. Calcolare $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$;

la firma è autentica se $v = s_1$.

Gli algoritmi di cifratura a chiave asimmetrica hanno la caratteristica di non essere molto veloci nelle fasi di cifratura e decifratura. Per questa loro caratteristica vengono generalmente utilizzati soltanto nella fase di autenticazione tra le due entità che vogliono comunicare tra loro. In genere il mittente invia un messaggio casuale detto **challenge** *challenge* e lo cifra con la chiave pubblica del destinatario. Solo il destinatario è così in grado di decifrarlo con la sua chiave privata (e magari rinviarlo al mittente, anche in chiaro, per conferma). Il mittente è così sicuro di parlare con il destinatario.

Dopo aver superato la fase di autenticazione, il mittente invia al destinatario, nella maniera illustrata precedentemente, una chiave simmetrica con la quale cifrare i messaggi che verranno scambiati successivamente nella comunicazione. In questo modo si sfrutta la velocità degli algoritmi di cifratura simmetrica, superandolo scoglio relativo al passaggio della chiave.

Un noto protocollo per lo scambio di chiavi è stato ideato da W. Diffie e M. Hellman (v. W. Diffie and M. E. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory 22, 1976).

???

23.3.4 Protocollo Diffie-Hellman

Il protocollo Diffie-Hellman (dal nome degli autori) è utilizzato per lo scambio di chiavi simmetriche attraverso un canale insicuro. Si basa sulle seguenti considerazioni: siano p un numero primo e g un numero intero (generatore) tale che $g < p$ e $\forall n \in [1, p-1] \exists z \in \mathbb{N} : n = g^z \bmod p$. Tali valori sono conosciuti da chiunque (pubblici). Sia $V = \{1, \dots, p-1\} \subset \mathbb{N}$.

Si supponga che due persone A e B vogliano scambiarsi un messaggio in maniera sicura su un canale insicuro. Secondo il protocollo Diffie-Hellman, A sceglie un numero $a_s \in V$ (valore privato) con il quale genera un valore pubblico

$$a_p = g^{a_s} \bmod p$$

In maniera analoga, B sceglie un numero $b_s \in V$ (valore privato) con il quale genera un valore pubblico

$$b_p = g^{b_s} \bmod p$$

Quindi A e B si scambiano i relativi valori pubblici a_p e b_p . In questo modo A può calcolare

$$b_p^{a_s} = g^{a_s b_s} \bmod p = k$$

e B può calcolare

$$a_p^{b_s} = g^{a_s b_s} \bmod p = k$$

ovvero entrambi sono in grado di ricavare la chiave comune k .

La sicurezza dipende dal fatto che è computazionalmente complesso calcolare $k = g^{a_s b_s} \bmod p$ dati i valori pubblici $a_p = g^{a_s} \bmod p$ e $b_p = g^{b_s} \bmod p$, se p è un numero primo sufficientemente grande (Maurer ha mostrato che rompere la cifratura Diffie-Hellman equivale al calcolo dei logaritmi discreti⁴).

Il protocollo di scambio di chiavi Diffie-Hellman è vulnerabile ad attacchi di tipo man-in-the-middle. Infatti, si supponga che C intercetti il valore pubblico di A , a_p ed invii a B il suo valore pubblico c_p in modo che B creda che sia a_p . Allo stesso modo, se C intercetta il valore pubblico di B , b_p , inoltra ad A il suo valore pubblico c_p in modo che A creda che sia b_p . In questo modo A e C si scambiano una chiave e C e B se ne scambiano un'altra e C può decifrare i messaggi che A crede di inviare a B .

⁴v. U. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, Advances in Cryptology - Crypto '94, Springer-Verlag (1994), 271-281.

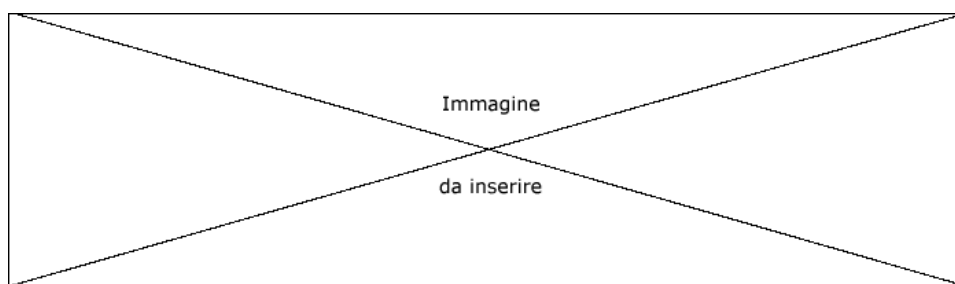


Figura 23.4: Schematizzazione della comunicazione col protocollo Diffie-Hellman.

(e magari inoltrarli a B cifrati con la chiave scambiata con B) ed i messaggi che B crede di inviare ad A (e magari inoltrarli ad A cifrati con la chiave scambiata con A). Il protocollo Diffie-Hellman di base non fornisce nessun meccanismo di autenticazione dei partecipanti.

Esiste una versione modificata del protocollo, detta *authenticated Diffie-Hellman key agreement protocol* o protocollo STS (Station-to-Station), sviluppato da W. Diffie, P. Van Oorschot e M. Wiener nel 1992 ed utilizza la firma digitale ed i certificati digitali.

Prima di eseguire lo scambio di chiavi con il protocollo precedentemente descritto, A e B sono ognuno in possesso di una coppia di chiavi (pubblica/privata) ed un certificato digitale relativo alla chiave pubblica. Durante lo scambio, A esegue una firma dei messaggi inviati che contengono la sua chiave pubblica a_p e lo stesso fa B . In questo modo anche se C riceve i messaggi di A e B non è in grado di firmarli a nome degli altri, quindi se li modifica, A e B si accorgono che i messaggi che ricevono non corrispondono a quelli inviati dal mittente.

???

23.3.5 Funzioni hash

Esistono dei meccanismi di cifratura che permettono soltanto di “cammuffare” un’informazione senza prevedere una chiave per poter successivamente decodificare il messaggio. Tali metodi sono detti **funzioni hash** o cifratura a senso unico (*one-way hash function*) e sono utilizzati per la cifratura delle password o per la generazione di impronte digitali o *message digest* (v. sez. 23.4).

Le password, infatti, non sono memorizzate in chiaro sul sistema, ma viene memorizzata soltanto la loro cifratura, dalla quale non è possibile ritornare alla password in chiaro. Quando un utente vuole accedere al sistema, il meccanismo di autenticazione richiede la password, la cifra con l’algoritmo di hashing considerato e confronta la password cifrata con quella memorizzata sul sistema per l’utente in questione. Se le due password cifrate coincidono, all’utente è permesso accedere al sistema.

Il message digest invece è un valore caratteristico di un messaggio o documento, calcolato applicando la funzione hash al messaggio stesso, che varia sensibilmente al variare del contenuto del messaggio.

L’algoritmo di hashing deve essere in grado di generare un messaggio cifrato quanto più scorrelato possibile dall’informazione originale e tale che sia estremamente improbabile che due messaggi diversi diano luogo allo stesso messaggio cifrato.

???

Algoritmi di cifratura

Di seguito sono riportati alcuni tra gli algoritmi più utilizzati per la cifratura con funzioni hash.

MD5 (Message Digest 5) creato da R. Rivest nel 1991, è nato per sostituire MD4, che a sua volta era il successore di MD2. Si tratta essenzialmente di MD4 con “cinture di sicurezza”: leggermente più lento di MD4 ma più sicuro.

Produce valori hash di 128 bit, accettando messaggi di input con lunghezza massima di 2^{64} bit e scomponendoli in blocchi di 512 byte (viene eventualmente effettuato un padding dei bit mancanti nell'ultimo blocco – il padding comprende comunque un valore di 64 bit che indica la lunghezza del messaggio originale). Una descrizione dell'algoritmo è riportata nella RFC 1321.

SHA (Secure Hash Algorithm) è un algoritmo sviluppato nel 1994 dal NIST (National Institute of Standard and Technology), FIPS 180, e dalla NSA (National Security Agency), molto simile a MD4.

Fornisce valori hash di 160 bit. La cifratura viene effettuata su messaggi con lunghezza massima di 2^{64} bit, scomponendoli in blocchi di 264 bit. Risulta un po' più lento di MD5, ma produce un message digest più grosso che lo rende più sicuro agli attacchi per forza bruta.

SHA-1 è una revisione del 1995 dell'algoritmo SHA, che ne corregge alcune falle (v. FIPS 180-1). Una sua descrizione è contenuta in nello standard ANSI X9.30 (part 2).

SHA-2 è una nuova versione dell'algoritmo SHA, pubblicata nel 2002, che fornisce valori hash di 256 bit.

RIPEMD

(Race Integrity Primitives Evaluation Message Digest) è un algoritmo sviluppato da H. Dobbertin, A. Bosselaers e B. Preneel nel 1996. È nato per rimpiazzare MD4. Esistono varie versioni: RIPEMD-128, RIPEMD-160, RIPEMD-256 e RIPEMD-320 con lunghezza della chiave rispettivamente di 128, 160, 256 e 320 bit. Si tratta essenzialmente di un algoritmo con caratteristiche e velocità paragonabili a quelle di SHA-1.

???

23.3.6 PKI - Public Key Infrastructure

La **PKI** (Public Key Infrastructure) è un insieme di software che permette ad un utente di cifrare/decifrare messaggi con la sua chiave pubblica/privata e di firmare i messaggi con la sua chiave privata. PKI

Un esempio di PKI è GPG (GNU Privacy Guard) (v. sez. ??) o PGP (Pretty Good Privacy). Questi software, come molti altri sistemi PKI, usano certificati firmati da sé stessi.

Molte aziende utilizzano PKI che si basano su gerarchie di certificati che determinano l'identità delle varie entità, sia dell'azienda che esterne ad essa.

Le PKI possono essere utilizzate per vari scopi, tra i quali

- cifratura e/o autenticazione del mittente della posta elettronica;
- cifratura e/o autenticazione dei documenti;
- autenticazione degli utenti e dei client con SSL;
- autenticazione per l'instaurazione di canali sicuri;

23.4 La firma digitale

I meccanismi descritti in precedenza permettono di garantire la riservatezza delle informazioni, ma non certificano che il messaggio sia stato effettivamente inviato dal mittente ovvero l'autore. Mettendo insieme i vari meccanismi di base descritti precedentemente, si può definire un ulteriore meccanismo che garantisca la autenticità del messaggio: la **firma digitale** (*digital signature*).

firma digitale

Si supponga che un mittente S debba inviare un messaggio M ad un destinatario D, in maniera tale che solo D sia in grado di leggere (riservatezza). Dal suo punto di

vista, il destinatario D vuole essere sicuro che il messaggio che riceverà sia effettivamente stato inviato dal mittente S e non sia stato modificato successivamente (autenticità). Per ottenere tutto ciò, S invierà il messaggio M cifrato con la chiave pubblica di D, in maniera tale che soltanto D (che è l'unico che dovrebbe possedere la propria chiave privata) possa decifrarlo. Inoltre, S crea un'impronta digitale (message digest) del messaggio in chiaro M, per mezzo di una funzione hash, e la cifra con la propria chiave privata, ottenendo così la firma digitale che aggiunge al messaggio cifrato. In questo modo D può decifrare la firma digitale con la chiave pubblica di S, ottenendo l'impronta di M inviata da S, e con la stessa funzione hash usata da S può ricalcolare l'impronta digitale del messaggio ricevuto e verificare che le due impronte coincidano. Questo garantisce l'integrità del messaggio (v. fig. 23.5).

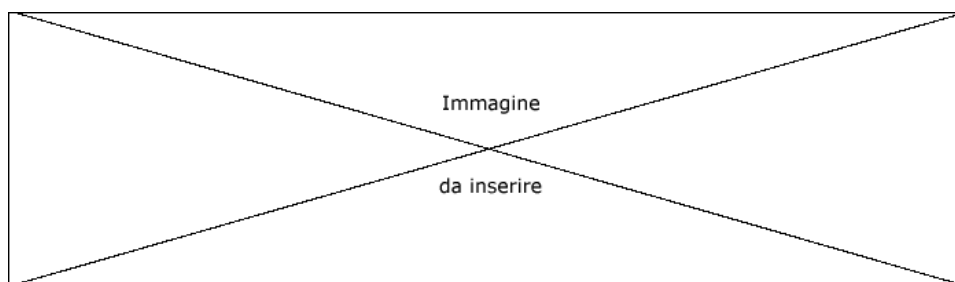


Figura 23.5: Schematizzazione della comunicazione utilizzando la firma digitale.

Questo meccanismo garantisce anche l'autenticità e la non ripudiabilità del messaggio da parte del suo autore (S), una volta che si è sicuri che la chiave pubblica relativa a quella privata con cui è stata cifrata l'impronta di M sia effettivamente la chiave pubblica di S. Tale sicurezza è garantita da una terza parte: l'autorità di certificazione (v. sez. ??).

La firma digitale diviene dunque una garanzia del messaggio o documento elettronico, alla stregua della sottoscrizione di un documento cartaceo, attestandone con certezza l'integrità, l'autenticità e la non ripudiabilità anche dal punto di vista legale, poiché la legislazione italiana attribuisce ad un documento elettronico con firma digitale lo stesso valore dello stesso in forma cartacea sottoscritto con firma autografa (v. art. 15 comma 2 della legge n. 59 del 15/3/1997 "Bassanini-1", D.P.R. n. 445 del 28/12/2000, D.P.C.M. 08/02/1999).

???

23.4.1 Il MAC - Message Authentication Code

MAC

Il **MAC** (Message Authentication Code) è un valore, talvolta riferito anche come checksum, che è utilizzato come codice di autenticazione di un messaggio, assieme ad una chiave privata. A differenza delle firme digitali, i MAC sono generati e verificati con la stessa chiave (simmetrica). Esistono 4 tipi di MAC, di seguito descritti.

MAC unconditionally secure

Il MAC si basa sulla cifratura del messaggio per mezzo di una chiave. Il messaggio cifrato si autentica per il fatto che nessun altro conosce la chiave.

MAC hash function-based (HMAC)

Il MAC si basa sull'utilizzo di una chiave e di una funzione hash. Il messaggio viene dato in input ad una funzione hash che genera il message digest relativo, quindi un algoritmo di cifratura, cifra il message digest per mezzo di una chiave. Il valore così ottenuto viene aggiunto al messaggio stesso. Chi riceve il messaggio può verificarne l'integrità decifrando il HMAC con l'apposita chiave e confrontando il message digest ricevuto con quello calcolato sul momento relativamente al messaggio in chiaro ricevuto.

MAC stream cipher-based

Il messaggio viene suddiviso in due sottomessaggi ed ogni sottomessaggio è inserito in un LFSR (Linear Feedback Shift Register), ovvero un registro che ad ogni colpo di clock sposta tutti i bit verso le cifre più significative ed alcuni degli output (*tap*) sono combinati in XOR in modo da realizzare un feedback.⁵ Il MAC è lo stato finale dei due LFSR.

MAC block cipher-based

Il messaggio viene cifrato con un algoritmo simmetrico a blocchi (es. DES-CBC). Il valore dell'ultimo blocco viene aggiunto al messaggio ed utilizzato come MAC.

23.5 I certificati digitali

Un **certificato digitale** è un documento che attesta la relazione di appartenenza di una chiave pubblica ad certa una entità (una persona, una azienda, una macchina, ...). Tale legame è garantito dall'ente emittente il certificato, ovvero una terza parte fidata che costituisce l'autorità di certificazione o *Certification Authority* (CA).

certificato digitale

Un certificato digitale contiene la chiave pubblica ed il nominativo dell'entità di cui viene garantita la corrispondenza, indicazioni relative all'algoritmo utilizzato per la generazione della chiave, una data di scadenza, il nome della CA che ha rilasciato il certificato, il suo numero di serie e la firma digitale della CA stessa a garanzia del fatto che il certificato digitale è stato rilasciato proprio da tale CA. In questo modo, chiunque può verificare l'autenticità del certificato con la chiave pubblica della CA e quindi essere sicuro che la chiave pubblica contenuta nel certificato appartenga proprio all'entità specificata dal certificato stesso.

In effetti la verifica della firma del certificato può essere effettuata con la chiave pubblica della CA che lo ha emesso, che deve a sua volta essere certificata da un altro certificato emesso da un'altra CA e così via.

Il formato dei certificati digitali più diffuso è definito dallo standard ITU-T X.509

Una CA può anche fornire un servizio di marcatura temporale (*timestamp*), ovvero attribuisce con certezza la data e l'ora alla quale è stato redatto il documento.

A tale scopo, l'autore deve generare l'impronta digitale del documento da marcare temporalmente ed inviarla alla CA che provvede ad aggiungervi la data ed ora corrente e quindi la cifra con la propria firma digitale, ottenendo così la marcatura temporale relativa al documento in questione. La marcatura viene rinviata all'autore che la può accludere al documento in modo tale che chiunque lo riceve può verificare la data di creazione dello stesso, per mezzo della chiave pubblica della CA.

23.6 Riferimenti

- A. Menezes, P. van Oorschot, and S. Vanstone "Handbook of Applied Cryptography", CRC Press, 1996
<http://www.cacr.math.uwaterloo.ca/hac>
- Elenco dei FIPS
<http://www.itl.nist.gov/fipspubs/>
- Sorgenti degli algoritmi di cifratura
<http://the-other.wiretapped.net/security/cryptography/algorithms/>
- Sorgenti delle funzioni hash
<http://the-other.wiretapped.net/security/cryptography/hashes/>
- Dettagli sugli algoritmi di cifratura
<http://www.wikipedia.org>

⁵gli LSFR sono generalmente utilizzati per la generazione veloce di pattern pseudocasuali.

- Legislatura italiana sulla firma digitale
<http://www.cnipa.gov.it>

Capitolo 24

Comunicazioni sicure

“Le persone oneste e intelligenti difficilmente fanno una rivoluzione, perchè sono sempre in minoranza.”

– Aristotele

Utilizzando i protocolli illustrati nel cap. 19, tutte le informazioni scambiate tra il client ed il server sono trasmesse in chiaro (*plain text*), ovvero tutte le macchine che si trovano lungo il percorso seguito dai pacchetti potrebbero intercettare tali pacchetti facendone una copia e chi ha l'accesso a tali macchine può leggere le informazioni in essi contenute.

In questo capitolo saranno illustrati dei protocolli che permettono di avere la sicurezza più o meno garantita che le informazioni scambiate tra il client ed il server non siano leggibili da terzi. Tali protocolli si basano su meccanismi di cifratura, ovvero le informazioni che transitano tra il client ed il server non sono in chiaro, ma sono cifrate secondo opportuni algoritmi, che permettono comunque di riottenere l'informazione in chiaro, per mezzo di una chiave segreta, che ovviamente il ricevente deve conoscere, altrimenti neanche quest'ultimo può leggere l'informazione trasmessa.

L'utilizzo dei protocolli qui presentati è subordinato alle vigenti leggi sulla crittografia del Paese considerato e alle patent relative agli algoritmi di crittografia utilizzati dalle implementazioni dei protocolli stessi.

Nel seguito sarà fatto riferimento all'architettura riportata in fig. 24.1.

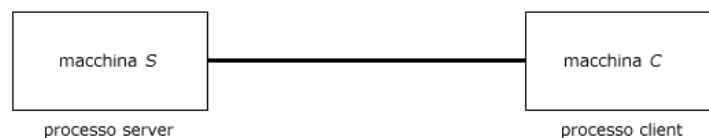


Figura 24.1: Architettura client server.

24.1 SSL/TLS - Secure Sockets Layer / Transport Layer Security

???

24.2 SSH - Secure SHell

Il protocollo SSH permette di accedere in maniera sicura alla shell di una macchina remota. Questo fa sì che tale protocollo sia molto utilizzato dagli amministratori di rete. Di tale protocollo ne esistono attualmente 2 versioni: la SSH Version 1 (SSH1),

sviluppata nel 1995, e la SSH Version 2 (SSH2), sviluppata nel 2001 ed in fase di standardizzazione presso il *Secure Shell Working Group* dell'IETF (Internet Engineers Task Force). SSH1 ed SSH2 sono due protocolli diversi e pertanto incompatibili: SSH1 ed SSH2 cifrano parti diverse dei pacchetti, e per l'autenticazione del client SSH1 usa le chiavi del server e del client mentre SSH2 usa soltanto le chiavi del client. SSH2 è un protocollo che dà maggiori garanzie sulla sicurezza delle informazioni scambiate, rispetto a SSH1.

Una versione libera del protocollo SSH è implementata dalla suite *OpenSSH*¹ che mette a disposizione una serie di programmi per la connettività: **ssh**, **scp** e **sftp**. Il server che gestisce la comunicazione cifrata è il daemon **sshd**. Esso viene lanciato in genere al boot del sistema e rimane in attesa di una connessione da parte di un eventuale client. Non appena riceve una richiesta di connessione da parte di un client, esso genera un processo figlio che si occuperà di gestire l'autenticazione e la comunicazione con il client ed il processo principale ritornerà in attesa di un'altra eventuale richiesta di connessione.

Comando: **sshd**

Path: **/usr/sbin/sshd**

SINTASSI

sshd [option]

DESCRIZIONE

option indica la modalità di funzionamento di **sshd**. Può assumere i seguenti valori:

-b nbit

indica il numero di bit (*nbit*) utilizzati per la chiave del server (SSH1) (per default è 768);

-d

attiva la modalità di debug: nel system log vengono memorizzati molti più messaggi del normale, e sshd non crea processi figli, ma gestisce esso stesso le comunicazioni con gli eventuali client. Tale opzione può essere ripetuta fino a 3 volte ed ogni ripetizione aumenta il livello di verbosità dei messaggi nel system log;

-e

indica di inviare l'output sullo standard error anziché nel system log;

-f config_file

specifica il nome del file di configurazione (per default viene utilizzato il file **/etc/ssh/sshd.config**);

-g login_grace_time

specifica il tempo entro il quale il client si deve autenticare (per default è 120 secondi). Il valore 0 indica nessun limite;

-h host_key_file

specifica il file da dal quale deve essere letta la chiave della macchina (per default il file è **/etc/ssh/ssh_host_key** per SSH1 e **/etc/ssh/ssh_host_rsa_key** (RSA) e **/etc/ssh/ssh_host_dsa_key** (DSA) per SSH2);

-i

indica che **sshd** deve essere gestito e lanciato da **inetd**;

-k key_gen_time

specifica ogni quanto tempo deve essere rigenerata la chiave di server (per default è 3600 secondi). Il valore 0 indica che la chiave non deve mai essere rigenerata;

-o option

specifica eventuali opzioni espresse nel formato utilizzato nel file di configurazione;

-p port

specifica la porta sulla quale il server rimane in ascolto per ricevere le richieste da eventuali client (la porta di default è la 22);

-q

attiva la modalità silenziosa (*quiet*): nessun messaggio viene scritto nel system log;

¹v. <http://www.openssh.org>.

- t attiva la modalità di test: viene effettuato soltanto il controllo sulla validità del file di configurazione e delle chiavi;
- u *len* specifica la lunghezza del campo nella struttura utmp che memorizza il nome della macchina remota (nel caso in cui il nome della macchina sia maggiore di *len* viene memorizzato l'indirizzo IP nel formato dotted decimal);
- D indica a **sshd** di non avviarsi come daemon (il processo non va in background);
- 4 forza il solo utilizzo di IPv4;
- 6 forza il solo utilizzo di IPv6;

Il file di configurazione relativo a **sshd** (per default `/etc/ssh/sshd_config`) è costituito da righe che contengono delle direttive con la seguente sintassi

keyword [*args*]

dove *keyword* esprime una specifica direttiva e *args* indica gli eventuali argomenti relativi alla direttiva considerata. *keyword* può assumere uno dei seguenti valori (le *keyword* sono case insensitive ma *args* sono case sensitive)

AFSTokenPassing

specifica se un token AFS può essere inoltrato al server (per default **no**);

AllowGroups

questa parola chiave può essere seguita da un elenco di nomi di gruppi separati da spazi (i simboli '*' e '?' possono essere utilizzati come wildcard nei nomi dei gruppi). L'accesso sarà permesso soltanto agli utenti il cui gruppo corrisponde ad uno tra quelli elencati (per default l'accesso è consentito a qualunque gruppo);

AllowTcpForwarding

indica se permettere o meno l'inoltro dei TCP segment (per default **yes**);

AllowUsers

questa parola chiave può essere seguita da un elenco di username separati da spazi (i simboli '*' e '?' possono essere utilizzati come wildcard negli username). L'accesso sarà permesso soltanto agli utenti il cui username corrisponde ad uno tra quelli elencati (per default l'accesso è consentito a qualunque utente). Nel caso in cui uno username sia espresso nella forma *username@hostname* il controllo sullo username e sullo hostname viene effettuato in maniera separata ???;

AuthorizedKeysFile

indica il file che contiene le chiavi pubbliche utilizzate per l'autenticazione. Il nome del file può contenere dei token particolari, riportati in tab. 24.1 che vengono interpretati durante la connessione. Il nome del file può essere un path assoluto (se inizia per '/') o relativo alla home directory dell'utente in questione (per default è `.ssh/authorized_keys`);

Token	Significato
%%	%
%h	home directory dell'utente
%u	username dell'utente

Tabella 24.1: Token di **AuthorizedKeysFile** interpretati da **sshd**.

Banner specifica il file il cui contenuto deve essere visualizzato prima di effettuare l'autenticazione. Tale direttiva è possibile soltanto per SSH2 (per default non viene visualizzato nessun messaggio);

ChallengeResponseAuthentication

indica se è permessa un'autenticazione a risposta di *challenge* (il default è **yes**);

Ciphers

Specifica il tipo di cifratura utilizzato con SSH2. Possono essere specificate più algoritmi di cifratura separati da una virgola (per default è `aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes256-cbc,aes128-ctr,aes192-ctr,aes256-ctr`);

ClientAliveInterval

indica il tempo in secondi, entro il quale `sshd`, se non riceve nessuna informazione dal client, invia un messaggio al client per richiedere una risposta. È applicabile soltanto a SSH2 (per default è 0, che indica che non deve essere inviato nessun messaggio da `sshd`);

ClientAliveCountMax

imposta il numero massimo di messaggi client alive che possono essere ignorati prima che il client risponda, relativi al fatto che il client non invia informazioni (per default è 3);

Compression

indica se è permessa la compressione delle informazioni (per default è `yes`);

DenyGroups

analogo a `AllowGroups`, ma esclude l'accesso ai gruppi elencati, anziché consentirlo;

DenyUsers

analogo a `AllowUsers`, ma esclude l'accesso agli utenti elencati, anziché consentirlo;

GatewayPorts

indica se altre interfacce di rete possono effettuare una connessione alle porte riservate per il port forwarding (v. sez. 24.2.4) (per default è `no`, le porte su cui è stato effettuato il port forwarding di ssh sono accessibili solo dall'interfaccia di loopback);

HostbasedAuthentication

indica se è permessa l'autenticazione per mezzo dei file `.rhosts` o `/etc/hosts.equiv` (oltre al controllo della chiave pubblica della macchina C). Questa direttiva è simile alla `RhostsRSAAuthentication` e viene applicata soltanto ad SSH2 (per default è `no`);

HostKey

specifica il file contenente la chiave privata della macchina S (per default è `/etc/ssh/ssh_host_key` per SSH1, `/etc/ssh/ssh_host_rsa_key` per RSA SSH2 e `/etc/ssh/ssh_host_dsa_key` per DSA SSH2). Il file deve avere i permessi di accesso solo per l'utente proprietario, altrimenti `sshd` non lo considera;

IgnoreRhosts

indica di non considerare i file `.rhosts` e `.shosts` nell'autenticazione indicata da `RhostsAuthentication`, `RhostsRSAAuthentication` o `HostbasedAuthentication` (per default è `yes`);

IgnoreUserKnownHosts

indica se `sshd` deve ignorare il file `~/.ssh/known_hosts` nell'autenticazione indicata da `RhostsRSAAuthentication` o `HostbasedAuthentication` (per default è `no`);

KeepAlive

specifica se il sistema deve inviare segnali di keepalive al client (per default è `yes`);

KerberosAuthentication

indica se è permessa l'autenticazione Kerberos (un ticket o KDC nel caso in cui `PasswordAuthentication` sia impostata) (per default è `no`);

KerberosOrLocalPasswd

indica che nel caso in cui l'autenticazione via Kerberos non vada a buon fine, l'autenticazione viene effettuata per mezzo di un meccanismo di password (file `/etc/passwd`) (per default è **yes**);

KerberosTgtPassing

specifica se un TGT Kerberos può essere inoltrato al server (per default è **no**);

KerberosTicketCleanup

indica se il file relativo alla cache del ticket utente deve essere cancellato al termine della connessione (per default è **yes**);

KeyRegenerationInterval

specifica il tempo in secondi dopo il quale la chiave del processo server deve essere rigenerata (SSH1). Il valore 0 indica che la chiave non deve mai essere rigenerata (per default è 3600);

ListenAddress

specifica l'indirizzo IP e porta sui quali **sshd** deve stare in attesa di eventuali connessioni. La sintassi utilizzata è la seguente

```
ListenAddress [hostname | IPv4_addr | IPv6_addr][:port]
```

Se *port* non è specificato, la porta utilizzata è quella specificata dalla direttiva **Port** specificata precedentemente (per default **sshd** rimane in ascolto su tutti gli indirizzi IP relativi alla macchina sulla quale è in esecuzione);

LoginGraceTime

specifica il tempo in secondi entro il quale il client deve completare la procedura di connessione, altrimenti **sshd** lo disconnette. Il valore 0 indica che non vi è alcun limite (per default è 120);

LogLevel

specifica il livello di verbosità dei messaggi inseriti nel system log. I valori possibili, con verbosità crescente, sono i seguenti: **QUIET**, **FATAL**, **ERROR**, **INFO**, **VERBOSE**, **DEBUG**, **DEBUG1**, **DEBUG2** e **DEBUG3** (per default è **INFO**);

MACs

specifica il l'algoritmo per il calcolo del MAC (Message Authentication Code), utilizzati con SSH2 per garantire l'integrità delle informazioni trasmesse. Possono essere specificati più algoritmi separati tra loro da una virgola (per default è **hmac-md5,hmac-sha1,hmac-ripemd160,hmac-sha1-96,hmac-md5-96**);

MaxStartups

specifica il numero massimo di connessioni contemporanee non autenticate. Le connessioni che eccedono tale numero vengono interrotte, a meno che non siano state autenticate (per default è 10). Può essere specificato anche un insieme di tre valori separati dal simbolo ':' che specificano nell'ordine il numero massimo iniziale di connessioni contemporanee non autenticate, la probabilità (percentuale) con la quale il tentativo di una ulteriore connessione sarà rifiutato ed il numero massimo definitivo di connessioni contemporanee non autenticate: la probabilità di rifiuto cresce linearmente con l'aumentare del numero di connessioni non autorizzate, fino a diventare totale quando il numero delle connessioni arriva alla quantità specificata dal terzo valore;

PasswordAuthentication

indica se l'autenticazione tipo password è consentita (per default è **yes**);

PermitEmptyPasswords

indica, nel caso in cui sia attivata l'autenticazione di tipo password, se è consentita l'autenticazione per gli utenti che hanno una password vuota (per default è **no**);

PermitRootLogin

indica se il superuser può accedere al sistema tramite **ssh**. Il valore dell'argomento può essere uno tra quelli riportati in tab. 24.2 (per default è **yes**);

args	Significato
yes	il superuser non può accedere al sistema tramite ssh con l'autenticazione di tipo password
without-password	l'autenticazione di tipo password è disabilitata per il superuser
forced-commands-only	il superuser può accedere al sistema tramite l'autenticazione di tipo publickey ma soltanto nel caso in cui sia stata specificata un'opzione sulla riga di comando
no	il superuser non può accedere al sistema tramite ssh

Tabella 24.2: Possibili valori di *args* per **PermitRootLogin**.

PermitUserEnvironment

specifica se il file `~/.ssh/environment` e le direttive **environment=** contenute nel file `~/.ssh/authorized_keys` devono essere considerati da **sshd** (per default è **no**);

PidFile

specifica il nome del file che contiene il PID di **sshd** (per default è `/var/run/sshd.pid`);

Port

specifica il numero della porta sulla quale **sshd** rimane in ascolto di eventuali richieste da parte di client (per default è 22) v. anche **ListenAddress**;

PrintLastLog

indica se **sshd** deve visualizzare la data e l'ora relative all'ultimo accesso effettuato dall'utente considerato (per default è **yes**);

PrintMotd

indica se **sshd** deve visualizzare il contenuto del file `/etc/motd` al login di ogni utente (per default è **yes**);

Protocol

specifica le versioni del protocollo SSH che **sshd** deve supportare. I valori possibili sono 1 (SSH1) e 2 (SSH2). Possono essere specificate più versioni separate tra loro da una virgola – l'ordine dell'elenco delle versioni non indica l'ordine di preferenza, poiché è il client che sceglie il protocollo fra quelli supportati dal server (per default è 2,1);

PubkeyAuthentication

indica se l'autenticazione di tipo publickey è consentita. Questa direttiva si applica soltanto a SSH2 (per default è **yes**);

RhostsAuthentication

indica se l'autenticazione tramite il file `.rhosts` o `/etc/hosts.equiv` è sufficiente – questa direttiva si applica soltanto a SSH1 (per default è **no**);

RhostsRSAAuthentication

indica se l'autenticazione deve essere effettuata per mezzo del file `.rhosts` o `/etc/hosts.equiv` e della chiave RSA – questa direttiva si applica soltanto a SSH1 (per default è **no**);

RSAAuthentication

indica se l'autenticazione deve essere effettuata in base alla chiave RSA – questa direttiva si applica soltanto a SSH1 (per default è **yes**);

ServerKeyBits

specifica il numero di bit che devono costituire la chiave del server – il minimo è 512 (per default è 768);

StrictModes

indica se **sshd** deve controllare che la home directory ed i file non siano accessibili ad altri, prima di accettare un login (per default è **yes**);

Subsystem

specifica un sottosistema esterno (ad esempio un daemon per il trasferimento di file) – questa direttiva è valida soltanto per SSH2 (per default non è specificato nessun sottosistema);

SyslogFacility

specifica la *facility* relativa ai messaggi da memorizzare nel system log. I valori possibili sono `DEAMON`, `USER`, `AUTH`, `LOCAL0`, `LOCAL1`, `LOCAL2`, `LOCAL3`, `LOCAL4`, `LOCAL5`, `LOCAL6` e `LOCAL7` (per default è `AUTH`);

UseDNS indica se `sshd` deve verificare che il nome della macchina *C* corrisponda veramente al suo indirizzo IP (per default è `yes`);

UseLogin

indica se per l'accesso remoto degli utenti deve essere utilizzato `login` (per default è `no`);

UsePrivilegeSeparation

indica se `sshd`, dopo aver autenticato un utente, deve creare un altro processo con i permessi dell'utente autenticato (per default è `yes`);

X11DisplayOffset

specifica il valore del numero di display per l'inoltro del protocollo X11 (per default è 10);

X11Forwarding

indica se i pacchetti del protocollo X11 possono essere inoltrati con port forwarding a `sshd` (per default è `no`);

X11UseLocalhost

indica se il server di X11 deve essere necessariamente associato all'indirizzo di loopback (per default è `yes`);

XAuthLocation

specifica il path assoluto di `xauth` (per default è `/usr/X11R6/bin/xauth`);

???

24.2.1 Autenticazione

Il meccanismo di autenticazione tra client e server della suite di *OpenSSH* dipende dal protocollo utilizzato.

SSH1

Ogni macchina ha una coppia di chiavi (pubblica/privata) RSA (in genere di 1.024 bit) che la identificano. Inoltre quando `sshd` viene avviato, viene generata automaticamente una chiave RSA (per default di 768 bit) che identifica il processo server.

Quando un client si connette da una macchina *C*, `sshd` risponde con la sua chiave pubblica RSA di sistema e la chiave RSA di server. Il client confronta la chiave pubblica della macchina *S* con quella presente sulla macchina *C* per vedere se la riconosce. Quindi il client genera un numero casuale di 256 bit, lo cifra con la chiave RSA pubblica della macchina *S* e con quella del server e lo invia a `sshd`. Entrambe le parti (client e server) utilizzeranno quindi tale valore numerico come chiave (simmetrica) di sessione, utilizzata per cifrare il resto della comunicazione con uno dei possibili algoritmi di cifratura (3DES o Blowfish – 3DES è il default). Il client utilizzerà l'algoritmo di cifratura fra quelli proposti dal server.

I metodi di autenticazione possibili sono quelli riportati di seguito

rhosts

l'accesso alla macchina *S* è consentito se la macchina *C*, dalla quale l'utente accede, è elencata nel file `/etc/hosts.equiv` o nel file `/etc/shosts.equiv` sulla macchina *S* e lo username dell'utente sulla macchina *C* è quello richiesto per il

login sulla macchina *S* e sulla macchina *S* tale username esiste. Se i file sopra indicati non esistono, l'accesso alla macchina *S* è consentito se sulla macchina *S* esiste il file `~/.rhosts` o `~/.shosts` e contiene una riga che riporta il nome della macchina *C* e lo username dell'utente sulla macchina *C* è quello richiesto per il login sulla macchina *S* e sulla macchina *S* tale username esiste.

rhosts + RSA host key

l'autenticazione si basa sul controllo effettuato nel caso *rhosts* ed in più la chiave pubblica della macchina *C* viene ricercata all'interno del file `/etc/ssh/ssh_known_hosts` (e se non presente viene ricercata anche nel file `~/.ssh/known_hosts`);

RSA challenge-response

l'autenticazione si basa sul riconoscimento della chiave pubblica e sullo scambio di un *challenge* per mezzo di cifratura asincrona: il client segnala al server la chiave pubblica dell'utente ed il server la ricerca nel file `~/.ssh/authorized_keys` e nel caso la trovi, invia un *challenge* al client, ovvero un valore casuale cifrato con la chiave pubblica dell'utente, cosicché solo il client che è in esecuzione con i privilegi dell'utente considerato può decifrare il *challenge* per mezzo della chiave privata dell'utente, contenuta nel file `~/.ssh/identity`.

password

l'autenticazione avviene sulla base di una password digitata dall'utente e trasmessa cifrata dal client al server;

SSH2

Ogni macchina ha una coppia di chiavi (pubblica/privata) RSA o DSA (in genere di 1024 bit) che la identificano.

La comunicazione viene cifrata con uno dei possibili algoritmi a chiave simmetrica (AES a 128, 192 o 256 bit, Blowfish, 3DES, CAST128, Arcfour). Il client utilizzerà l'algoritmo di cifratura fra quelli proposti dal server. Inoltre l'integrità dell'informazione sarà garantita da un'impronta crittografica (hash MAC – Message Authentication Code) SHA1 o MD5.

I metodi di autenticazione possibili sono quelli riportati di seguito

hostbased

l'autenticazione si basa su un controllo analogo a quello effettuato nel caso *rhosts* + *RSA host key* con SSH1;

public key

l'autenticazione è analoga a quella descritta nel caso *RSA challenge-response* con SSH1 ma la chiave di sessione deriva da un valore condiviso Diffie-Hellman. Il client genera un'impronta della chiave di sessione con la chiave privata dell'utente in questione, contenuta nel file `~/.ssh/id_rsa` (RSA) o `~/.ssh/id_dsa` (DSA), e invia il messaggio al server. Il server autentica il client se riconosce la sua chiave pubblica e se l'impronta corrisponde al messaggio inviato.

challenge-response

l'autenticazione si basa su un controllo analogo a quello effettuato nel caso *RSA challenge-response* con SSH1;

password

l'autenticazione avviene sulla base di una password digitata dall'utente e trasmessa cifrata dal client al server;

24.2.2 Login

Quando un utente accede con successo al sistema remoto (macchina *S*), `sshd` effettua i seguenti passi

1. Se il login avviene su una TTY (e non è stato specificato nessun comando) visualizza l'ultima data/ora di accesso dell'utente in questione ed il contenuto del file `/etc/motd` (a meno che non esista il file `~/.hushlogin`);
2. Se il login avviene su una TTY memorizza la data/ora relativa all'accesso;
3. Se esiste il file `/etc/nologin` ne visualizza il contenuto ed scollega l'utente (a meno che non sia il superuser);
4. Imposta i privilegi dell'utente in questione;
5. Imposta l'ambiente di base;
6. Imposta l'ambiente con le variabili impostate nel file `~/.ssh/environment`;
7. Imposta la working directory con la home directory dell'utente in questione;
8. Se esiste il file `~/.ssh/rc` lo esegue, altrimenti se esiste il file `/etc/ssh/sshrc` lo esegue, altrimenti esegue `xauth ???`;
9. Esegue il comando che lancia la shell di default dell'utente in questione;

24.2.3 Generazione di chiavi asimmetriche

Una coppia di chiavi asimmetriche (RSA o DSA) può essere generata per mezzo del comando `ssh-keygen` (man page `ssh-keygen(1)`).

24.2.4 ssh

Il comando `ssh` (secure shell - man page `ssh(1)`) è un client di comunicazione che, utilizzando il protocollo SSH (SSH1 o SSH2), permette di effettuare il login su una macchina remota. Se il login va a buon fine si avrà l'accesso alla shell definita per l'utente che ha effettuato il login e si potrà così interagire con il sistema remoto come se si impartissero i comandi direttamente sulla sua tastiera.

Quando `ssh` si connette a `sshd`, controlla la chiave pubblica di sistema della macchina *S* con quelle presenti nei file `/etc/ssh/ssh_known_hosts` e `~/.ssh/known_hosts`. Nel caso in cui la chiave non sia presente `ssh` richiede interattivamente se deve essere effettuata la connessione alla macchina *S* ed in caso affermativo memorizza la relativa chiave nel file `~/.ssh/known_hosts`.

L'autenticazione tra il client (macchina *C*) ed il server (macchina *S*) può avvenire, dipendentemente dal protocollo utilizzato (SSH1 o SSH2), per mezzo di uno dei metodi nell'ordine di seguito riportati

- SSH1**
1. rhosts;
 2. rhosts + RSA host key;
 3. RSA challenge-response;
 4. password;

- SSH2**
1. hostbased;
 2. public key;
 3. challenge-response;
 4. password;

Comando: `ssh`
 Path: `??*/ssh`

SINTASSI

`# ssh [option] {host | username@host} [command]`

DESCRIZIONE

option indica la modalità di funzionamento di **ssh**. Può assumere i seguenti valori:

- a disabilita l'inoltro della connessione all'authentication agent;
- A abilita l'inoltro della connessione all'authentication agent;
- b *bind_address*
 specifica l'indirizzo dell'interfaccia di rete dalla quale trasmettere;
- c *encryption_algorithm*
 specifica l'algoritmo da utilizzare per la cifratura della sessione di comunicazione. Le scelte possibili sono **blowfish**, **des** e **3des** – nel caso di SSH2 può essere specificato un elenco di algoritmi, nell'ordine di preferenza, separati tra loro da una virgola (per default è **3des**);
- e *escape_character*
 specifica il carattere di escape da utilizzare nella comunicazione. Può essere un qualunque carattere standard **c**, un carattere di controllo espresso come **^c**, o **none** (nessun carattere di escape). Il carattere di escape di default è **~** e le sequenze di escape sono riportate in tab. 24.3;

Sequenza	Significato
~~	~
~.	chiude la connessione
~~Z	manda in background ssh
~#	elena le connessioni redirette
~&	manda in background ssh alla chiusura della connessione
~?	visualizza un elenco dei caratteri di escape
~B	invia un BREAK alla macchina <i>S</i> (solo SSH2)
~C	apre una riga di comando per la definizione di redirezioni di porte
~R	richiede il passaggio di chiave della connessione (solo SSH2)

Tabella 24.3: Possibili sequenze di escape utilizzate nella comunicazione client-server.

- f indica a **ssh** di andare in background prima dell'esecuzione del comando;
- g permette al sistema remoto di connettersi alle porte locali redirette;
- i *identity_file*
 specifica il file che contiene le chiavi private RSA o DSA (per default i file sono **~/.ssh/identity** per le chiavi RSA (SSH1), **~/.ssh/id_rsa** per le chiavi RSA (SSH2) e **~/.ssh/id_dsa** per le chiavi DSA (SSH1)).
- I *smartcard_device*
 specifica il file di dispositivo relativo al lettore di smartcard per la lettura della chiave privata RSA;
- k disabilita l'inoltro dei ticket e token AFS di Kerberos;
- l *username*
 specifica lo username da utilizzare per autenticarsi sulla macchina remota;
- m *mac*
 specifica l'elenco degli algoritmi di MAC (message authentication code) in ordine di preferenza, separati tra loro da una virgola;
- n redirige lo standard input a **/dev/null**;
- N indica di non eseguire il comando remoto (solo per SSH2). È utilizzato per la redirezione delle porte;
- o *option*
 specifica le opzioni utilizzando la sintassi del file di configurazione;
- p *port*
 indica la porta sulla quale è in ascolto il processo server (**sshd**) sulla macchina *S*;
- q attiva la modalità silenziosa (quiet): tutti i messaggi di attenzione o diagnostici vengono soppressi;
- s specifica la richiesta di sottosistema sulla macchina *S* (solo per SSH2). Il sottosistema viene indicato come **command**;
- t abilita l'allocazione una pseudo-TTY;

- T disabilita l'allocazione una pseudo-TTY;
 - v abilita la modalità verbosa: *ssh* visualizzerà messaggi relativi al suo progresso. Possono essere specificate fino a 3 -v per incrementare il livello di verbosità;
 - V mostra la versione di *ssh*;
 - x disabilita l'inoltro dei pacchetti X11;
 - X abilita l'inoltro dei pacchetti X11;
 - C richiede la compressione delle informazioni trasmesse. L'algoritmo di compressione è quello utilizzato da *gzip* (il livello di compressione è controllato dalla direttiva *CompressionLevel* per SSH1);
 - F *config_file*
specifica il file di configurazione (per default vengono considerati i file di configurazione */etc/ssh/ssh.config*, a livello di sistema, e *~/.ssh/config*, a livello utente);
 - L *fwport*
specifica la redirectione della porta della macchina *C*. Il parametro *fwport* è specificato secondo la sintassi *port:host:hostport* (IPv4) o *port/host/hostport* (IPv6) che indica di redirigere la porta *port* sulla macchina *C* (*host*) alla porta *hostport* sulla macchina *S*. Viene allocato un socket in ascolto sulla porta *port* della macchina *C* e quando viene effettuata una connessione a tale porta la connessione viene rediretta su di un canale sicuro e viene effettuata una connessione a *hostport* della macchina *S*;
 - R *fwport*
specifica la redirectione della porta della macchina *S*. Il parametro *fwport* è specificato secondo la sintassi *port:host:hostport* (IPv4) o *port/host/hostport* (IPv6) che indica di redirigere la porta *port* sulla macchina *S* (*host*) alla porta *hostport* sulla macchina *C*. Viene allocato un socket in ascolto sulla porta *port* della macchina *S* e quando viene effettuata una connessione a tale porta la connessione viene rediretta su di un canale sicuro e viene effettuata una connessione a *hostport* della macchina *C*;
 - D *port*
specifica una redirectione dinamica della porta sulla macchina *C*. Viene allocato un socket in ascolto sulla porta *port* della macchina *C* e quando viene effettuata una connessione a tale porta la connessione viene rediretta su di un canale sicuro ed il protocollo di livello application è utilizzato per determinare l'interfaccia di rete remota della connessione (attualmente è supportato il protocollo SOCKS4);
 - 1 forza *ssh* ad utilizzare soltanto SSH1;
 - 2 forza *ssh* ad utilizzare soltanto SSH2;
 - 4 forza *ssh* ad utilizzare soltanto IPv4;
 - 6 forza *ssh* ad utilizzare soltanto IPv6;
- host* è il nome dell'interfaccia (od il suo indirizzo IP) alla quale si desidera connettersi (macchina *S*);
- username* è lo username con il quale si desidera autenticarsi sulla macchina *S*;
- command* è l'eventuale comando da far eseguire sulla macchina *S*;

Il comando *ssh* utilizza un file di configurazione (per default esiste un file di configurazione a livello di sistema, */etc/ssh/ssh.config*, ed uno a livello utente, *~/.ssh/config*) contenente le direttive per il funzionamento di *ssh*, secondo la seguente sintassi (v. man page *ssh.config(5)*)

keyword [*args*]

dove *keyword* indica una specifica direttiva e *args* sono gli eventuali argomenti relativi alla direttiva considerata. I possibili valori di *keyword* sono di seguito riportati

Host indica di applica le direttive seguenti per le connessioni con le macchine *S* il cui nome è elencato nella parte *args*, costituita da un elenco di nomi di macchine separati da spazi (i simboli ‘*’ e ‘?’ possono essere utilizzati come wildcard);

AddressFamily

specifica l’address family del protocollo utilizzato. I possibili valori sono quelli riportati in tab. ??;

args	Significato
inet	indica di utilizzare soltanto IPv4
inet6	indica di utilizzare soltanto IPv6
any	indica di utilizzare qualunque versione del protocollo IP

Tabella 24.4: Possibili valori di *args* per la direttiva **AddressFamily**.

AFSTokenPassing

indica se passare i token AFS al server – questa direttiva si applica soltanto a SSH1;

BatchMode

indica se la richiesta interattiva di password o passphrase devono essere disabilitate (per default è **no**);

BindAddress

specifica l’interfaccia di rete da utilizzare per la trasmissione delle informazioni (tale direttiva non funziona se è attivata la direttiva **UsePrivilegedPort**);

ChallengeResponseAuthentication

indica se deve essere utilizzata l’autenticazione challenge-response (per default è **yes**);

CheckHostIP

indica se **ssh** deve controllare l’indirizzo IP della macchina *S* (per default è **yes**);

Cipher specifica l’algoritmo di cifratura da utilizzare nella sessione di comunicazione. Le opzioni possibili sono **blowfish**, **des** e **3des** – questa direttiva si applica soltanto a SSH1 (per default è **3des**);

Ciphers

specifica l’elenco degli algoritmi di cifratura da utilizzare, in ordine di preferenza, nella sessione di comunicazione, separati uno dall’altro da una virgola – questa direttiva si applica soltanto a SSH2 (per default è **aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfb**);

ClearAllForwardings

indica se annullare tutte le redirezioni delle porte (per default è **no**);

Compression

indica se utilizzare la compressione (per default è **no**);

CompressionLevel

specifica il livello di compressione delle informazioni trasmesse. I valori vanno da 1 (compressione minima) a 9 (compressione massima) – questa direttiva si applica soltanto a SSH1 (per default è **6**);

ConnectionAttempts

specifica il numero di tentativi di connessione (uno al secondo) prima di considerare la connessione con il server non possibile (per default è **1**);

ConnectTimeout

specifica il tempo in secondi (al posto del timeout del TCP) dopo il quale considerare il tentativo connessione fallito;

DynamicForward

specifica la porta relativa alla macchina *C* che deve essere rediretta sul canale sicuro ed il protocollo di livello application viene utilizzato per determinare l'indirizzo a cui connettersi (al momento è supportato il protocollo SOCKS4 e `ssh` agisce da server SOCKS4);

EnableSSHKeySign

indica se deve essere utilizzato il comando `ssh-keysign` durante l'autenticazione hostbased (per default è `no`);

EscapeChar

specifica il carattere di escape (per default è `~`).;

ForwardAgent

indica se la connessione all'authentication agent deve essere rediretta alla macchina remota (per default è `no`);

ForwardX11

indica se le connessioni X11 devono essere automaticamente redirette sul canale sicuro (per default è `no`);

GatewayPorts

indica se le macchine remote possono connettersi alle porte localmente redirette (per default è `no`);

GlobalKnownHostsFile

specifica il file dal quale leggere le chiavi pubbliche dei sistemi conosciuti (per default è `/etc/ssh/ssh_known_hosts`);

HostbasedAuthentication

indica se tentare un'autenticazione di tipo rhosts nel caso in cui sia abilitata l'autenticazione di tipo public key – questa direttiva si applica soltanto a SSH2 (per default è `no`);

HostKeyAlgorithms

specifica l'elenco degli algoritmi, in ordine di preferenza, relativi alla generazione delle chiavi con le quali effettuare l'autenticazione, separati l'uno dall'altro da una virgola (per default è `ssh-rsa,ssh-dsa`);

HostKeyAlias

specifica un alias ;

???

La direttiva viene presa in considerazione secondo l'elenco nell'ordine di seguito riportato

1. opzione sulla riga di comando;
2. file di configurazione a livello utente (`~/.ssh/config`);
3. file di configurazione a livello di sistema (`/etc/ssh/ssh_config`);

Il file `~/.ssh/known_hosts` contiene le chiavi delle macchine remote alle quali l'utente ha acceduto tramite `ssh` e che non sono elencate nel file `/etc/ssh/ssh_known_hosts`.

Chiavi private

Il file `~/.ssh/identity` contiene la chiave privata RSA dell'utente. Viene utilizzato dal protocollo SSH1.

Il file `~/.ssh/id_rsa` contiene la chiave privata RSA dell'utente. Viene utilizzato dal protocollo SSH2.

Il file `~/.ssh/id_dsa` contiene la chiave privata DSA dell'utente. Viene utilizzato dal protocollo SSH2.

È possibile cifrare tramite 3DES il contenuto del file per mezzo di una passphrase in modo che soltanto l'utente sia in grado di leggerne il contenuto (il superuser infatti, sebbene possa accedere al file, non può in tal caso visualizzarne in chiaro il contenuto).

`ssh` ignora il file contenente la chiave privata dell'utente se esso è accessibile dagli altri (se è abilitato uno dei diritti di accesso per gli altri utenti non proprietari del file).

Chiavi pubbliche

Il file `~/.ssh/identity.pub` contiene la chiave pubblica RSA dell'utente. Viene utilizzato dal protocollo SSH1.

Il file `~/.ssh/id_rsa.pub` contiene la chiave pubblica RSA dell'utente. Viene utilizzato dal protocollo SSH2.

Il file `~/.ssh/id_dsa.pub` contiene la chiave pubblica DSA dell'utente. Viene utilizzato dal protocollo SSH2.

Le chiavi pubbliche devono essere inserite nel file `~/.ssh/authorized_keys` sulla macchina remota alla quale si vuole accedere.

Il file `~/.ssh/config` contiene la configurazione di `ssh` per l'utente in questione.

Il file `~/.ssh/authorized_keys` contiene l'elenco delle chiavi pubbliche (RSA/DSA) relative agli utenti ai quali è consentito l'accesso da remoto alla macchina considerata (v. `sshd(8)`).

Il file `/etc/ssh/ssh_known_hosts` contiene l'elenco delle chiavi delle macchine remote a cui si vuole accedere. È composta da righe secondo la seguente sintassi

```
system_name public_key [comment]
```

Il file `/etc/ssh/ssh_config` contiene la configurazione di `ssh` a livello di sistema.

Il file `/etc/ssh/ssh_host_key` contiene la chiave privata RSA del sistema, utilizzata dal protocollo SSH1 (è accessibile in lettura soltanto dal superuser).

Il file `/etc/ssh/ssh_host_rsa_key` contiene la chiave privata RSA del sistema, utilizzata dal protocollo SSH2.

Il file `/etc/ssh/ssh_host_dsa_key` contiene la chiave privata DSA del sistema, utilizzata dal protocollo SSH2.

Il file `~/.rhosts` contiene l'elenco delle macchine client e degli utenti che possono accedere da remoto, utilizzato dall'autenticazione `hostbased`.

Il file `~/.shosts` è identico al file `~/.rhosts` tranne per il fatto che mentre `~/.rhosts` è accessibile anche da `rsh` e `rlogin` `~/.shosts` è utilizzato soltanto da `ssh`.

Il file `/etc/hosts.equiv` contiene i nomi delle macchine client da cui può essere effettuato l'accesso alla macchina server in questione, utilizzato nel caso di autenticazione `hostbased`.

Il file `/etc/shosts.equiv` è identico al file `/etc/hosts.equiv` tranne per il fatto che mentre `/etc/hosts.equiv` è accessibile anche da `rsh` e `rlogin` `/etc/shosts.equiv` è utilizzato soltanto da `ssh`.

Il file `/etc/ssh/sshr` è uno script (a livello di sistema) lanciato automaticamente da `ssh` non appena qualunque utente accede al sistema da remoto, prima ancora che l'utente abbia l'accesso alla shell.

Il file `~/.ssh/rc` è uno script (a livello utente) lanciato automaticamente da `ssh` non appena l'utente in questione accede al sistema da remoto, prima ancora che abbia l'accesso alla shell.

Il file `~/.ssh/environment` contiene eventuali definizioni di variabili d'ambiente.
???

Port forwarding

Il comando `ssh` permette anche di rendere sicure comunicazioni che utilizzano protocolli non sicuri, effettuando quello che viene definito un *port forwarding*, ovvero uno *spostamento di porte* di comunicazione. Ad esempio, si supponga di voler rendere sicura una comunicazione in chiaro dalla macchina *C*, porta *c* alla macchina *S*, porta *s*. Innanzitutto è necessario che lato server (macchina *S*) sia in esecuzione il daemon `sshd` e la relativa porta di ascolto (per default la 22) sia aperta. Lato client (macchina *C*) con il comando `ssh` si può instaurare un canale sicuro (cifrato) sul quale vengano redirette le

richieste di comunicazione sulla porta *c* della macchina *C*, quindi la macchina *S* inoltrerà i pacchetti ricevuti alla porta *s*. E viceversa. Ovvero, quando la macchina *C* aprirà una connessione a sé stessa sulla porta *c*, questa verrà reindirizzata, in maniera cifrata, alla porta sulla quale è in ascolto **sshd** (22) sulla macchina *S*, che poi reinoltrerà i pacchetti ricevuti alla porta *s* della macchina *S* stessa.

??? figura ???

Si può anche utilizzare il port forwarding nel caso in cui sulla macchina *S* non sia in esecuzione **sshd**. In questo caso ci si può appoggiare su una terza macchina (macchina *I*) sulla quale sia in esecuzione **sshd**, in modo tale che il canale sicuro (cifrato) sul quale vengano redirette le richieste di comunicazione sulla porta *c* della macchina *C* porti i pacchetti alla macchina *S*, che a sua volta inoltrerà i pacchetti ricevuti alla porta *i* della macchina *I*. E viceversa. È chiaro che la sicurezza della comunicazione sarà in questo caso limitata soltanto al tratto che collega la macchina *C* con la macchina *I*.

??? figura ???

Ad esempio, se si vuole rendere sicuro lo scarico di posta elettronica effettuato tramite il protocollo POP (porta 110) tra la macchina **mypc.mydomain.it**, sulla quale c'è il client di posta elettronica (ad esempio *Evolution*), ed il server di posta (macchina **mail.mydomain.it**), si può operare come segue. Innanzitutto sulla macchina **mail.mydomain.it** deve essere in esecuzione **sshd** e sulla macchina **mypc.mydomain.it** si può impartire il comando seguente:

```
$ ssh -L 1100:mail.mydomain.it:110 mail.mydomain.it
```

Quindi si devono modificare le impostazioni del client di posta elettronica (su **mypc.mydomain.it**) in modo tale che per scaricare la posta non si colleghi più alla porta 110 di **mail.mydomain.it**, ma si colleghi alla porta 1100 di **mypc.mydomain.it**. In questo modo, quando il client di posta si conatterà alla porta 1100 di **mypc.mydomain.it**, la comunicazione verrà rediretta su un canale sicuro verso **mail.mydomain.it** che, ricevuti i pacchetti, li inoltrerà a sé stesso sulla porta 110. Qui ci sarà il POP server che risponderà al mittente, anch'esso attraverso il canale di comunicazione sicuro.

Nel caso in cui sulla macchina **mail.mydomain.it** non sia in esecuzione **sshd**, si può reindirizzare la comunicazione sicura verso un'altra macchina sulla quale c'è **sshd** (ad esempio **other.mydomain.it**) e quindi impartire sulla macchina **mypc.mydomain.it** il comando seguente

```
$ ssh -L 1100:mail.mydomain.it:110 other.mydomain.it
```

Quindi si devono modificare le impostazioni del client di posta elettronica (su **mypc.mydomain.it**) come descritto precedentemente. Quando il client di posta si conatterà alla porta 1100 di **mypc.mydomain.it**, la comunicazione verrà rediretta su un canale sicuro verso **other.mydomain.it** che, ricevuti i pacchetti, li inoltrerà a **mail.mydomain.it** sulla porta 110. Qui ci sarà il POP server che risponderà al mittente, anch'esso attraverso lo stesso meccanismo. In questo caso però la comunicazione avviene in modo sicuro soltanto tra **mypc.mydomain.it** e **other.mydomain.it**. ???

24.2.5 scp

Il comando **scp** (secure **copy** - man page **scp(1)**) permette di copiare, in modo sicuro, file e directory tra due macchine.

???

24.2.6 sftp

Il comando **sftp** (secure **FTP** - man page **sftp(1)**) è un client FTP che permette di scambiare, in modo sicuro, file e directory tra due macchine secondo il protocollo FTP.

???

???

24.3 GNU Privacy Guard

GNU Privacy Guard, o più comunemente GNUPG o ancora GPG, è una suite per la gestione di chiavi crittografiche e della cifratura stessa. Tale strumento costituisce il backend utilizzabile da qualunque altra applicazione che desideri far uso di chiavi per cifrare/decifrare informazioni crittografate.

GPG è un sostituto di PGP (Pretty Good Privacy), un'applicazione creata da P. Zimmermann nel 1991, implementa il protocollo OpenPGP (RFC 2440) utilizzando soltanto algoritmi di cifratura non coperti da patent (DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 e TIGER). Le chiavi sono memorizzate in insiemi detti portachiavi o *keyring* che sono rappresentati da file all'interno della directory `~/.gnupg`.

??? ID ??? ??? fingerprint ???

Il comando principale di GNU Privacy Guard è `gpg` (man page `gpg(1)`).

Comando: `gpg`

Path: `???/gpg`

SINTASSI

`gpg [option] command`

DESCRIZIONE

option indica la modalità di funzionamento di `gpg`. Può assumere i seguenti valori:

`-b nbit`

indica il numero di bit (*nbit*) utilizzati per la chiave del server (SSH1) (per default è 768);

command indica la modalità di funzionamento di `gpg`. Può assumere i seguenti valori:

`-s` | `--sign`

esegue un'operazione di firma;

`--clearsign`

esegue un'operazione di firma in chiaro;

`-b` | `--detach-sign`

esegue un'operazione di firma non inglobata nel messaggio;

`-e` | `--encrypt`

esegue un'operazione di cifratura del messaggio;

`-c` | `--symmetric`

esegue un'operazione di cifratura a chiave simmetrica del messaggio;

`--store`

solo memorizzazione (v. RFC 1991);

`--decrypt [filename]`

esegue un'operazione di decifratura dello standard input o del file *filename*. L'output verrà visualizzato sullo schermo o memorizzato in un file specificato dal comando `--output`. Se il messaggio è firmato, viene verificata anche la firma;

`--verify [sigfile [signed-files]]`

verifica la firma contenuta nel file *sigfile* (o dallo standard input, se *sigfile* non è specificato). Nel caso in cui sia specificato *signed-files* questo viene considerato come il messaggio firmato, la cui firma è contenuta nel file *sigfile*, altrimenti viene cercato un file con estensione *.sig* o *.asc*;

`--verify-files [files]`

esegue la verifica della firma come `--verify` ma lavora su file che contengono sia il messaggio firmato che la firma;

`--list-keys [names]` | `--list-public-keys [names]`

elenca le chiavi del portachiavi pubblico o soltanto quelle specificate da *names*;

`--list-secret-keys [names]`

elenca le chiavi del portachiavi privato (segreto) o soltanto quelle specificate da *names*;


```

--list-sigs [names]
    come --list-keys ma elenca anche le firme relative alle chiavi;
--check-sigs [names]
    come --list-sigs ma verifica anche le firme relative alle chiavi;
--fingerprint [names]
    come --list-keys ma elenca anche le fingerprint relative alle chiavi
    (se specificato 2 volte, vengono visualizzate le fingerprint relative
    anche alle chiavi secondarie);
--list-packets
    elenca la sequenza dei pacchetti (utile per il debug);
--gen-key
    genera una nuova coppia di chiavi (pubblica/privata);
--edit-key name
    attiva la modalità di gestione delle chiavi. In questa modalità sono
    possibili vari comandi:
    sign    esegue un'operazione di firma sulla chiave dell'utente il cui
            username è name;
    lsign   come sign ma la firma è impostata come non esportabile,
            ovvero la firma non sarà utilizzabile da altri. Questo è utile
            per fare in modo che le chiavi siano valide soltanto in un
            ambiente locale;
    revsig  richiede la revoca della firma;
    trust   cambia l'affidabilità relativa alla veridicità del rapporto tra
            la chiave ed chi afferma di esserne il proprietario;
    disable disabilita una chiave;
    enable  abilita una chiave;
    adduid  crea un altro user ID;
    deluid  elimina uno user ID;
    addkey  aggiunge una sottochiave alla chiave considerata;
    delkey  rimuove una sottochiave;
    revkey  revoca una sottochiave;
    expire  cambia la data di termine della validità della chiave;
    passwd  cambia la passphrase relativa alla chiave privata;
    uid n  imposta la selezione dello user id con indice n (n = 0 li
            deselecta tutti);
    key n  imposta la selezione della sottochiave con indice n (n = 0 le
            deselecta tutte);
    check   controlla tutti gli user id selezionati;
    pref    elenca le preferenze;
    toggle  cambia la modalità di visualizzazione tra le chiavi pubbliche
            e quelle private;
    save    memorizza i cambiamenti effettuati sulle chiavi ed esce;
    quit    esce senza memorizzare le variazioni effettuate sulle chiavi;

```

???

Eseguendo **gpg** da un utente diverso dal superuser è possibile che venga visualizzato un messaggio di avvertimento analogo a **'gpg: Warning: using insecure memory!'**, in quanto gli utenti che non hanno i privilegi del superuser non possono bloccare le pagine di memoria (il kernel può scaricare sul filesystem – swap – le pagine di memoria di qualunque processo, a meno che queste non siano state bloccate e per farlo il processo in questione deve avere i privilegi di superuser).

Potrebbero essere visualizzati anche altri messaggi di avvertimento, come **'gpg: WARNING: unsafe permissions on configuration file "/home/username/.gnupg/gpg.conf"'**, che indica che il file in questione è leggibile da altri utenti (oltre che dal superuser), ed analogamente il messaggio **'gpg: WARNING: unsafe enclosing directory permissions on configuration file'** indica che la directory contenente la configurazione di **gpg** è leggibile da altri utenti.

24.4 Cifratura della posta elettronica

???

24.5 VPN - Virtual Private Network

???

24.6 Kerberos

Kerberos

Nel 1983, dalla collaborazione del MIT, *Digital Equipment Corp.* ed *IBM*, è stato sviluppato il sistema di autenticazione di rete **Kerberos**. Come il nome ricorda il cane a tre teste della mitologia greca a guardia degli inferi (Cerbero), tale simbolo si rifà ai tre scopi principali del progetto: autenticazione, autorizzazione e accounting. Al momento Kerberos è arrivato alla versione 5 (v. RFC 1510).

Kerberos utilizza un sistema di autenticazione a terza parte fidata. I *principal* (utenti e macchine) si autenticano tramite un KDC (Key Distribution Center) che rappresenta la terza parte fidata. Questo è possibile grazie al fatto che ogni principal condivide una chiave con il KDC.

L'autenticazione può essere sia one-way che two-way, ovvero un principal si autentica ad un altro che se lo desidera può autenticarsi a sua volta a lui.

Ogni autenticazione ha una periodo di vita limitato (circa la durata di una sessione di login)

La richiesta delle credenziali avviene soltanto nel momento di inizio della sessione, ovvero una gestione di tipo *single-sign-on*.

???

principal

Un **principal** è un'entità alla quale Kerberos può assegnare ticket. Gli identificativi dei principal sono suddivisi generalmente in tre parti, secondo la seguente sintassi:

primary/instance@realm

dove:

primary

è la prima parte del principal, ovvero quella che lo identifica più in dettaglio. Ad esempio per un utente il principal può essere il suo username, per una macchina il suo nome, ...;

instance

specifica ulteriormente la parte *primary* aggiungendo delle informazioni, ma può anche essere omessa. Ad esempio per un utente tale informazione può non esistere (john@KERB.MYDOMAIN.COM) oppure essere specificata per qualificarne le mansioni (john/admin@KERB.MYDOMAIN.COM). I principal john@kerb.mydomain.com e john/admin@kerb.mydomain.com sono considerati a tutt'oggi effetti diversi, con permessi e password separate. Per una macchina può rappresentare il suo FQDN (host/zeus.mydomain.com@KERB.MYDOMAIN.COM);

realm

indica l'ambiente (rete) gestito dal KDC considerato. Nella maggior parte dei casi è il nome del dominio in lettere maiuscole;

Un client (un utente o un servizio) invia una richiesta di autenticazione al KDC. Il KDC crea un'apposito ticket, detto TGT (Ticket Granting Ticket), che invia al client. Poiché il TGT è cifrato con la password del client, che il KDC deve conoscere, il client è in grado di decifrarlo e può utilizzarlo per ottenere i ticket per i servizi di rete che utilizzano tale protocollo. Da questo momento, qualora il client desideri autenticarsi ad un server, invia il TGT al KDC che risponde con un ticket valido per l'autenticazione. La richiesta e la concessione dei ticket per i vari servizi tramite il TGT avviene in

maniera trasparente all'utente. Quindi il KDC fornisce due servizi, in quanto funge da AS (Authentication Server), poiché esso è il repository delle chiavi dei principal, e da TGS (Ticket Granting Server), poiché risponde con i TGT.

Ogni messaggio cifrato è contenuto in un involucro (ASN.1) che, oltre a contenere il messaggio cifrato, indica l'algoritmo di cifratura utilizzato e la versione della chiave.

Un client (un utente o un servizio) invia una richiesta di autenticazione ad un AS, richiedendo le credenziali per un certo server. L'AS, che contiene l'elenco dei principal e le relative chiavi, risponde con le credenziali cifrate con la chiave del client. Le credenziali consistono in un ticket per il server ed una chiave di cifratura temporanea (detta anche chiave di sessione). Il client trasmette un ticket contenente l'identità del client ed una copia della chiave di sessione, il tutto cifrato con la chiave del server al server che vuole utilizzare. In questo modo il client ed il server condividono la stessa chiave di sessione che viene utilizzata per autenticare il client ed eventualmente il server. Essa può essere anche utilizzata per cifrare la comunicazione successiva o per scambiare altre chiavi di sessione con le quali cifrare la comunicazione.

??? figura ???

Per verificare l'identità dei principal in una comunicazione, il client trasmette il ticket al server. Il ticket contiene una parte detta authenticator che include un timestamp ed è cifrata con la chiave di sessione. In questo modo il client prova al server che il ticket è stato richiesto di recente e che il ticket è stato generato da un principal che conosce la chiave di sessione.

L'integrità dei messaggi scambiati è garantita per mezzo di un checksum (un message digest) che è cifrato con la chiave di sessione.

Kerberos può essere utilizzato anche per autenticazioni tra ambienti (*realm*) diversi. Ogni realm ha un proprio KDC che autentica i propri principal. Il nome di ogni principal comprende anche quello del realm a cui appartiene.

Per far sì che un principal di un realm si possa autenticare con un principal di un altro realm, si devono stabilire delle chiavi inter-realm relative ai KDC dei realm interessati. Nel KDC del realm *A* viene memorizzata la chiave inter-realm del KDC del realm *B* e viceversa, in maniera tale che il KDC del realm *A* diviene un principal del realm *B* ed il KDC del realm *B* un principal del realm *A*. Un principal del realm *A* può quindi richiedere l'autenticazione verso un principal del realm *B* ed il KDC del realm *A* fornisce al principal che desidera autenticarsi un TGT valido per il realm *B* (cifrato con la chiave inter-realm in maniera tale che il KDC del realm *B* possa decifrarlo).

??? figura ???

24.7 Riferimenti

???

Capitolo 25

Protezione del sistema

“Essere preparati alla guerra è uno dei mezzi più efficaci di preservare la pace.”
– G. Washington

Un sistema è sicuro quando non c'è alcuna possibilità di accedervi da parte di una persona che non sia a conoscenza di un account utente. Poiché il software gestisce le interazioni tra le persone e le macchine ed il software è realizzato da persone, le quali non sono infallibili, va da sé che nessun sistema può essere dichiarato assolutamente sicuro, a meno che l'accesso allo stesso non sia fisicamente impedito alle persone non autorizzate.

È talvolta necessario però poter far accedere il sistema dall'esterno, attraverso la rete. Si pensi infatti ai server web (HTTP), ai DNS, ai server FTP, che sono di grande utilità in una rete all'interno di un'azienda o magari tra più aziende distribuite sul territorio nazionale o magari in tutto il mondo. Nasce quindi l'esigenza di limitare gli accessi possibili ai processi che girano su una macchina dall'esterno. La risposta a queste esigenze sono le tecniche di firewalling.

C'è sempre comunque la possibilità che un attacco riesca a passare le misure di sicurezza presenti su un sistema. Questo è il campo dei sistemi di rilevamento delle intrusioni (IDS – Intrusion Detection System).

25.1 Gli attacchi

Sostanzialmente nel software sono presenti delle falle (*bug*) che lo fanno agire in maniera non prevista. Tali falle possono essere opportunamente sfruttate da persone che ne sono a conoscenza per accedere al sistema come se si trattasse di un utente dello stesso. Si parla in tal caso di **exploit** del programma. In particolare la persona esterna accede al sistema con i privilegi dell'utente con i quali viene eseguito il processo con la falla. *exploit*

La sicurezza di un sistema dipende quindi dalla qualità del software che gira sul sistema stesso, sia quindi il sistema operativo che le applicazioni che gestiscono i servizi messi a disposizione dalla macchina verso l'esterno (sulla rete).

I cosiddetti *cracker*¹ lanciano dei veri e propri attacchi ai sistemi per cercare di penetrarvi o per comprometterne comunque l'utilizzo (**DoS** – Denial of Service), adottando tecniche che si basano sulle falle del software dovute sia a cattiva scrittura del codice (chiamate a funzioni di sistema non controllate, ...) o addirittura codice errato (controlli errati, condizioni di funzionamento non testate, ...). È evidente che i bug possono essere di varia importanza per quanto riguarda la sicurezza: da quelli che comunque non permettono all'utente di accedere a funzionalità a lui non permesse a quelli che permettono all'utente di accedere anche ad una funzionalità a lui non permessa che seppur minima può aprire un varco nel sistema. *DoS*

¹È bene distinguere tra *hacker* e *cracker*: un *hacker* è una persona che ha la passione di conoscere, scoprire, analizzare il codice ed i programmi e magari anche di trovarne i bug, mentre un *cracker* sfrutta i bug a proprio vantaggio per “intrufolarsi” nei sistemi altrui, accedere alle informazioni private e magari bloccare l'utilizzo del sistema stesso.

Una delle tecniche più utilizzate per tentare di penetrare in un sistema è il *buffer overflow* che permette di scrivere in una cella di memoria “casuale” un valore desiderato. A titolo di esempio, un programma scritto in C che chiama la funzione `scanf` è soggetto a tale tipo di attacco (o come si dice in gergo *exploit*). La funzione `scanf` prende una stringa inserita con la tastiera e la memorizza in una variabile (in memoria). Poiché `scanf` non fa nessun controllo sulla lunghezza della stringa inserita da tastiera, è possibile inserire una stringa di lunghezza superiore a quella contenibile all’interno della variabile in cui questa verrà memorizzata e quindi i dati inseriti verranno memorizzati all’interno della variabile finché questa ne può contenere ed il resto viene memorizzato nelle celle di memoria successive a quelle relative alla variabile. In questo modo si può riuscire a sovrascrivere valori presenti in parti della memoria altrimenti inaccessibili che magari si riferiscono a variabili che controllano il funzionamento del processo. Quindi si può riuscire a far funzionare il processo in modo non previsto per l’utente in questione, permettendo all’utente di compiere operazioni non previste.

???

I sistemi operativi attuali sono dei software generalmente ben testati e rodati che presentano ben poche falle di sicurezza (considerando soltanto le funzionalità di base). Visto che i bug dipendono dal software, ragionando anche semplicemente da un punto di vista statistico, più applicazioni sono in esecuzione in un sistema, più è probabile che vi siano bug che possono compromettere la sicurezza del sistema stesso. In genere quindi, più servizi vengono offerti da un sistema (web server, FTP server, SSH server, ...), più questo è soggetto ad avere falle di sicurezza. In riferimento ai sistemi GNU/Linux, sono i processi (demoni) relativi ai servizi come `httpd`, `ftpd`, `sshd`, ..., che possono presentare falle che possono compromettere la sicurezza del sistema. Per questo motivo è consigliabile offrire all’esterno il minor numero di servizi possibile, o per lo meno controllare chi è che comunica da remoto con il sistema considerato (anche se esistono comunque dei meccanismi di “falsificazione” dei pacchetti relativi alla comunicazione). A tal proposito si può decidere di non attivare determinati servizi non lanciando in esecuzione i relativi demoni o se si desidera utilizzare certi servizi su una rete che si ritiene sicura (una rete privata che si ha sotto controllo) e non renderli disponibili per chiunque (*Internet*) si possono definire delle politiche di accesso ai servizi offerti dal sistema tramite un *firewall*.

Il meccanismo di comunicazione stessa permette tra l’altro la possibilità di procurare dei malfunzionamenti a sistemi remoti a causa di un suo uso non “canonico”. Esistono quindi “attacchi” a sistemi, i più noti dei quali sono di seguito riportati, che mirano a compromettere l’utilizzo dei servizi da essi offerti.

SYN flooding

per com’è definito il protocollo TCP, all’inizio di una connessione viene effettuato il three-way handshake: il client invia un TCP segment al server con il SYN attivo (SYN segment) al quale il server risponde con un TCP segment con i bit SYN e ACK attivi. Il client risponde a sua volta per confermare la ricezione della risposta del server inviando al server stesso un altro TCP segment con solo l’ACK attivo e da quel momento la connessione è stabilita. Durante il periodo tra l’invio del TCP segment SYN-ACK e l’invio del relativo TCP segment ACK, la connessione è detta mezza aperta (half-open connection) e per questo il server utilizza parte della memoria nella quale vengono memorizzate le strutture dati relative alla connessione che sta per essere stabilita.

L’attacco consiste nell’inviare un numero molto elevato di richieste di connessioni fittizie, ovvero un numero elevato di SYN segment (da cui il nome SYN flooding), in maniera tale che il server non riesca più a rispondere ad altre richieste poiché è troppo impegnato a rispondere a quelle che sta ricevendo. Si crea così un DoS. Il server, che magari non è stato pensato per gestire un numero di connessioni troppo elevato, potrebbe persino comportarsi in maniera anomala e/o contenere dei bug che l’attaccante può sfruttare per accedere al sistema.

IP spoofing

???

Smurf

???

Sniffing

è una delle tecniche di base più utilizzate da un attaccante. Consiste nel ricevere una copia di tutti i pacchetti che transitano da e verso un computer bersaglio, sul quale successivamente sferrare un attacco.

Hijacking

???

E-mail bombing (spamming)

???

???

25.2 L'installazione del software

Il software costituisce l'elemento fondamentale per l'uso del sistema, ma bisogna essere sicuri che esso effettui esattamente le operazioni che da esso ci aspettiamo. Ad esempio si potrebbe avere

???

25.3 I firewall

Un **firewall** (porta taglia-fiamma) è un dispositivo che permette di filtrare il traffico di rete tra due (o più) interfacce, in base a determinate politiche (*policy*). L'utilizzo di un firewall rende possibile controllare chi accede ai servizi messi a disposizione dal sistema e a quali altri sistemi si può accedere, offrendo la sicurezza desiderata ed eventualmente tracciando il traffico in un file, per poterlo controllare successivamente. *firewall*

In commercio esistono apparecchi dedicati a svolgere il compito di firewall, spesso denominati *firewall hardware* per contraddistinguerli dai software che svolgono tale compito su macchine con sistemi operativi multipurpose, utilizzabili cioè per poterci lavorare in generale, che possono avere, come GNU/Linux, un *firewall software*. Questo non deve trarre in inganno poiché la politica di firewalling è comunque gestita attraverso un insieme di regole che vengono attuate attraverso un software.

Esistono essenzialmente due tipologie di firewall

Packet filter esegue un filtraggio sui pacchetti che lo attraversano, dal livello fisico fino al livello di trasporto dello stack OSI (v. cap. 16). Ad esempio, un firewall di questo tipo può scartare i pacchetti che arrivano da interfacce di rete con un determinato indirizzo IP, o far passare soltanto il traffico relativo a determinate porte (TCP o UDP).

Proxy server esegue un filtraggio sui pacchetti che lo attraversano, ai livelli più alti dello stack OSI. Un firewall di questo tipo (spesso denominato soltanto *proxy server*), può permettere, oltre alla gestione di caching delle pagine visitate, anche l'accesso o meno a determinate pagine web basandosi sul contenuto delle stesse.

Un firewall può essere caratterizzato in base alla possibilità di gestire il controllo sui pacchetti che lo attraversano dipendentemente dallo stato della relativa comunicazione. Si identificano così i seguenti tipi di firewall

Stateless il firewall non tiene conto dello stato della comunicazione, ma analizza i pacchetti isolatamente l'uno dall'altro, senza tener conto di nessuna correlazione tra essi.

Stateful il firewall tiene conto dello stato della comunicazione, permettendo l'analisi dei pacchetti in relazione, per esempio, al fatto che la comunicazione tra il mittente ed il destinatario sia già stata iniziata o meno. Un firewall di questo tipo è generalmente più sicuro rispetto ad uno *stateless* poiché permette di filtrare i pacchetti in maniera più selettiva.

NAT
PAT

I firewall in genere possono implementare funzionalità di **NAT** (Network Address Translation) e **PAT** (Port Address Translation), che consentono al firewall di modificare automaticamente l'indirizzo IP o la porta del mittente o destinatario del pacchetto. Queste funzionalità permettono a più sistemi collegati in una rete privata di poter accedere ad *Internet* attraverso il firewall.

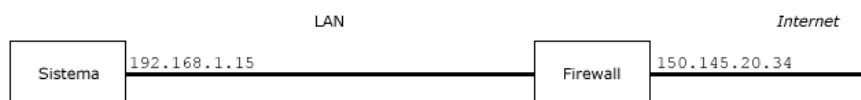


Figura 25.1: Schematizzazione del meccanismo di NAT.

Il firewall infatti provvede a modificare opportunamente gli indirizzi IP del pacchetto. Si supponga che un sistema della rete privata con un'interfaccia che ha l'indirizzo IP 192.168.1.15 voglia accedere al sito *Internet* con indirizzo IP 216.239.37.99. Se il firewall fungesse soltanto da router, il pacchetto inviato dall'interfaccia 192.168.1.15 avrebbe appunto 192.168.1.15 come indirizzo IP del mittente e 216.239.37.99 come indirizzo IP di destinazione. Questo funziona fintantoché il pacchetto arriva al server con interfaccia avente indirizzo IP 216.239.37.99. Ma quando questo provvede a rispondere al mittente (192.168.1.15), il pacchetto viene scartato (letteralmente buttato via) dal primo router ben configurato, poiché l'indirizzo IP 192.168.1.15, come illustrato nel cap. 16, si riferisce ad una rete privata e non può essere utilizzato come indirizzo pubblico (per *Internet*). Quindi, per far comunicare l'interfaccia 192.168.1.15 con il sito 216.239.37.99 il firewall deve essere configurato per funzionare con il meccanismo di NAT: quando il pacchetto con indirizzo IP mittente 192.168.1.15 e destinatario 216.239.37.99 arriva al firewall, questo viene modificato dal firewall stesso che sostituisce l'indirizzo dell'interfaccia del mittente con l'indirizzo IP della sua interfaccia connessa ad internet (150.145.20.34). Quindi inoltra il pacchetto. Quando l'interfaccia 216.239.37.99 riceve il pacchetto, essa lo elabora e quindi risponde al mittente, cioè all'indirizzo 150.145.20.34 (il firewall), che ricevendo il pacchetto di risposta e ricordandosi che quello si riferisce al pacchetto di richiesta precedentemente inviato dall'interfaccia 192.168.1.15 (questa informazione il firewall se l'era salvata da qualche parte) sostituisce l'indirizzo IP del destinatario del pacchetto con 192.168.1.15 (cioè l'indirizzo IP dell'interfaccia che aveva inviato il primo pacchetto) e quindi inoltra il pacchetto sulla rete privata. A questo punto l'interfaccia 192.168.1.15 riceverà la risposta alla sua richiesta.

In questo modo è possibile far accedere ad *Internet* un insieme di computer connessi in una rete privata, senza dover assegnare ad ognuna delle loro interfacce un indirizzo IP valido per *Internet*. È chiaro che questo meccanismo permette a più macchine l'accesso ad *Internet*, ma questo avviene attraverso un'unica interfaccia e quindi le varie macchine si divideranno la banda di comunicazione a disposizione per tale accesso, cioè se si tratta di un accesso acon un modem a 56 kbit/s e le macchine collegate ad internet sono 3, ognuna di esse avrà un collegamento a circa 18.7 kbit/s (questo è vero se tutte e 3 le macchine stanno inviando o ricevendo dei pacchetti contemporaneamente, altrimenti la banda che ogni macchina può sfruttare è maggiore).

L'utilizzo di NAT implica anche il PAT sulla porta sorgente poiché più macchine comunicano con l'esterno. Si supponga ad esempio che una delle macchine presenti nella rete privata invii un pacchetto verso *Internet* con indirizzo IP del mittente 192.168.1.43 e porta mittente 5012. Il firewall esegue il NAT sostituendo l'indirizzo IP del mittente con quello della sua porta esterna (es. 194.143.24.15) e quindi tenta di inviare il pacchetto

verso il destinatario. Se però un'altra macchina della rete privata avesse già instaurato una comunicazione con lo stesso server remoto dalla stessa porta mittente, il firewall deve cambiare la porta mittente (si ricorda che una connessione è univocamente identificata dalla quaterna $\langle \text{indirizzo_IP_mittente}, \text{porta_mittente}, \text{indirizzo_IP_destinatario}, \text{porta_destinatario} \rangle$).

25.3.1 Packet filtering

Un firewall che effettua il *packet filtering* (filtraggio dei pacchetti) gestisce i pacchetti che lo attraversano in base ad opportune regole che si riferiscono ai protocolli di più basso livello del modello OSI, dal livello fisico fino a quello di trasporto.

Il meccanismo di firewalling sui sistemi GNU/Linux (dal kernel 2.4 in poi) è *Netfilter*². Si tratta di un *packet filter* dotato di *stateful inspection*. Il packet filtering è implementato nel kernel, ma la parte relativa al filtraggio avviene tramite dei moduli: in questo modo il sistema di filtraggio risulta dinamicamente estensibile.

Poiché il meccanismo di firewalling è un'attività del kernel, per il suo funzionamento è necessario che il kernel sia stato opportunamente compilato e accompagnato dagli eventuali moduli necessari.

Per default un sistema GNU/Linux non permette ai pacchetti di transitare da un'interfaccia di rete ad un'altra, cioè non fa da *router* (ovvero non permette il *forwarding* dei pacchetti), ma questa caratteristica può essere abilitata facendo contenere '1' al file `/proc/sys/net/ipv4/ip_forward`. Ciò può essere fatto con il comando seguente

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

Questo va bene per IPv4, ma per la versione IPv6 è necessario inserire il valore '1' nel file `/proc/sys/net/ipv6/conf/all/forwarding`, che, analogamente a quanto visto prima, può essere fatto con il comando seguente

```
# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

Poiché tale impostazione è relativa al filesystem virtuale `/proc`, non viene memorizzata sulla memoria di massa, pertanto se si desidera che tale impostazione sia attivata ad ogni avvio del sistema, è necessario impartire tale comando ogni volta che il sistema viene riavviato (magari per mezzo di un apposito script).

I pacchetti che transitano su *Netfilter* vengono esaminati, secondo un insieme di **regole** (*rule*) raggruppate in **chain** (catene o *punti di controllo*) assegnate a **tabelle** (*table*). Le regole determinano la politica di filtraggio dei pacchetti.

regole
chain
tabelle

Ogni regola specifica una *condizione* ed un *target* (obiettivo), cioè l'azione che *Netfilter* deve intraprendere sul pacchetto in transito, quando questo soddisfa la condizione indicata dalla regola considerata.

Un pacchetto può essere destinato alla macchina che funge da firewall o ad un'altra macchina collegata in rete a quest'ultima. L'instradamento corretto del pacchetto è compito del meccanismo di routing (v. cap. 16).

Il funzionamento di *Netfilter* si basa sull'insieme di tabelle e chain illustrato in fig. 25.2; le tabelle e le chain predefinite sono elencate rispettivamente in tab. 25.1 e tab. 25.2. Si possono comunque definire altre chain per gestire in maniera più modulare le politiche di sicurezza, ovvero le regole di filtraggio dei pacchetti.

La tabella **mangle** supporta le chain INPUT, FORWARD e POSTROUTING dal kernel 2.4.18.

Netfilter è un meccanismo di firewalling dotato di *stateful inspection* dei pacchetti, reso possibile dal **connection tracking**, ovvero un meccanismo con il quale *Netfilter* tiene traccia dello stato delle connessioni. Tale ruolo è implementato da *Conntrack* che risiede nel kernel (oppure può essere caricato come modulo `ip_conntrack`) e consente di gestire gli stati elencati in tab. 25.3.

connection tracking

²v. <http://www.netfilter.org>.

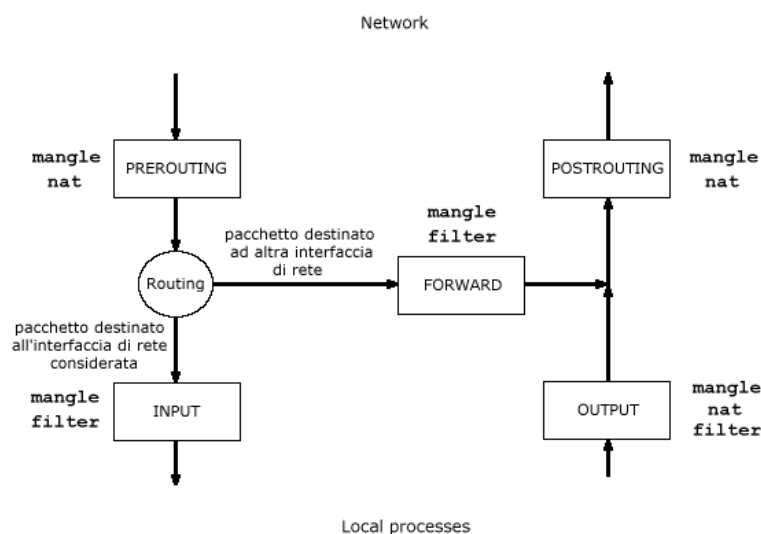


Figura 25.2: Schematizzazione del funzionamento di *Netfilter*.

Tabella	Descrizione
filter	tabella relativa al filtraggio dei pacchetti (tabella di default).
nat	tabella relativa al NAT. Viene consultata quando un pacchetto tenta di effettuare una nuova connessione.
mangle	tabella relativa alle direttive per l'alterazione di parti dei pacchetti che passano dal firewall.

Tabella 25.1: Tabelle utilizzate da *Netfilter*.

Chain	Descrizione
PREROUTING	contiene le regole relative ai pacchetti che arrivano in ingresso al firewall.
FORWARD	contiene le regole relative ai pacchetti che vengono inoltrati dal firewall verso altre interfacce di rete.
INPUT	contiene le regole relative ai pacchetti che arrivano in ingresso al firewall (ai processi locali).
OUTPUT	contiene le regole relative ai pacchetti che vengono generati localmente dal firewall.
POSTROUTING	contiene le regole relative ai pacchetti che escono dal firewall.

Tabella 25.2: Chain predefinite di *Netfilter*.

Stato	Descrizione
NEW	indica che il pacchetto è relativo ad una nuova connessione.
ESTABLISHED	indica che il pacchetto appartiene ad una connessione già instaurata.
RELATED	indica che il pacchetto si riferisce ad una connessione già instaurata (ESTABLISHED) – ad es. un pacchetto ICMP che segnala un errore della comunicazione TCP o il flusso dati di un server FTP (la connessione è sulla porta 21 ma il flusso dati avviene sulla porta 20).
INVALID	indica che non è possibile identificare lo stato relativo alla comunicazione alla quale si riferisce il pacchetto (in genere è una buona idea scartare tutti i pacchetti la cui comunicazione è in questo stato).
SNAT	indica che l'indirizzo IP del mittente del pacchetto precedentemente ricevuto è diverso da quello di destinazione del pacchetto in uscita.
DNAT	indica che l'indirizzo IP del destinatario del pacchetto precedentemente inviato è diverso da quello del mittente del pacchetto ricevuto.

Tabella 25.3: Stati su cui si basa *Conntrack*.

Poiché il conection tracking non funziona a dovere nel caso di frammentazione dei pacchetti, *Conntrack*, se attivato, provvede automaticamente a disabilitare la deframmentazione dei pacchetti.

La gestione del connection tracking viene effettuata nelle chain PREROUTING e OUTPUT della tabella nat.

Lo stato delle connessioni è consultabile attraverso il file `/proc/net/ip_conntrack`. Il suo contenuto ha la seguente sintassi

protocol proto_num exp.time status details

dove

protocol è il protocollo di più alto livello (massimo livello di trasporto) riconosciuto del pacchetto;

proto_num è il numero identificativo del protocollo;

exp.time è il numero di secondi di “vita” rimanenti per l’annotazione (timeout). Tale valore viene decrementato automaticamente. I tempi di default sono quelli riportati in tab. 25.4.

Status	Timeout di default
NONE	30 minuti
ESTABLISHED	5 giorni
SYN_SENT	2 minuti
SYN_RECV	60 secondi
FIN_WAIT	2 minuti
TIME_WAIT	2 minuti
CLOSE	10 secondi
CLOSE_WAIT	12 ore
LAST_ACK	30 secondi
LISTEN	2 minuti

Tabella 25.4: Valori di default per *exp.time*.

Tali valori sono comunque impostabili attraverso i file `/proc/sys/net/ipv4/netfilter/ip_ct.*`. I valori sono espressi in **jiffies** (centesimi di secondo), quindi il valore 3000 significa 30 secondi. In particolare, il valore contenuto nel file `/proc/sys/net/ipv4/netfilter/ip_ct_generic_timeout` (che per default corrisponde a 10 minuti) è il valore utilizzato per le connessioni relative a protocolli che *Conntrack* non comprende;

status è lo stato attuale della connessione relativa (es. `SYN_SENT`);

details riporta i dettagli della comunicazione (es. indirizzo IP e porta del mittente ed indirizzo IP e porta del destinatario) sia per i messaggi di richiesta che per quelli di risposta (i messaggi di risposta avranno i dati relativi al mittente scambiati con quelli relativi al destinatario);

Nel caso di PAT implicito, le porte del mittente sono divise in tre classi: 0 - 511, 512 - 1023, 1024 - 65535. Una porta che appartiene ad una di tali classe sarà sempre mappata in un'altra appartenente alla stessa classe.

Le annotazioni vengono rimosse quando scade il loro *exp.time*, tranne quelle marcate come `[ASSURED]`. Queste vengono eliminate soltanto quando viene raggiunto il massimo numero di connessioni tracciabili. Tale valore dipende dalla quantità di memoria centrale presente sulla macchina (per macchine con 256 MB di RAM è 16376) ed è contenuto nel file `/proc/sys/net/ipv4/ip_conntrack_max`.

Netfilter è gestito con il comando `iptables` (man page `iptables(8)`)³.

³con i kernel antecedenti alla versione 2.4 la gestione del firewall avveniva con il comando `ipchains`.

Comando: **iptables**

Path: **/sbin/iptables**

SINTASSI

iptables [-h | -v | {-t | --table} *table*] [*command*] [*option*] [*match*] [*target*]

DESCRIPTION

-h visualizza un aiuto sommario di **iptables**;

-v visualizza la versione di **iptables**;

-t *table* indica la tabella di riferimento **filter**, **nat** o **mangle** (se non viene specificato, viene considerata la tabella **filter**);

command indica il comando da far eseguire a **iptables**. Può assumere i seguenti valori:

{-A | --append} *chain rule*

aggiunge una o più regole (*rule*) alla *chain*;

{-D | --delete} *chain rule*

elimina una o più regole (*rule*) dalla *chain* (*rule* può essere in forma di regola o semplicemente il numero relativo alla posizione della regola da eliminare all'interno della *chain* - la prima regola è la numero 1);

{-I | --insert} *chain* [*rule*] *rule*

inserisce una o più regole (*rule*) nella *chain* con il numero di ordine *rule* (*rule*) (se non viene specificato *rule*, la regola viene inserita come prima regola della *chain*);

{-R | --replace} *chain rule* *rule*

sostituisce la regola alla posizione (*rule*) nella *chain* con quella specificata da *rule*;

{-L | --list} [*chain*]

elenca le regole contenute nella *chain*. Se *chain* non è specificata, vengono elencate tutte le *chain*;

{-F | --flush} [*chain*]

elimina tutte le regole contenute nella *chain*. Se *chain* non è specificata, vengono eliminate tutte le *chain* relative alla tabella considerata;

{-Z | --zero} [*chain*]

azzerà i contatori dei pacchetti e dei byte relativi a tutte le *chain*;

{-N | --new-chain} *chain*

crea una nuova *chain* con il nome specificato da *chain*;

{-X | --delete-chain} [*chain*]

elimina la *chain* specificata da *chain* (la *chain* deve essere vuota e non può essere una di quelle predefinite). Se *chain* non viene specificato, vengono eliminate tutte le *chain* relative alla tabella considerata;

{-P | --policy} [*chain*] [*target*]

imposta la politica di default per la *chain* specificata da *chain* relativamente all'obiettivo *target* (soltanto le *chain* predefinite hanno una politica di default);

{-E | --rename-chain} *old-chain new-chain*

rinomina la *chain* *old-chain* in *new-chain* (non funziona con le *chain* predefinite).

option indica l'opzione di funzionamento di **iptables**. Può assumere i seguenti valori:

-v | --verbose

(utilizzato con i comandi **-L**, **-A**, **-I**, **-D** e **-R**) indica di visualizzare un output più verboso;

-x | --exact

(utilizzato con il comando **-L**) indica di visualizzare il numero esatto relativamente ai contatori senza le lettere **K**, **M** e **G** (le lettere **K**, **M** e **G** relative ai contatori si riferiscono rispettivamente a kilo-, mega- e giga- del sistema internazionale, cioè sono moltiplicatori di 10^3 , 10^6 e 10^9);

-n | **--numeric**
 (utilizzato con il comando **-L**) indica di visualizzare valori numerici relativamente a indirizzi e porte invece di usare i nomi degli host e delle applicazioni;

--line-numbers
 (utilizzato con il comando **-L**) indica di visualizzare i numeri relativi alle regole;

{-c | --set-counters} packet-counter byte-counter
 (utilizzato con i comandi **-I**, **-A** e **-R**) imposta i contatori al valore specificato da *packet-counter* e *byte-counter*;

--modprobe
 indica il modulo da utilizzare per aggiungere al kernel;

match specifica la condizione da verificare. Le condizioni possono essere dalle più generali a quelle specifiche per i vari protocolli TCP, UDP, ICMP, a quelle particolari relative allo stato, al proprietario ed ai limiti. *Netfilter* effettuerà i controlli sulle parti del pacchetto coinvolte dalle condizioni specificate.

{-p | --protocol} [!] protocol
 è verificata dai pacchetti relativi al protocollo *protocol* che è un elenco di nomi di protocolli (o i relativi valori numerici) contenuti nel file */etc/protocols* separati dal carattere *,*. Può contenere anche il valore *all* che si riferisce a qualunque protocollo (v. tab. 25.5). Il simbolo *!* indica a *Netfilter* di considerare la negazione della condizione specificata (es. **--protocol ! tcp** indica qualunque protocollo diverso dal TCP);

Protocollo	Keyword	Valore	Descrizione
IP	ip	0	internet protocol, pseudo protocol number
ICMP	icmp	1	internet control message protocol
IGMP	igmp	2	Internet Group Management
GGP	ggp	3	gateway-gateway protocol
IP-ENCAP	ipencap	4	IP encapsulated in IP (officially "IP")
ST	st	5	ST datagram mode
TCP	tcp	6	transmission control protocol
EGP	egp	8	exterior gateway protocol
PUP	pup	12	PARC universal packet protocol
UDP	udp	17	user datagram protocol
HMP	hmp	20	host monitoring protocol
XNS-IDP	xns-idp	22	Xerox NS IDP
RDP	rdp	27	"reliable datagram" protocol
ISO-TP4	iso-tp4	29	ISO Transport Protocol class 4
XTP	xtp	36	Xpress Transfer Protocol
DDP	ddp	37	Datagram Delivery Protocol
IDPR-CMTP	idpr-cmtp	39	IDPR Control Message Transport
RSPF	rsfp	73	Radio Shortest Path First
VMTP	vmtp	81	Versatile Message Transport
OSPF	ospf	89	Open Shortest Path First IGP
IPIP	ipip	94	Yet Another IP encapsulation
ENCAP	encap	98	Yet Another IP encapsulation

Tabella 25.5: Protocolli utilizzabili da *netfilter*.

{-s | --source | --src} [!] address[/mask]
 è verificata dai pacchetti il cui indirizzo IP del mittente (sorgente) è specificato da *address[/mask]*. Può essere sia l'indirizzo di un'interfaccia specifica che quello di una rete (con la relativa *mask* che può essere specificata con un numero, che indica il numero di bit di cui si compone la parte rete dell'indirizzo, o in forma dotted decimal). Il simbolo *!* indica a *Netfilter* di considerare la negazione della condizione specificata;

{-d | --destination | --dst} [!] address[/mask]
 è verificata dai pacchetti il cui indirizzo IP del destinatario è specificato da *address[/mask]*. Può essere sia l'indirizzo di un'interfaccia specifica che quello di una rete (con la relativa *mask* che può essere specificata con un numero, che indica il numero di bit di cui si compone la parte rete dell'indirizzo, o in forma dotted decimal). Il simbolo

‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

{-i |--in-interface} [!] *name*
 è verificata dai pacchetti che arrivano a *Netfilter* attraverso l’interfaccia di rete specificata da *name* (questo *match* è valido soltanto per le chain PREROUTING, INPUT e FORWARD). Se *name* termina con il simbolo ‘+’, è verificata dai pacchetti che arrivano attraverso tutte le interfacce di rete il cui nome inizia con la parte di *name* che precede il simbolo ‘+’. Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

{-o |--out-interface} [!] *name*
 è verificata dai pacchetti che devono essere inviati da *Netfilter* attraverso l’interfaccia di rete specificata da *name* (questo *match* è valido soltanto per le chain FORWARD, OUTPUT e POSTROUTING). Se *name* termina con il simbolo ‘+’, è verificata dai pacchetti che devono uscire attraverso tutte le interfacce di rete il cui nome inizia con la parte di *name* che precede il simbolo ‘+’. Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

[!] {-f |--fragment}
 è verificata dai pacchetti che sono frammenti di pacchetti IP successivi al primo (un frammento o un pacchetto ICMP non hanno la specificazione dell’indirizzo IP del mittente e del destinatario). Il simbolo ‘!’ indica a *Netfilter* che la condizione è applicabile soltanto al primo frammento dei pacchetti frammentati o a pacchetti non frammentati;

{--source-port |--sport} [!] *port*
 (applicabile soltanto ai protocolli di trasporto come TCP e UDP) è verificata dai pacchetti la cui porta del mittente (sorgente) è specificata da *port*. Questo può specificare anche un intervallo di porte con la sintassi *from_port:to_port* in cui *from_port* indica il numero della porta che rappresenta l’estremo inferiore dell’intervallo e *to_port* indica il numero di quella che rappresenta l’estremo superiore (nel caso in cui vengano omessi, al posto di *from_port* viene considerato il valore 0 ed al posto di *to_port* viene considerato il valore 65535). Nel caso in cui *from_port* sia maggiore di *to_port*, i due valori vengono automaticamente scambiati. Le porte possono essere specificate sia con un numero che con una stringa, contenuta nel file */etc/services*, che rappresenta l’applicazione che risponde su tale porta. Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

{--destination-port |--dport} [!] *port*
 (applicabile soltanto ai protocolli di trasporto come TCP e UDP) è verificata dai pacchetti a cui porta del destinatario è specificata da *port*. Questo può specificare anche un intervallo di porte con la sintassi *from_port:to_port* in cui *from_port* indica il numero della porta che rappresenta l’estremo inferiore dell’intervallo e *to_port* indica il numero di quella che rappresenta l’estremo superiore (nel caso in cui vengano omessi, al posto di *from_port* viene considerato il valore 0 ed al posto di *to_port* viene considerato il valore 65535). Nel caso in cui *from_port* sia maggiore di *to_port*, i due valori vengono automaticamente scambiati. Le porte possono essere specificate sia con un numero che con una stringa, contenuta nel file */etc/services*, che rappresenta l’applicazione che risponde su tale porta. Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

--tcp-flags [!] *mask comp*
 (applicabile soltanto al protocollo TCP) è verificata dai pacchetti TCP i cui flag specificati da *mask*, sono impostati secondo quanto specificato da *comp*. I parametri *mask* e *comp* sono elenchi di stringhe separate dal carattere ‘,’ (SYN, ACK, FIN, RST, URG, PSH, ALL, NONE). Ad esempio **--tcp-flags SYN,ACK,FIN SYN** si riferisce ai pacchetti TCP che hanno il flag SYN impostato ed i flag ACK e FIN non impostato.

Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

- [!] **--syn**
(applicabile soltanto al protocollo TCP) è verificata dai pacchetti con il flag SYN impostato ed i flag ACK e FIN non impostato. Tali particolari pacchetti sono utilizzati dal protocollo TCP per instaurare una connessione (comunicazione). È equivalente a **--tcp-flags SYN,RST,ACK SYN**. Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;
- tcp-option [!] number**
(applicabile soltanto al protocollo TCP) è verificata dai pacchetti TCP il cui campo Options ha il valore specificato da *number*. Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;
- mss value**
(applicabile soltanto al protocollo TCP) è verificata dai pacchetti TCP (SYN o SYN/ACK) il cui campo MSS contiene il valore specificato da *value* (*value* può essere anche un intervallo di valori della forma *[min-value]:[max-value]*);
- icmp-type [!] type**
(applicabile soltanto al protocollo ICMP) è verificata dai pacchetti ICMP il cui tipo è specificato da *type*. Questo può essere un valore numerico o una stringa che identifica il tipo di pacchetto ICMP secondo quanto riportato dal comando **iptables -p icmp -h** (v. tab. ??). Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

Esiste la possibilità di specificare delle indicando un match esplicito con la sintassi

{-m | **--match**} *explicit-match*

dove *explicit-match* indica il tipo di match esplicito.

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **mac**

- mac-source [!] address**
(applicabile soltanto a pacchetti che si basano su Ethernet relativamente alla chain PREROUTING, INPUT o FORWARD) è verificata dai pacchetti il cui indirizzo MAC del mittente è specificato da *address* nella forma *XX:XX:XX:XX:XX:XX*. Il simbolo ‘!’ indica a *Netfilter* di considerare la negazione della condizione specificata;

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **limit**

- limit rate**
indica il massimo numero di pacchetti che verificano la condizione specificata all'interno di un determinato periodo temporale. Il valore è specificato da *rate* che è espresso nella forma *value[/unit]* dove *value* è un valore numerico e *unit* è una unità temporale (**second**, **minute**, **hour**, **day**) che se non specificata è **minute**. Il valore di default è **3/hour**;
- limit-burst number**
indica il massimo numero di pacchetti consecutivi (*number*) che soddisfano la condizione specificata nell'unità di tempo impostata con **--limit rate**. Il valore di default di *number* è 5;

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **mark**

- mark value[/mask]**
è verificata dai pacchetti il cui campo mark contiene il valore *value* (tutti i pacchetti che attraversano *Netfilter* hanno associato un campo *mark* che contiene un valore numerico intero senza segno - da 0 a 4294967296). Se *mask* è specificato, viene effettuato un AND tra *value* e *mask* prima di effettuare il confronto;

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **multiport**

{--source-ports |--sports} port[,port[,port...]]
 (applicabile soltanto ai protocolli di trasporto come TCP e UDP) indica l'elenco di porte del mittente (sorgente) specificato da *port[,port[,port...]]*.
{--destination-ports |--dports} port[,port[,port...]]
 (applicabile soltanto ai protocolli di trasporto come TCP e UDP) indica l'elenco di porte del destinatario specificato da *port[,port[,port...]]*.
--ports port[,port[,port...]]
 (applicabile soltanto ai protocolli di trasporto come TCP e UDP) è verificata se le porte del mittente e del destinatario sono le stesse e se cadono nell'elenco specificato da *port[,port[,port...]]*.

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **owner**

--uid-owner userid
 è verificata dai pacchetti che sono stati creati da un processo con l'effective user id specificato da *userid*;
--gid-owner groupid
 è verificata dai pacchetti che sono stati creati da un processo con l'effective group id specificato da *groupid*;
--pid-owner processid
 è verificata dai pacchetti che sono stati creati da un processo con il process id specificato da *processid*;
--sid-owner sessionid
 è verificata dai pacchetti che sono stati creati da un processo con il session id specificato da *sessionid*;
--cmd-owner name
 è verificata dai pacchetti che sono stati creati da un processo il cui comando è specificato da *name*;

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **state**

[!] --state state
 è verificata dai pacchetti che sono negli stati specificati da *state* (un elenco di stringhe separate dal carattere ',' relative agli stati del *Conntrack*, v. tab. 25.3). Il simbolo '!' indica a *Netfilter* di considerare la negazione della condizione specificata;

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **tos**

--tos tos
 è verificata dai pacchetti il cui campo TOS (Type Of Service) dell'header dell'IP datagram contiene il valore specificato da *tos*, che può essere anche una stringa secondo quanto riportato da *iptables -m tos -h* (v. tab. 25.6);

Valore	Keyword
0	Normal-Service
2	Minimize-Cost
4	Maximize-Reliability
8	Maximize-Throughput
16	Minimize-Delay

Tabella 25.6: Valori per filtro sul campo TOS.

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **ah**

--ahspi [!] spi
 è verificata dai pacchetti IPsec il cui header AH contiene il valore specificato da *spi* (che può specificare un unico valore o un intervallo di valori secondo la sintassi *min_spi:max_spi*). Il simbolo '!' indica a *Netfilter* di considerare la negazione della condizione specificata;

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a **esp**

--espspi [!] spi
 è verificata dai pacchetti IPsec il cui header AH contiene il valore specificato da *spi* (che può specificare un unico valore o un intervallo di valori secondo la sintassi *min_spi:max_spi*). Il simbolo '!' indica a *Netfilter* di considerare la negazione della condizione;

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a *length*

--length *length*

è verificata dai pacchetti che hanno una lunghezza uguale al valore specificato da *length* (che può specificare un unico valore o un intervallo di valori secondo la sintassi *min_length:max_length*)

Le seguenti opzioni sono specificabili soltanto con *explicit-match* uguale a *ttl*

--ttl *ttl*

è verificata dai pacchetti il cui campo TTL (Time To Live) dell'header IP contiene il valore specificato da *ttl*;

target indica l'azione da intraprendere nel caso in cui il pacchetto verifichi la condizione indicata da *match*. Può essere specificato per mezzo della sintassi seguente

{-j | --jump} *action*

dove *action* specifica l'azione da intraprendere, come descritto di seguito, o il nome di una chain (appartenente alla tabella corrente) da analizzare. Nel caso in cui venga specificato il nome di una chain (*chain_B*), *Netfilter* prenderà in considerazione tutte le regole presenti in tale chain per controllare il pacchetto ed una volta scorso l'elenco di tali regole, viene ripreso il controllo dalla regola (nella chain corrente) successiva a quella che specificava la *chain_B* come target (v. fig. 25.3).

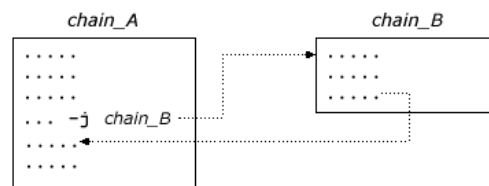


Figura 25.3: Esempio di chain espressa come *action*.

Le *action* possibili sono

ACCEPT indica di accettare il pacchetto. Per tale pacchetto non saranno controllate ulteriori condizioni specificate in altre chain della tabella considerata (le condizioni contenute in altre chain appartenenti ad altre tabelle vengono comunque applicate al pacchetto).

DNAT (Destination Network Address Translation) indica di modificare l'indirizzo IP del destinatario del pacchetto. Questa *action* è disponibile soltanto per le chain PREROUTING e OUTPUT della tabella nat (e per le chain chiamate da queste) e le chain che contengono tale *action* non possono essere chiamate da altre chain che non siano queste.

Con **-j DNAT** possono essere utilizzate le seguenti ulteriori opzioni

--to-destination *ipaddr[:port]*

indica uno specifico indirizzo IP con cui sostituire quello del destinatario (o un intervallo di indirizzi IP specificato con la sintassi *first_ipaddr-last_ipaddr*). Nel caso di un intervallo di indirizzi IP, *Netfilter* provvede ad assegnare al pacchetto uno di essi scelto a caso, il che permette la gestione del load balancing di più macchine. Per i protocolli a livello di trasporto, come TCP o UDP, può essere specificata anche la porta (o un'intervallo di porte secondo la sintassi *first_port-last_port*) con la quale sostituire il valore della porta del destinatario;

DROP indica di scartare il pacchetto senza inviare nessun messaggio di ritorno al mittente (questo può avere l'effetto di lasciare "appeso" il socket sul sistema che ha inviato il pacchetto).

- LOG** indica di scrivere nel system log (v. cap. 5) un messaggio che riporta alcune informazioni relative al pacchetto. L'esame del pacchetto continua comunque con le regole successive contenute nelle chain che questo deve attraversare.
Con `-j LOG` possono essere utilizzate le seguenti ulteriori opzioni
- `--log-level level`
indica il livello di verbosità del log, specificato da *level* (v. man page `syslog.conf(5)`);
 - `--log-prefix prefix`
indica di inserire nel messaggio di log una stringa (*prefix*) da premettere alle informazioni relative al pacchetto;
 - `--log-tcp-sequence prefix`
indica di inserire nel messaggio di log i numeri di sequenza del TCP;
 - `--log-tcp-options`
indica di inserire nel messaggio di log le options dall'header TCP;
 - `--log-ip-options`
indica di inserire nel messaggio di log le options dall'header IP;
- MARK** indica di impostare il campo *mark* che *Netfilter* associa ad ogni pacchetto.
Questa *action* è disponibile soltanto per la tabella mangle.
Con `-j MARK` possono essere utilizzate le seguenti ulteriori opzioni
- `--set-mark mark`
indica il valore (*mark*) con cui impostare il campo *mark*;
- MASQUERADE**
indica di sostituire l'indirizzo IP del mittente del pacchetto con quello dell'interfaccia attraverso la quale vengono inviati (come *SNAT*) ma la regola non viene considerata quando l'interfaccia non esiste (la connessione non è attiva - connessioni dialup). Questa *action* è disponibile soltanto per la chain POSTROUTING della tabella nat.
Con `-j MASQUERADE` possono essere utilizzate le seguenti ulteriori opzioni
- `--to-ports port`
indica il valore della porta (o dell'intervallo di porte con la sintassi *first_port-last_port*) da utilizzare per sostituire la porta del mittente, secondo quanto specificato da *port* (soltanto per i protocolli a livello di trasporto come TCP o UDP);
- MIRROR** indica di invertire l'indirizzo IP del mittente con quello del destinatario e ritrasmettere il pacchetto.
Questa *action* è disponibile soltanto per le chain PREROUTING, INPUT e FORWARD (e per le chain chiamate da queste).
- QUEUE** indica di accodare i pacchetti verso applicazioni che girano in user-space estranee a *Netfilter* come quelle che si occupano di gestione centralizzata degli accessi o di filtri tipo proxy. Lo stato della coda è gestito tramite il file `/proc/net/ip_queue`, mentre la sua lunghezza massima è memorizzata nel file `/proc/sys/net/ipv4/ip_queue_maxlen` (il suo valore di default è 1024). La documentazione relativa si trova in "Netfilter Hacking HOW-TO".
- REDIRECT**
indica di modificare l'indirizzo IP del destinatario in modo da risultare il destinatario del pacchetto (l'indirizzo IP del destinatario dei pacchetti entranti viene sostituito con quello dell'interfaccia di rete attraverso la quale sono stati ricevuti, mentre a quelli generati localmente, viene sostituito con 127.0.0.1).
Questa *action* è disponibile soltanto per le chain PREROUTING e OUTPUT (e per le chain chiamate da queste).
Con `-j REDIRECT` possono essere utilizzate le seguenti ulteriori opzioni

--to-ports *port*

indica la porta (o l'intervallo di porte con la sintassi *first.port-last.port*) con cui impostare la porta del destinatario del pacchetto (soltanto per i protocolli a livello di trasporto come TCP o UDP);

REJECT indica di scartare il pacchetto inviando al mittente un messaggio di errore.

Questa *action* è disponibile soltanto per le chain INPUT, FORWARD e OUTPUT (e per le chain chiamate da queste) e le chain contenenti tali *action* non possono essere chiamate da altre chain che non siano queste.

Con **-j REJECT** possono essere utilizzate le seguenti ulteriori opzioni

--reject-with *type*

indica il tipo (*type*) di messaggio di errore da inviare al mittente. Questo può essere uno dei messaggi ICMP riportati in tab. 25.7;

Type	Messaggio ICMP
icmp-net-unreachable	Network unreachable
icmp-host-unreachable	Host unreachable
icmp-port-unreachable	Port unreachable (default)
icmp-proto-unreachable	Protocol unreachable
icmp-net-prohibited	Destination network administratively prohibited
icmp-host-prohibited	Destination host administratively prohibited

Tabella 25.7: I possibili messaggi di errore da ritornare al mittente.

--tcp-reset

indica di inviare al mittente il messaggio di errore costituito da un pacchetto TCP RST (è valido soltanto per pacchetti del protocollo TCP);

RETURN indica di non considerare la chain corrente per il controllo delle condizioni sul pacchetto. Se la regola che contiene tale *action* è una sub-chain (cioè una chain specificata come *action* di una regola contenuta in un'altra chain) *Netfilter* ritornerà a prendere in considerazione la chain principale (da cui quella corrente è stata chiamata). Se la regola che contiene tale *action* è una chain principale, le regole seguenti non vengono considerate e *Netfilter* applicherà la regola relativa alla policy di default per quella chain.

SNAT (Source Network Address Translation) indica di modificare l'indirizzo IP del mittente del pacchetto. Questa *action* è disponibile soltanto per la chain POSTROUTING della tabella nat (e per le chain chiamate da queste) e le chain che contengono tale *action* non possono essere chiamate da altre chain che non siano queste.

Con **-j SNAT** possono essere utilizzate le seguenti ulteriori opzioni

--to-source *ipaddr[:port]*

indica uno specifico indirizzo IP con cui sostituire quello del mittente (o un intervallo di indirizzi IP specificato con la sintassi *first.ipaddr-last.ipaddr*). Per i protocolli a livello di trasporto, come TCP o UDP, può essere specificata anche la porta (o un'intervallo di porte secondo la sintassi *first.port-last.port*) con la quale sostituire il valore della porta del mittente;

TCPMSS indica di modificare il campo MSS dei pacchetti TCP SYN che imposta la massima dimensione dei pacchetti che transitano sulla rete (usualmente è impostato a MTU - 40). Questa *action* è utile per limitare le dimensioni dei pacchetti inviati su *Internet* attraverso dei server che scartano i pacchetti ICMP Fragmentation Needed.

Questa *action* è disponibile soltanto per pacchetti TCP.

Con **-j TCPMSS** può essere utilizzata una delle seguenti ulteriori opzioni

	<p>--set-mss <i>value</i> indica di impostare il campo MSS del pacchetto TCP con il valore specificato da <i>value</i>;</p> <p>--clamp-mss-to-pmtu indica di impostare il campo MSS del pacchetto TCP con il valore MTU - 40;</p>
TOS	<p>indica di modificare il campo TOS (Type Of Service) del pacchetto IP per specificare la politica di routing relativa al pacchetto stesso, anche se tale campo non è sempre considerato (soprattutto in maniera corretta) dai router.</p> <p>Questa <i>action</i> è disponibile soltanto per la tabella mangle.</p> <p>Con -j TOS possono essere utilizzate le seguenti ulteriori opzioni</p> <p>--set-tos <i>tos</i> indica di impostare il campo TOS del pacchetto con il valore indicato da <i>tos</i>, che può essere anche una stringa secondo quanto riportato da iptables -j TOS -h (v. tab. 25.6);</p>
TTL	<p>indica di impostare il valore del campo TTL (Time To Live) del pacchetto IP. Questo può essere utile per impostare tutti i pacchetti che vengono inviati su <i>Internet</i> da un NAT, con lo stesso valore (in modo tale che l'ISP non possa accorgersi facilmente che sono pacchetti relativi a più macchine: potrebbe non gradire che più macchine utilizzano la stessa connessione a <i>Internet</i>);</p> <p>Questa <i>action</i> è disponibile soltanto per la tabella mangle.</p> <p>Con -j TTL possono essere utilizzate le seguenti ulteriori opzioni</p> <p>--ttl-set <i>ttl</i> indica di impostare il campo TTL del pacchetto con il valore indicato da <i>ttl</i>;</p> <p>--ttl-dec <i>ttl</i> indica di decrementare il campo TTL del pacchetto con il valore indicato da <i>ttl</i>;</p> <p>--ttl-inc <i>ttl</i> indica di incrementare il campo TTL del pacchetto con il valore indicato da <i>ttl</i>;</p>
ULOG	<p>indica di registrare un messaggio di log con applicazioni che girano in user-space, inviando il messaggio stesso in multicast tramite un netlink socket. Le applicazioni che devono gestire il log possono essere agganciate ai netlink group relativi all'indirizzo multicast utilizzato, per ricevere il pacchetto.</p> <p>Con -j ULOG possono essere utilizzate le seguenti ulteriori opzioni</p> <p>--ulog-nlgroup <i>nlgroup</i> indica il netlink group al quale il pacchetto è inviato, secondo quanto specificato da <i>nlgroup</i>, che può assumere un valore compreso tra 1 e 32 (il valore di default è 1);</p> <p>--ulog-prefix <i>prefix</i> indica la stringa da anteporre al messaggio da inserire nel log (massimo 32 caratteri);</p> <p>--ulog-cprange <i>size</i> indica il numero di byte da copiare nello user-space, secondo quanto specificato da <i>size</i> (il valore 0 indica di copiare l'intero pacchetto - valore di default);</p> <p>--ulog-qthreshold <i>size</i> indica il numero di pacchetti da accodare in kernel-space prima di inviarli come messaggio multipart del netlink, secondo quanto specificato da <i>size</i> (per default è 1);</p>

Nel caso di firewall stateless è opportuno aprire in ingresso le porte al di sopra della 1024 in modo da poter ricevere i pacchetti di risposta ad eventuali pacchetti di richiesta che escono dal firewall. Questo non è necessario nel caso di firewall stateful in quanto si può permettere l'ingresso di pacchetti con porta di destinazione maggiore di 1024 soltanto per i pacchetti di risposta, senza essere obbligati ad aprire in ingresso tutte le porte sopra la 1024 a tutti i pacchetti.

???

La configurazione del firewall impostata per mezzo di **iptables** non viene memorizzata sul filesystem e quindi viene perduta al successivo riavvio del sistema. È comunque possibile creare uno script in cui inserire i comandi necessari alla configurazione desiderata del firewall in modo da lanciarlo in esecuzione durante la procedura di avvio del sistema.

Si può optare anche per il salvataggio delle impostazioni del firewall per mezzo degli appositi comandi **iptables-save** (man page **iptables-save(8)**) e **iptables-restore** (man page **iptables-restore(8)**) che servono, rispettivamente, per salvare le impostazioni correnti del firewall e per ripristinare il firewall con le informazioni precedentemente salvate. Le impostazioni vengono gestite in un formato leggermente diverso da quello utilizzato dalla linea di comando con **iptables**.

L'utilizzo di **iptables-save** e **iptables-restore** velocizza considerabilmente il salvataggio ed il caricamento di un consistente insieme di regole poiché il passaggio tra user-space e kernel-space (dove vanno a finire le regole) viene effettuato una volta soltanto e non regola per regola come avverrebbe utilizzando uno script con varie chiamate a **iptables**.

```
Comando: iptables-save
Path: /sbin/iptables-save

SINTASSI
# iptables-save [-c] [-t table]
```

DESCRIZIONE

```
-c indica a iptables-save di salvare anche i contatori;
-t table indica la tabella della quale effettuare il salvataggio. Se non è specificato,
vengono salvate tutte le tabelle;
```

Per default **iptables-save** visualizza l'output sullo standard output (terminale). Per salvare le impostazioni su un file si può utilizzare la redirectione, come nell'esempio seguente

```
# iptables-save -c > iptables.conf
```

che salva le impostazioni relative a tutte le tabelle ed i contatori nel file **iptables.conf**.

Di seguito è riportato un esempio dell'output di **iptables-save**.

```
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*filter
:INPUT DROP [1:229]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth1 -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -m state --state NEW,RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*mangle
:PREROUTING ACCEPT [658:32445]
:INPUT ACCEPT [658:32445]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [891:68234]
:POSTROUTING ACCEPT [891:68234]
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
# Generated by iptables-save v1.2.6a on Wed Apr 24 10:19:55 2002
*nat
```

```
:PREROUTING ACCEPT [1:229]
:POSTROUTING ACCEPT [3:450]
:OUTPUT ACCEPT [3:450]
-A POSTROUTING -o eth0 -j SNAT --to-source 195.233.192.1
COMMIT
# Completed on Wed Apr 24 10:19:55 2002
```

Le righe relative alle tabelle hanno la seguente sintassi

**table-name*

Le righe che specificano le regole sono del tipo

:chain-name chain-default-policy [packet-counter:byte-counter]

Le direttive (regole) hanno la sintassi seguente

[[packet-counter:byte-counter]] rule

dove *rule* è una regola come imparita con il comando `iptables`, tranne per il fatto che non riporta l'indicazione del comando `iptables` né quella della tabella di riferimento. La parte relativa ai contatori è presente soltanto se è stata specificata l'opzione `-c`.

Ogni dichiarazione di tabella viene conclusa dalla parola chiave `COMMIT`.

Comando: `iptables-restore`
Path: `/sbin/iptables-restore`

SINTASSI

```
# iptables-restore [-c | --counters] [-n | --noflush]
```

DESCRIZIONE

`-c | --counters` indica a `iptables-restore` di ripristinare anche i contatori;

`-n | --noflush` indica a `iptables-restore` di non eliminare le regole già presenti nelle tabelle (il comportamento di default è quello di eliminare prima tutte le regole presenti nelle tabelle e quindi inserirci quelle specificate).

Poiché `iptables-restore` imposta il firewall con le direttive impostate tramite lo standard input (tastiera), si può utilizzare la redirectione per ripristinare le impostazioni salvate in un file, come nell'esempio seguente

```
# cat iptables.conf | iptables-restore -c
```

oppure

```
# iptables-restore -c < iptables.conf
```

che imposta il firewall ripristinando i contatori e le regole contenute nel file `iptables.conf` precedentemente creato con `iptables-save`.

La procedura di avvio del sistema, provvede ad impostare automaticamente il firewall con le regole contenute nel file `/etc/sysconfig/iptables`, tramite il comando `iptables-restore`.

Si noti che `iptables-restore` non gestisce il caricamento di regole con valori parametrizzati (es. l'indirizzo IP dinamico dell'interfaccia di rete) e per questo tipo di regole si deve ricorrere comunque ad uno script.

Esempi

Nell'ambito della sicurezza è una buona regola bloccare l'accesso a tutti i pacchetti (policy di default DROP) e aggiungere delle regole che permettano l'accesso soltanto per determinati pacchetti.

???

È consigliabile abilitare il filtraggio dei pacchetti tramite la routing policy in modo tale che il router stesso scarti i pacchetti che arrivano da un'interfaccia inaspettata. Ad esempio, se un pacchetto con indirizzo IP relativo alla rete privata arriva all'interfaccia esterna, questo viene scartato. Questo può essere fatto per l'interfaccia ppp0 con il seguente comando

```
# echo "1" > /proc/sys/net/ipv4/conf/ppp0/rp_filter
```

Uno script di esempio che inserisce messaggi nel log di sistema man mano che i pacchetti ICMP attraversano *Netfilter* è il seguente

```
#!/bin/bash
#
# rc.test-iptables - test script for iptables chains and tables.
#
# Copyright (C) 2001 Oskar Andreasson <blueflux@koffein.net>
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program or from the site that you downloaded it
# from; if not, write to the Free Software Foundation, Inc., 59 Temple
# Place, Suite 330, Boston, MA 02111-1307 USA
#

#
# Filter table, all chains
#
iptables -t filter -A INPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A INPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter INPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter OUTPUT:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-request \
-j LOG --log-prefix="filter FORWARD:"
iptables -t filter -A FORWARD -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="filter FORWARD:"

#
# NAT table, all chains except OUTPUT which don't work.
#
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat PREROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-request \
```

```

-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A POSTROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat POSTROUTING:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="nat OUTPUT:"
iptables -t nat -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="nat OUTPUT:"

#
# Mangle table, all chains
#
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -A PREROUTING -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle PREROUTING:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I FORWARD 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle FORWARD:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -I INPUT 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle INPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -A OUTPUT -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle OUTPUT:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-request \
-j LOG --log-prefix="mangle POSTROUTING:"
iptables -t mangle -I POSTROUTING 1 -p icmp --icmp-type echo-reply \
-j LOG --log-prefix="mangle POSTROUTING:"

```

Gli utenti di un sistema GNU/Linux con l'accesso ad *Internet* attraverso una connessione dialup (modem), spesso desiderano che nessuno possa accedere dall'esterno al loro sistema. Di seguito c'è uno script che funziona in questo modo

```

#
# Inserisci i moduli del connection-tracking
# (non necessari se inclusi nel kernel)
#
insmod ip_conntrack
insmod ip_conntrack_ftp

#
# Crea una catena che blocchi le nuove connessioni,
# eccetto quelle provenienti dall'interno.
#
iptables -N blockchain
iptables -A blockchain -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A blockchain -m state --state NEW -i ! ppp0 -j ACCEPT
iptables -A blockchain -j DROP

#
# Dalle catene INPUT e FORWARD salta a questa catena
#
iptables -A INPUT -j blockchain
iptables -A FORWARD -j blockchain

```

Si può sentire la necessità di effettuare un NAT per le macchine presenti su una rete privata e poterle fare accedere ad *Internet* tramite un'unica connessione dialup effettuata dalla macchina che funge da gateway/firewall, tramite uno script analogo a quello seguente (che deve essere eseguito sulla macchina che funge da firewall)


```
# Abilita l'IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# Carica il modulo NAT.
# (non necessario se incluso nel kernel)
modprobe iptable_nat

# NAT su ppp0 (la connessione dialup)
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

È evidente che le macchine sulla rete protetta dal firewall che effettua anche il NAT, devono avere come gateway di default una macchina della rete o il firewall stesso.⁴

Una cosa utile è quella di tracciare i tentativi di recapito di pacchetti “strani”, che può essere fatta, ad esempio, con il seguente script

```
iptables -N no-conns-from-ppp0
iptables -A no-conns-from-ppp0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A no-conns-from-ppp0 -m state --state NEW -i ! ppp0 -j ACCEPT
iptables -A no-conns-from-ppp0 -i ppp0 -m limit -j LOG --log-prefix "Bad packet from ppp0:"
iptables -A no-conns-from-ppp0 -i ! ppp0 -m limit -j LOG --log-prefix "Bad packet not from ppp0:"
iptables -A no-conns-from-ppp0 -j DROP

iptables -A INPUT -j no-conns-from-ppp0
iptables -A FORWARD -j no-conns-from-ppp0
```

???

25.4 Proxy server

???

25.5 IDS - Intrusion Detection System

???

25.6 Riferimenti

- “Iptables Tutorial”
<http://iptables-tutorial.frozentux.net/iptables-tutorial.html>
- “Netfilter Hacking HOW-TO”
<http://www.netfilter.org/unreliable-guides/netfilter-hacking-HOWTO/index.html>

⁴v. cap. 17 e sez. 20.4.

Parte IV

Appendici

Appendice A

Il software e le licenze

“Le idee migliori sono proprietà di tutti.”
– Seneca

L’informatica ha aperto nuovi orizzonti, nuove prospettive per lo sviluppo sia tecnologico che scientifico. Se la ricerca da un lato produce processori con prestazioni sempre più elevate e hardware sempre più avanzato, le operazioni compiute dalle macchine sono governate dal software.

Il software quindi riveste un ruolo fondamentale nell’informatica: i sistemi operativi, i driver, le varie applicazioni, sono tutti dei software che istruiscono la macchina di come deve comportarsi in corrispondenza di determinate situazioni.

Il software è un’opera dell’ingegno umano e pertanto è vincolato alle leggi sul diritto di autore, analogamente a quanto avviene per i libri, o altro. Nelle sezioni seguenti viene fatta una panoramica sui vari vincoli che possono essere imposti sul software stesso per mezzo delle *licenze*.

A.1 Le licenze

Quando si parla di software in realtà si sta parlando di qualcosa di astratto, un prodotto della mente umana concretizzato nella sequenza di istruzioni che lo costituisce. In genere, per qualunque opera, si identifica un proprietario, ovvero l’entità che ne detiene i diritti di autore o copyright (indicato dal simbolo ©), che può essere colui che ha effettivamente realizzato l’opera o altri che magari ha acquisito tali diritti tramite la sottoscrizione di un apposito contratto. Le leggi sul diritto di autore stabiliscono quali sono i diritti del detentore del copyright (il proprietario) e di chi la utilizza (utente).

Per il software è stato inoltre introdotto un contratto, non sottoscritto direttamente dall’utente, che generalmente limita ulteriormente i diritti dell’utente stesso. A tale contratto si fa riferimento in genere con il termine **licenza**. Pertanto, prima di utilizzare un software è opportuno leggere la relativa licenza per conoscere i termini di utilizzo del software stesso. Il software acquisito gratuitamente o a pagamento è utilizzabile dall’utente soltanto secondo quanto previsto dalle indicazioni contenute nella relativa licenza.

licenza

Poiché il software una volta in linguaggio macchina¹ è direttamente comprensibile all’elaboratore, ma non è intellegibile dagli esseri umani, in vari paesi le leggi che tutelano il diritto di autore relativo al software consentono la distribuzione del software soltanto in linguaggio macchina, ovvero il codice eseguibile e considerano illecito qualunque tentativo di ricondursi al codice sorgente dall’analisi del codice eseguibile.

Molti paesi, come gli Stati Uniti, prevedono anche la possibilità di brevettare gli algoritmi, rendendo così illecito l’uso di uno stesso algoritmo da parte di altri programmatori, a meno che non sia esplicitamente autorizzato dal detentore del brevetto. Immaginiamoci cosa succedrebbe se qualcuno decidesse di brevettare un algoritmo di

¹v. sez. ??

ordinamento come il bubble sort, merge sort, quick sort o l'ordinamento "accademico". Nessun programmatore potrebbe più ordinare gli elementi di un vettore.

A.2 Il software libero

software libero

Con il termine *free software*² o **software libero** viene indicata una categoria di software, che chiunque può utilizzare, copiare, modificare e distribuire come meglio crede, sia a pagamento che gratuitamente. Si parla comunque di software libero soltanto se lo stesso è distribuito corredato dei sorgenti, in modo che chi lo desidera possa agevolmente modificarlo.

open source

La distribuzione di un software corredato di sorgenti non è di per sé sufficiente per affermare che il software in questione appartenga al software libero. È infatti possibile che i sorgenti siano forniti ma che la licenza non permetta la modifica né la redistribuzione. Si parla in tal caso di software **open source**³ indicando in questo modo che del software in questione sono disponibili i sorgenti.

Il software libero garantisce i seguenti livelli di libertà

0. la libertà di eseguire il programma per qualunque scopo;
1. la libertà di studiare il funzionamento del programma per poterlo adattare alle proprie esigenze;
2. la libertà di distribuire copie del programma a chiunque;
3. la libertà di modificare il programma e di distribuirne delle copie modificate a chiunque;

copyleft

Esistono anche altre categorie di software come il software protetto da **copyleft** che è un software libero coperto da una licenza tale che oltre a garantire il diritto d'autore, allo stesso tempo garantisce anche il fatto che il software stesso e le sue eventuali derivazioni rimangano libere: il copyleft non rende possibile aggiungere delle restrizioni alla licenza che deve comunque seguire il software anche nelle sue derivazioni. Un esempio di licenza di questo tipo è la GNU GPL (v. sez. A.4).

software di dominio pubblico

Un caso particolare è il **software di dominio pubblico**. Per esso infatti non è definito un detentore dei diritti di autore e quindi si tratta sì di software libero, poiché non è di nessuno e quindi non esiste nemmeno una licenza che vincola l'utilizzo dello stesso, ma chiunque può farne quello che vuole, compreso il fatto di appropriarsi dei diritti.

software proprietario

Il **software proprietario** è tutto il software che viene distribuito generalmente dietro pagamento e senza il codice sorgente, con una licenza tale impedisce la copia, modifica e distribuzione dello stesso. A questa categoria appartiene la maggior parte del software ad oggi in commercio.

software freeware

Esiste poi il **software freeware** che viene inteso generalmente come software gratuito ma del quale non viene fornito il codice sorgente e pertanto sebbene non venga limitata di fatto la copia, modifica e la distribuibilità dello stesso, non si ha la libertà di poterlo modificare.

software shareware

Il **software shareware** è generalmente riferito a software proprietario (distribuito senza codice sorgente) che è possibile copiare e distribuire, e per il quale viene richiesto il pagamento di una certa quantità di denaro dopo un certo "periodo di prova".

La figura fig. A.1 riporta le relazioni fra le varie categorie di software elencate.

A.3 Garanzie del software

La maggior parte delle software house, ovvero quelle aziende che producono software, forniscono soltanto il codice eseguibile, ovvero la versione in linguaggio macchina del

²in inglese *free* può voler dire anche gratuito, per questo si è soliti aggiungere a *free software* la frase "Free in the sense of freedom" cioè "libero nel senso di libertà".

³v. <http://www.opensource.org>

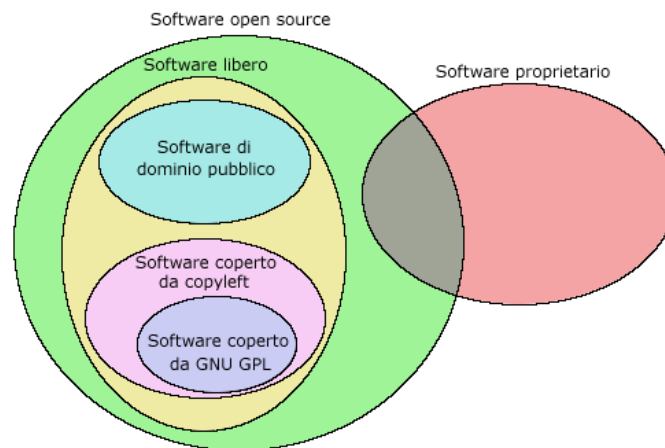


Figura A.1: Relazioni fra alcune categorie di software.

software, senza fornire i sorgenti che lo hanno prodotto. È vero che l'importante, per un utente finale, è avere il codice eseguibile per poter far eseguire il codice fruendo così del prodotto, ma in questo modo chi può verificarne la bontà dello stesso? Si supponga infatti che per qualche motivo nel software vi siano dei meccanismi che fanno sì che il software fornito non si comporti esattamente come crede l'utente ...

???

Di seguito è riportata la GNU General Public License della FSF, comunemente detta anche GPL, come esempio di licenza per il software libero.

A.4 The GNU General Public License

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they,

too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
- (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that

you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

A.5 Licenza Pubblica Generica GNU

Quella che segue è una traduzione italiana non ufficiale della Licenza Pubblica Generica GNU, curata dal gruppo Pluto, ILS e dal gruppo italiano di traduzione GNU. Non è pubblicata dalla Free Software Foundation e non ha valore legale nell'esprimere i termini di distribuzione del software che usa la licenza GNU General Public License (GPL). Solo la versione originale in inglese della licenza (v. sez. A.4) ha valore legale.

La traduzione è riportata con lo scopo di facilitare, alle persone di lingua italiana, la comprensione del significato della GNU General Public License.

Versione 2, Giugno 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Chiunque può copiare e distribuire copie letterali di questo documento di licenza, ma non ne è permessa la modifica.

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 Software 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

0. 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 essere 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 copiatura, 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.

1. 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.

2. 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 nella sua interezza in licenza gratuita 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 complesso. 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.

3. 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 del 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.

4. 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.
5. L’acquirente non è tenuto 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.
6. 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.
7. 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.

8. 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.

9. 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.

10. 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.

NESSUNA GARANZIA

11. 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.
12. 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

A.6 Riferimenti

???

Appendice B

Utilizzo pratico

“Se c’è soluzione, perché ti preoccupi? Se non c’è soluzione, perché ti preoccupi?”
– Aristotele

B.1 Dispositivi

B.1.1 Configurazione flash pen

Una *flash pen* è un dispositivo di memorizzazione che si connette al sistema attraverso la porta USB ed il sistema è in grado di “vedere” tale dispositivo come se fosse un’unità a disco di tipo SCSI. Quindi, prima di collegare la flash pen al sistema, si può visualizzare il contenuto del file `/proc/bus/usb/devices` con il comando

```
$ cat /proc/bus/usb/devices
T: Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=12 MxCh= 2
B: Alloc= 0/900 us ( 0%), #Int= 0, #Iso= 0
D: Ver= 1.10 Cls=09(hub ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0000 ProdID=0000 Rev= 0.00
S: Product=USB OHCI Root Hub
S: SerialNumber=c8049000
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr= 0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E: Ad=81(I) Atr=03(Int.) MxPS= 2 Iv1=255ms
```

per poi confrontarlo con lo stesso dopo averla collegata

```
$ cat /proc/bus/usb/devices
T: Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=12 MxCh= 2
B: Alloc= 0/900 us ( 0%), #Int= 0, #Iso= 0
D: Ver= 1.10 Cls=09(hub ) Sub=00 Prot=00 MxPS= 8 #Cfgs= 1
P: Vendor=0000 ProdID=0000 Rev= 0.00
S: Product=USB OHCI Root Hub
S: SerialNumber=c8049000
C:* #Ifs= 1 Cfg#= 1 Atr=40 MxPwr= 0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub
E: Ad=81(I) Atr=03(Int.) MxPS= 2 Iv1=255ms
T: Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 2 Spd=12 MxCh= 0
D: Ver= 1.10 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs= 1
P: Vendor=0ea0 ProdID=6803 Rev= 1.00
S: Manufacturer=USB
S: Product=Solid state disk
S: SerialNumber=230760A43E33747B
C:* #Ifs= 1 Cfg#= 1 Atr=80 MxPwr=100mA
I: If#= 0 Alt= 0 #EPs= 3 Cls=08(stor.) Sub=06 Prot=50 Driver=usb-storage
E: Ad=81(I) Atr=02(Bulk) MxPS= 64 Iv1=0ms
```

```
E: Ad=02(0) Atr=02(Bulk) MxPS= 64 Iv1=0ms
E: Ad=83(I) Atr=03(Int.) MxPS= 2 Iv1=1ms
```

Le righe in più, cioè quelle successive alla riga

```
T: Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 2 Spd=12 MxCh= 0
```

si riferiscono alla flash pen (**Solid state disk**). Per poterla utilizzare è necessario montare il filesystem ad essa relativo, in un mount point nell'albero delle directory. Il file di dispositivo relativo alla flash pen è la prima partizione del primo disco SCSI non ancora utilizzato dal sistema (nel caso in cui non sia presente nessun disco SCSI, il file di dispositivo relativo alla flash pen è `/dev/sda1`).

Quindi, si crea una directory (da superuser) che fungerà da mount point (es. `/mnt/usbdisk`) con il comando

```
# mkdir /mnt/usbdisk
```

e si inserisce all'interno del file `/etc/fstab` una riga analoga a quella seguente

```
/dev/sda1          /mnt/usbdisk      auto      noauto,user      0 0
```

A questo punto qualunque utente potrà digitare il comando

```
$ mount /mnt/usbdisk
```

per montare il filesystem nella directory `/mnt/usbdisk`, mentre il comando

```
$ umount /mnt/usbdisk
```

servirà per smontare lo stesso prima di scollegare il dispositivo dal sistema.

???

B.2 Interfaccia grafica

B.2.1 Personalizzazione dei cursori del mouse

I cursori (o puntatori) del mouse sono i simboli caratteristici (spesso frecce) che si spostano sull'interfaccia grafica seguendo il movimento del dispositivo di puntamento (mouse) e permettono all'utente di impartire i comandi. Ogni simbolo viene visualizzato in corrispondenza di un'opportuno stato: ad esempio, se il cursore è in prossimità del bordo di una finestra, esso assume un aspetto diverso da quando è lontano da esso, in modo da far capire all'utente che è possibile spostare il bordo della finestra per ridimensionarla.

Dunque i file dei cursori sono raggruppati in insiemi detti **temi** (*theme*), che contengono un'immagine per ogni stato in cui può trovarsi il cursore.

Ogni cursore è caratterizzato da un punto sensibile (*hot spot*), che è il pixel dell'immagine del cursore preso come riferimento, cioè il pixel che determina la posizione dello schermo individuato in ogni momento dal cursore. Ad esempio, nel caso di una freccia generalmente l'*hot spot* è posizionato sulla punta della freccia.

XFree86 gestisce le immagini da associare al dispositivo di puntamento, ricercandole, nell'ordine indicato, nelle seguenti directory

1. `~/icons`
2. `/usr/share/icons`
3. `/usr/share/pixmaps`
4. `/usr/X11R6/lib/X11/icons`



Figura B.1: I cursori del dispositivo di puntamento.

In ognuna delle directory viene ricercata la directory che ha il nome del tema desiderato ed all'interno di tale directory, viene ricercata la directory **cursors**, che conterrà i file relativi ai cursori. Ad esempio, se si vuole utilizzare il tema "MyTheme", verrà ricercata la directory `~/.icons/MyTheme` e se non esiste viene tentato con la directory `/usr/share/icons/MyTheme`, e così via. Non appena viene trovata la directory **MyTheme**, vengono considerati i file presenti nella sottodirectory **cursors**. Per il cursore viene sempre utilizzato il relativo file presente nella prima directory trovata nell'elenco precedentemente riportato. Se qualcuna delle directory relative al tema non esiste, vengono utilizzati i cursori di default.

Se esiste un file **index.theme** all'interno di una directory relativa ad un tema, esso deve contenere i riferimenti ad un altro tema da cui si vogliono ereditare i cursori. La sintassi di tale file è la seguente

```
[Icon Theme]
Inherits=theme.name
```

dove *theme.name* è il nome del tema (directory) a cui fare riferimento nel caso in cui alcuni file relativi ai cursori non esistano nella sottodirectory **cursors**.

Se nessun tema è stato indicato, viene considerato il tema "default".

I file che rappresentano i cursori hanno un formato particolare (formato Xcursor), che può essere ottenuto trasformando una o più immagini in formato PNG in un file Xcursor, tramite il comando **xcursorgen**.

```
Comando: xcursorgen
Path: /usr/X11R6/bin/xcursorgen
SINTASSI
# xcursorgen [config-file] [Xcursor-file]
```

DESCRIZIONE

config-file è il nome del file di configurazione per la generazione del file in formato Xcursor;

Xcursor-file è il nome del file in formato Xcursor che verrà prodotto da **xcursorgen**;

Il file di configurazione contiene righe con il seguente formato

size Xhot Yhot filename [delay]

dove

size è la dimensione in pixel del lato dell'immagine a cui la riga si riferisce (immagine quadrata);

Xhot è la coordinata orizzontale della posizione dell'*hot spot* dell'immagine con valori crescenti da sinistra verso destra (inizia da 0);

Yhot è la coordinata verticale della posizione dell'*hot spot* dell'immagine con valori crescenti dall'alto verso il basso (inizia da 0);

filename

è il nome del file che contiene l'immagine relativa ad un frame dell'animazione del cursore;

delay è il ritardo espresso in millisecondi tra il frame relativo alla riga corrente e quello successivo nell'animazione del cursore (se non è specificato, il cursore è rappresentato da una sola immagine fissa);

I file che rappresentano i cursori nei loro vari stati, devono rispettare la nomenclatura riportata in fig. B.1.

Si supponga dunque di voler sostituire il cursore visualizzato nello stato “standard”, ovvero in quello meno particolare di tutti, che è spesso rappresentato da un freccia rivolta verso l'alto a sinistra. La prima cosa da fare è quella di creare un'immagine quadrata (ad esempio 24×24 pixel) avente l'*hot spot* posizionato nel pixel più in alto a sinistra, con la quale sostituire il cursore e salvarla in un file in formato PNG (ad es. `mycursor.png`). Quindi si crea un file di testo (ad es. `mycursor.conf`) contenente la seguente riga

```
24 0 0 mycursor.png
```

e si crea il file `left_ptr` con il comando

```
$ xcursorgen mycursor.conf left_ptr
```

Adesso non rimane altro creare una directory (ad es. `~/icons/mycursors/cursors`) nella quale copiare il file `left_ptr` appena creato. Dopodiché si deve indicare al sistema di utilizzare il tema “mycursors”, creando il file di testo `~/icons/default/index.theme`, inserendovi le seguenti righe

```
[Icon Theme]
```

```
Inherits=mycursors
```

Dopo aver riavviato il server grafico, al successivo accesso all'interfaccia grafica da parte dell'utente considerato, si vedrà il cursore creato.

Se si avesse voluto creare un cursore animato, si sarebbero dovuti creare più file contenenti altrettante immagini in formato PNG che rappresentano i fotogrammi (*frame*) dell'animazione. Si consideri ad esempio un cursore animato composto da 3 frame, ognuno di 24×24 pixel (ad es. `mycursor1.png`, `mycursor2.png` e `mycursor3.png`). Il corrispondente file di configurazione (ad es. `mycursor.conf`) potrebbe essere composto dalle seguenti righe

```
24 0 0 mycursor1.png 400
```

```
24 0 0 mycursor2.png 300
```

24 0 0 mycursor3.png 300

così da far visualizzare, nell'animazione del cursore, il primo frame per una durata di 400 ms ed il secondo ed il terzo per 300 ms.

B.2.2 Personalizzazione dei menù di Gnome

Il menù di sistema di Gnome è composto da un elenco di voci, cliccando sulle quali si possono lanciare delle applicazioni o aprire altri elenchi di voci (sotto-menù). Un esempio è riportato in fig. B.2.

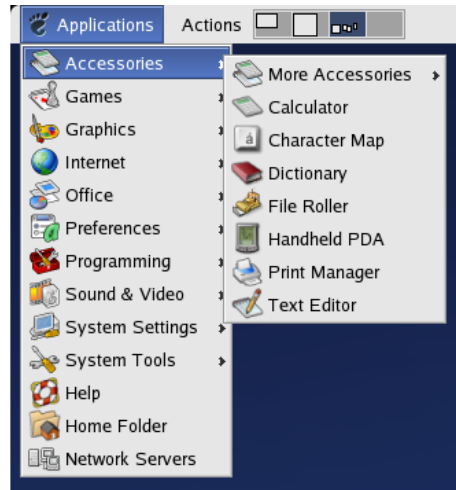


Figura B.2: Esempio de menù di sistema di Gnome.

Le voci nell'elenco sono la rappresentazione dei file con estensione **desktop** contenuti nella directory `/usr/share/applications` (a livello di sistema) e `???` (a livello utente).

Tali file hanno la struttura ...
 ???

B.3 Interfaccia carattere

B.3.1 Disabilitare lo shutdown con Ctrl-Alt-Del

???

Appendice C

Licenza

Il presente documento, come riportato nella pagina precedente all'indice, è coperto dalla licenza GNU FDL (GNU Free Documentation License), della quale si riporta una copia della versione originale (in lingua inglese) ed una copia di una versione non ufficiale in lingua italiana, utile per meglio comprendere il senso della licenza stessa.

C.1 GNU Free Documentation License

Version 1.2, November 2002
Copyright ©2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **“Modified Version”** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **“Secondary Section”** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **“Invariant Sections”** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **“Cover Texts”** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **“Transparent”** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called **“Opaque”**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **“Title Page”** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section **“Entitled XYZ”** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **“Acknowledgements”**, **“Dedications”**, **“Endorsements”**, or **“History”**.) To **“Preserve the Title”** of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and

disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

C.2 GNU Free Documentation License (italiano)

Quella presentata in questa sezione è una traduzione italiana non ufficiale della GNU Free Documentation License (Licenza per Documentazione Libera GNU). Non è pubblicata dalla Free Software Foundation e non ha valore legale nell’esprimere i termini di distribuzione delle opere che la utilizzano. Solo la versione originale in inglese della licenza (riportata in sez. C.1) ha valore legale. La presente traduzione è riportata soltanto nella speranza che possa aiutare le persone di lingua italiana a capire meglio il significato della licenza.

Versione 1.2, November 2002

Copyright ©2000,2001,2002 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

A chiunque è permesso copiare e distribuire copie letterali di questa licenza, ma senza apportare alcuna modifica.

Preambolo

Lo scopo di questa licenza è di rendere un manuale, un testo o altri funzionali ed utili documenti “liberi” nel senso di assicurare a tutti la libertà effettiva di copiarli e redistribuirli, con o senza modifiche, a fini di lucro o meno. In secondo luogo questa licenza prevede per autori ed editori il modo per ottenere il giusto riconoscimento del proprio lavoro, preservandoli dall’essere considerati responsabili per modifiche apportate da altri.

Questa licenza è un “copyleft”: ciò vuol dire che i lavori che derivano dal documento originale devono essere ugualmente liberi. È il complemento alla Licenza Pubblica Generale GNU, che è una licenza di tipo “copyleft” pensata per il software libero.

Abbiamo progettato questa licenza al fine di applicarla alla documentazione del software libero, perché il software libero ha bisogno di documentazione libera: un programma libero dovrebbe accompagnarsi a manuali che forniscano la stessa libertà del software. Ma questa licenza non è limitata alla documentazione del software; può essere utilizzata per ogni testo che tratti un qualsiasi argomento e al di là dell'avvenuta pubblicazione cartacea. Raccomandiamo principalmente questa licenza per opere che abbiano fini didattici o per manuali di consultazione.

1. APPLICABILITÀ E DEFINIZIONI

Questa licenza si applica a qualsiasi manuale o altra opera, in qualsiasi formato, che contenga una nota messa dal detentore del copyright che dica che si può distribuire nei termini di questa licenza. Tale nota assicura una licenza d'uso dell'opera, riconosciuta in tutto il mondo, libera da royalty, di durata illimitata, secondo le condizioni qui elencate. Con il termine **“documento”**, in seguito ci si riferisce a qualsiasi manuale o opera alla quale si applica questa licenza. Ogni fruitore è un destinatario della licenza e viene indicato con il termine **“voi”**. La licenza viene automaticamente accettata se si copia, modifica o distribuisce il documento in modo tale da richiedere il permesso relativo alle leggi sul copyright.

Una **“versione modificata”** di un documento è ogni opera contenente il documento stesso o parte di esso, sia riprodotto alla lettera che con modifiche, oppure traduzioni in un'altra lingua.

Una **“sezione secondaria”** è un'appendice cui si fa riferimento o una premessa del documento e riguarda esclusivamente il rapporto dell'editore o dell'autore del documento con l'argomento generale del documento stesso (o argomenti affini) e non contiene nulla che possa essere compreso nell'argomento principale (per esempio, se il documento è in parte un manuale di matematica, una sezione secondaria non può contenere spiegazioni di matematica). Il rapporto con l'argomento può essere un tema collegato storicamente con il soggetto principale o con soggetti affini, o essere costituito da argomentazioni legali, commerciali, filosofiche, etiche o politiche pertinenti.

Le **“sezioni non modificabili”** sono alcune sezioni secondarie i cui titoli sono esplicitamente dichiarati essere sezioni non modificabili, nella nota che indica che il documento è realizzato sotto questa licenza. Se una sezione non corrisponde alla definizione di sezione secondaria sopra espressa, non è permesso designarla come non modificabile. Il documento può non contenere sezioni non modificabili. Se il documento non identifica nessuna sezione non modificabile significa che non ne contiene alcuna.

I **“testi copertina”** sono dei brevi brani di testo che sono elencati nella nota che indica che il documento è realizzato sotto questa licenza. Un testo di fronte copertina può contenere al più 5 parole, ed un testo di retro copertina può contenere al massimo 25 parole.

Una copia **“trasparente”** del documento indica una copia leggibile da un calcolatore, codificata in un formato le cui specifiche sono disponibili pubblicamente (preferibile per effettuare revisioni dei documenti), i cui contenuti possono essere visti e modificati direttamente, ora e in futuro, con generici editor di testi o (per immagini composte da pixel) con generici editor di immagini o (per i disegni) con qualche editor di disegni ampiamente diffuso, e la copia deve essere adatta al trattamento per la formattazione o per la conversione in una varietà di formati atti alla successiva formattazione. Una copia fatta in un altro formato di file trasparente il cui markup (o la cui assenza di markup) sia stato effettuato per intralciare o scoraggiare modifiche future da parte dei lettori non è trasparente. Un formato di immagine non è trasparente se è utilizzato per rappresentare qualsiasi porzione sostanziale di testo. Una copia che non è trasparente è detta **“opaca”**.

Esempi di formati adatti per copie trasparenti sono l'ASCII puro senza markup, il formato di input per Texinfo, il formato di input per \LaTeX SGML o XML accoppiati ad una DTD pubblica e disponibile, e semplice HTML, PostScript o PDF conforme agli standard e progettato per essere modificato manualmente. Esempi di formati immagini trasparenti sono PNG, XCF e JPG. I formati opachi sono formati proprietari che possono

essere letti e modificati solo con word processor proprietari, SGML o XML per cui non è in genere disponibile la DTD o gli strumenti per il trattamento, e HTML, PostScript o PDF generato automaticamente da qualche word processor per il solo output.

La “**pagina del titolo**” di un libro stampato indica la pagina del titolo stessa, più qualche pagina seguente per quanto necessario a contenere in modo leggibile, il materiale che la licenza prevede che compaia nella pagina del titolo. Per opere in formati in cui non sia contemplata esplicitamente la pagina del titolo, con “pagina del titolo” si intende il testo prossimo al titolo dell’opera, precedente l’inizio del corpo del testo.

Una sezione “**intitolata XYZ**” indica una parte del documento il cui titolo è esattamente XYZ o contiene XYZ in parentesi che segue il testo XYZ tradotto in un’altra lingua (qui XYZ sta per uno specifico nome di sezione menzionato di seguito, come “**Ringraziamenti**”, “**Dediche**”, “**Approvazioni**” o “**Storia**”). Per “**conservare il titolo**” di tale sezione, qualora si modificasse il documento, si intende che il titolo deve rimanere “intitolato XYZ” in accordo con tale definizione.

Il documento può includere dinieghi di garanzie (Warranty Disclaimers) in prossimità della nota che attesta l’applicazione di questa licenza al documento. Questi dinieghi di garanzie sono considerati parte della licenza, ma soltanto per quello che riguarda il diniego di garanzie: qualunque altra implicazione che essi possono avere è nulla e non ha effetto sull’applicazione della licenza.

2. COPIE LETTERALI

Si può copiare e distribuire il documento con l’ausilio di qualsiasi mezzo, per fini di lucro e non, fornendo per tutte le copie questa licenza, le note sul copyright e l’avviso che questa licenza si applica al documento, e che non si aggiungono altre condizioni al di fuori di quelle della licenza stessa. Non si possono usare misure tecniche per impedire o controllare la lettura o la produzione di copie successive alle copie che si producono o distribuiscono. Però si possono ricavare compensi per le copie fornite. Se si distribuiscono un numero sufficientemente elevato di copie si devono seguire anche le condizioni della sezione 3.

Si possono anche prestare copie e con le stesse condizioni sopra menzionate possono essere utilizzate in pubblico.

3. COPIARE IN NOTEVOLI QUANTITÀ

Se si pubblicano a mezzo stampa (o si copiano su altri supporti che in genere sono provvisti di copertine) più di 100 copie del documento, e la nota della licenza indica che esistono uno o più testi copertina, si devono includere nelle copie, in modo chiaro e leggibile, tutti i testi copertina indicati: il testo della prima di copertina in prima di copertina e il testo di quarta di copertina in quarta di copertina. Ambedue devono identificare l’editore che pubblica il documento. La prima di copertina deve presentare il titolo completo con tutte le parole che lo compongono egualmente visibili ed evidenti. Si può aggiungere altro materiale alle copertine. Il copiare con modifiche limitate alle sole copertine, purché si preservino il titolo e le altre condizioni viste in precedenza, è considerato alla stregua di copiare alla lettera.

Se il testo richiesto per le copertine è troppo voluminoso per essere riprodotto in modo leggibile, se ne può mettere una prima parte per quanto ragionevolmente può stare in copertina, e continuare nelle pagine immediatamente seguenti.

Se si pubblicano o distribuiscono copie opache del documento in numero superiore a 100, si deve anche includere una copia trasparente leggibile da un calcolatore per ogni copia o menzionare per ogni copia opaca un indirizzo di una rete di calcolatori pubblicamente accessibile in cui vi sia una copia trasparente completa del documento, spogliato di materiale aggiuntivo, e a cui si possa accedere anonimamente e gratuitamente per scaricare il documento usando i protocolli standard e pubblici generalmente usati. Se si adotta l’ultima opzione, si deve prestare la giusta attenzione, nel momento in cui si inizia la distribuzione in quantità elevata di copie opache, ad assicurarsi che la copia

trasparente rimanga accessibile all'indirizzo stabilito fino ad almeno un anno di distanza dall'ultima distribuzione (direttamente o attraverso rivenditori) di quell'edizione al pubblico.

È caldamente consigliato, benché non obbligatorio, contattare l'autore del documento prima di distribuirne un numero considerevole di copie, per metterlo in grado di fornire una versione aggiornata del documento.

4. MODIFICHE

Si possono copiare e distribuire versioni modificate del documento rispettando le condizioni delle precedenti sezioni 2 e 3, purché la versione modificata sia realizzata seguendo scrupolosamente questa stessa licenza, con la versione modificata che svolga il ruolo del documento, così da estendere la licenza sulla distribuzione e la modifica a chiunque ne possieda una copia. Inoltre nelle versioni modificate si deve:

- A. Usare nella pagina del titolo (e nelle copertine se ce ne sono) un titolo diverso da quello del documento, e da quelli di versioni precedenti (che devono essere elencati nella sezione storia del documento ove presenti). Si può usare lo stesso titolo di una versione precedente se l'editore di quella versione originale ne ha dato il permesso.
- B. Elencare nella pagina del titolo, come autori, una o più persone o gruppi responsabili in qualità di autori delle modifiche nella versione modificata, insieme ad almeno cinque fra i principali autori del documento (tutti gli autori principali se sono meno di cinque) a meno che essi non permettano esplicitamente di non rispettare questo requisito.
- C. Dichiarare nella pagina del titolo il nome dell'editore della versione modificata in qualità di editore.
- D. Conservare tutte le note sul copyright del documento originale.
- E. Aggiungere un'appropriata licenza per le modifiche di seguito alle altre licenze sul copyright.
- F. Includere immediatamente dopo la nota di copyright, un avviso di licenza che dia pubblicamente il permesso di usare la versione modificata nei termini di questa licenza, nella forma mostrata nell'addendum alla fine di questo testo.
- G. Preservare in questo avviso di licenza l'intera lista di sezioni non modificabili e testi copertina richieste come previsto dalla licenza del documento.
- H. Includere una copia non modificata di questa licenza.
- I. Conservare la sezione intitolata "Storia", conservare il suo titolo, e aggiungere a questa un elemento che riporti al minimo il titolo, l'anno, i nuovi autori, e gli editori della versione modificata come figurano nella pagina del titolo. Se non ci sono sezioni intitolate "Storia" nel documento, createne una che riporti il titolo, gli autori, gli editori del documento come figurano nella pagina del titolo, quindi aggiungete un elemento che descriva la versione modificata come detto in precedenza.
- J. Conservare l'indirizzo in rete riportato nel documento, se c'è, al fine del pubblico accesso ad una copia trasparente, e possibilmente l'indirizzo in rete per le precedenti versioni su cui ci si è basati. Questi possono essere collocati nella sezione "Storia". Si può omettere un indirizzo di rete per un'opera pubblicata almeno quattro anni prima del documento stesso, o se l'originario editore della versione cui ci si riferisce ne dà il permesso.
- K. Per ogni sezione intitolata "Ringraziamenti" o "Dediche", si conservino il titolo, il senso, il tono della sezione stessa.

- L. Si conservino inalterate le sezioni non modificabili del documento, nei propri testi e nei propri titoli. I numeri della sezione o equivalenti non sono considerati parte del titolo della sezione.
- M. Si cancelli ogni sezione intitolata “Approvazioni”. Solo questa sezione può non essere inclusa nella versione modificata.
- N. Non si modifichi il titolo di sezioni esistenti come “Approvazioni” o per creare confusione con i titoli di sezioni non modificabili.
- O. Si conservi qualsiasi diniego di garanzia.

Se la versione modificata comprende nuove sezioni di primaria importanza o appendici che ricadono in sezioni secondarie, e non contengono materiale copiato dal documento, si ha facoltà di rendere non modificabili quante sezioni si voglia. Per fare ciò si aggiunga il loro titolo alla lista delle sezioni immutabili nella nota di copyright della versione modificata. Questi titoli devono essere diversi dai titoli di ogni altra sezione.

Si può aggiungere una sezione intitolata “Approvazioni”, a patto che non contenga altro che le approvazioni alla versione modificata prodotte da vari soggetti—per esempio, affermazioni di revisione o che il testo è stato approvato da una organizzazione come la definizione normativa di uno standard.

Si può aggiungere un brano fino a cinque parole come Testo Copertina, e un brano fino a 25 parole come Testo di Retro Copertina, alla fine dell’elenco dei Testi Copertina nella versione modificata. Solamente un brano del Testo Copertina e uno del Testo di Retro Copertina possono essere aggiunti (anche con adattamenti) da ciascuna persona o organizzazione. Se il documento include già un testo copertina per la stessa copertina, precedentemente aggiunto o adattato da voi o dalla stessa organizzazione nel nome della quale si agisce, non se ne può aggiungere un altro, ma si può rimpiazzare il vecchio ottenendo l’esplicita autorizzazione dall’editore precedente che aveva aggiunto il testo copertina.

L’autore/i e l’editore/i del documento non ottengono da questa licenza il permesso di usare i propri nomi per pubblicizzare la versione modificata o rivendicare l’approvazione di ogni versione modificata.

5. UNIONE DI DOCUMENTI

Si può unire il documento con altri realizzati sotto questa licenza, seguendo i termini definiti nella precedente sezione 4 per le versioni modificate, a patto che si includa l’insieme di tutte le Sezioni Invarianti di tutti i documenti originali, senza modifiche, e si elenchino tutte come Sezioni Invarianti della sintesi di documenti nella nota relativa alla licenza, e che si preservino tutti i relativi dinieghi di garanzie.

Nella sintesi è necessaria una sola copia di questa licenza, e multiple sezioni invarianti possono essere rimpiazzate da una singola copia se identiche. Se ci sono multiple Sezioni Invarianti con lo stesso nome ma contenuti differenti, si renda unico il titolo di ciascuna sezione aggiungendovi alla fine e fra parentesi, il nome dell’autore o editore della sezione, se noti, o altrimenti un numero distintivo. Si facciano gli stessi aggiustamenti ai titoli delle sezioni nell’elenco delle Sezioni Invarianti nella nota di copyright della sintesi.

Nella sintesi si devono unire le varie sezioni intitolate “Storia” nei vari documenti originali di partenza per formare un’unica sezione intitolata “Storia”; allo stesso modo si proceda con le sezioni intitolate “Ringraziamenti”, e “Dediche”. Si devono eliminare tutte le sezioni intitolate “Approvazioni”.

6. RACCOLTE DI DOCUMENTI

Si può produrre una raccolta che consista del documento e di altri realizzati sotto questa licenza; e rimpiazzare le singole copie di questa licenza nei vari documenti con una sola inclusa nella raccolta, solamente se si seguono le regole fissate da questa licenza per le copie alla lettera come se si applicassero a ciascun documento.

Si può estrarre un singolo documento da una raccolta e distribuirlo individualmente sotto questa licenza, solo se si inserisce una copia di questa licenza nel documento estratto e se si seguono tutte le altre regole fissate da questa licenza per le copie alla lettera del documento.

7. RACCOGLIERE INSIEME A LAVORI INDIPENDENTI

Una raccolta del documento o sue derivazioni con altri documenti o lavori separati o indipendenti, all'interno di o a formare un archivio o un supporto per la distribuzione, è detta "aggregato" se il copyright per l'intera raccolta non è utilizzato per limitare i diritti legali degli utenti della raccolta oltre a quelli che ogni lavoro individuale permette. Quando il documento è incluso in un aggregato, questa licenza non si applica agli altri lavori in esso contenuti, qualora non siano però loro stessi lavori derivati dal documento.

Se le esigenze del Testo Copertina della sezione 3 sono applicabili a queste copie del documento allora, se il documento è inferiore alla metà dell'intero aggregato i Testi Copertina del documento possono essere posti in copertine che delimitano solo il documento all'interno dell'aggregato, o l'equivalente elettronico delle copertine se il documento è in formato elettronico. Altrimenti devono apparire sulla copertina stampata relativa all'intero aggregato.

8. TRADUZIONI

La traduzione è considerata un tipo di modifica, e di conseguenza si possono distribuire traduzioni del documento seguendo i termini della sezione 4. Rimpiazzare sezioni non modificabili con traduzioni richiede un particolare permesso da parte dei detentori del diritto d'autore, ma si possono includere traduzioni di una o più sezioni non modificabili in aggiunta alle versioni originali di queste sezioni immutabili. Si può fornire una traduzione della presente licenza, e tutte le note ad essa relative presenti nel documento, ed ogni diniego di garanzia, a patto che si includa anche l'originale versione inglese di questa licenza e le versioni originali delle note relative e dei dinieghi di garanzia. In caso di discordanza fra la traduzione e la versione originale di questa licenza o nota o diniego, la versione originale prevale sempre.

Se una sezione del documento è intitolata "Riconoscimenti", "Dediche" o "Storia", il requisito (sezione 4) di conservarne il titolo (sezione 1) richiederà di modificarne l'attuale titolo.

9. TERMINI

Non si può applicare un'altra licenza al documento, copiarlo, modificarlo, o distribuirlo al di fuori dei termini espressamente previsti da questa licenza. Ogni altro tentativo di applicare un'altra licenza al documento, copiarlo, modificarlo, o distribuirlo è deprecato e pone fine automaticamente ai diritti previsti da questa licenza. Comunque, per quanti abbiano ricevuto copie o abbiano diritti coperti da questa licenza, essi non ne cessano se si rimane perfettamente coerenti con quanto previsto dalla stessa.

10. REVISIONI FUTURE DI QUESTA LICENZA

La Free Software Foundation può pubblicare nuove versioni rivedute della Licenza per Documentazione Libera GNU volta per volta. Qualche nuova versione potrebbe essere simile nello spirito alla versione attuale ma differire in dettagli per affrontare nuovi problemi e concetti. Si veda <http://www.gnu.org/copyleft>.

Ad ogni versione della licenza viene dato un numero che distingue la versione stessa. Se il documento specifica che si riferisce ad una versione particolare della licenza contraddistinta dal numero o "ogni versione successiva", si ha la possibilità di seguire termini e condizioni sia della versione specificata che di ogni versione successiva pubblicata (non come bozza) dalla Free Software Foundation. Se il documento non specifica un numero di versione particolare di questa licenza, si può scegliere ogni versione pubblicata (non come bozza) dalla Free Software Foundation.

Indice analitico

- ' , 297
- (, 294, 298
-) , 294, 298
- * , 206, 297
- ** , 298
- + , 300
- , , 297
- , 300
- . , 297
- / , 34, 92, 93, 297
- : , 297
- ; , 294, 296
- ;; , 297
- < , 294, 302
- = , 300
- > , 294, 302
- >> , 302
- ? , 298
- [, 298
- [, 299
- # , 279, 281, 296
- \$, 279, 281, 298
- % , 294, 300
- & , 294, 300
- &> , 303
- && , 300
- &> , 299
-] , 298
-]] , 299
- ‘ , 297
- { , 298
- } , 298
- ~ , 300
- ~+ , 300
- ~- , 300
- " , 297
- < , 299
- > , 299
- >> , 299
- \ , 297
- \< , 299
- \> , 299
- ^ , 300
- .. , 92
- . , 92
- 2> , 303
- abstraction layer, 122
- accesso diretto, 92
- accesso sequenziale, 92
- account, 205
- active close, 408
- active open, 407
- address bus, 22
- agetty, 229, 231
- AGP, 24
- albero, 34, 91, 92
 - foglia, 92
 - freccia, 91
 - nodo, 91, 92
 - radice, 92
- albero dei processi, 263
- albero delle directory, 91, 92
- algebra di Boole, 12
- algoritmi di cifratura, 212
- alias, 290
- alias, 290
- aliasing, 317
- ambiente, 267, 280, 285
- amministratore, 206
- amministratore del sistema, 16
- amministratori dei gruppi, 214
- anacron, 311, 313, 314
- anacrontab, 314
- analogici, 315
- and, 12
- anonymous, 227
- ANSI, 13
- anti-aliasing, 343
- append, 302
- applicazioni, 36
- applicazione, 3, 36
- apropos, 46
- apt-get, 364
- architettura a 32 bit, 22
- ARP, 396
- arp, 396
- ASCII, 13
- Assisted LBA, 29
- assoluto, 34
- AT, 25
- at, 218, 311
- ATA, 18, 24
 - cable select, 25
 - canale, 24
 - master, 24

- primary, 24
 - secondary, 24
 - slave, 24
- ATAPI, 24
- attacco per dizionario, 429
- attacco per forza bruta, 429
- autonegoiazione, 384
- background, 303, 304
- backslash, 204, 226, 285, 287
- backslash continuation, 75
- backspace, 287
- backup, 57
- badblocks**, 182
- banda passante, 320
- base, 5
- basename, 285
- Bash, 17, 283–288, 291–296, 300
- bash, 288
- bash**, 282–285, 287, 289, 300, 301
- batch, 294
- batch**, 311
- best-effort, 389
- bg**, 304
- bias, 10
- big endian, 387
- bin, 217
- BIOS, 50
- bit, 12
- bit-shift, 29
- blank, 112, 282
- block cipher, 434
- boot, 49
 - chain-load, 56
- boot loader, 29, 50, 52, 93
- boot manager, 52
- boot sector, 50, 52
- bootable, 50
- bootstrap, 49
- break, 231
- bridge, 385
- broadcast, 383
- browser, 42
- buffer, 31, 193
- buffer di shell, 279
- buffer overflow, 466
- bug, 465
- built-in command, 291
- bus, 18, 21
 - di sistema, 22
- bus parallelo, 21
- bus seriale, 21
- button, 324
- byte, 12
- bzip2**, 362
- cache, 19, 31, 54
 - livello, 19
- cache ARP, 396
- cal**, 310
- campionamento, 316
- campo, 348
- caratteri jolly, 293
- caret, 280
- caret notation, 195, 304
- carriage return, 285
- case, 17
- case insensitive, 226
- case sensitive, 204
- case sensitive, 92
- cat**, 190, 191
- cavi di rete, 39, 375
- CD, 20, 29
 - CD-R, 21
 - CD-RW, 21
- cd**, 140
- cella, 19
- centro stella, 381
- Centronics, 26
- Certification Authority, 443
- certificato digitale, 443
- chage**, 213, 214
- Chain, 470
- chain, 472
- chain_B, 477
- challenge, 439, 452
- Channel Access Method, 381
- chassis, 17
- chattr**, 102
- chfn**, 208, 209
- chgrp**, 118, 119
- chiave, 353, 428
- chiave primaria, 353
- chiave privata, 436
- chiave pubblica, 436
- chkconfig**, 74, 75
- chkfontpath**, 344
- chmod**, 108, 116, 117, 186
- chown**, 117, 118
- Christmas tree packet, 406
- chroot**, 150
- chroot jail, 150
- CHS, 28
- CHS to CHS, 29
- chsh**, 208
- chunk, 169
- ciclo di clock, 18
- ciclica, 267
- CIDR, 391
- cifratura a chiave pubblica, 436
- cifratura a chiave segreta, 433
- cifra, 4
- cilindro, 28
- classe A, 390

- classe B, 390
- classe C, 390
- classe di congruenza, 430
- classi, 471
- clean, 119
- client, 323, 375, 376
- client grafici, 324
- clock, 310
- clock tick, 18
- cmd, 192
- code, 397
- code di messaggi, 276
- code segment, 114
- codice macchina, 36
- cold boot, 50
- collegamento
 - simbolico, 105
 - diretto, 105
 - fisico, 105
- comandi interni, 291, 294
- command, 240
- command history, 280, 291
- command interpreter, 279
- command line editing, 285
- commutazione, 381
- compilatore, 36
- compilazione, 36
- compiler, 36
- compress, 362
- computer, 3
- condizione, 469
- congruenti modulo n , 430
- connection, 404
- connection tracking, 469
- connectionless, 389
- consistente, 119
- contesa, 381
- control bus, 22
- controller, 27
- controller MIDI, 318
- cooperative multitasking, 270
- copyleft, 490
- core, 269
- core dump, 269, 273
- cp, 152, 186–188
- CPU, 18
- cracker, 465
- crashare, 228
- crittografia, 428
- cron, 218, 310, 312, 313
- crond, 311
- crontab, 311–314
- crontab, 221, 311
- crypt, 212, 232
- csh, 216, 282
- CSLIP, 383
- ctags, 195
- curse, 58
- cursore, 279, 280
- daemon, 68, 70
- data, 99
- data block, 100
- data bus, 22
- data encapsulation, 376
- data segment, 268
- database, 42, 347
 - campo, 352
 - colonna, 352
 - sequenza, 352
 - gerarchico, 347
 - record, 352
 - relazionale, 348
 - reticolare, 347
 - riga, 352
 - sequenza, 352
 - tabella, 352
- datagram, 78
- date, 134, 308, 309
- dati persistenti, 256
- debugger, 269, 273
- decimal dotted notation, 389
- default route, 392
- default shell, 205, 206, 208
- delivered, 272
- dentry, 104
- desktop, 17, 85
- desktop manager, 342
- Destination, 393
- device, 18
- device driver, 235
- df, 180, 181
- digit, 4
- digital signature, 441
- digitali, 316
- digitalizzato, 316
- DIN, 25
- dinamico, 151
- dircolors, 135
- direct mode, 250
- directed broadcast, 391
- directory, 34, 91, 92, 104
- directory cache, 125
- directory corrente, 34, 92
- directory di lavoro, 34, 92
- directory radice, 34
- directory tree, 91, 92
- dischetto, 20
- disco fisso, 20
- disco rigido, 20
- disco
 - ATA, 30
 - ESDI, 30
 - formattazione, 89

- MBR, 29
- partition table, 29
- partizionamento, 87
- partizione, 29, 87, 95
- partizioni estese, 30
- partizioni logiche, 30
- piatto, 28
- SCSI, 30
- settore, 28
- testina, 28
- traccia, 28
- XT, 30
- disk swapping, 60
- display, 323
- dispositivo, 18
- dispositivo di input, 17
- dispositivo di output, 17
- dispositivo di puntamento, 17
- distribuzioni, 39
- distro, 39
- DLC Address, 382
- dmesg, 244
- dnsdomainname, 414
- domini di collisione, 385
- doppiamente concatenata, 267
- DoS, 465
- dot, 320
- dot per inch, 320
- dot pitch, 320
- double precision, 10
- double quotes, 294
- double-word, 12
- dpi, 320
- dpkg, 363
- Draft Standard, 379
- drive, 20
- driver, 36, 87, 109, 235
- dselect, 364
- dtksh, 283
- DTS, 307
- du, 181, 182, 289
- dump, 128
- durata, 213
- DVD, 20, 29
- E²PROM, 19
- e2fsadm, 175
- e2label, 90
- EBCDIC, 13
- echo, 286, 287
- Echo reply, 398
- Echo request, 398
- ECHS, 29
- editor, 192
- edquota, 155, 156
- EEPROM, 19
- effective, 266
- effective GID, 265, 266
- effective UID, 80, 265, 266
- EIDE, 24
- EISA, 22
- elaboratore elettronico, 3
- emacs, 199, 202
- endianess, 388
- entità, 348
- environment, 267, 280, 285, 289
- environment variable, 280
- epoch, 307
- ERD, 355
- error, 267, 397
- escape character, 293
- ESDI, 27
- eseguibile, 36
- esponente, 5, 10
- espressioni regolari, 203
- ex, 200, 201
- exit, 216, 293
- exit status, 37, 127, 172, 269, 295
- exploit, 465, 466
- export, 288
- ext, 128
- ext2, 96, 97, 120–122, 183
 - blocco, 97
 - block, 97
 - cylinder group, 97
 - data block, 99
 - data block bitmap, 99
 - group descriptor, 98
 - inode, 100
 - inode bitmap, 99
 - inode table, 98, 99
- ext3, 120–122, 183
- extent, 95
- fake, 125
- fase, 315
- FAT, 53, 54
- fc-cache, 343
- fdisk, 30, 88
- fg, 304
- FHS, 93
- FIFO, 108
- file, 34, 91, 92, 99
 - atime, 100
 - ctime, 100
 - data –, 111
 - device –, 109
 - FIFO, 107
 - di dispositivo, 107, 109
 - speciale, 107
 - formato, 110
 - formato binario, 110
 - formato testo, 110
 - formattazione, 110

- GID, 100
- metadata, 99
- metadati, 99
- mode, 100
- mtime, 100
- named pipe, 107
- permessi, 113
- regular –, 103
- socket, 107
- sparse –, 96, 100
- tipo di –, 103
- UID, 100
- file**, 111
- file descriptor, 275
- file descriptor, 267, 275
- file sorgente, 36
- filesystem, 34, 91, 266
 - blocco allocato, 95
 - blocco, 95
 - block, 95
 - blocks bitmap, 98
 - ext2, 95, 96
 - ext3, 95
 - file, 99
 - fragment, 103
 - group descriptor, 98
 - inode, 100
 - inode bitmap, 98
 - inode table, 98
 - blocco libero, 95
 - shareable, 93
 - static, 93
 - struttura fisica, 91
 - struttura logica, 91
 - superblock, 97
 - unshareable, 93
 - variable, 93
- filesystem GID, 266
- filesystem UID, 265
- find**, 58, 202
- firewall, 466, 467
 - chain, 469
 - regole, 469
 - tabelle, 469
- firewall hardware, 467
- firewall software, 467
- firewire, 26
- firma digitale, 441
- firmware, 28, 50
- flag, 113
- flash, 20
- flash pen, 20, 501
- flat cable, 25, 27
- floating point, 10
- floppy disk, 20
- font, 343
- Fontconfig, 343
- footer, 376
- foreground, 303, 304
- foreign key, 354
- formattazione, 121, 190
- forwarding, 469
- fraction, 10
- fragments, 384, 395
- frame, 342, 382, 504
- frame buffer, 250
- frammentazione esterna, 95
- frammentazione interna, 103
- frammenti, 395
- free**, 151, 269
- free software, 490
- frequenza, 18, 315
- frequenza di refresh verticale, 321
- fsck**, 86, 119, 120, 122, 128, 129, 245
- funzione di Eulero, 432
- funzione hash, 266
- funzioni hash, 440
- funzioni modulari, 431
- funzioni modulari esponenziali, 432
- gateway, 385
- gedit**, 324, 325
- getty**, 83, 84, 229–231
- GID, 210
- glifo, 343
- globbing, 293
- glyph, 343
- GNU/Linux, xiii
- gpg**, 460
- grace period, 156
- grafo orientato, 91
- greeter, 84
- grep**, 202
- group account, 210
- groupdel**, 210, 211
- groupmod**, 210
- groupname, 210
- groups**, 211
- grpconv**, 215
- grpunconv**, 215
- GRUB, 55
 - stage, 56
- grub**, 58
- gruppo, 210
- gruppo A, 212
- gruppo B, 212
- GUI, 17
- gview**, 201
- gvim**, 201
- gzip**, 362
- hacker, 465
- hardware clock, 310
- half-byte, 12

- hard disk, 20
- hard limit, 156
- hard link, 100
- hard real-time, 271
- hardware, 18, 36
- hardware clock, 307–310
- hash, 266
- hdparm, 31, 32, 34
- head, 191
- header, 376
- help, 58, 201, 291
- help pwd, 140
- hexdump, 198, 199
- highcolor, 321
- hohup, 240
- home directory, 94, 140, 205–207
- hop, 392
- host id, 389, 391
- hostname, 328, 414
- hot spot, 502, 504
- hot-swap, 33
- hotkey, 147
- how, 47
- how-to, 47
- http root directory, 94
- httpd, 72
- hub, 384, 385
- hwclock, 308–310
- icon, 17
- icona, 17
- id, 211
- IDE, 24
- idle, 246
- IEEE, 26
- IETF, 379
- ifconfig, 389, 415
- immagine, 176, 236, 269
- impulsi di clock, 18
- in fase di sviluppo, 236
- incapsulazione, 376
- index node, 100
- indice, 353
- indirizzi privati, 391
- indirizzi pubblici, 390
- indirizzo, 19
- indirizzo di memoria, 19
- indirizzo fisico, 379, 396
- indirizzo IP, 389
- inetd, iv, 75, 76, 79, 446
- info, 43, 45
- info page, 45
- init, 63–70, 84, 86, 221, 255, 261, 263, 269, 340
- initd, 146
- inode, 100
 - direct block, 100
 - double indirect block, 100
 - indirect block, 100
 - triple indirect block, 100
- inode cache, 125
- input, 36, 37, 267
- insert, 286
- insmod, 239, 256
- integrità referenziale, 354
- intercettazione, 273
- interfacce di rete, 375
- interfaccia di rete, 39
- interfaccia a riga di comando, 16
- interfaccia di rete, 379
- interfaccia grafica, 17
- interfaccia testuale, 16
- interfaccia utente, 16
- internal command, 291
- Internet suite, 378, 389
- interpretazione, 36
- interprete, 36, 37
- interprete dei comandi, 279
- interpreter, 36
- interrupt, 38
- interrupt hardware, 39
- interrupt software, 39
- interruptible, 274
- intersezione, 12
- intersezione logica, 12
- IP datagram, 389, 394
- IP forwarding, 393
- IPC, 272
- ipchains, 471
- iptables, 471, 472, 481, 482
- iptables-restore, 481, 482
- iptables-save, 481, 482
- IrDA, 27
- ISA, 22
- ISN, 406
- ISO, 15
- issue, 229
- JFS, 120
- jiffies, 471
- job, 303
- job control, 303
- job corrente, 303
- jobs, 303
- journal, 120
- journaled filesystem, 120
- journaling, 120
- jsh, 283
- jumper, 27
- Kamikaze packet, 406
- Kerberos, 462
- kernel, 38, 50, 93, 235
 - immagine del –, 52

- kernel modulare, 236
- kernel monolitico, 236
- kernel panic, 248
- kernel-space, 236
- kernel**d, 256
- keyring, 460
- kill, 272, 273
- klogd, 220, 221, 246
- ksh, 282
- ksyms, 257, 261
- lamp test segment, 406
- LAN, 39
- laptop, 26
- last, 234
- lastlog, 233, 234
- layout, 35
- lazy unmount, 130
- LBA, 29
- less, 44, 45, 191, 193–198, 303
- lesskey, 194
- LFS, 101
- libreria, 94
- licenza, 489
- LILO
 - map file, 60
- lilo, 60–62
- linear mapping, 171
- linear mode, 163
- linguaggio di programmazione, 36
- linguaggio macchina, 36
- link, 105
 - dangling –, 106
 - fast –, 107
 - hard –, 105
 - physical –, 105
 - soft –, 105
 - symbolic –, 105
- linker, 37
- Linux, xiii
- Linux RAID, 164
- Linux-PAM, 221
- lista, 267
- little endian, 388
- live-CD, 40
- livelli di maturità, 379
- livello di applicazione, 378
- livello di collegamento, 377
- livello di presentazione, 378
- livello di rete, 377
- livello di sessione, 378
- livello di trasporto, 378
- livello fisico, 377
- 11, 290
- 1n, 106, 107
- LOADLIN, 53
- Local Area Network, 377
- log, 120
- log di sistema, 218
- logger, 220
- Logical Extent, 170
- Logical Volume, 170
- login, 83, 84, 206, 228, 229, 231–233
- login banner, 229
- logname, 209
- logout, 240
- logrotate, 221
- logwatch, 221
- long form, 282
- look-ahead, 32
- loopback, 391
- ls, 106, 131–134, 303
- lsattr, 102
- LSB, 12
- lspci, 22–24, 141
- lspci -x, 23, 24
- lsraid, 163, 167
- lvcreate, 174
- lvextend, 175
- LVM, 170
- lvremove, 175
- MAC, 442
- MAC address, 382
- macchina a stati, 36
- magic number, 111
- mail, 218
- make, 237, 238
- makefile, 363
- man, 43–45
- man page, 22, 43
- mandatory locking, 114
- Mandrake, 40
- Mandrake Linux, 40
- mantissa, 10
- map, 60
- mark, 475, 478
- match, 474
- MBP, 50
- MBR, 51
- MCA, 22
- md, 163
- md device, 163
- mdadd, 163, 168, 169
- mdcreate, 168, 169
- mdop, 170
- mdrun, 163, 169
- mdstop, 163, 169, 170
- meccanismo di routing, 392
- media, 20
- memoria, 18, 19
 - non volatile, 19
 - volatile, 19
- memoria cache, 31

- memoria centrale, 18, 19
 - RAM, 19
 - ROM, 19
- memoria condivisa, 276
- memoria di massa, 19
- memoria virtuale, 268
- mesh, 381
- message digest, 212, 440
- message queues, 276
- metacaratteri, 293, 294
 - ' , 294
 - *, 293
 - , 293
 - ., 293
 - :, 293
 - ?, 293
 - [, 293
 - {, 293
 - }, 293
 -], 293
 - ", 294
 - \, 294
 - ~, 293
- microprocessore, 18
- microswitch, 27
- MIDI message, 318
- mingetty, 229, 231
- minidistribuzioni, 40
- minix, 96
- MIPS, 18
- mixer, 317
- mkdir, 186
- mkfifo, 108
- mkfs, 89
- mknod, 107–109
- mkraid, 163, 165
- mkswap, 151
- MO, 20, 29
- mode, 113
- modem, 375
- modo promiscuo, 383
- modprobe, 260
- moduli, 38, 236, 255
- monitor, 17
- montare, 52
- more, 191–193, 196
- motherboard, 18
- mount, 44
- mount, iv, 125, 127–130
- mount point, 90, 94
- mount-point, 125
- mounting, 94, 119, 125, 223
- mouse, 17
- MSB, 12
- MTU, 384
- multicast, 390, 393
- multihomed, 375

- mv, 188, 189
- name completion, 290
- named pipe, 108, 141, 276
- nastygram, 406
- NAT, 468
- negazione, 12
- negazione logica, 13
- net id, 389, 391
- net prefix, 391, 392
- netstat, 409
- network, 39
- network byte order, 387
- newgrp, 211, 212
- newline, 285, 287
- nibble, 12
- NIC, 26, 39
- nice, 271
- nm, 199
- no_access list, 78
- nodo, 34
- nohup, 240
- normalizzazione, 10
- not, 13
- ntpd, 310
- ntsysv, 75
- numero
 - composto, 431
 - primi tra loro, 431
 - primo, 431
- objdump, 199
- od, 198, 199
- on-drive defect management, 32
- on-the-fly, 36, 121
- open source, 490
- operazione atomica, 266
- opzioni, 281, 282
- or, 13
- orologio CMOS, 307
- orologio del BIOS, 307
- orologio di sistema, 307
- otto caratteri, 212
- output, 36, 37, 267
- output device, 17
- overflow, 8, 12
- overwrite, 286
- pacchetti, 359
- pacchetto, 42, 376
- package, 359
- packet filter, 469
- packet filtering, 469
- page fault, 268
- pager, 44, 45, 58
- pagine, 235
- paging, 268

- PAM, 221
- parsing, 279
- parted, 88
- partimage, 177
- partimaged, 179
- partition table, 30, 34, 50, 51
- partizioni primarie, 29
- passive open, 407
- passo, 312
- passwd, 208
- password, 16, 83, 205, 208, 212, 229, 232
- PAT, 468
- PatchNumber, 38
- path, 34, 45, 93, 105
 - assoluto, 93
 - relativo, 93
- path assoluto, 34
- payload, 376
- PCI, 22
- PCMCIA, 26
- periferica
 - esterna, 21
 - interna, 21
- periferiche, 21
- periferica, 18
- peristenza, 320
- persistent data, 256
- Physical Extent, 170
- Physical Volume, 170
- pianificazione, 311
- PID, 261
- pidof, 263
- pin, 25
- ping, 398, 399
- pipe, 108, 193, 195, 219, 275
- pipeline, 302
- pippo, 94
- pixel, 321
- PKI, 441
- plain text, 212, 445
- PNG, 110
- PNP, 27
- npdump, 141
- policy, 467
- politica di scheduling, 270
- polling, 38
- port, 403
- port forwarding, 458
- porta, 403
- porta parallela, 18, 26
- porta seriale, 18, 26
- POST, 50
- PPID, 263
- PPP, 383
- pppd, 417
- preamble, 382
- preambolo, 382
- predicato, 12
- preemptive multitasking, 270
- principal, 462
- printer, 17
- priorità assoluta, 270
- priorità dinamica, 270
- privacy, 428
- procedura di boot, 49
- procedura di login, 205, 228
- process group, 66
- process tree, 263
- processo, 38, 261
- processo A, 263
- processo B, 263
- processo figlio, 263
- processo genitore, 263
- processo padre, 263
- processore, 18
- profondità di colore, 321
- programma, 36, 38
- programmi, 3, 36
- programma, 3
- promiscuous mode, 383
- promiscua, 378
- prompt, 16, 279–281
 - primario, 284
 - secondario, 284
- Proposed Standard, 379
- protected mode, 63
- protocollo, 39, 376
- prova, 217
- proxy server, 467
- ps, 64, 141, 264, 265, 270
- PS/2, 18, 25
- pstree, 263, 264
- punto di montaggio, 125
- pvccreate, 172, 174
- pvdisplay, 174
- pwconv, 214, 215
- pwd, 140
- pwunconv, 215
- query, 397
- quota, 153
- quota, 157, 158, 180
- quotacheck, 153, 154, 157
- quotaoff, 155
- quotaon, 154, 155
- quoting, 196, 293
- race condition, 275
- RAID, 158
 - stripe, 159
- raid0run, 166
- raidreconf, 166, 167
- raidstart, 163, 165, 166
- raidstop, 163, 166

- RAM, 19
 - banco, 19
- RAM disk, 176
- ramfs, 176
- rappresentazione in virgola mobile, 10
- rawrite.exe**, 41
- rawritewin.exe**, 42
- RDB, 348
- rdev**, 52, 244, 245
- read-ahead, 32
- real, 266
- real GID, 265, 266
- real mode, 63
- real UID, 265, 266
- real-time, 271
- reboot
 - cold –, 249
 - warm –, 249
- Red Hat, 40
- redhat-config-network**, 418
- redirezione dei comandi, 302
- regular expression, 203
- regular file, 103
- ReiserFS, 120
- relativo, 34
- relazioni, 354
- release, 37, 38
- remoto, 39
- renice**, 271, 272
- repquota**, 157
- restricted shell, 283
- rete, 39
- reti estese, 39, 377
- reti locali, 39, 377
- RFC, 379
- rgview**, 201
- rgvim**, 201
- riga di comando, 16
 - argomento, 16
 - parametro, 16
- riservatezza, 428
- risoluzione, 322
- rksh**, 283
- rm**, 106, 189, 190
- rmmod**, 260
- rollback, 122
- ROM, 19
- root, 16, 41, 94, 217
- root directory, 34, 53, 92, 93, 127, 128, 150
- route**, 392, 393, 418
- router, 385, 392, 469
- routing, 392
- routing table, 385, 392
- rpm**, 364
- RS-232, 26
- RS-422, 26
- rsh**, 283
- RTC, 21, 307
- run-parts**, 313
- runlevel, 64, 68, 85
 - ondemand runlevel, 65
- runlevel**, 69
- runnable, 271
- running, 274
- runt, 383, 385
- rview**, 201
- rvim**, 201
- salt, 212
- SAS, 25
- SATA, 25
- saturazione, 321
- saved GID, 265, 266
- saved UID, 265, 266
- SBP, 50
- scanpci**, 141
- scheda di rete, 375, 379
- scheda madre, 18
- schede di rete, 39
- schedulazione dei job, 311
- scheduler, 270
- scp**, 459
- screen, 323
- script, 37, 70, 280, 292–295
- scripting language, 280, 292, 294
- scroll, 279
- scrollback buffer, 250, 279
- SCSI, 25
 - Fast Wide SCSI, 25
 - ID, 25
 - LUN, 25
 - Ultra SCSI, 25
 - Wide SCSI, 25
- segmentation fault, 269
- segmenti, 268
- semafori, 276
- serbatoio, 31
- server, 375, 376
- server grafico, 323
- servizi, 70
- set**, 286, 288, 300
- setpci**, 22–24
- setquota**, 155
- settore, 28, 29
- sftp**, 459
- sgid, 113
- sh**, 282, 284
- shadow password, 206, 213
- shared library, 221
- shared memory, 276
- shell, 16, 37, 205, 279
- shell di login, 216, 283, 284
- shell interattiva, 283, 284

- shell interattiva di login, 283
- short form, 282
- shred**, 190
- shredding, 190
- shutdown, 119
- shutdown**, 85, 86, 119
- SI, xix
- sibling, 276
- signal handler, 273
- single precision, 10
- single quotes, 294
- single-sign-on, 462
- sintetizzazione, 317
- sistema di numerazione, 4
 - base, 4
 - binario, 6
 - decimale, 5
 - esadecimale, 6
 - esadecimale, 7
 - ottale, 6
 - posizionale, 4
- skeleton, 207
- skeleton directory, 206, 207
- slash, 34, 92
- SLIP, 383
- slot, 21
- smoothing, 343
- sniffer, 378
- socket, 403, 404
- socket**, 108
- soft limit, 156
- soft real-time, 271
- software, 18, 36
- software clock, 307–310, 312
- software di dominio pubblico, 490
- software freeware, 490
- software libero, 490
- software proprietario, 490
- software shareware, 490
- sorgente, 36
- source file, 36
- spare disk, 163
- sparse, 188
- spazio di indirizzi, 268
- SQL, 355
- ssh**, 453–455, 458
- ssh-keygen**, 453
- sshd**, 446–451, 453, 454, 459
- stabile, 236
- stack di protocolli, 376
- stampante, 17
- Standard, 379
- standard error, 267
- standard input, 267
- standard output, 267
- start of frame, 382
- startx**, 339
- state cipher, 434
- stateful inspection, 469
- stateless, 468
- statica, 270
- statico, 151
- steganografia, 428
- sticky, 114
- stored procedure, 355
- stream, 78
- stream cipher, 434
- strftime**, 285
- stringhe, 13
- strings**, 198
- striped mapping, 171
- su**, 215, 216
- subnet mask, 391
- successful, 37, 269
- suid, 113
- super daemon, 75
- superfloppy, 29
- superuser, 16, 41, 64, 94, 114, 116, 189, 206, 279
- swap, 128, 150, 268
- swap file, 151
- swap filesystem, 151
- swap in, 150
- swap out, 150
- swap partition, 151
- swapon**, 152, 153
- swapon**, 152
- switch, 385
- switching hub, 384
- symbolic link, 70
- symlink, 105
- SYN segment, 406
- sync**, 181, 182, 264
- SYSLINUX, 54
- syslog, 218
- syslogd**, 147, 218, 219, 221
- system call, 38
- system clock, 18, 307
- system group, 216
- system log, 76, 218
- system status file, 69, 221
- system user, 216
- tab folder, 324
- Tabelle, 470
- tail**, 191
- tar**, 359, 360
- tarball, 359
- target, 469
- task, 261
- task vector, 267
- tastiera, 17
- tavola di verità, 13
- tavoletta grafica, 17

- TCP segment, 404
- tcpdump, 409
- tcsh, 216, 282
- telinit, 66, 69, 85
- tema, 502
- testina, 28
- text editor, 199
- text segment, 268
- text viewer, 190
- theme, 502
- thread, 266
- three-way handshake, 407
- time slice, 270
- time zone, 134
- time-stamp, 61
- timestamp, 443
- tinta, 321
- to, 47
- token, 223, 280, 381
- token passing, 381
- tokenizzazione, 280
- toolbar, 324
- top, 141, 264, 265
- topologia fisica, 380
- topologia logica, 380
- touch, 185, 186
- touch pad, 17
- traceroute, 400, 401
- trackball, 17
- trailing, 320
- transceiver, 379
- transfer mode, 33
- transfer rate, 25
- trasduttori, 315
- tristate, 33
- truecolor, 321
- ttmkfdir, 344
- tune2fs, 97
- tutti, 217
- type, 397
- UDP datagram, 403
- UID, 16, 205
- umask, 80, 149
- umask, 149
- umount, iv, 125, 129, 130
- una, 212
- unalias, 290
- underflow, 11
- underscore, 279
- Unicode, 15
- unione, 12
- unione logica, 13
- Unix domain socket, 110
- unmaskirq, 33
- unmounted, 119
- unreliable, 389
- unset, 289
- urgent condition, 274
- urgent point, 405
- USB, 18, 26
- user account, 205, 207
- user friendly, 3
- user-space, 236
- useradd, 207
- userdel, 207, 208
- usermod, 207
- username, 16, 83, 94, 205, 207, 209, 229, 231, 232
- users, 233
- UT, 307
- UTC, 307
- utente, 205
- utente A, 216
- utente B, 216
- utenti, 16
- UTF-8, 15
- valore, 321
- vanilla kernel, 237
- variabili d'ambiente, 280
- variabili di ambiente, 285
- versioning, 38
- VFS, 122
- vgchange, 175
- vgcreate, 173
- vgdisplay, 174
- vgextend, 174
- vgreduce, 174
- vgremove, 173, 174
- vgscan, 175
- vi, 45, 192, 196, 199, 200, 239
- view, 201
- vim, 199–201
- vimdiff, 200
- violazione di segmento, 269
- virtual console, 83
- virtual memory, 150
- virtual memory manager, 150
- virtual terminal, 83
- virtuale, 268
- VLB, 22
- vmstat, 150, 269
- Volume Group, 170
- volumi, 170
- w, 233
- WAN, 39
- warm boot, 50
- warnquota, 158
- wavetable, 317
- well known port numbers, 408
- what is, 46
- which, 202

- who, 234
- whoami, 209
- Wide Area Network, 377
- wildcard, 293
- window, 17, 342
- window manager, 342
- wireless, 26
- word, 12
- working directory, 34, 92, 130, 140, 205,
206
- workstation, 376
- write-caching, 33

- X protocol, 323
- X Window System, 17, 323
- xcursorgen, 503
- xdm, 84, 340, 341
- XFree86, 325, 339
- XFS, 120
- xfs, 344
- xinetd, iv, 75–81
- xinit, 339
- Xlib, 324
- XT, 27

- zip disk, 20

Bibliografia

- [1] D. Giacomini, *Appunti di informatica libera*, <http://www.a2.prosa.it>, 21/07/2002
- [2] S. Piccardi, *GaPiL - Guida alla Programmazione in Linux*, <http://gapil.firenze.linux.it>, 10/12/2002
- [3] M. Stutz, *The Linux Cookbook: Tips and Techniques for Everyday Use*, Jan 2002
- [4] Andrew G. Morgan, *The Linux-PAM System Administrators' Guide*, <http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/pam.html>, Jun 2002
- [5] D. A. Rusling, *The Linux Kernel*, <http://www.linuxhq.com/guides/TLK/tlk.html>, 1999
- [6] T. Aivazian, *Linux Kernel 2.4 Internals*, <http://www.moses.uklinux.net/patches/lki.html>, 07 Aug 2002
- [7] L. Wirzenius, J. Oja, and S. Stafford, *The Linux System Administrators' Guide*, Nov 2001
- [8] O. Kirch and T. Dawson, *The Linux Network Administrator's Guide, Second Edition*, Jul 2000
- [9] G. Mourani, *Securing & Optimizing Linux: The Ultimate Solution*, 07 Aug 2002
- [10] M. Cooper, *Advanced Bash-Scripting Guide*, <http://tldp.org/LDP/abs/html/index.html>, 05 Jan 2003
- [11] W. R. Stevens, *Advanced Programming in the UNIX Environment*, Prentice Hall PTR, 1995
- [12] W. R. Stevens, *TCP/IP Illustrated - The protocols*, Addison Wesley, 1994
- [13] KnowledgeWorks Inc., *Linux tutorial*, <http://www.linuxfree.net>