Mandrake Linux 8.2

Manuale di riferimento

MandrakeSoft

Marzo 2002

http://www.mandrakelinux.com/

Mandrake Linux 8.2 : Manuale di riferimento
MandrakeSoft

Sommario

Lista delle Tabelle

Lista delle Figure

Prefazione

1. Note legali

2. Informazioni su Mandrake Linux

Mandrake Linux è una distribuzione *GNU/Linux* creata dalla **MandrakeSoft** S.A. La **MandrakeSoft** è nata su Internet nel 1998 con l'intento primario di sviluppare un sistema *GNU/Linux* facile da installare e da usare. I due principi guida della **MandrakeSoft** sono la filosofia di sviluppo *open source* e il lavoro di gruppo.

2.1. Contattare la comunità Mandrake

Quelli che seguono sono gli indirizzi Internet di alcune risorse relative a **Mandrake Linux**. Se desiderate avere altre informazioni sulla **MandrakeSoft**, visitate il suo sito web (`http://www.mandrakesoft.com`). Vi segnaliamo inoltre il sito dedicato alla distribuzione **Mandrake Linux** (`http://www.mandrakelinux.com`) e a tutto ciò che la riguarda.

Innanzitutto, la **MandrakeSoft** è orgogliosa di presentare il suo nuovo sistema di aiuto cooperativo: MandrakeExpert (`http://www.mandrakeexpert.com/`) non è un altro di quei siti web in cui alcune persone aiutano chi ha problemi informatici in cambio di un compenso, che deve essere pagato indipendentemente dalla qualità del servizio ricevuto. Questo sito, invece, offre una nuova esperienza che si basa sulla fiducia e sulla soddisfazione di ricompensare le altre persone per i loro contributi.

Oltre al sistema di aiuto vi segnaliamo MandrakeCampus (`http://www.mandrakecampus.com/`), un sito dove la comunità *GNU/Linux* può trovare informazioni e corsi gratuiti su tutti gli argomenti e le tecnologie relative al software libero, e che rappresenta per insegnanti, tutor e studenti un posto dove scambiare reciprocamente le proprie conoscenze.

Esiste inoltre un sito per i "mandrakofili" denominato Mandrake Forum (`http://www.mandrakeforum.com`): è il sito principale per suggerimenti su **Mandrake Linux**, trucchi, voci di corridoio, anticipazioni, notizie semi-ufficiali e altro. È anche l'unico sito web interattivo ospitato dalla **MandrakeSoft**, quindi se avete qualcosa da dirci, o se volete condividere qualcosa con altri utenti, non cercate altrove: questo è il posto giusto per farlo!

Seguendo la filosofia *open source*, **MandrakeSoft** offre molte forme di supporto (`http://www.mandrakelinux.com/en/ffreesup.php3`) per le distribuzioni **Mandrake Linux**. Vi invitiamo, in particolare, a partecipare alle varie mailing list (`http://www.mandrakelinux.com/en/flists.php3`), nelle quali la comunità di **Mandrake Linux** mostra la propria vivacità e disponibilità.

Infine, non dimenticate di visitare MandrakeSecure (`http://www.mandrakesecure.net/`): questo sito raccoglie tutto il materiale relativo alla sicurezza riguardante le distribuzioni **Mandrake Linux**. In particolare, vi troverete avvisi su eventuali bug e problemi di sicurezza, oltre ad articoli riguardanti la sicurezza e la privacy. È una tappa obbligatoria per chiunque amministri un server o sia anche solo interessato alle problematiche della sicurezza.

2.2. Supportare Mandrake

A grande richiesta, la **MandrakeSoft** offre la possibilità agli utenti soddisfatti dei suoi prodotti di fare una donazione (`http://www.mandrakelinux.com/donations/`) per sostenere gli sviluppi futuri del sistema **Mandrake Linux**. Un vostro eventuale contributo aiuterà la **MandrakeSoft** nel compito di creare per i propri utenti una distribuzione ancora migliore, più sicura, più facile, più aggiornata e tradotta in un maggior numero di lingue.

Se avete talento, inoltre, le vostre capacità possono essere utilissime per uno dei tanti compiti necessari alla preparazione di un sistema **Mandrake Linux**:

- Assemblaggio: un sistema *GNU/Linux* è costituito principalmente da programmi presi da Internet; questi programmi devono essere assemblati in modo che possano funzionare correttamente insieme.
- Programmazione: la **MandrakeSoft** supporta in modo diretto moltissimi progetti, cercate quello che più vi interessa e offrite il vostro aiuto allo sviluppatore principale.
- Localizzazione: traduzione delle pagine web, dei programmi e della relativa documentazione.
- Documentazione: infine, ma non ultimo per importanza, il libro che state leggendo in questo momento richiede molto impegno per essere costantemente adeguato alla rapida evoluzione del sistema.

Visitate la pagina delle collaborazioni (`http://www.mandrakesoft.com/labs/`) per avere più informazioni su come partecipare all'evoluzione di **Mandrake Linux**.

Il giorno 3 agosto 2001, dopo essersi affermata come una delle principali aziende di software *GNU/Linux* e Open Source a livello mondiale, la **MandrakeSoft** è stata la prima società in ambito Linux a essere quotata su un mercato di scambio europeo. Sia che siate già un azionista **MandrakeSoft**, o che desideriate diventarlo, le nostre pagine per gli azionisti (`http://www.mandrakesoft.com/company/investors/`) vi forniranno le migliori informazioni finanziarie sulla società.

2.3. Acquistare i prodotti Mandrake

Per la gioia dei fan di **Mandrake Linux** che desiderano approfittare della comodità degli acquisti on-line, la **MandrakeSoft** ora vende i suoi prodotti in tutto il mondo tramite il sito di commercio elettronico MandrakeStore (`http://www.mandrakestore.com/`). Nel sito potete trovare, oltre ai prodotti software **Mandrake Linux** — sistemi operativi e strumenti per la rete (Single Network Firewall), anche offerte speciali per gli abbonamenti, assistenza, software e licenze di terze parti, testi per la formazione, libri su *GNU/Linux* e altri prodotti riguardanti la **MandrakeSoft**.

3. Autori e traduttori

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

- Yves Bailly
- Camille Bégnis
- Marco De Vitis
- Francis Galiègue
- Hinrich Goehlmann

- Carsten Heiming
- Fabian Mandelbaum
- Joël Pomerleau
- Peter Rait
- Roberto Rosselli Del Turco
- Christian Roy
- Stefan Siegel

Inoltre hanno contribuito anche: Philippe Ambon, Jay Beale, Hoyt Duff, Joël Flores-Carpio, Giuseppe Ghibò, Till Kampetter, Alexander Sasha Kirillov, Damien Dams Krotkine, Robert Kulagowski, Kevin Lecouvey, François Pons, Guillaume Poulin, Pascal Pixel Rigaux, John Rye e Laurence Tricon.

4. Strumenti usati per la stesura di questo manuale

Questo manuale è stato impaginato con *DocBook*. Sono stati utilizzati il linguaggio *perl* e *GNU make* per gestire i file relativi. I sorgenti in XML sono stati elaborati con `openjade` e `jadetex`, facendo uso dei fogli di stile di Norman Walsh. Le immagini sono state catturate con `xwd` e *GIMP*, e convertite con `convert` (quest'ultimo programma fa parte del pacchetto *ImageMagick*). I file `PostScript` sono stati generati con il programma `dvips`. Tutti questi programmi sono presenti nella vostra distribuzione **Mandrake Linux**, e sono tutti liberamente distribuibili.

5. Nota del curatore

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

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

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

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

6. Convenzioni usate in questo manuale

6.1. Convenzioni tipografiche

Al fine di rendere immediatamente evidenti e distinte rispetto al testo normale alcune parole di tipo speciale, gli autori di questa documentazione hanno utilizzato diverse forme di evidenziazione del testo. La tabella che segue vi propone un esempio per ciascun tipo o gruppo di parole speciali, con la speciale formattazione grafica e il significato relativi.

Esempio formattato	Significato
inodo	Questo tipo di formattazione ha lo scopo di mettere in evidenza un termine tecnico spiegato nel *Glossario*.
`ls -lta`	Rappresenta comandi o argomenti necessari a questi ultimi. Questa formattazione è applicata a comandi impartiti da linea di comando, alle loro opzioni e ai nomi di file. Si veda anche la sezione riguardo la " *Sintassi dei comandi*, pag. iv ".
ls(1)	Rappresenta il riferimento a una pagina di manuale (pagina "man"). Per richiamare la stessa pagina da linea di comando potete digitare semplicemente `man 1 ls`.

Esempio formattato	Significato
`$ ls *.pid im-wheel.pid`	Indica brani di testo che dovrebbero comparire sul vostro schermo. Include esempi di interazione con il computer, testo generato da programmi, etc.
`localhost`	Si tratta di qualche tipo di dato letterale che, in genere, non rientra in nessuna delle categorie definite in precedenza. Ad esempio, una parola chiave contenuta in un file di configurazione.
`Apache`	Questa formattazione è usata per indicare i nomi delle applicazioni. Non è il caso dell'esempio che abbiamo usato, ma in particolari contesti il nome dell'applicazione e il nome di un comando che ne fa parte potrebbero coincidere, e in questi casi la loro diversa formattazione provvederà a tenerli distinti.
File	Rappresenta le voci di menu e, più in generale, il testo degli elementi delle interfacce grafiche. La lettera sottolineata indica una scorciatoia da tastiera, se presente.
`Bus SCSI`	Indica un componente del computer, o anche il computer stesso.
Le petit chaperon rouge	Rappresenta testo appartenente a un lingua diversa rispetto a quella in cui è scritto il manuale.
Attenzione!	Questa formattazione, come è ovvio, è riservata ad avvertimenti particolari, e ha la funzione di enfatizzare le parole, come se fossero gridate ;-).

Questa icona indica una nota; in genere si tratta di un commento che aggiunge informazioni al contesto.

Questa invece indica un suggerimento; può trattarsi di un consiglio su come eseguire un'azione particolare, o di una caratteristica interessante che vi potrebbe far comodo.

Prestate molta attenzione quando vedete quest'icona: indica sempre informazioni molto importanti su argomenti specifici.

6.2. Convenzioni generiche

6.2.1. Sintassi dei comandi

L'esempio che segue mostra i simboli che useremo in questo manuale per la descrizione degli argomenti di un comando:

```
comando <arg. non letterale> [--opzione={arg1,arg2}] [arg. opzionale ...]
```

Questa è una simbologia standard, e la troverete utilizzata allo stesso modo altrove, come ad esempio nelle pagine `man`.

I caratteri "<" (minore di) e ">" (maggiore di) indicano un argomento **obbligatorio** che non deve essere digitato alla lettera così come riportato, ma che dipende dalle vostre necessità. Ad esempio, `<nome_di_un_file>` si riferisce al nome di un file effettivamente esistente: se il nome in questione è `pippo.txt`, dovreste digitare `pippo.txt`, e non `<pippo.txt>` o `<nome_di_un_file>`.

Le parentesi quadre "[ ]" indicano argomenti opzionali, che possono anche non essere inclusi nella linea di comando.

I puntini di sospensione "..." indicano che in quel punto è possibile inserire un numero qualsiasi di elementi.

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

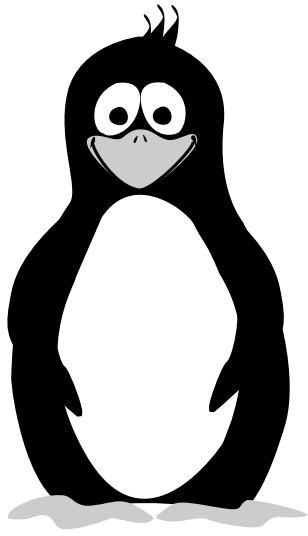
6.2.2. Notazioni particolari

In alcuni casi vi verrà chiesto di premere, ad esempio, la combinazione di tasti Ctrl+R. Questo vuol dire che dovrete premere il tasto R mentre tenete premuto il tasto Ctrl. Lo stesso principio vale per i tasti Alt e Shift.

Per quanto riguarda i menu, invece, selezionare la voce di menu **File**→**Ricarica configurazione utente** (**Ctrl+R**) significa: cliccare sul testo **File** nella barra dei menu (in genere è una barra orizzontale nella parte superiore della finestra) e poi, una volta comparso il relativo menu verticale, cliccare sulla voce **Ricarica configurazione utente**. Questa notazione, inoltre, vi informa del fatto che per ottenere lo stesso risultato potete usare la combinazione di tasti Ctrl+R, come descritto in precedenza.

6.2.3. Utenti generici del sistema

Tutte le volte che è stato possibile, abbiamo usato per i nostri esempi due utenti generici:

Maria Pinguino		Rappresenta l'utente che viene creato al momento dell'installazione
Pietro Pinguino		Rappresenta un utente creato successivamente dall'amministratore del sistema

7. Introduzione

Benvenuti, e grazie per aver scelto **Mandrake Linux**! Questo libro è rivolto a quelle persone che desiderano immergersi a fondo nel proprio sistema *GNU/Linux*, sfruttandone le enormi potenzialità. Il libro è diviso in tre parti:

- *Introduzione a Linux*: in questa prima parte vi presenteremo la linea di comando, con le sue utili applicazioni, e i principi basilari dell'elaborazione di testi, fondamentale su *GNU/Linux*.

  Inizieremo con un capitolo che servirà da introduzione al mondo di *UNIX* e, in particolare, a quello di *GNU/Linux*. È necessario comprendere bene i concetti presentati in questo capitolo prima di affrontare il successivo, dedicato alla linea di comando, nel quale vi illustreremo l'uso dei programmi più comuni per la gestione dei file e alcune utili caratteristiche della *shell*.

  Un altro capitolo è dedicato all'elaborazione di testi. Dato che la maggior parte dei file di configurazione in *UNIX* sono file di testo, è probabile che dovrete modificarli con un ***editor di testo***. Imparerete a usare due degli editor di testo più famosi nel mondo di *UNIX* e *GNU/Linux*: il poderoso *Emacs* e il moderno (!) *Vi*.

  Arrivati a questo punto, dovreste essere in grado di compiere semplici operazioni di manutenzione ordinaria del vostro sistema. I due capitoli successivi trattano di applicazioni pratiche della linea di comando, e del controllo dei processi in generale.

- *Linux in profondità*: qui troverete alcuni dettagli sull'architettura del filesystem e del kernel di *Linux*.

  Nel primo capitolo vedrete come è organizzata la struttura ad albero dei file. I sistemi *UNIX* tendono ad avere dimensioni considerevoli, ma ogni file ha il proprio posto in una particolare directory; dopo aver letto questo capitolo saprete dove cercare un file in base al suo ruolo all'interno del sistema.

  Il capitolo successivo parla del ***filesystem*** e dei ***punti di mount***. Qui imparerete cosa significano questi termini e vedrete un esempio pratico.

  Proseguiremo con un capitolo interamente dedicato ai filesystem di *GNU/Linux*: dopo avervi presentato i filesystem esistenti, vi insegneremo qualcosa in più sui tipi di file e su alcuni altri concetti che potrebbero risultarvi nuovi. Un altro capitolo ancora vi presenterà uno speciale filesystem di *GNU/Linux*: `/proc`.

  Imparerete poi a conoscere la procedura di avvio di **Mandrake Linux**, e scoprirete come utilizzarla al meglio.

- *Uso avanzato*: infine troverete dei capitoli rivolti agli utenti più esperti, tra cui un utilissimo capitolo sulla risoluzione dei problemi più comuni.

  Innanzitutto due capitoli dedicati alla stampa e alla gestione delle stampanti.

  Proseguiremo poi parlando dei livelli di sicurezza disponibili in **Mandrake Linux**, e a seguire due capitoli per apprendisti stregoni: come compilare e installare un nuovo kernel e un programma distribuito in forma di sorgenti.

  Il manuale si chiude con una guida alla risoluzione dei problemi più comuni; se doveste avere problemi con il vostro sistema, questo è un buon punto di partenza.

Buon divertimento!

I. Introduzione a Linux

Capitolo 1. Concetti base di UNIX

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

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

1.1. Utenti e gruppi

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

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

- il "vero nome" dell'utente (qualunque nome vogliate, in realtà);
- un nome di ***login***;
- una ***password*** (ne avete scelta una, vero? **:-)**).

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

Un'altra operazione effettuata quando si aggiunge un nuovo utente al sistema è la creazione di un gruppo. Come opzione predefinita, il programma di installazione crea un gruppo per ciascun utente. Come vedremo, i gruppi sono utili quando si devono condividere file fra un certo numero di utenti. Un gruppo può contenere tutti gli utenti che volete, ed è una caratteristica utilizzata spesso sui sistemi più estesi per separare i diversi tipi di utenti. In una università, ad esempio, può esserci un gruppo per ciascuna facoltà, uno per i docenti, e così via. Ma è vero anche il contrario: un utente può far parte di diversi gruppi allo stesso tempo, fino a un massimo di 32. Un professore di matematica, ad esempio, può essere membro del gruppo dei docenti, e, contemporaneamente, di quello dei suoi amati studenti di matematica.

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

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

Figura 1-1. Schermata di login in modalità grafica

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

Se invece il vostro sistema si avvia in modalità console (solo testo), la schermata sarà simile a questa (Figura 1-2):

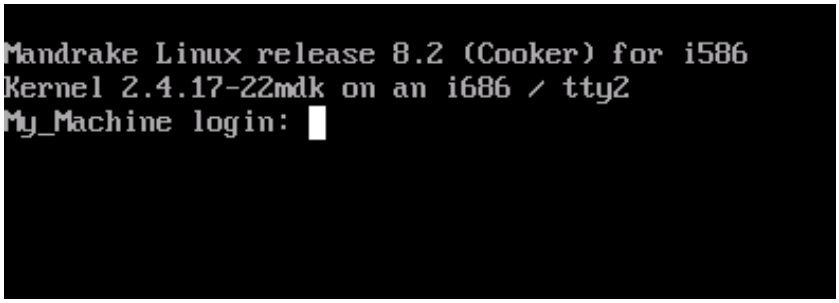


Figura 1-2. Schermata di login in modalità console

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

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

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

Se vi collegate spesso come `root` è facile fare degli errori che possono rendere il sistema inutilizzabile: basta un solo sbaglio perché ciò avvenga. In particolare, se non avete protetto l'account `root` con una password, qualsiasi utente potrà utilizzarlo e modificare qualsiasi parte del vostro sistema, persino altri sistemi operativi presenti sulla stessa macchina. Non utilizzare una password per `root` è, come avrete capito, una pessima idea.

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

1.2. Principi di base riguardo i file

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

La differenza principale è una diretta conseguenza del fatto che *`GNU/Linux`* è un sistema multiutente: ogni file è proprietà esclusiva di un utente e di un gruppo. Quando abbiamo accennato agli utenti, poco fa, non abbiamo citato un fatto importante: ogni utente ha una sua cartella (nota come la sua ***directory home***), ed è il proprietario di questa directory e di tutti i file che lui stesso creerà in futuro. Solo lui, nessun altro.

Tutto questo non avrebbe grande utilità, tuttavia, se si limitasse al solo concetto di proprietà dei file. Infatti c'è dell'altro: in quanto proprietario di un file, l'utente può impostare dei **permessi** che riguardano il file stesso. Questi permessi sono distinti in tre diverse categorie: permessi del proprietario del file, di qualsiasi utente membro del gruppo associato al file (il cosiddetto ***owner group***) escluso l'utente che ne è il proprietario, e gli altri, ovvero qualsiasi utente che non rientri nei primi due casi.

Ci sono tre diversi permessi:

1. Permesso di lettura *Read* (`r`): per un file, questo permesso consente di leggerne il contenuto; per una directory, di elencarne il contenuto (cioè i file che essa contiene).
2. Permesso di scrittura *Write* (`w`): per un file, questo permesso consente di modificarne il contenuto. Per una directory, permette a un utente di modificare e cancellare i file che essa contiene, anche se non è il proprietario di quei file.
3. Permesso di esecuzione *eXecute* (`x`): per un file, questo permesso consente di lanciarlo in esecuzione; di conseguenza, normalmente solo i file eseguibili dovrebbero avere questo permesso. Per una directory, questo permesso consente di ***attraversarla***, cioè di spostarsi dentro o attraverso di essa. Notate che questa caratteristica è indipendente dal permesso di lettura: potrebbe capitare che abbiate il permesso per attraversare una directory, ma che non possiate leggerne il contenuto!

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

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

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

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

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

Diamo un'occhiata più da vicino ai permessi associati a ciascuno di questi file: prima di tutto, saltiamo il primo carattere relativo al tipo di file; per il file `un_file` abbiamo i seguenti diritti: `rw-r------`. Questi caratteri vanno interpretati come segue:

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

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

- pietro, come proprietario della directory, può elencare i file contenuti in essa (`r`), può aggiungere o rimuovere file al suo interno (`w`), e può attraversarla (`x`);

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

Ricordate, c'è **un'eccezione** a questa regola: l'account `root` può modificare gli attributi (permessi, proprietario e gruppo di appartenenza) di tutti i file, anche se non ne è il proprietario. Questo significa che può anche diventarne proprietario. `root` può leggere i file per cui non ha il permesso di lettura, attraversare le directory che normalmente non potrebbe attraversare, e così via. E se gli manca un permesso, non deve far altro che assegnarselo...

Infine, vale la pena di notare le differenze relative ai nomi dei file nel mondo *UNIX* rispetto a *Windows*; sotto *UNIX*, infatti, questi sono molto meno limitati e molto più flessibili rispetto a quanto avviene sotto *Windows*:

- il nome di un file può contenere qualsiasi carattere (ad eccezione del carattere corrispondente al codice ASCII 0, che indica la fine di una stringa di testo, e del carattere **/**, che costituisce il separatore di directory), anche quelli non stampabili. Ma soprattutto, *UNIX* distingue tra maiuscole e minuscole: i file `readme` e `Readme` sono diversi, perché `r` e `R` sono due **caratteri** del tutto diversi;
- Come forse avrete notato, un nome di file non deve necessariamente avere un'estensione, a meno che voi non preferiate così. Su *GNU/Linux*, come pure su ogni altro sistema operativo, le estensioni non identificano realmente il contenuto di un file. Tali "estensioni", tuttavia, sono molto utili. Il carattere "punto" (`.`) in *UNIX* non è che un carattere come tutti gli altri; è importante sapere, comunque, che in *UNIX* i nomi di file che cominciano con un punto sono "file nascosti".

1.3. Processi

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

La differenza più importante è, ancora una volta, legata al concetto di utente: ogni processo viene eseguito con i diritti dell'utente che lo ha lanciato. Il sistema identifica i processi in modo univoco utilizzando un numero per ciascun processo, che viene chiamato *process ID*, o PID. Grazie al PID il sistema sa, fra le altre cose, chi (ovvero quale utente) ha lanciato il processo in esecuzione: a questo punto deve soltanto controllare che quanto richiesto dal processo sia del tutto "legale". Quindi, tornando all'esempio del file `un_file`, un processo lanciato dall'utente pietro potrà aprire questo file in modalità ***read-only*** (sola lettura), ma non in modalità ***read-write*** (lettura e scrittura), perché i permessi associati al file lo impediscono. Ancora una volta, `root` fa eccezione a questa regola...

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

Esistono dei programmi anti-virus anche per *GNU/Linux*, ma servono solo per i file *DOS*/*Windows*... Questo perché si trovano sempre più spesso file server *GNU/Linux* che operano con client *Windows* tramite il software *Samba*.

Sotto *Linux* il controllo dei processi è semplice; un modo per controllarli consiste nell'usare i segnali: grazie ad essi, infatti, potete sospendere o terminare un processo, ad esempio, semplicemente inviandogli il segnale corrispondente. Questo vale solo per i processi che voi stessi avete lanciato, tuttavia, e non per quelli di un altro utente (l'unica eccezione a questa regola è costituita, di nuovo, da `root`). Nel capitolo *Controllo dei processi*, pag. 33 imparerete come ottenere il PID di un processo e inviargli dei segnali.

1.4. Breve introduzione alla linea di comando

La linea di comando è il modo più diretto per impartire dei comandi al sistema. Se utilizzate la linea di comando di *GNU/Linux* vi accorgerete presto che è molto più potente e flessibile di qualsiasi prompt dei comandi che possiate aver utilizzato in passato. Questo perché essa offre accesso diretto non solo a tutte le applicazioni

X, ma anche a migliaia di programmi che operano in modalità console (invece che in modalità grafica) e che non hanno un equivalente grafico, programmi che spesso dispongono di numerose opzioni, combinabili fra di loro, che non sarebbero difficilmente accessibili sotto forma di menu o pulsanti.

Ma, ammettiamolo, per iniziare ci vuole un po' di aiuto. Questo è lo scopo del presente capitolo. La prima cosa da fare, se utilizzate *KDE*, è lanciare un emulatore di terminale. C'è un'icona che lo indica chiaramente nel pannello (Figura 1-3).

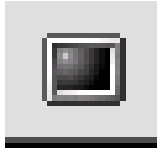


Figura 1-3. L'icona del terminale nel pannello di KDE

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

```
[maria@localhost maria]$
```

Questo presuppone che il vostro nome utente sia maria e che il nome del vostro computer sia `localhost` (il che in genere corrisponde alla realtà, se se la vostra macchina non fa parte di una rete). Tutto ciò che compare dopo il prompt è quello che dovrete digitare. Notate che quando siete `root`, il simbolo `$` del prompt diventa un `#` (questo è vero soltanto nella configurazione standard del sistema, dato che con *GNU/Linux* potete personalizzare questi dettagli). Il comando per "diventare" `root` dopo aver lanciato una shell come utente normale è `su`:

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

In tutto il resto del manuale, il prompt verrà rappresentato dal simbolo `$`, sia che siate `root`, sia che siate un utente comune. Vi diremo quando dovrete essere `root`, quindi ricordatevi di usare il comando `su` come appena visto. Un simbolo `#` all'inizio di una riga di codice rappresenta invece un commento.

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

```
$ pwd
/home/maria
```

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

1.4.1. cd: Change Directory

Il comando `cd` è proprio come quello del *DOS*, solo con qualcosa in più. Fa proprio quello che dice il suo nome: cambia la directory di lavoro. Potete anche usare `.` e `..`, che indicano, rispettivamente, la directory attuale e la directory immediatamente superiore (*parent directory*). Digitando `cd` senza argomenti, ritornerete alla vostra home directory. Digitando `cd -` tornerete indietro all'ultima directory che avete attraversato. Infine, potete specificare la home directory di un utente, ad esempio pietro, digitando `cd ~pietro` (~ da solo indica la vostra home directory). Notate che, come utente normale, di solito non potete andare nelle home directory di altri utenti, a meno che questi non lo autorizzino o che questa non sia la regola generale sul vostro sistema; oppure, a meno che non siate `root`; quindi diventiamo `root` e facciamo un po' di pratica:

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

Adesso torniamo a essere utenti normali.

1.4.2. Alcune variabili di ambiente e il comando echo

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

1. `HOME`: questa variabile contiene una stringa che rappresenta il percorso della vostra home directory.
2. `PATH`: questa variabile contiene la lista di tutti i percorsi in cui la shell cerca gli eseguibili quando digitate un comando. Notate che, a differenza del *DOS*, come impostazione predefinita una *shell* **non** cercherà i programmi nella directory corrente!
3. `USERNAME`: questa variabile contiene il vostro nome di login.
4. `UID` Contiene il vostro user ID.
5. `PS1`: contiene la definizione del vostro prompt. Spesso è una combinazione di sequenze speciali, per avere maggiori informazioni potete leggere la ***pagina di manuale*** bash(1).

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

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

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

1.4.3. cat: visualizza il contenuto di uno o più file sullo schermo

Non c'è molto da aggiungere, il comando fa proprio questo: visualizza sullo standard output, normalmente lo schermo, il contenuto di uno o più file:

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

1.4.4. less: un visualizzatore a pagine

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

```
less /etc/termcap
```

Per scorrere il file, usate le frecce in su e in giù. Premete **q** per uscire. `less`, tuttavia, può fare molto più di questo: digitate `h` per l'aiuto, e guardate. Lo scopo di questo paragrafo, in ogni caso, era solo quello di permettervi di leggere file lunghi, e l'abbiamo raggiunto **:-)**

1.4.5. ls: elencare file

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

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

Se non si specifica un file o una directory, `ls` visualizza la lista dei file nella directory corrente. Le sue opzioni sono moltissime, ne descriveremo soltanto alcune:

- `-a`: elenca tutti i file, inclusi i ***file nascosti*** (in *UNIX* i file nascosti sono quelli il cui nome comincia con `.`); l'opzione `-A` elenca "quasi" tutti i file, il che significa: tutti i file che l'opzione `-a` visualizzerebbe tranne "." e "..";
- `-R`: elenca in modo ricorsivo, cioè elenca anche tutti i file contenuti in tutte le sottodirectory della directory indicata sulla linea di comando;
- `-s`: visualizza la dimensione in kilobyte accanto a ciascun file;

- `-l`: visualizza informazioni aggiuntive sui file;
- `-i`: visualizza il numero di inode (il numero identificativo di un file nel filesystem, consultate in proposito il capitolo Il filesystem di Linux) accanto a ciascun file;
- `-d`: visualizza le directory specificate sulla linea di comando come file normali, invece di elencare il loro contenuto.

Alcuni esempi:

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

1.4.6. Scorciatoie utili da tastiera

Ci sono molte utili combinazioni di tasti che vi possono far risparmiare parecchio lavoro, e in questo paragrafo vedremo alcune tra le più comuni. Questa sezione presuppone che stiate utilizzando la *shell* predefinita fornita con **Mandrake Linux**, *bash*, ma le scorciatoie dovrebbero funzionare anche in altre *shell*.

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

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

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

Infine, `Ctrl+s` e `Ctrl+q`: questi due comandi servono rispettivamente a sospendere e a riattivare il flusso di caratteri su un terminale. Vengono usati molto raramente, ma potrebbe accadere che digitiate `Ctrl+s` per errore (dopo tutto, **s** e **d** sono molto vicini sulla tastiera). Per cui, se premete dei tasti ma non vedete apparire niente sullo schermo, provate a premere `Ctrl+q`, e attenzione: tutti i caratteri che avete digitato tra `Ctrl+s` e `Ctrl+q` verranno scritti sullo schermo tutti in una volta.

Capitolo 2. Introduzione alla linea di comando

Nel capitolo *Concetti base di UNIX*, pag. 1 vi abbiamo spiegato come aprire una *shell*. In questo capitolo vi mostreremo come utilizzarla.

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

2.1. Comandi per la gestione dei file

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

2.1.1. mkdir, touch: creazione di directory e file vuoti

Il comando `mkdir` (*MaKe DIRectory*) viene utilizzato per creare le directory. La sua sintassi è semplice:

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

Solo un'opzione merita di essere commentata: l'opzione `-p`. Questa ha due usi particolari:

1. creerà le directory superiori se in precedenza non esistevano. Se non la si specifica, `mkdir` non avrà successo e si lamenterà per il fatto che le directory indicate non esistono;
2. terminerà senza messaggi di errore se la directory che intendiamo creare esiste già. Come prima, se si omette l'opzione `-p`, `mkdir` non avrà successo, e si lamenterà per il fatto che tale directory è già presente.

Ecco alcuni esempi:

- `mkdir foo`: crea la directory `foo` nella directory corrente;
- `mkdir -p images/misc docs`: crea la directory `misc` nella directory `images` creando per prima quest'ultima, se non esiste già (`-p`); inoltre crea una directory `docs` nella directory corrente.

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

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

Quindi lanciando il comando

```
touch file1 images/file2
```

verranno creati un file vuoto chiamato `file1` nella directory corrente, e un file vuoto `file2` nella directory `images`.

2.1.2. rm: cancellazione di file o directory

Il comando `rm` (*ReMove*) sostituisce due comandi *DOS*, `del` e `deltree`, rispetto ai quali dispone di più opzioni. Ecco la sua sintassi:

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

Le sue opzioni includono:

- `-r`, o `-R`: cancella ricorsivamente. Questa opzione è **obbligatoria** per cancellare una directory, vuota o meno. Per cancellare directory vuote, comunque, si può usare anche il comando `rmdir`.
- `-i`: richiede una conferma prima di ogni cancellazione. Si noti che, come opzione predefinita, per motivi di sicurezza in **Mandrake Linux** `rm` non è che un ***alias*** per `rm -i`; lo stesso vale per i comandi `cp` e `mv`.

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

L'utilità di questi alias varia a seconda dell'utente (sono comunque molto utili per i principianti). Se volete rimuoverli, è sufficiente modificare il file `.bashrc` aggiungendo questa riga: `unalias rm cp mv`.

- `-f`: l'opposto dell'opzione `-i`, forza la cancellazione di file e directory, anche se l'utente non ha i diritti di scrittura sui file[2].

Alcuni esempi:

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

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

2.1.3. mv: spostare o rinominare dei file

La sintassi del comando `mv` (*MoVe*) è la seguente:

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

Alcune opzioni:

- `-f`: forza lo spostamento dei file – non avvisando se qualche file preesistente viene sovrascritto dall'operazione.
- `-i`: l'opposto – richiede all'utente una conferma prima di sovrascrivere un file esistente.
- `-v`: modalità ***prolissa*** (ingl. *verbose*), riporta tutte le modifiche effettuate.

Alcuni esempi:

- `mv -i /tmp/pics/*.png .`: sposta tutti i file nella directory `/tmp/pics/` il cui nome finisce con `.png` nella directory corrente (`.`), richiedendo una conferma prima di sovrascrivere qualsiasi file.
- `mv foo bar`: rinomina il file `foo` come `bar`. Se esisteva già una directory `bar`, l'effetto di questo comando sarebbe stato di spostare l'intera directory `foo` (la directory stessa più tutti i file e le directory che contiene, ricorsivamente) all'interno della directory `bar`.
- `mv -vf file* images/ trash/`: sposta dalla directory corrente, senza chiedere conferma, tutti i file il cui nome comincia per `file` e l'intera directory `images/` nella directory `trash/`, e mostra tutte le operazioni effettuate.

2.1.4. cp: copiare file e directory

Il comando `cp` (*CoPy*) corrisponde a due comandi *`DOS`*, `copy` e `xcopy`, rispetto ai quali dispone di più opzioni. Questa è la sua sintassi:

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

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

- `-R`: copia ricorsiva; **obbligatorio** se si vuole copiare una directory, anche se vuota.
- `-i`: richiede una conferma prima di sovrascrivere qualsiasi file preesistente.
- `-f`: l'opposto dell'opzione `-i`, sostituisce qualsiasi file esistente senza chiedere conferma.

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

- -v: modalità prolissa, visualizza tutte le azioni compiute dal comando cp.

Alcuni esempi:

- cp -i /tmp/images/* images/: copia tutti i file nella directory /tmp/images alla directory images/ che si trova all'interno della directory attuale, richiedendo una conferma se un file sta per essere sovrascritto.
- cp -vR docs/ /shared/mp3s/* mystuff/: copia l'intera directory docs più tutti i file della directory /shared/mp3s nella directory mystuff.
- cp foo bar: crea una copia del file foo con il nome bar nella directory corrente.

2.2. Gestione degli attributi dei file

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

2.2.1. chown, chgrp: cambiare il proprietario o il gruppo di uno o più file

La sintassi del comando chown (*CHange OWNer*) è la seguente:

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

Le opzioni includono:

- -R: modalità ricorsiva; per cambiare il proprietario di tutti i file e le directory presenti in una directory.
- -v: modalità prolissa; descrive tutte le operazioni compiute dal comando chown; riporta quali file hanno cambiato proprietario e quali sono rimasti invariati.
- -c: simile all'opzione -v, ma riporta solo quali file sono cambiati.

Alcuni esempi:

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

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

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

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

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

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

2.2.2. chmod: modificare i permessi di file e directory

Il comando chmod (*CHange MODe*) ha una sintassi molto particolare. La sintassi generica è:

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

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

1. in ottale: in questo caso i permessi dell'utente proprietario sono indicati con delle cifre che seguono lo schema <x>00, dove <x> corrisponde al permesso assegnato: 4 per il permesso di lettura, 2 per quello di scrittura e 1 per quello di esecuzione; allo stesso modo i permessi del gruppo proprietario vengono

espressi come `<x>0` e i permessi per gli altri utenti ("others") nella forma `<x>`. Pertanto tutto quello che dovete fare è sommare insieme i permessi per ottenere la cifra corretta. I permessi `rwxr-xr--`, ad esempio, corrispondono a 400+200+100 (diritti del proprietario, `rwx`) +40+10 (diritti del gruppo, `r-x`) +4 (diritti degli altri, `r--`) = 754; in questo modo, i permessi vengono espressi in forma assoluta. Questo significa che i permessi assegnati in precedenza vengono sovrascritti;

2. con delle espressioni: in questo caso i permessi vengono indicati da una sequenza di espressioni separate da virgole; un'espressione, quindi, ha il seguente formato: `[categoria]<+|-|=><permessi>`.

   La categoria può essere indicata da una o più lettere:

   - `u` (*User*, permessi del proprietario);
   - `g` (*Group*, permessi per il gruppo proprietario);
   - `o` (*Others*, permessi per "gli altri").

   Se non viene specificata una categoria, il cambiamento si applica a tutte. Un segno `+` assegna un permesso, un segno `-` lo rimuove e un segno `=` lo assegna. Per finire, il permesso è uno o più dei seguenti:

   - `r` (*Read*);
   - `w` (*Write*);
   - `x` (*eXecute*).

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

- `-R`: cambia i permessi in modo ricorsivo.
- `-v`: modalità prolissa, riporta le operazioni effettuate per ciascun file.
- `-c`: simile all'opzione `-v`, ma mostra solo i file per cui sono stati modificati i permessi.

Esempi:

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

2.3. I caratteri speciali (meta-caratteri) e le espressioni regolari nella shell

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

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

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

prima di tutto la linea di comando viene suddivisa in parole (`ls` e `*.txt` in questo esempio). Quando la shell trova un carattere `*` in una parola, interpreterà la parola intera come un modello di meta-espansione e la sostituirà con i nomi di tutti i file che corrispondono al modello. Pertanto la linea di comando che viene

effettivamente eseguita sarà `ls readme.txt recipe.txt`, che darà i risultati che abbiamo mostrato. Ci sono altri caratteri che vengono interpretati in modo simile dalla shell:

- `?`: sostituisce un carattere qualsiasi, ma soltanto uno;
- `[...]` :stabilisce una corrispondenza con qualsiasi carattere si trovi fra le parentesi quadre; in questo caso è possibile definire un ambito di caratteri (ad esempio `1-9`) o dei ***valori discreti***, o addirittura entrambi. Ad esempio: `[a-zBE5-7]` comprende tutti i caratteri da `a` a `z`, una `B`, una `E`, un `5`, un `6` o un `7`;
- `[!...]`: stabilisce una corrispondenza con qualsiasi carattere che **non** sia indicato tra le parentesi quadre. `[!a-z]`, ad esempio, verrà sostituito da qualsiasi carattere che non sia una lettera minuscola[3];
- `{c1,c2}`: viene sostituito da `c1` o `c2`, dove `c1` e `c2` sono a loro volta delle espressioni jolly, il che significa che potete scrivere `{[0-9]*,[acr]}`, ad esempio.

Ecco alcuni esempi con il relativo significato:

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

2.4. La redirezione e le pipe

2.4.1. Ancora a proposito dei processi

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

2.4.2. La redirezione

Immaginate di voler visualizzare una lista di tutti i file che terminano con `.png` [4] nella directory `images`. Visto che l'elenco è molto lungo, volete salvarlo in un file per consultarlo con comodo in seguito. Per farlo potete usare questo comando:

```
$ ls images/*.png 1>lista_file
```

Questo significa che lo standard output di questo comando (`1`) viene rediretto (`>`) verso il file chiamato `lista_file`. Il carattere `>` è l'operatore di redirezione dell'output. Se il file specificato nella redirezione non esiste, viene creato al momento, mentre se esiste il suo contenuto viene sovrascritto. Se in un comando di redirezione che usa questo operatore non viene specificato un descrittore, la shell assume che vogliate utilizzare lo standard output; volendo quindi potreste scrivere, più semplicemente:

```
$ ls images/*.png >lista_file
```

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

3. Attenzione! Se questo è vero sotto *GNU/Linux*, potrebbe non esserlo per altri sistemi operativi basati su *UNIX*. Questa caratteristica dipende dall'**ordine di raggruppamento** dei caratteri. Su alcuni sistemi `[a-z]` corrisponde a a, A, b, B, (...) , z. E non abbiamo neanche menzionato il fatto che alcune lingue hanno dei caratteri accentati...
4. Potreste pensare che sia strano parlare di "file il cui nome termina per `.png` " piuttosto che "immagini PNG". Tuttavia, ricordiamolo ancora, in un sistema *UNIX* i file non sono necessariamente identificati dalla loro estensione: in questo ambiente le estensioni non definiscono il tipo di file. Un file il cui nome termina per `.png` potrebbe tranquillamente essere un'immagine JPEG, un programma, un file di testo o di qualsiasi altro tipo. Notate che lo stesso vale per *Windows*

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

```
wc -l 0<lista_file
```

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

```
wc -l <lista_file
```

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

```
sed -e 's/\.png$//g' <lista_file >la_lista
```

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

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

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

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

2.4.3. Le pipe

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

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

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

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

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

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

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

L'uso delle pipe e della redirezione non è limitato ai file di testo, comprensibili per gli esseri umani. Ad esempio, lanciando questo comando da un *`terminale`*:

```
xwd -root | convert - ~/il_mio_desktop.png
```

viene creata un'istantanea del vostro desktop che viene salvata nel file `il_mio_desktop.png` [5] nella vostra directory personale.

5. E in questo caso sarà veramente un'immagine PNG **:-)** (supponendo che voi abbiate installato il pacchetto " `ImageMagick` "...).

2.5. Completamento automatico

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

2.5.1. Esempio

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

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

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

```
$ less file_
```

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

```
less file_w<TAB>
```

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

```
less file_dal_nome_troppo_lungo_per_digitarlo
```

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

2.5.2. Altri metodi di completamento

Il tasto `TAB` non è il solo modo di attivare il completamento, anche se è il più utilizzato. In linea di massima, la parola da completare sarà il nome di un comando se è la prima parola della linea di comando (`nsl<TAB>` verrà espanso in `nslookup`), e un nome di file per le altre parole, a meno che la parola non sia preceduta da un carattere "magico" da scegliere tra `~`, `@` o `$`; in questo caso, la shell cercherà di completare la parola assumendo che sia, rispettivamente, il nome di un utente, il nome di una macchina o di una variabile di ambiente [6]. Esiste anche un carattere magico per completare il nome di un comando (`!`) o il nome di un file (`/`).

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

Come conseguenza, un sistema per visualizzare tutte le variabili di ambiente definite è digitare `Ctrl+x $` su una linea vuota. Un altro esempio: se volete visualizzare la pagina di manuale del comando `nslookup`, potete digitare `man nsl` seguito da `Esc-!`, e la shell completerà automaticamente la linea di comando in `man nslookup`.

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

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

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

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

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

Questo comando potrebbe richiedere anche diversi minuti prima che l'operazione sia terminata. A questo punto avete due possibilità: la prima, più distruttiva, consiste nell'arrestare l'esecuzione (kill) e ripetere il comando quando avrete il tempo di aspettare; per farlo dovreste digitare `Ctrl+c`: questo comando vi riporta al prompt. Ma aspettate un momento, non fatelo! Continuate a leggere.

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

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

e vi ritroverete di nuovo al prompt. A questo punto il processo è in standby, in attesa che voi lo riattiviate (come risulta dalla dicitura `Stopped`). Ed è ciò che vorrete fare, ma in modo tale che il processo riparta in background. Digitate `bg` (che sta per *BackGround*) per ottenere questo risultato:

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

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

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

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

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

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

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

2.7. Conclusione

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

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

Anche nel caso in cui esistano applicazioni grafiche con funzionalità equivalente, molto spesso si tratta di interfacce grafiche che si basano su questi stessi programmi. I programmi da linea di comando, inoltre, presentano il vantaggio di poter operare in modo non interattivo: potete cominciare a masterizzare un CD e poi uscire dal sistema con la certezza che il processo di masterizzazione verrà portato a termine (si veda la pagina di manuale nohup(1)).

Capitolo 3. Elaborazione testi: Emacs e Vi

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

3.1. Emacs

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

3.1.1. Una breve introduzione

Avviare *Emacs* è relativamente semplice:

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

Emacs aprirà in un buffer ogni file immesso come argomento, con un massimo di due buffer visibili contemporaneamente; se non specificate alcun file, verrà aperto il buffer `*scratch*`. Se state usando *X* avrete a disposizione anche dei menu, ma noi qui ci occuperemo delle operazioni da tastiera.

3.1.2. I primi passi

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

```
$ emacs file1 file2
```

per ottenere la finestra mostrata in Figura 3-1.

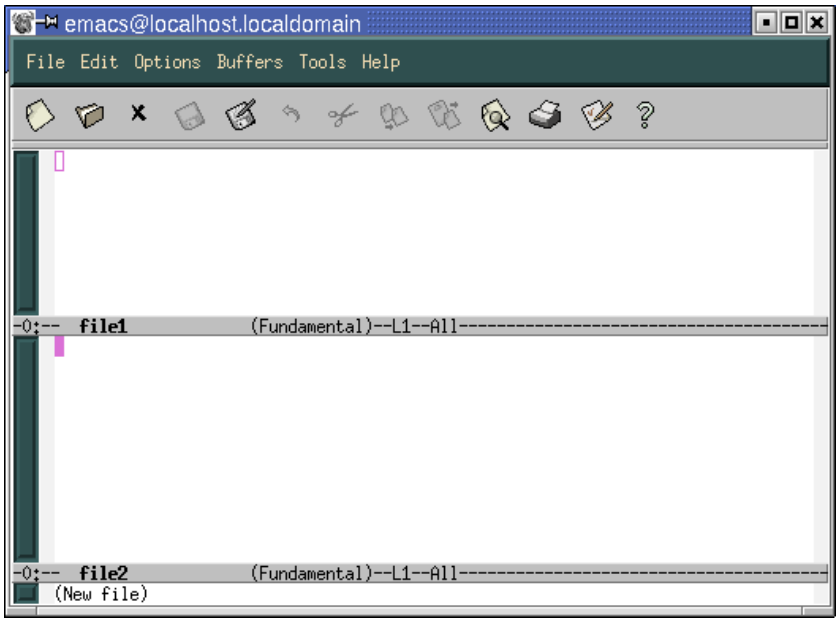


Figura 3-1. `Emacs`**, modifica simultanea di due file**

Come potete vedere, sono stati creati due buffer: uno per ciascun file. Ne è visibile anche un terzo, in fondo allo schermo (dove è scritto `(New file)`): questo è il mini-buffer; non potete andare di vostra iniziativa in

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

questo buffer, dovete essere invitati a farlo da *Emacs* stesso durante le operazioni interattive. Per cambiare buffer digitate `Ctrl+x o`. Potete scrivere del testo come in un editor "normale", e cancellarlo con i tasti `Canc` o `Backspace`.

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

Quando volete salvare i cambiamenti effettuati in un file digitate `Ctrl+x Ctrl+s`, se invece volete salvare il contenuto del buffer in un altro file digitate `Ctrl+x Ctrl+w`, e *Emacs* vi chiederà il nome del file in cui salvarlo. Per inserirlo potete aiutarvi con il ***completamento automatico***.

3.1.3. La gestione dei buffer

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

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

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

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

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

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

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

3.1.4. Copia, taglia, incolla, cerca

Supponiamo di essere nella situazione di Figura 3-2.

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

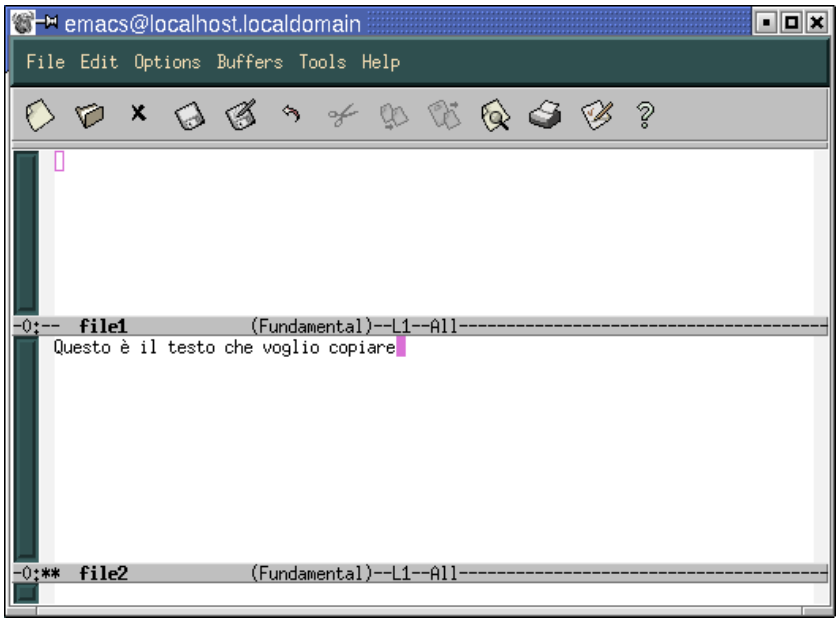


Figura 3-2. `Emacs`**, prima di copiare il blocco di testo**

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

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

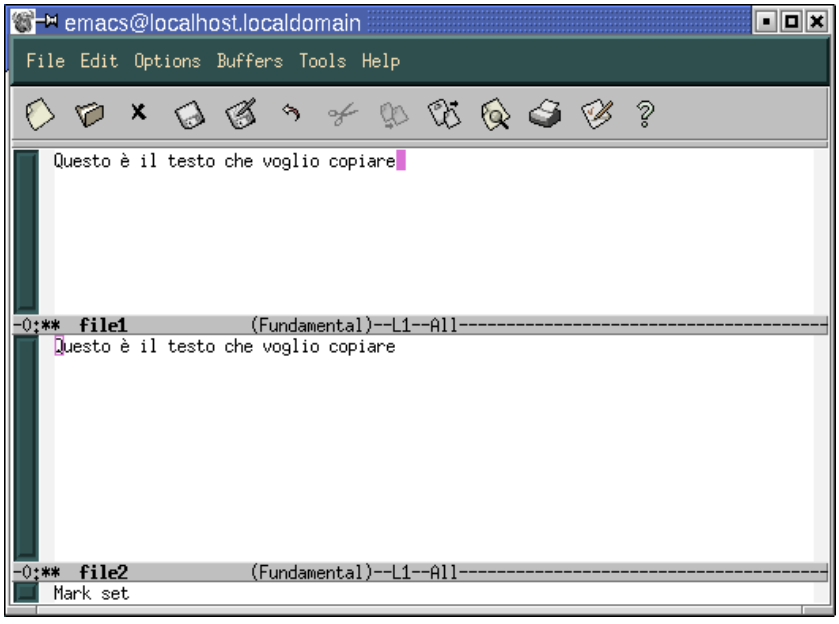


Figura 3-3. `Emacs`**, dopo la copia del blocco di testo**

In effetti, ciò che avete appena fatto è stato copiare il testo nel ***"kill ring"*** di *Emacs*: questo "kill ring" contiene tutti i blocchi di testo copiati o tagliati da quando *Emacs* è stato avviato. **Qualsiasi** blocco, appena copiato o

tagliato, viene inserito in cima al kill ring. La combinazione `Ctrl+y` incolla solo il blocco che si trova in cima: se volete accedere agli altri blocchi premete `Ctrl+y` e poi `Alt+y` finché non raggiungete il testo desiderato.

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

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

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

3.1.5. Uscire da Emacs

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

3.2. Vi: il capostipite

Vi è stato il primo editor a tutto schermo. Rappresenta uno dei principali punti criticati dai detrattori di *UNIX*, ma è allo stesso tempo uno dei principali argomenti dei suoi difensori: sebbene sia complicato da imparare, è anche uno strumento estremamente potente una volta che ci si abitua a usarlo. Con poche pressioni dei tasti un utente di *Vi* può smuovere le montagne e, a parte *Emacs*, di pochi editor di testo può essere detta la stessa cosa.

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

3.2.1. Modalità inserimento, modalità comandi, modalità `ex`...

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

```
$ vi file1 file2
```

A questo punto vi troverete di fronte a una finestra simile a quella in Figura 3-4.

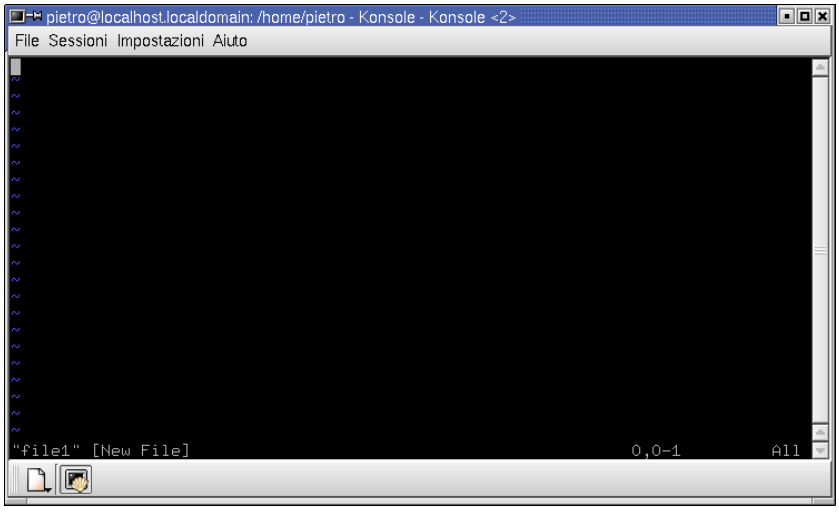


Figura 3-4. Situazione iniziale in `vim`

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

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

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

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

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

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

3.2.2. La gestione dei buffer

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

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

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

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

```
Non salvato dopo modifica (usa ! per eseguire comunque)
```

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

3.2.3. Elaborazione del testo e comandi di movimento

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

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

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

- **h**, **j**, **k**, **l**: rispettivamente un carattere a sinistra, in giù, in su e a destra[3];

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

- **e**, **b**, **w**: fino alla fine e fino all'inizio della parola attuale, fino all'inizio della parola successiva;
- **^**, **0**, **$**: fino al primo carattere non vuoto della riga attuale, fino all'inizio della riga attuale, fino alla fine della riga attuale;
- `f<x>`: fino alla prossima occorrenza del carattere `<x>`; ad esempio, `fe` sposta il cursore sulla prossima occorrenza del carattere **e**;
- `/<stringa>`, `?<stringa>`: fino alla successiva occorrenza della stringa o espressione regolare `<stringa>`, e la stessa cosa a ritroso nel file; ad esempio, `/pippo` sposta il cursore fino alla prossima occorrenza della parola `pippo`;
- **{**, **}**: fino all'inizio e fino alla fine del paragrafo attuale;
- **G**, **H**: fino alla fine del file, fino all'inizio dello schermo.

Ciascuno di questi caratteri di area interessata o comandi di movimento può essere preceduto da un numero di ripetizioni; nel caso di **G**, invece, esso indica un numero di riga all'interno del file. Partendo da queste basi, potete creare qualsiasi combinazione.

Alcuni esempi:

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

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

3.2.4. Taglia, copia, incolla

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

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

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

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

Diamo un'occhiata all'esempio di Figura 3-5.

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

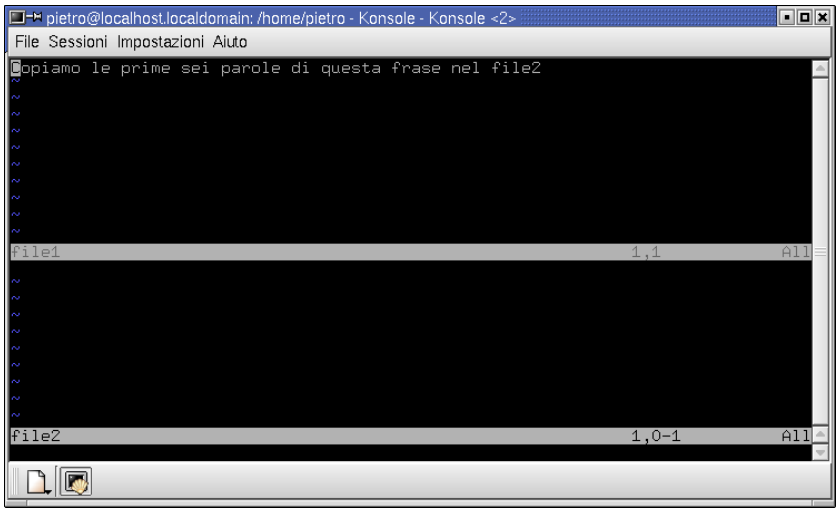


Figura 3-5. `vim`**, prima di copiare il blocco di testo**

Per compiere questa operazione seguiremo questi passi:

- copiamo le prime 6 parole della frase nella memoria **r** (per esempio): `"ry6w` [5];
- andiamo nel buffer `file2`, che si trova sotto: `Ctrl+w j`;
- incolliamo il contenuto della memoria **r** prima del cursore: `"rp`.

Otteniamo così il risultato che ci aspettavamo, come mostrato in Figura 3-6.

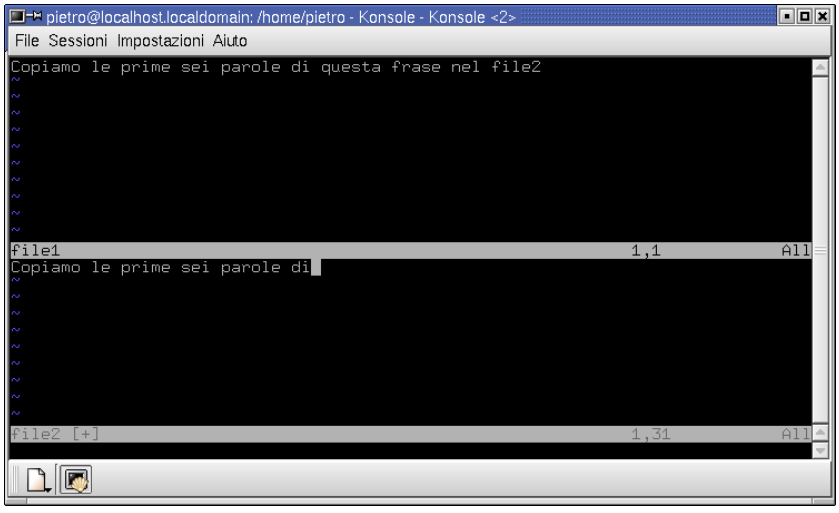


Figura 3-6. `vim`**, dopo la copia del blocco di testo**

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

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

3.2.5. Uscire da Vi

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

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

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

3.3. Un'ultima parola...

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

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

Capitolo 4. Strumenti da linea di comando

Lo scopo di questo capitolo è presentare alcuni strumenti da linea di comando che possono tornare utili nell'uso quotidiano. Naturalmente potete evitare di leggerlo se intendete usare unicamente l'ambiente grafico, ma un rapido sguardo potrebbe farvi cambiare idea **:-)**.

Questo capitolo non ha una vera e propria struttura, i comandi sono elencati così come capita, dal più comunemente usato al più oscuro. Ciascun comando sarà accompagnato da un esempio, ma la ricerca di altri usi interessanti per gli stessi comandi sarà lasciata a voi, come esercizio.

4.1. grep: ricerca di stringhe all'interno di file

Va bene, il nome del programma non è molto intuitivo, e non lo è neanche il suo significato (`grep` sta per "General Regular Expression Parser"), ma il suo scopo è semplice: cercare in uno o più file un modello (ingl. *pattern*) di testo immesso come argomento. La sua sintassi è:

```
grep [opzioni] <modello> [uno o più file]
```

Se indicate più file insieme, prima di ciascuna riga del risultato sarà mostrato il nome del file corrispondente. Usate l'opzione `-h` per evitare che i nomi dei file siano mostrati; usate invece l'opzione `-l` per ottenere solo i nomi dei file in cui sono state trovate corrispondenze. Il modello è una espressione regolare, sebbene la maggior parte delle volte si tratti di una semplice parola. Le opzioni usate più di frequente sono le seguenti:

- `-i`: fa una ricerca *case insensitive*, cioè che non distingue fra maiuscole e minuscole;
- `-v`: inverte la ricerca: mostra le righe che **non** contengono corrispondenze;
- `-n`: mostra il numero di riga per ogni riga in cui viene trovata una corrispondenza;
- `-w`: dice a *grep* che il modello deve corrispondere a una parola intera.

Ecco un esempio di come usarlo su un file di nome `alex`:

```
$ cat alex
Ciao Ale
Salve Alessandro
Ci vediamo, Ale

  # Cerca la stringa "ciao", nessuna differenza tra maiuscole e minuscole
$ grep -i ciao alex
Ciao Ale

  # Cerca la stringa "Ale" come parola intera, e stampa
  # il numero di riga prima di ogni occorrenza
$ grep -nw Ale alex
1:Ciao Ale
3:Ci vediamo, Ale

  # Adesso vogliamo tutte le righe che non cominciano con "C"
$ grep -v "^C" alex
Salve Alessandro
$
```

Se volete usare *grep* in una pipe non dovete indicare il nome di un file, poiché come comportamento predefinito il comando preleva il suo ingresso dallo ***standard input***. Allo stesso modo, come comportamento predefinito, il comando stampa il risultato sullo ***standard output***, quindi potete indirizzare l'output di un *grep* verso un altro programma senza alcun problema. Ad esempio:

```
$ cat /usr/share/doc/HOWTO/Parallel-Processing-HOWTO | \
  grep -n thread | less
```

4.2. find: cerca file in base a determinati criteri

`find` è un veterano tra i comandi *UNIX*. Il suo scopo è analizzare ricorsivamente una o più directory e cercare all'interno di esse i file che corrispondono a determinati criteri. Sebbene sia molto utile, la sua sintassi è veramente complessa, ed è necessario un certo impegno per utilizzarlo. La sintassi generale è:

```
find [opzioni] [directory] [criterio] [azione]
```

Se non specificate alcuna directory, `find` cercherà nella directory attuale. Se non specificate il criterio di ricerca, questo sarà considerato equivalente a "true", e quindi saranno trovati come corrispondenti tutti i file. Le opzioni, i criteri e le azioni sono così tanti che qui ne citeremo solo alcuni. Iniziamo con le opzioni:

- `-xdev`: esclude dalla ricerca le directory che risiedono su altri filesystem.
- `-mindepth <n>`: effettua la ricerca solo sui livelli a partire dal livello `n` al di sotto della directory specificata.
- `-maxdepth <n>`: cerca i file che si trovano al massimo `n` livelli al di sotto della directory specificata.
- `-follow`: segue i link simbolici corrispondenti a directory. Come comportamento predefinito, `find` non li segue.
- `-daystart`: in caso di controlli relativi al tempo (vedi sotto), considera come orario l'inizio del giorno attuale invece del valore predefinito (24 ore prima dell'orario attuale).

Un criterio può essere costituito da uno o più controlli ***atomici*** fra i molti disponibili; alcuni controlli utili sono:

- `-type <tipo>`: cerca un determinato tipo di file; `<tipo>` può essere uno dei seguenti: `f` (file normale), `d` (directory), `l` (collegamento simbolico), `s` (socket), `b` (file in modalità a blocchi), `c` (file in modalità a caratteri) o `p` (pipe con nome).
- `-name <modello>`: cerca i file i cui nomi corrispondono al `<modello>` indicato. Con questa opzione, il `<modello>` è considerato come un modello di ***meta-espansione*** (si veda il capitolo *I caratteri speciali (meta-caratteri) e le espressioni regolari nella shell*, pag. 12).
- `-iname <modello>`: come `-name`, ma non distingue tra maiuscole e minuscole.
- `-atime <n>`, `-amin <n>`: cerca i file che sono stati letti `<n>` giorni fa (`-atime`) o `<n>` minuti fa (`-amin`). Potete anche usare `+<n>` o `-<n>`, e in questo caso saranno cercati i file letti rispettivamente al massimo o al minimo `<n>` giorni/minuti fa.
- `-anewer <file>`: cerca i file che sono stati letti più recentemente del file `<file>`.
- `-ctime <n>`, `-cmin <n>`, `-cnewer <file>`: sono equivalenti a `-atime`, `-amin` e `-anewer`, ma considerano la data dell'ultima modifica del file.
- `-regex <modello>`: come `-name`, ma il `modello` viene considerato come una ***espressione regolare***.
- `-iregex <modello>`: come `-regex`, ma non distingue tra maiuscole e minuscole.

Esistono molti altri tipi di controlli, fate riferimento a `find`(1) per ulteriori dettagli. Per combinare i controlli potete scegliere fra:

- `<c1> -a <c2>`: è vero se sono veri sia `<c1>` che `<c2>`; `-a` è implicito, quindi potete scrivere `<c1> <c2> <c3> ...` se volete che siano verificati tutti i controlli `<c1>`, `<c2>`, `...`
- `<c1> -o <c2>`: è vero se almeno uno tra `<c1>` e `<c2>` è vero. Ricordate che `-o` ha una ***precedenza*** inferiore a `-a`, e quindi se volete, ad esempio, cercare i file che corrispondono ad almeno uno dei criteri `<c1>` e `<c2>` oltre che al criterio `<c3>`, dovrete usare le parentesi e scrivere `( <c1> -o <c2> ) -a <c3>`. Dovrete usare una ***sequenza escape*** per le parentesi (disattivandole), perché altrimenti saranno interpretate dalla *shell*!
- `-not <c1>`: inverte il controllo `<c1>`, pertanto `-not <c1>` è vero se `<c1>` è falso.

Infine, potete specificare una azione per ciascun file trovato. Le più frequentemente usate sono le seguenti:

- `-print`: stampa il nome di ciascun file sullo standard output. È l'azione predefinita.
- `-ls`: stampa sullo standard output l'equivalente del comando `ls -ilds` per ogni file trovato.
- `-exec <comando>`: esegue il comando `<comando>` per ogni file trovato. La linea di comando `<comando>` deve terminare con un "`;`", per il quale deve essere utilizzata una sequenza escape in modo che la shell non

lo interpreti; la posizione del file è rappresentata da `{}`. Per maggiori chiarimenti si vedano gli esempi di utilizzo.

- `-ok <comando>`: come `-exec`, ma chiede conferma per ogni comando.

Ci siete ancora? Bene, ora facciamo un po' di pratica, è sempre il modo migliore per capire questo mostro. Supponiamo che vogliate trovare tutte le directory contenute in `/usr/share`. Dovrete scrivere:

```
find /usr/share -type d
```

Supponiamo che abbiate un server HTTP, e che tutti i vostri file HTML siano in `/var/www/html`, che è anche la vostra directory attuale. Volete cercare tutti i file il cui contenuto non è stato modificato da un mese a questa parte. Poiché possedete pagine create da diversi autori, alcuni file hanno l'estensione `html` mentre altri hanno l'estensione `htm`. Volete creare dei collegamenti a questi file nella directory `/var/www/vecchi`. Dovrete quindi scrivere[1]:

```
find \( -name "*.htm" -o -name "*.html" \) -a -ctime -30 -exec ln {} \
 /var/www/vecchi \;
```

Va bene, questo esempio è un po' complicato ed è necessaria una breve spiegazione. Il criterio è questo:

```
\( -name "*.htm" -o -name "*.html" \) -a -ctime -30
```

e fa quello che noi vogliamo: cerca tutti i file i cui nomi finiscono in `.htm` o `.html` " `\( -name "*.htm" -o -name "*.html" \)` ", **e che** (`-a`) non sono stati modificati negli ultimi 30 giorni, che equivalgono più o meno a un mese (`-ctime -30`). Fate attenzione alle parentesi: in questo caso sono necessarie, perché `-a` ha una precedenza maggiore; se non le avessimo usate, sarebbero stati trovati tutti i file il cui nome finisce per `.htm`, più tutti i file il cui nome finisce per `.html` e che non sono stati modificati nell'ultimo mese, e questo non è quello che avevamo in mente. Notate anche che le parentesi sono disattivate dalla shell: se noi avessimo scritto `( .. )` invece di `\( .. \)`, la shell le avrebbe interpretate e avrebbe cercato di eseguire `-name "*.htm" -o -name "*.html"` in una sotto-shell... Un'altra soluzione poteva essere quella di racchiudere le parentesi fra virgolette doppie o singole, ma una barra inversa (\) qui è preferibile perché dobbiamo isolare un solo carattere.

E, per finire, abbiamo il comando che deve essere eseguito su ogni file:

```
-exec ln {} /var/www/vecchi \;
```

Anche qui è necessario disattivare il " `;` " dalla *`shell`*, altrimenti questa lo interpreterebbe come un separatore di comandi. Se non lo fate, `find` lamenterà la mancanza di un argomento dell'opzione `-exec`.

Un ultimo esempio: supponiamo che abbiate una directory `/shared/images` enorme, contenente tutti i tipi di immagini possibili, e supponiamo che di solito voi usiate il comando *`touch`* per aggiornare la data di un file di nome `stamp` contenuto nella stessa directory, in modo da avere un riferimento temporale. Vogliamo cercare in questa directory tutte le immagini ***JPEG*** più recenti del file `stamp`, e poiché possedete immagini provenienti da diverse fonti, questi file potranno avere estensione `jpg`, `jpeg`, `JPG` o `JPEG`. Vogliamo anche evitare di cercare nella directory `old`. Vogliamo infine che questa lista di file ci venga spedita via e-mail, e il nostro nome utente è `pietro`:

```
find /shared/images -cnewer     \
     /shared/images/stamp       \
     -a -iregex ".*\.jpe?g"     \
     -a -not -regex ".*/old/.*" \
       | mail pietro -s "Nuove immagini"
```

Ed ecco fatto! Naturalmente questo comando non è molto utile se dovete riscriverlo per intero ogni volta, e inoltre potreste volere che sia eseguito periodicamente in automatico... Per fare ciò, leggete il prossimo paragrafo.

1. Notate che per questo esempio è necessario che `/var/www` e `/var/www/vecchi` siano sullo stesso filesystem!

4.3. crontab: analizzare o modificare il file crontab

`crontab` è un comando che vi permette di eseguire dei comandi a intervalli di tempo regolari, con il vantaggio di non dover essere necessariamente collegati al sistema, e di poter ricevere il risultato dei comandi via e-mail. Potete indicare l'intervallo di tempo in minuti, ore, giorni e anche mesi. In base alle opzioni utilizzate, `crontab` si comporterà in diversi modi:

- `-l`: stampa il vostro file `crontab` attuale.
- `-e`: modifica il vostro file `crontab`.
- `-r`: elimina il vostro file `crontab` attuale.
- `-u <utente>`: applica una delle opzioni di cui sopra all'utente `<utente>`. Solo `root` può farlo.

Iniziamo modificando un file `crontab`. Se scrivete `crontab -e`, se avete impostato le variabili d'ambiente `EDITOR` o `VISUAL` vi troverete di fronte al vostro editor di testi preferito, altrimenti sarà utilizzato *Vi*. Una riga del file `crontab` è composta da sei campi. I primi cinque campi sono gli intervalli di tempo in minuti, ore, giorni del mese, mesi e giorni della settimana. Il sesto campo è il comando che deve essere eseguito. Le righe che iniziano con un `#` sono considerate come commenti e saranno ignorate da `crond` (il programma che si occupa dell'esecuzione dei file `crontab`). Ecco un esempio di `crontab`:

per poterlo stampare con un carattere leggibile abbiamo dovuto spezzare le righe più lunghe in più parti; pertanto, alcune parti devono essere digitate su una sola riga. Quando una riga termina con il carattere \, significa che quella riga continua sotto. Questa convenzione è valida anche nei file `Makefile`, nella *shell* e in altre situazioni.

```
# Se non volete ricevere posta, fate precedere la
#   riga seguente dal segno di commento (#)
#MAILTO=""
#
# Ogni due 2 giorni, alle 14.00 in punto, fa un elenco delle
#   nuove immagini come nell'esempio precedente - dopo di che
#   "ritocca" il file "stamp". Il carattere "%" viene considerato
#   come un ritorno a capo, questo vi permette di inserire più
#   comandi sulla stessa linea.
0 14 */2 * *  find /shared/images              \
  -cnewer /shared/images/stamp                 \
  -a -iregex ".*\.jpe?g"                       \
  -a -not -regex                               \
    ".*/old/.*"%touch /shared/images/stamp
#
# Suona una musica ogni volta che è Natale :)
0 0 25 12 * mpg123 $HOME/musica/buon-natale.mp3
#
# Ogni martedì alle 17.00 stampa la lista della spesa...
0 17 * * 2 lpr $HOME/lista-della-spesa.txt
```

Ci sono parecchi modi per specificare gli intervalli, oltre a quelli mostrati in questo caso. Ad esempio, potete indicare un insieme di ***valori discreti*** separati da virgole (`1,14,23`) o un intervallo di valori (`1-15`), o anche combinare le due cose insieme (`1-10,12-20`), specificando eventualmente un passo (`1-12,20-27/2`). Ora tocca a voi trovare dei comandi utili da inserire in questo file!

4.4. at: programmare un comando per una sola esecuzione

Potreste anche voler avviare un comando in un giorno particolare, e non periodicamente. Ad esempio, supponiamo che vogliate vi sia ricordato che avete un appuntamento oggi pomeriggio alle 6:00; state usando *X*, e volete essere avvertiti alle 5:30 del pomeriggio che è ora di andare. In questo caso, `at` è quello che vi serve:

```
$ at 5:30pm
  # Ora siete di fronte al promp di "at"
at> xmessage "È ora di andare! Appuntamento alle 18.00"
  # Premete Ctrl-d per uscire
at> <EOT>
$
```

Potete indicare il tempo in diversi modi:

- `now + <intervallo>`: significa "adesso" più un intervallo di tempo (opzionale, se non lo indicate significa semplicemente "adesso"). La sintassi per l'intervallo è `<n> (minutes|hours|days|weeks|months)`. Ad esempio, potete scrivere `now + 1 hour` (fra un'ora), `now + 3 days` (fra tre giorni), e così via.
- `<orario> <giorno>`: specifica esattamente la data. Il parametro `<orario>` è obbligatorio. `at` è molto flessibile riguardo al formato: ad esempio, potete scrivere `0100`, `04:20`, `2am`, `0530pm`, `1800`, o uno fra i tre seguenti valori speciali: `noon` (mezzogiorno), `teatime` (l'ora del tè, le 16.00) o `midnight` (mezzanotte). Il parametro `<giorno>` è opzionale; anche questo può essere scritto in diversi modi: `12/20/2001`, ad esempio, che indica il 20 dicembre del 2001, oppure nel formato europeo, `20.12.2001`. Potete omettere l'anno, ma in questo caso viene accettato solo il formato europeo: `20.12`. Potete anche indicare il mese per esteso: `Dec 20` e `20 Dec` sono entrambe espressioni consentite.

`at` dispone anche di alcune opzioni:

- `-l`: stampa la lista dei compiti attualmente accodati; il primo campo è il numero associato al compito. È equivalente al comando `atq`.
- `-d <n>`: rimuove il compito numero `<n>` dalla coda. Potete vedere i numeri dei compiti usando `atq`. È equivalente a `atrm <n>`.

Come sempre, consultate la `pagina di manuale` at(1) per conoscere ulteriori opzioni.

4.5. tar: Tape ARchiver

Pur avendo già usato `tar` nel capitolo *La compilazione e l'installazione di software libero*, pag. 83, non vi abbiamo spiegato come funziona; lo scopo di questa sezione è proprio questo. Come `find`, anche `tar` è un antico strumento di *UNIX*, e come tale ha una sintassi un po' particolare. La sua sintassi è:

```
tar [opzioni] [file...]
```

Eccovi ora un elenco di opzioni. Ricordate che esse possiedono tutte un'opzione estesa equivalente, ma per queste dovrete fare riferimento alla relativa pagina man, perché qui non saranno indicate. E, naturalmente, qui non saranno indicate neanche tutte le opzioni disponibili **:-)**.

l'uso del trattino iniziale (-) per le opzioni brevi di `tar` è da evitare, fatta eccezione per quelle che seguono un'opzione estesa.

- `c`: crea nuovi archivi.
- `x`: estrae file da un archivio.
- `t`: elenca i file contenuti in un archivio.
- `v`: elenca semplicemente i file mentre vengono aggiunti o estratti da un archivio, oppure, insieme all'opzione `t` (vedi sopra), mostra una lista estesa dei file invece di una breve.
- `f <file>`: crea un archivio di nome `<file>`, estrae dall'archivio `<file>` oppure elenca i file dell'archivio `<file>`. Se questo parametro non viene indicato, il file predefinito sarà `/dev/rmt0`, che generalmente è un file speciale associato a uno ***streamer*** (unità a nastri). Se al posto del nome del file viene usato `-` (un trattino), l'uscita o l'ingresso (a seconda se state estraendo da un archivio o creandone uno) saranno associati a standard output o standard input.
- `z`: impone a `tar` di comprimere con `gzip` l'archivio da creare, oppure indica che l'archivio da cui estrarre è compresso con `gzip`.
- `j`: come `z`, ma il programma usato per la compressione sarà `bzip2`.
- `p`: quando vengono estratti file da un archivio conserva tutti i loro attributi, compresi proprietario, ultima data d'accesso e così via. Molto utile per copie di interi filesystem;
- `r`: accoda a un archivio esistente i file indicati nella linea di comando. Ricordate che l'archivio al quale accodate i file **non** deve essere compresso!
- `A`: aggiunge gli archivi indicati nella linea di comando a quello specificato con l'opzione `f`. Come per `r`, affinché questo funzioni gli archivi non devono essere compressi.

Ci sono tante, tante, tante altre opzioni, potete fare riferimento alla pagina man tar(1) per una lista completa; guardate ad esempio l'opzione d. Ora facciamo un po' di pratica: supponiamo che vogliate creare un archivio di tutte le immagini contenute in /shared/images, compresso con bzip2, di nome images.tar.bz2 e posizionato nella vostra home directory. Dovrete quindi scrivere:

```
 #
 # Attenzione: dovete trovarvi nella directory contenente
 #   i file che volete archiviare!
 #
$ cd /shared
$ tar cjf ~/images.tar.bz2 images/
```

Come potete vedere, abbiamo usato tre opzioni: c per dire a tar che volevamo creare un archivio, j per dirgli che lo volevamo compresso con bzip2, e f ~/images.tar.bz2 per dirgli che l'archivio doveva essere creato nella nostra directory home, con il nome images.tar.bz2. Ora potremmo voler controllare che l'archivio sia corretto; possiamo farlo semplicemente vedendo l'elenco dei suoi file:

```
 #
 # Torniamo alla nostra directory home
 #
$ cd
$ tar tjvf images.tar.bz2
```

Qui abbiamo detto a tar di elencare (t) i file dell'archivio images.tar.bz2 (f images.tar.bz2), avvertendolo che questo archivio era compresso con bzip2 (j), e richiedendo una lista estesa (v). Adesso supponiamo che voi abbiate cancellato la directory delle immagini: fortunatamente il vostro archivio è intatto, e ora volete estrarre di nuovo i file nelle loro posizioni originali, in /shared. Ma, poiché volete che il vostro comando find per cercare nuove immagini continui a funzionare correttamente, dovete preservare tutti i loro attributi:

```
 #
 # spostatevi nella directory dove volete estrarre
 # i file contenuti nell'archivio
 #
$ cd /shared
$ tar jxpf ~/images.tar.bz2
```

Ed ecco fatto!

Ora, supponiamo che vogliate estrarre dall'archivio la directory images/cars, e nient'altro. Potete scrivere così:

```
$ tar jxf ~/images.tar.bz2 images/cars
```

Nel caso ve ne stiate preoccupando, state tranquilli: se provate ad archiviare file speciali, tar li considererà per quello che sono, file speciali, e non ne copierà il contenuto. Perciò, sì, potete tranquillamente includere /dev/mem in un archivio **:-)**. Ah, si comporta correttamente anche con i collegamenti, quindi non preoccupatevi neanche di questo. Per quel che riguarda i collegamenti simbolici, date uno sguardo anche all'opzione h nella relativa pagina man.

4.6. bzip2 e gzip: programmi per la compressione di dati

Avrete notato che abbiamo già parlato di questi due programmi quando ci siamo occupati di tar. A differenza di winzip su *Windows*, qui l'archiviazione e la compressione vengono eseguite usando due strumenti separati – tar per l'archiviazione, e i due programmi di cui stiamo per parlare per comprimere i dati: bzip2 e gzip. Anche su *GNU/Linux* esistono altri programmi di compressione, come zip, arj o rar, ma vengono usati di rado.

Inizialmente bzip2 era stato scritto per sostituire gzip: i suoi rapporti di compressione sono generalmente migliori, sebbene richieda più memoria. Tuttavia gzip viene ancora utilizzato, a causa della maggiore diffusione che ha avuto in passato rispetto a bzip2.

I due comandi hanno una sintassi simile:

```
gzip [opzioni] [uno o più file]
```

Se non viene indicato alcun file, sia gzip che bzip2 si mettono in attesa di dati dallo standard input e inviano il risultato sullo standard output. Quindi potete usare entrambi i programmi nelle pipe. Possiedono anche alcune opzioni in comune:

- `-1`, ..., `-9`: imposta il rapporto di compressione. Più alto il numero, migliore la compressione; ma migliore significa anche più lenta: non si può avere la botte piena e la moglie ubriaca.
- `-d`: decomprime i file. Equivale ad usare `gunzip` o `bunzip2`.
- `-c`: invia sullo standard output il risultato della compressione o decompressione dei file indicati come argomenti.

Come comportamento predefinito, sia `gzip` che `bzip2` cancellano i file che hanno compresso (o decompresso) se non usate l'opzione `-c`. Con `bzip2` potete evitarlo usando l'opzione `-k`, ma `gzip` non la possiede!

Ora vediamo alcuni esempi. Supponiamo che vogliate comprimere con `bzip2` tutti i file della directory attuale il cui nome finisce con `.txt`; userete quindi:

```
$ bzip2 -9 *.txt
```

Supponiamo che vogliate condividere il vostro archivio di immagini con qualcuno che non possiede `bzip2`, ma solo `gzip`. Non è necessario decomprimere l'archivio e poi ricomprimerlo, potete semplicemente decomprimerlo sullo standard output, usare una pipe, comprimere dallo standard input e indirizzare l'uscita verso il nuovo archivio:

```
bzip2 -dc images.tar.bz2 | gzip -9 >images.tar.gz
```

Ecco fatto. Avreste anche potuto scrivere `bzcat` invece di `bzip2 -dc`. Esiste anche un equivalente per `gzip`, ma il suo nome è `zcat`, e non **gzcat**. Se volete visualizzare direttamente un file compresso, senza doverlo prima decomprimere, potete anche usare `bzless` (e rispettivamente `zless`). Come esercizio, provate a trovare il comando da usare per visualizzare file compressi senza prima decomprimerli, e senza usare `bzless` o `zless` **:-)**.

4.7. Tanti, tanti altri...

Ci sono così tanti comandi che un libro completo su di essi sarebbe grande come un'enciclopedia. Questo capitolo non ha coperto neanche un decimo dell'argomento, ma potete comunque fare molte cose con quello che avete imparato qui. Se lo desiderate, potete leggere alcune pagine `man`: sort(1), sed(1), zip(1) (sì, è proprio quello che pensate: potete estrarre o creare archivi `.zip` con *GNU/Linux*), convert(1), e così via. Il modo migliore per prendere confidenza con questi strumenti è fare pratica e sperimentare, e probabilmente troverete per loro moltissime applicazioni, anche le più inaspettate. Buon divertimento!

Capitolo 5. Controllo dei processi

5.1. Ancora sui processi

Nel capitolo *Processi*, pag. 4 abbiamo visto che è possibile monitorare i processi: questo è l'argomento di cui ci occuperemo ora. Per comprendere meglio le operazioni che effettueremo, è opportuno introdurre qualche altra informazione su di essi.

5.1.1. L'albero dei processi

Come per i file, tutti i processi che girano in un sistema *GNU/Linux* sono organizzati in una struttura ad albero, e la radice di quest'albero è `init`. Ogni processo ha un numero (il suo PID, *Process ID*), insieme al numero del proprio processo padre (PPID, *Parent Process ID*). Il PID di `init` è 1, e così pure il suo PPID: `init` è il padre di se stesso.

5.1.2. Segnali

Ogni processo in un sistema *UNIX* può reagire ai segnali inviatigli. Vi sono 64 differenti segnali, che possono essere identificati in base al numero (a partire da 1) o al loro nome simbolico. I 32 segnali "più alti" (da 33 a 64) sono segnali in tempo reale, e non rientrano nell'ambito di questo capitolo. Per ciascuno di essi, il processo può definire una diversa reazione, fatta eccezione per due segnali: il segnale numero 9 (`KILL`), e il segnale numero 19 (`STOP`).

Il segnale 9 uccide un processo in modo irrevocabile, senza lasciargli il tempo di terminare l'esecuzione in maniera appropriata. Questo è il segnale che dovreste inviare a un processo bloccato, o che manifesta altri problemi. Un elenco completo dei segnali è disponibile digitando il comando `kill -l`.

5.2. Informazioni sui processi: i comandi ps e pstree

Questi due comandi visualizzano una lista dei processi attualmente in esecuzione sul sistema, secondo i criteri da voi stabiliti.

5.2.1. ps

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

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

Come d'abitudine per molti programmi di utilità *UNIX*, `ps` dispone di un gran numero di opzioni, ve ne segnaliamo alcune tra le più comuni:

- `a`: visualizza anche i processi avviati da altri utenti;
- `x`: visualizza anche i processi che non hanno un terminale di controllo o che ne hanno uno diverso da quello che state usando;
- `u`: visualizza per ciascun processo il nome dell'utente che lo ha avviato e l'ora in cui è partito.

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

L'output di questo comando è suddiviso in due diversi campi: quello che vi interesserà di più è il campo `PID`, che contiene il numero identificativo del processo. Il campo `CMD` contiene il nome del comando eseguito. Un modo molto comune di lanciare `ps` è il seguente:

```
$ ps ax | less
```

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

5.2.2. pstree

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

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

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

5.3. Inviare segnali ai processi: kill, killall e top

5.3.1. kill, killall

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

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

```
$ kill 785
```

Se intendete inviargli il segnale 9, digitate:

```
$ kill -9 785
```

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

```
$ killall -9 netscape
```

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

5.3.2. top

`top` è un programma onnicomprensivo: svolge contemporaneamente le funzioni di `ps` e `kill`. Il programma funziona in modalità console, quindi viene lanciato in un terminale, come viene mostrato qui (Figura 5-1).

```
  1:22pm  up 23:26,  5 users,  load average: 0.01, 0.02, 0.00
50 processes: 47 sleeping, 2 running, 1 zombie, 0 stopped
CPU states:  1.5% user,  1.1% system,  0.0% nice, 97.2% idle
Mem:  128008K av, 124936K used,   3072K free,  40000K shrd,   1912K buff
Swap: 136512K av,   1772K used, 134740K free                 24700K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
 9837 root      15   0 60716  59M  1940 R       0  1.5 47.4   3:10 X
10164 fg         9   0  1060 1060   860 R       0  0.5  0.8   0:00 top
 9855 fg         1   0  2192 2192  1564 S       0  0.1  1.7   0:00 xterm
10066 fg         1   0 20592  20M  8312 S       0  0.1 16.0   0:09 ld-linux.so.
10168 fg        19   0   440  440   360 S       0  0.1  0.3   0:00 sleep
    1 root       0   0   168  168    92 S       0  0.0  0.1   0:04 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:01 kupdate
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kpiod
    5 root       0   0     0    0     0 SW      0  0.0  0.0   0:03 kswapd
    6 root     -20 -20     0    0     0 SW<     0  0.0  0.0   0:00 mdrecoveryd
  133 root       0   0    72   60     0 S       0  0.0  0.0   0:00 apmd
  268 bin        0   0    76   52     0 S       0  0.0  0.0   0:00 portmap
  320 root       0   0   292  244   168 S       0  0.0  0.1   0:00 syslogd
  330 root       0   0   496  164   120 S       0  0.0  0.1   0:00 klogd
  345 root       0   0   184  176    80 S       0  0.0  0.1   0:00 crond
  360 root       0   0   224  216   124 S       0  0.0  0.1   0:00 inetd
```

Figura 5-1. Esempio di esecuzione di `top`

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

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

II. Linux in profondità

Capitolo 6. Organizzazione della struttura del filesystem

Al giorno d'oggi, un sistema *UNIX* può raggiungere dimensioni notevoli, davvero notevoli. Questo è particolarmente vero per *GNU/Linux*: la grande quantità di software disponibile lo renderebbe un sistema impossibile da gestire, se non esistessero delle linee guida per il posizionamento dei file nella struttura gerarchica del filesystem.

Lo standard riconosciuto in questo campo è il FHS (*Filesystem Hierarchy Standard*), che ha raggiunto la versione 2.2 al momento della stesura di questo manuale. Il documento che descrive lo standard è disponibile su Internet, in vari formati, sul sito pathname (`http://www.pathname.com/fhs/`). Questo capitolo ne costituisce solo un breve riassunto, ma dovrebbe essere sufficiente per capire in quale directory cercare (o collocare) un dato file.

6.1. Dati condivisibili e non, statici e variabili

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

Si noti che queste classificazioni sono solamente consigliate; non siete obbligati ad adottarle, ma possono esservi di grande aiuto nella gestione del vostro sistema. Si noti ancora che che la distinzione statico/variabile si applica solo all'uso di un sistema, e non alla sua configurazione: durante l'installazione di un programma dovranno necessariamente essere modificate directory che "normalmente" sono statiche, ad esempio `/usr`.

6.2. La directory radice: /

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

- `/bin`: file eseguibili essenziali. Questa directory contiene i comandi di base che saranno utilizzati da tutti gli utenti e che sono necessari per le normali operazioni del sistema: `ls`, `cp`, *login*, etc. Statica, non condivisibile.
- `/boot`: contiene i file necessari al programma che gestisce il boot di *GNU/Linux* (*grub* o *LILO* per le piattaforme **Intel**). Questa directory può contenere il kernel, e se questo non si trova qui allora deve trovarsi nella directory `root`. Statica, non condivisibile.
- `/dev`: contiene i file dei dispositivi di sistema (`dev` sta per *DEVices*). Statica, non condivisibile.
- `/etc`: questa directory contiene tutti i file di configurazione caratteristici del sistema. Statica, non condivisibile.
- `/home`: contiene le directory personali degli utenti del sistema. Questa directory può essere condivisibile o meno (in alcune reti molto grandi è resa condivisibile attraverso NFS). Variabile, condivisibile.
- `/lib`: questa directory contiene le librerie essenziali per il funzionamento del sistema e, in `/lib/modules`, i moduli del kernel. Tutte le librerie necessarie ai programmi che si trovano nelle directory `/bin` e `/sbin` devono trovarsi in questa directory, come pure il linker `ld.so`. Statica, non condivisibile.
- `/mnt`: directory contenente i punti di mount per i filesystem temporanei. Variabile, non condivisibile.
- `/opt`: questa directory contiene pacchetti non indispensabili per il funzionamento del sistema. È consigliabile mettere i file statici (eseguibili, librerie, pagine di manuale, etc.) relativi a questi pacchetti in `/opt/nome_del_pacchetto`, e i loro file di configurazione in `/etc/opt`.
- `/root`: directory personale dell'utente `root`. Variabile, non condivisibile.
- `/usr`: si veda la prossima sezione. Statica, condivisibile.
- `/sbin`: contiene i file eseguibili necessari all'avvio del sistema, utilizzabili solo da `root`. Anche un utente normale può eseguirli, ma senza andare troppo lontano. Statica, non condivisibile.
- `/tmp`: directory destinata a contenere i file temporanei che alcuni programmi possono creare. Variabile, non condivisibile .

- /var: directory per i dati che possono essere modificati in tempo reale dai programmi (ad es. dal server della posta elettronica, da programmi di controllo, dal server di stampa, etc.). Tutto il contenuto di /var è variabile, ma le sue varie sottodirectory possono essere o non essere condivisibili.

6.3. /usr: la più grossa

La directory /usr è la directory destinata ad accogliere la maggior parte dei programmi installati nel sistema. Tutti i file eseguibili che si trovano in questa directory non devono essere necessari all'avvio del sistema o alla sua manutenzione, dal momento che la gerarchia /usr molto spesso si trova su un filesystem separato. Date le sue dimensioni, spesso notevoli, /usr ha una propria gerarchia di sottodirectory; ne citiamo solo alcune:

- /usr/X11R6: l'intera gerarchia di *X Window System*. Tutti gli eseguibili necessari al funzionamento di *X* (inclusi i server *X*) e tutte le relative librerie devono essere qui. La directory /usr/X11R6/lib/X11 contiene tutte le impostazioni di *X* comuni a tutti i sistemi. Le impostazioni specifiche per ogni sistema, invece, dovrebbero trovarsi in /etc/X11.
- /usr/bin: questa directory contiene la maggior parte dei file eseguibili presenti nel sistema. **Qualsiasi** eseguibile che non sia necessario alla manutenzione e/o all'amministrazione del sistema deve trovarsi in questa directory, ad eccezione dei file relativi a programmi installati da voi stessi, i quali dovrebbero invece essere in /usr/local.
- /usr/lib: questa directory contiene tutte le librerie necessarie all'esecuzione dei programmi che si trovano in /usr/bin e /usr/sbin. Contiene anche un collegamento simbolico /usr/lib/X11 che fa riferimento alla directory contenente le librerie di *X Window System*, /usr/X11R6/lib (sempre che *X* sia installato, naturalmente).
- /usr/local: questa è la directory dove andrebbero installate le applicazioni personali. Durante l'installazione dovrebbe essere stata creata la gerarchia necessaria: lib/, bin/, etc.
- /usr/share: questa directory contiene tutti i dati, indipendenti dall'architettura del sistema, necessari alle applicazioni che si trovano in /usr. Fra le altre cose, in questa directory si trovano le informazioni sul fuso orario e sulla localizzazione del sistema (zoneinfo e locale).

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

6.4. /var: dati modificabili durante l'uso

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

- /var/log: contiene i file di log del sistema;
- /var/spool: contiene i file di lavoro dei demoni di sistema. Ad esempio, /var/spool/lpd contiene i file di lavoro del server di stampa, e /var/spool/mail i file di lavoro del server di posta elettronica (cioè tutti i messaggi in arrivo e in partenza dal sistema);
- /var/run: questa directory viene usata per tenere nota di tutti i processi usati dal sistema, in maniera che si possa agire su di essi nell'eventualità di un cambio di ***runlevel*** nel sistema (si veda il capitolo *I file di avvio del sistema: init sysv*, pag. 59).

6.5. /etc: file di configurazione

La directory /etc è una delle directory fondamentali in qualunque sistema *UNIX*: contiene tutti i principali file di configurazione del sistema. **Non cancellatela mai** per risparmiare spazio! Un altro suggerimento: se volete estendere la struttura gerarchica del filesystem su più partizioni, ricordate che la directory /etc non deve essere messa su una partizione separata, poiché è necessaria all'inizializzazione del sistema.

Alcuni file particolarmente importanti sono:

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

Esistono anche alcune sottodirectory per programmi che hanno bisogno di un gran numero di file per la loro configurazione. È il caso di *`X Window System`*, ad esempio, al quale è dedicata l'intera directory `/etc/X11`.

Capitolo 7. Filesystem e punti di mount

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

7.1. Principi

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

Proprio come *Windows* ha bisogno di un "disco C:", *GNU/Linux* deve poter montare la radice del suo albero di file e directory (`/`) in qualche luogo, e lo fa su una partizione che contiene il ***filesystem di root***. Una volta montata la radice, è possibile montare altri filesystem della struttura ad albero su altri ***punti di mount*** all'interno di tale struttura. Qualunque directory sotto la directory radice può fungere da punto di mount. Notate, inoltre, che potete montare lo stesso filesystem più volte in punti diversi.

Questo consente una grande flessibilità nelle configurazioni possibili. Generalmente, ad esempio, nei server web un'intera partizione viene destinata specificamente alla directory che ospita i dati del server. La directory che li ospita normalmente è `/home/httpd`, che di conseguenza funge da punto di mount per la partizione. In Figura 7-1 e in Figura 7-2 potete vedere la situazione del sistema prima e dopo aver montato il filesystem.

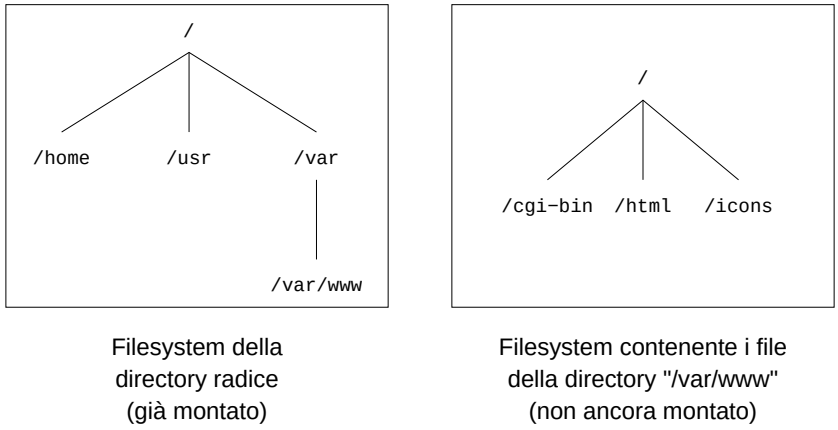


Figura 7-1. Un filesystem non ancora montato

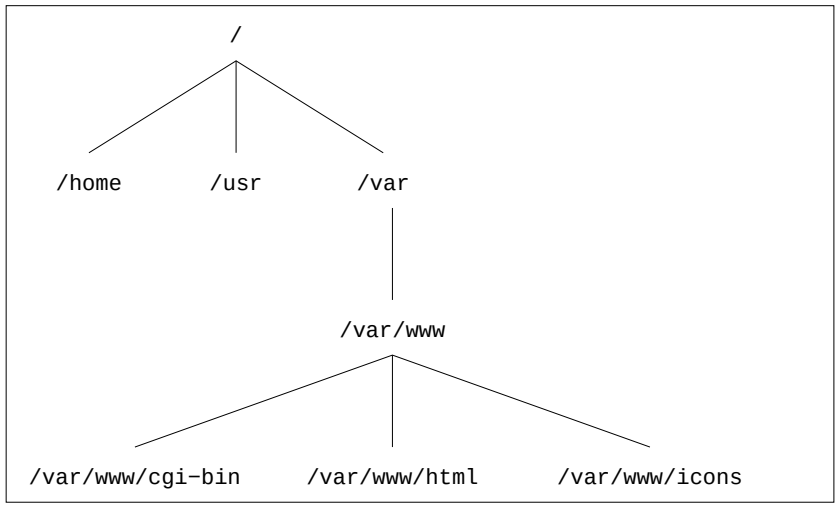


Figura 7-2. Il filesystem ora è montato

Come potete immaginare, questa caratteristica offre diversi vantaggi: la struttura ad albero rimane sempre la stessa, sia che stia su un solo filesystem, sia che si estenda su svariate dozzine[1]. Quando lo spazio comincia a mancare, è sempre possibile spostare fisicamente parti importanti della struttura delle directory, cosa che stiamo per fare in questa sezione.

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

1. la directory che funge da punto di mount deve esistere;
2. inoltre questa directory **dovrebbe essere vuota**: se la directory scelta come punto di mount contiene già dei file, questi verranno soltanto "nascosti" dal filesystem appena montato, e non cancellati, ma non saranno più accessibili finché non verrà liberato il punto di mount.

7.2. Partizionamento di un disco rigido e formattazione di una partizione

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

Come prima cosa, dovete sapere il "nome" del vostro disco rigido, cioè quale file lo rappresenta. Se supponiamo che lo installiate come slave sulla vostra interfaccia IDE primaria, il nome sarà `/dev/hdb` [2]. Vi invitiamo a consultare la sezione *Gestione delle partizioni* del *Manuale dell'utente* per una spiegazione dei metodi usati per partizionare un disco rigido. Notate che *`DiskDrake`* provvederà anche alla creazione dei filesystem sulle nuove partizioni.

7.3. I comandi mount e umount

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

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

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

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

1. *`GNU/Linux`* può gestire fino a 64 filesystem montati contemporaneamente.
2. Come trovare il nome di un disco è spiegato nella *Guida all'installazione*.

L'opzione `-t` è usata per specificare il tipo di filesystem che la partizione deve ospitare. I filesystem che si incontrano più spesso sono: `ext2` (il filesystem di *GNU/Linux*), VFAT (per tutte le partizioni *DOS/Windows*: FAT 12, 16 o 32) e iso9660 (il filesystem per i CD-ROM). Se non specificate nessun filesystem, `mount` cercherà di capire qual è il filesystem di una partizione leggendo il primo blocco. In genere questo tentativo ha pieno successo.

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

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

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

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

```
umount <punto di mount|dispositivo>
```

Quindi, per smontare la partizione, digitate:

```
$ umount /mnt/
```

oppure:

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

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

7.4. Il file /etc/fstab

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

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

Ogni riga contiene, nell'ordine:

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

C'è **sempre** una voce per il filesystem radice. Le partizioni di *swap* sono speciali perché non sono visibili nella struttura delle directory, e il loro campo nel punto di mount presenta la parola chiave `swap`. Torneremo a parlare di `/proc` in maggior dettaglio nel capitolo *Il filesystem /proc*, pag. 55. Un altro filesystem particolare è `/dev/pts`.

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

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

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

Ci sono due opzioni speciali: `noauto` e `user`. L'opzione `noauto` specifica che il filesystem non deve essere montato all'avvio, ma soltanto al momento desiderato. L'opzione `user` specifica che qualunque utente può montare e smontare il filesystem. Tipicamente, queste due opzioni sono usate per i drive CD-ROM e per i floppy, ma valgono anche per i dischi rigidi. Ci sono anche altre opzioni, infatti il file `/etc/fstab` è persino corredato da una sua pagina di manuale (fstab(5)).

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

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

oppure:

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

Ora portiamo a termine il compito di spostare una partizione: avete copiato la gerarchia della directory `/usr` e modificato il file `/etc/fstab` in maniera tale che la nuova partizione sia montata all'avvio. Al momento, però, i file della vecchia directory `/usr` sono ancora lì! Dovrete, quindi, eliminarli per liberare un po' di spazio (cosa che, d'altronde, era il vostro scopo iniziale!); per far ciò è necessario, come primo passo, spostarvi in modalità monoutente digitando `telinit 1` sulla linea di comando, e poi:

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

e avete finito. Adesso potete tornare in modalità multiutente (`telinit 3` o `telinit 5`) e, se non c'è più lavoro amministrativo da fare, dovreste uscire dall'account di `root`.

7.5. Una nota sull'opzione supermount

I kernel più recenti, come quelli forniti con **Mandrake Linux**, presentano una caratteristica molto interessante per gli utenti che usano frequentemente floppy disk e CD. `supermount`, la cui installazione dipende dal livello di sicurezza prescelto, provvede a montare e smontare automaticamente i supporti al momento in cui sono inseriti o rimossi. Questa caratteristica è molto utile, poiché non è più necessario digitare il comando `mount`, o `umount`, ogni volta che si inserisce o si estrae un supporto rimovibile.

Capitolo 8. Il filesystem di Linux

Il vostro sistema *GNU/Linux*, naturalmente, risiede sul vostro disco rigido all'interno di un filesystem. In questo capitolo discuteremo le diverse caratteristiche di vari filesystem, e i vantaggi che ciascuno di essi offre.

8.1. Confronto fra alcuni filesystem

A un certo punto dell'installazione potete scegliere fra diversi **filesystem** per le vostre partizioni. Questo significa che potete formattare le partizioni in base a diversi algoritmi.

La scelta di uno fra i vari filesystem disponibili non è immediatamente evidente, a meno che voi non siate degli esperti. Pertanto vi proponiamo una rapida presentazione di tre dei filesystem più aggiornati, tutti e tre inclusi in **Mandrake Linux**.

8.1.1. I filesystem tra cui scegliere

8.1.1.1. Ext2FS

Il **Secondo Filesystem Esteso** (in forma abbreviata **Ext2FS** o, più semplicemente, **ext2**) è stato il filesystem predefinito di *GNU/Linux* per molti anni. È stato introdotto in sostituzione del **Filesystem Esteso** (il che spiega l'uso dell'aggettivo "Secondo"). Il "nuovo" filesystem ha corretto alcuni problemi e superato alcuni limiti della versione precedente.

Ext2FS aderisce agli standard previsti per i filesystem destinati ai sistemi Unix. Sin dalla nascita era stata prevista una sua evoluzione, nonostante offrisse già una grande solidità e buone prestazioni.

8.1.1.2. Ext3FS

Come suggerito dal nome, il **Terzo Filesystem Esteso** è il successore di Ext2FS. È compatibile con quest'ultimo, ma in più offre una caratteristica di notevole interesse: il ***journaling*** .

Uno dei difetti più gravi dei filesystem "tradizionali" come Ext2FS è costituito dalla loro scarsa tolleranza per quanto riguarda improvvisi blocchi del sistema (dovuti a interruzioni di corrente, ad esempio, o all'errato comportamento di qualche programma). Questi fenomeni comportano, in genere, un lungo e meticoloso esame della struttura del filesystem, e tentativi di correggere eventuali errori che, in alcuni casi, possono provocare ulteriori danni. Come conseguenza, si può verificare una perdita parziale o totale dei dati salvati sul disco.

Il journaling risolve questo problema. Diciamo, per semplificare, che il nostro obiettivo è memorizzare le operazioni (come l'archiviazione di un file) **prima** di effettuarle davvero. Possiamo paragonare questo meccanismo alla registrazione degli eventi giornalieri nel giornale di bordo del capitano di una nave. Il risultato è un filesystem sempre coerente. E, nel caso si verifichi un problema, il controllo è molto rapido e le eventuali riparazioni assai limitate. Il tempo necessario per la verifica di un filesystem, quindi, è proporzionale al suo uso, non alle sue dimensioni.

Ext3FS, pertanto, offre la tecnologia dei filesystem di tipo *journaling*, pur mantenendo la struttura generale di Ext2FS: questa caratteristica assicura un'ottima compatibilità.

8.1.1.3. ReiserFS

A differenza di Ext3FS, **ReiserFS** è stato creato *ex novo*. È un filesystem di tipo *journaling* come Ext3FS, ma la sua struttura interna è radicalmente differente. In particolare, fa uso di schemi ad albero binario ispirati dal software usato per gestire i database.

8.1.1.4. JFS

JFS è il filesystem di tipo *journaling* progettato e usato da IBM. In un primo momento si trattava di software proprietario e chiuso, ma recentemente IBM ha deciso di aprirne l'accesso alla comunità del software libero. La sua struttura interna è vicina a quella di ReiserFS.

8.1.2. Differenze tra i filesystem citati

	Ext2FS	Ext3FS	ReiserFS	JFS
Stabilità	Eccellente	Buona	Buona	Soddisfacente
Strumenti per recuperare file cancellati	Sì (procedura complessa)	Sì (procedura complessa)	No	No
Durata del riavvio dopo un blocco del sistema	Lungo, anche molto lungo	Veloce	Molto veloce	Molto veloce
Recupero dei dati in caso di blocco del sistema	Generalmente buono, ma alto rischio di perdita parziale o totale dei dati	N/A	Molto buono. Una perdita completa dei dati è estremamente rara	Molto buono

Tabella 8-1. Caratteristiche dei filesystem

Per quanto riguarda le dimensioni massime dei file, queste dipendono da numerosi parametri (ad esempio la dimensione dei blocchi per ext2 e ext3), ed è probabile che la situazione cambi a seconda della versione del kernel e dell'architettura hardware. Il minimo disponibile, comunque, al momento è vicino o superiore ai 2 Tb (1Tb=1024 Gb), e può arrivare fino ai 4Pb (1Pb=1024 Tb) per JFS. Questi valori, sfortunatamente, sono limitati anche dalla dimensione massima raggiungibile da un dispositivo a blocchi, che per i kernel dell'attuale serie `2.4.X`, e limitatamente all'architettura `X86`, non può andare oltre i 2Tb[1] anche in modo RAID. Per avere ulteriori informazioni, consultate Adding Support for Arbitrary File Sizes to the Single UNIX Specification (`http://ftp.sas.com/standards/large.file/x_open.20Mar96.html`).

8.1.3. E sul piano delle prestazioni?

Comparare le prestazioni è sempre una cosa assai delicata. Ogni tipo di test presenta dei limiti, e i risultati vanno interpretati con cautela. Al momento attuale Ext2FS è un filesystem molto maturo, ma il suo sviluppo non prevede grandi balzi in avanti; i filesystem di tipo *journaling* come Ext3FS e ReiserFS, al contrario, sono in rapida evoluzione. I test compiuti soltanto un paio di mesi o di settimane fa sono già troppo vecchi. Non dimentichiamo, inoltre, che le caratteristiche dell'hardware moderno, in particolare per quanto riguarda la capacità dei dischi rigidi, hanno contribuito grandemente ad appianare le differenze tra di loro. Al momento, tuttavia, JFS è il filesystem che offre le migliori prestazioni.

Ogni sistema comporta vantaggi e svantaggi. Tutto dipende, in effetti, dall'uso che intendete fare del vostro computer. Le esigenze di un semplice computer da ufficio saranno pienamente soddisfatto da Ext2FS, mentre per un server è preferibile usare un filesystem *journaling* come Ext3FS. ReiserFS, grazie anche alle sue origini, sarà la scelta più adatta per un server di database. JFS, infine, verrà scelto quando la velocità del trasferimento dati costituisce il requisito essenziale.

Per quanto riguarda un uso "normale", i quattro filesystem offrono più o meno gli stessi vantaggi. ReiserFS permette di accedere rapidamente ai file di piccole dimensioni, ma è piuttosto lento nella manipolazione di file piuttosto grandi (molti megabyte). Nella maggior parte dei casi, i vantaggi che caratterizzano ReiserFS rendono i suoi inconvenienti di scarsa importanza.

8.2. Tutto è un file

Il *Manuale dell'utente* ha introdotto i concetti di proprietà dei file e di permessi di accesso, ma per capire davvero il ***filesystem*** di *UNIX* (e questo vale anche per il filesystem *ext2fs* di *GNU/Linux*) è necessario ridefinire il concetto stesso di file.

In questo caso "tutto" significa **davvero** tutto: un disco rigido, una partizione sul disco, una porta parallela, una connessione a un sito web, una scheda *Ethernet*, tutte queste cose sono dei file. Persino le directory sono file. *GNU/Linux* è in grado di riconoscere molti tipi di file oltre ai file e alle directory standard. Notate che con

1. Vi chiederete com'è possibile raggiungere capacità così elevate con dischi rigidi che raggiungono a malapena i 180Gb. In effetti, se usate tre schede RAID, ciascuna delle quali controlla 8 dischi da 128 Gb cadauno, ecco che arrivate a 3Tb...

"tipo di file" qui non ci riferiamo al **contenuto** di un file: per *GNU/Linux* e qualsiasi altro sistema *UNIX*, un file, che si tratti di un'immagine GIF, di un file binario o di qualsiasi altro tipo, non è altro che una sequenza di byte. L'operazione di distinzione dei file in base al loro contenuto viene lasciata alle applicazioni.

8.2.1. I diversi tipi di file

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

1. ***File in modalità a caratteri***: questi sono file di sistema speciali (come `/dev/null`, che abbiamo già incontrato), o periferiche (porte parallele o seriali), che hanno in comune il fatto che il loro contenuto (se ne hanno) non è ***bufferizzato*** (in altre parole, non sono conservati in memoria). File di questo tipo sono identificati per mezzo della lettera `'c'`.
2. ***File in modalità a blocchi***: questi file sono periferiche e, a differenza dei file in modalità a caratteri, il loro contenuto **è** bufferizzato. File che rientrano in questa categoria sono, ad esempio, dischi rigidi, partizioni, lettori floppy, lettori CD-ROM e così via. I file `/dev/hda`, `/dev/sda5` sono esempi di file in modalità a blocchi. Nella visualizzazione dei risultati di `ls -l`, questi file sono identificati con la lettera `'b'`.
3. ***Collegamenti simbolici*** (noti anche come "link simbolici"): questi file sono molto diffusi, e ampiamente utilizzati nella procedura di avvio del sistema in **Mandrake Linux** (si veda in proposito il capitolo *I file di avvio del sistema: init sysv*, pag. 59). Come suggerisce il nome, il loro scopo è quello di collegare i file in modo simbolico, il che significa che file di questo tipo possono riferirsi a un file esistente o meno; l'argomento sarà affrontato più avanti in questo capitolo. Molto spesso (e a torto, come vedremo in seguito) sono anche detti ***"soft link"*** . Sono identificati con una `'l'`.
4. ***Pipe con nome***: nel caso ve lo stiate chiedendo, sì, queste pipe sono molto simili a quelle usate con i comandi *shell*, ma la loro particolarità è quella di avere dei nomi. Continuate a leggere per saperne di più. Sono molto rare, comunque, ed è assai improbabile che voi ne incontriate una durante le vostre esplorazioni del sistema; in caso questo succedesse, la lettera che le identifica è la `'p'`. Per saperne di più, date uno sguardo a *Pipe "anonime" e pipe con nome*, pag. 50.
5. ***Socket***: questo è il tipo di file usato per tutte le connessioni di rete. Soltanto alcune di esse hanno un nome, tuttavia. Inoltre esistono diversi tipi di socket, ma i collegamenti sono ammessi per un solo tipo; in ogni caso, questo argomento va ben oltre gli obiettivi di questo manuale. File di questo tipo vengono identificati con la lettera `'s'`.

Ecco un esempio di ciascun tipo file:

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

8.2.2. Gli inodi

Gli inodi costituiscono, insieme all'assioma "Tutto è un file", il nocciolo fondamentale di un qualsiasi filesystem *UNIX*. Il termine originale inglese ***"inode"*** sta per *Information NODE*.

Gli inodi sono archiviati su disco in una **tabella degli inodi**. Esiste un inodo per ogni tipo di file che può essere archiviato dal filesystem, incluse directory, pipe con nome, file in modalità a caratteri, e così via. Il che ci porta a un'altra affermazione famosa: "L'inodo è il file". Gli inodi costituiscono anche il modo in cui *UNIX* identifica un file in maniera univoca.

Ebbene sì, avete letto bene: sotto *UNIX*, un file **non viene identificato in base al suo nome**, ma per mezzo del suo numero di inodo[2]. La ragione di questo va ricercata nel fatto che uno stesso file può avere più nomi, o

2. Importante: notate che i numeri di inodo sono univoci **per ciascun filesystem**, il che significa che su di un altro filesystem può esistere un inodo con lo stesso numero. Questo ci porta a capire la differenza tra inodi su disco e inodi in

anche nessun nome. Il nome di un file, sotto *UNIX*, non è che una voce in un inodo di directory; questa voce viene detta **collegamento**, o link. Passiamo quindi a esaminare i collegamenti in maggior dettaglio.

8.3. Collegamenti

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

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

L'opzione `-i` del comando `ls` visualizza il numero di inode, che si trova nel primo campo dell'output. Come potete vedere, prima che noi creassimo il file `a` non esistevano altri file nella directory. L'altro campo che ci interessa è il terzo, quello che riporta il numero di collegamenti al file (in realtà sono collegamenti all'inodo).

L'azione del comando `touch a` può essere scomposta in due azioni separate:

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

Ma adesso, se digitiamo:

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

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

Ora, se noi digitiamo:

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

notiamo che, malgrado il "file originale" sia stato cancellato, l'inodo esiste ancora, ma ora l'unico collegamento che fa riferimento ad esso è il file di nome `/home/maria/example/b`.

Su *UNIX*, pertanto, un file non ha un nome; ha, invece, uno o più collegamenti (***link***) in una o più directory.

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

Le connessioni di rete costituiscono esempi tipici di file che non hanno collegamenti (cioè, che non hanno nome): non vedrete mai il file corrispondente alla vostra connessione al sito Mandrake Linux (`www.mandrakelinux.com`) nell'albero delle directory, in qualunque directory lo cerchiate. Allo stesso modo, quando usate una ***pipe*** in una *shell*, l'inodo che corrisponde alla pipe esiste, ma non è collegato.

memoria: mentre due inodi su disco possono avere lo stesso numero, a patto che si trovino su filesystem diversi, gli inodi in memoria devono poter disporre di un numero univoco su tutto il sistema. Per garantire l'univocità, si può usare, ad esempio, una tabella hash che associ il numero di inodo all'identificatore del dispositivo.

8.4. Pipe "anonime" e pipe con nome

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

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

Un fatto che non potrete osservare in questo esempio (in quanto avviene troppo rapidamente per essere visto) è che le azioni di scrittura sulle pipe creano un blocco. Questo significa che quando il comando `ls` scrive su una pipe, rimane bloccato finché un processo dal lato opposto legge dalla pipe. Per poter visualizzare questo effetto, potete creare delle pipe con nome, che, a differenza delle pipe usate dalle *shell*, hanno un proprio nome (cioè, sono visibili nel filesystem, mentre le pipe della *shell* non lo sono)[3]. Il comando per creare questo tipo di pipe è `mkfifo`:

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

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

```
$ head -5 <una_pipe
  #
  # Il programma si blocca, sospendiamone l'esecuzione: Ctrl-z
```

3. Esistono anche altre differenze tra i due tipi di pipe, ma sono al di là degli obiettivi di questo libro.

```
  #
zsh: 741 suspended  head -5 < una_pipe
  #
  # Mettiamolo in background...
  #
$ bg
[1]  + continued  head -5 < una_pipe
  #
  # ...e diamogli da mangiare :)
  #
$ ls -d /proc/[0-9] >la_stessa_pipe
$ /proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
[1]  + 741 done       head -5 < una_pipe
$
```

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

8.5. File "speciali": file in modalità a caratteri e file in modalità a blocchi

Come abbiamo detto in precedenza, si tratta di file creati dal sistema oppure da periferiche collegate alla vostra macchina. Abbiamo anche menzionato il fatto che il contenuto di file in modalità a blocchi è bufferizzato, mentre quello di file in modalità a caratteri non lo è. Per meglio illustrare questo concetto, inserite un floppy nel drive e digitate il comando che segue due volte di seguito:

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

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

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

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

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

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

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

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

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

8.6. I link simbolici e le limitazioni degli "hard" link

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

I collegamenti simbolici sono file di un tipo particolare, il cui unico contenuto è una stringa qualsiasi, che può riferirsi a un nome di file esistente o meno. Quando utilizzate un collegamento simbolico dalla riga di comando o in un programma, di fatto accedete al file al quale esso si riferisce, se esiste. Ad esempio:

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

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

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

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

- non è possibile creare un collegamento a un inodo che si trova su di un filesystem diverso rispetto a tale inodo, per un semplice motivo: il contatore di link è conservato all'interno dell'inodo stesso, e gli inodi non possono essere condivisi fra filesystem. I collegamenti simbolici, invece, permettono questa operazione;
- non è possibile collegare delle directory, poiché abbiamo visto che il contatore di link per una directory ha uno scopo particolare. Potete invece creare un collegamento simbolico a una directory e usarlo come se fosse effettivamente una directory.

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

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

8.7. Attributi dei file

Come il filesystem FAT dispone di attributi per i file (archivio, file di sistema, nascosto), anche *ext2fs* ne ha di propri, ma differenti. Ne parliamo qui per ragioni di completezza, ma sono usati molto raramente. Se desiderate davvero un sistema sicuro, comunque, proseguite nella lettura.

Esistono due comandi per modificare gli attributi dei file: `lsattr(1)` e `chattr(1)`. Come avrete probabilmente immaginato, `lsattr` elenca (*LiSt*) gli attributi, mentre `chattr` li cambia (*CHange*). Gli attributi di cui stiamo parlando possono essere assegnati solo a directory e file normali. Sono ammessi i seguenti attributi:

1. `A` (*no Access time*, niente data di accesso): se un file o una directory presenta questo attributo, tutte le volte che vi si accede, sia in lettura che in scrittura, l'indicazione cronologica relativa all'ultimo accesso

resta inalterata. Questo può tornare utile, ad esempio, nel caso di file o directory che vengono richiesti in lettura molto spesso, soprattutto se si considera che questo parametro è l'unico tra quelli di un inodo che cambia quando il file è aperto solo in lettura.

2. `a` (*append only*, aggiungi soltanto): se un file presenta questo attributo ed è accessibile in scrittura, l'unica operazione possibile sarà quella di aggiungere dati al suo precedente contenuto. Per una directory, invece, questo significa che è possibile soltanto aggiungervi file, ma non rinominare o cancellare uno dei file già presenti. Solo `root` può impostare o rimuovere questo attributo.
3. `d` (*no dump*, niente dump): `dump (8)` è il programma standard per effettuare il backup dei dati in *`UNIX`*. Effettua una copia di qualsiasi filesystem per il quale sia impostato 1 come valore del *dump counter* nel file `/etc/fstab` (si veda il capitolo *Filesystem e punti di mount*, pag. 43). Ma se un file o una directory presentano questo attributo, a differenza degli altri non verranno presi in considerazione quando verrà usato il comando dump. Si noti che, nel caso delle directory, questo riguarda anche tutte le sotto-directory e i file in esse contenuti.
4. `i` (*immutable*, non modificabile): un file o una directory che presentino questo attributo non possono in nessun modo essere modificati: non possono essere rinominati, nessun ulteriore collegamento ad essi può essere creato[4] e non possono essere cancellati. Soltanto `root` può impostare o disabilitare questo attributo. Notate che, come conseguenza, neanche la data di accesso può essere modificata, perciò quando questo attributo è attivo non è necessario ricorrere all'attributo `A`.
5. `s` (*secure deletion*, cancellazione sicura): quando viene cancellato un file o una directory che presenta questo attributo i blocchi che occupava su disco vengono sovrascritti con degli zeri.
6. `S` (*Synchronous mode*, modalità sincrona): se un file o una directory presentano questo attributo, tutte le modifiche apportate sono sincrone e vengono immediatamente registrate su disco.

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

4. Ormai dovreste sapere bene cosa significa "aggiungere un collegamento", sia per un file che per una directory **:-)**.

Capitolo 9. Il filesystem /proc

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

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

9.1. Informazioni sui processi

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

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

Notate, però, che come utenti voi potete (giustamente) visualizzare solo le informazioni relative ai vostri processi, e non a quelli di altri utenti. Quindi proviamo a diventare root e a vedere che informazioni possiamo ottenere dal processo 127:

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

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

1. cmdline: questo (pseudo-)file contiene l'intera linea di comando che è stata utilizzata per avviare il processo. Non è formattata: non ci sono spazi tra il nome del programma e i suoi argomenti, e non c'è neanche il ritorno a capo a fine riga. Per poterlo vedere potete digitare: perl -ple 's,\00, ,g' cmdline.
2. cwd: questo collegamento simbolico punta all'attuale directory di lavoro ("current working directory", da qui il nome) del processo.
3. environ: questo file contiene tutte le variabili d'ambiente definite per il processo, nella forma VARIABLE=valore. Come per cmdline, il contenuto non è minimamente formattato: nessun ritorno a capo, né per separare le diverse variabili, né alla fine. Un modo per vederlo è digitare perl -pl -e 's,\00,\n,g' environ.
4. exe: questo è un link simbolico che punta al file eseguibile corrispondente al processo in esecuzione.
5. fd: questa sottodirectory contiene la lista dei descrittori di file attualmente aperti dal processo. Si veda più avanti.

6. `maps`: stampando (ad esempio tramite `cat`) il contenuto di questa pipe denominata (o "pipe con nome") potrete vedere le sezioni dello spazio di indirizzamento del processo che sono attualmente assegnate a un file. I campi da sinistra a destra sono: lo spazio di indirizzamento relativo all'assegnazione, i permessi associati all'assegnazione, lo scostamento (ingl. "offset") da cui parte l' assegnazione rispetto all'inizio del file, il dispositivo su cui si trova il file a cui è assegnato lo spazio, il numero di inodo del file, e infine il nome del file stesso. Quando il dispositivo è 0 e non è presente né numero di inodo né il nome del file, si tratta di assegnazioni anonime. Si veda mmap(2).
7. `root` Questo è un collegamento simbolico che punta alla directory considerata come radice dal processo. Di solito questa sarà `/`, ma si veda anche chroot(2).
8. `status` Questo file contiene varie informazioni sul processo: il nome dell'eseguibile, il suo stato attuale, i suoi PID e PPID, i suoi UID e GID reali ed effettivi, il suo utilizzo di memoria, e altro ancora.

Visualizzando il contenuto della directory `fd`, sempre per il processo 127, otterremo quanto segue:

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

Questa, infatti, è la lista dei descrittori di file aperti dal processo. Ogni descrittore aperto è indicato da un collegamento simbolico, il cui nome è il numero del descrittore stesso, e il quale punta al file aperto tramite il descrittore[1]. Notate anche i permessi dei collegamenti simbolici: questo è l'unico caso in cui hanno senso, poiché rappresentano i permessi con i quali è stato aperto il file corrispondente al descrittore.

9.2. Informazioni sull'hardware

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

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

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

```
$ cat interrupts
           CPU0
  0:     127648          XT-PIC  timer
  1:       5191          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  5:       1402          XT-PIC  xirc2ps_cs
  8:          1          XT-PIC  rtc
 10:          0          XT-PIC  ESS Solo1
 12:       2631          XT-PIC  PS/2 Mouse
 13:          1          XT-PIC  fpu
 14:      73434          XT-PIC  ide0
 15:      80234          XT-PIC  ide1
NMI:          0
$ cat ioports
0000-001f : dma1
```

1. Se ricordate quello che abbiamo detto nella sezione *La redirezione e le pipe*, pag. 13, sapete già cosa rappresentano i descrittori 0, 1 e 2.

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

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

```
$ lsdev
Device            DMA   IRQ  I/O Ports
------------------------------------------------
cascade             4     2
dma                         0080-008f
dma1                        0000-001f
dma2                        00c0-00df
ESS                         1080-108f 10c0-10cf 10d4-10df 10ec-10ef
fpu                      13 00f0-00ff
ide0                     14 01f0-01f7 03f6-03f6 1050-1057
ide1                     15 0170-0177 0376-0376 1058-105f
keyboard                  1 0060-006f
Mouse                    12
pic1                        0020-003f
pic2                        00a0-00bf
rtc                       8 0070-007f
serial                      03f8-03ff
Solo1                    10
timer                     0 0040-005f
vga+                        03c0-03df
xirc2ps_cs                5 0300-030f
$
```

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

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

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

9.3. La sottodirectory /proc/sys

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

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

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

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

   Sostituite 1 con uno 0 se volete disabilitare il routing.

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

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

   e questo tipo di attacco diverrà impossibile.

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

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

Perché esse siano eseguite a ogni avvio del sistema potreste aggiungere queste due righe nel file `/etc/rc.d/rc.local`, in modo da evitare di doverle digitare ogni volta; un'altra soluzione è modificare il file `/etc/sysctl.conf`, si veda sysctl.conf(5).

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

Capitolo 10. I file di avvio del sistema: init sysv

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

10.1. In principio fu init

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

Il file di configurazione di `init` è `/etc/inittab`. C'è una pagina di manuale che ne descrive l'uso nei minimi dettagli (inittab(5)), ma qui vedremo solo alcune delle opzioni di configurazione.

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

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

Questa istruzione dice a `init` che, al momento dell'inizializzazione del sistema, lo script `/etc/rc.d/rc.sysinit` deve essere eseguito prima di ogni altra operazione (`si` sta per *System Init*). In seguito `init` controlla la riga che contiene l'indicazione `initdefault` per determinare il runlevel predefinito:

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

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

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

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

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

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

10.2. I runlevel

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

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

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

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

- 0: arresto completo della macchina;

1. Ora sapete perché mettere la directory `/sbin` su un filesystem diverso da root è una pessima idea **:-)**
2. Modificando il file `/etc/inittab` potete, se lo desiderate, aumentare fino a un massimo di 64 il numero delle console virtuali, oppure ridurlo, seguendo sempre la stessa sintassi. Ma non dimenticate che anche *X* gira su una console virtuale, quindi lasciategliene almeno una!

- 1: modalità***monoutente***;, va utilizzata in caso di gravi problemi del sistema o per operazioni di recupero in seguito a un crash.
- 2: modalità ***multiutente***, senza accesso alla rete;
- 3: modalità multiutente, stavolta con accesso alla rete;
- 4: inutilizzato;
- 5: come il 3, ma con il lancio dell'interfaccia grafica;
- 6: riavvio del sistema.

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

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

Come potete vedere, tutti i file di questa directory sono link simbolici, e tutti hanno un nome in un formato particolare. Questo formato corrisponde allo schema

```
<S|K><ordine><nome_servizio>
```

. La lettera `S` sta per *Start*, "avvia" un servizio, e la `K` per *Kill*, "ferma" il servizio. Gli script vengono eseguiti secondo un ordine ascendente, e se due script hanno lo stesso numero vengono eseguiti in ordine alfabetico. Noterete anche che ciascun link simbolico punta a uno script che si trova nella directory `/etc/rc.d/init.d` (fatta eccezione per `local`), ed è quest'ultimo a controllare effettivamente ogni singolo servizio.

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

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

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

III. Uso avanzato

Capitolo 11. La stampa

Questo capitolo è diviso in due parti: *Installazione e gestione delle stampanti*, pag. 63, per chi provvede alla normale amministrazione del proprio sistema; e *La stampa di documenti*, pag. 68, che spiega come configurare e utilizzare uno strumento di stampa avanzato: *XPP*.

11.1. Installazione e gestione delle stampanti

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

11.1.1. Installazione di `cups` e uso della sua interfaccia web

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

Potete gestire le vostre stampanti con *cups* in due modi diversi: tramite un'interfaccia web, oppure richiamando il programma di gestione delle stampanti di *KDE* (**Sistema→Gestore stampa**). Abbiamo deciso di descrivere l'interfaccia web perché è accessibile da qualunque piattaforma, e permette la gestione di stampanti remote. Troverete una documentazione esauriente in merito agli strumenti di stampa di *KDE* nel loro sito web (`http://printing.kde.org/`).

Una volta lanciato il vostro navigatore web preferito, digitate `http://localhost:631/` nel campo di testo relativo al percorso o all'URL. Vi verrà mostrato il menu principale di *cups* (Figura 11-1).

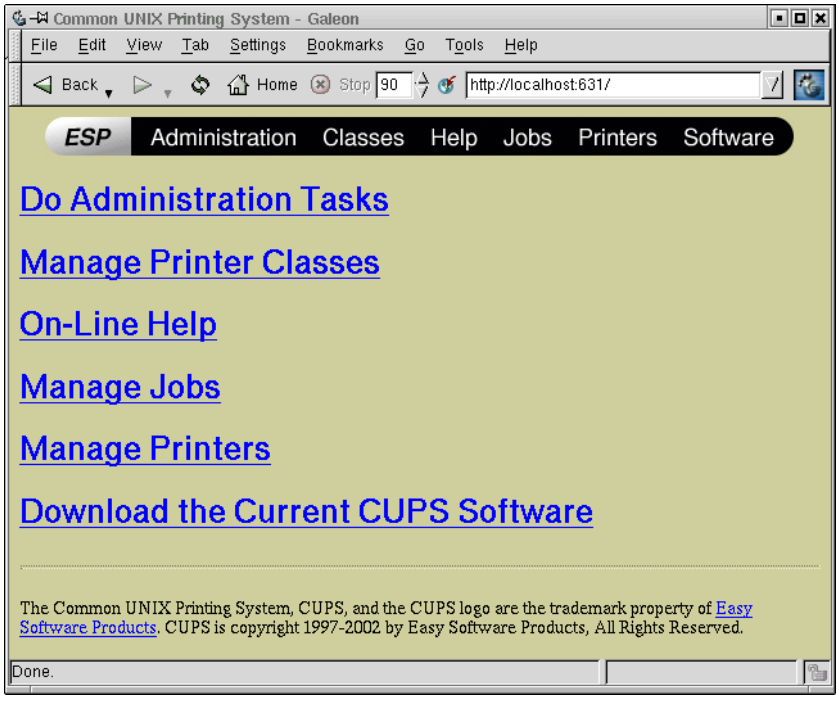


Figura 11-1. La pagina di benvenuto di `cups`

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

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

11.1.2. Configurazione di una nuova stampante

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

La pagina **Manage Printers** (Figura 11-2) dovrebbe quindi risultare vuota, per il momento.

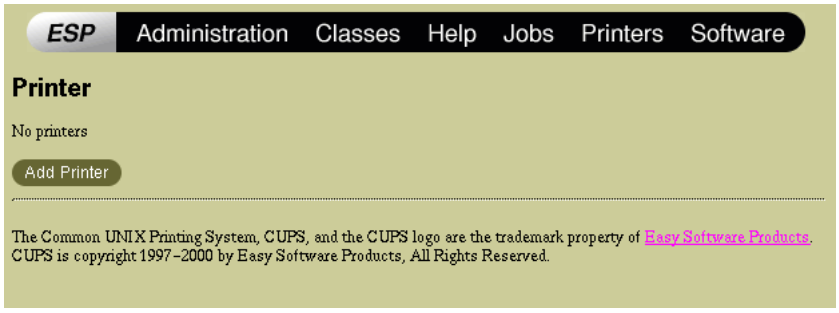


Figura 11-2. La lista priva di stampanti di cups

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

La prima volta che vorrete portare a termine una funzione amministrativa, come installare una nuova stampante, con *cups*, vi verrà richiesta la password di `root` (Figura 11-3). Per continuare, dovete inserire il login e la password di root.

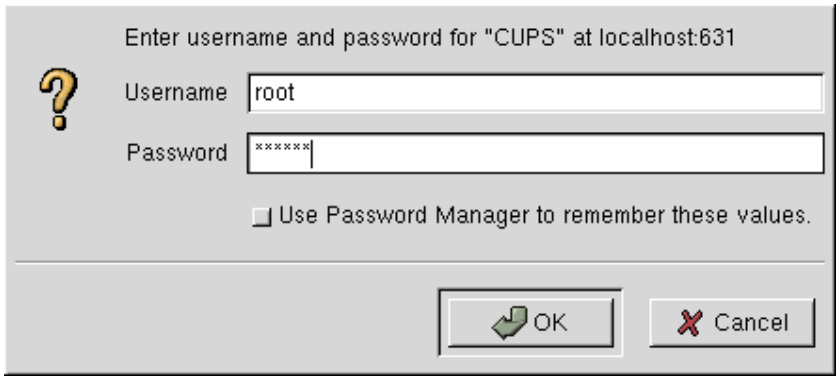


Figura 11-3. La finestra di login di cups

11.1.2.1. Informazioni informali riguardo la stampante

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

Figura 11-4. Installazione di una nuova stampante, passo 1

L'unico campo indispensabile è il nome della stampante.

Ricordatevi di collegare la stampante al computer e di accenderla, in modo che venga identificata automaticamente durante la configurazione.

11.1.2.2. Connessione della stampante

È necessario comunicare a *cups* dove si trova fisicamente la stampante. Per una stampante connessa direttamente al vostro computer, dovete scegliere **Parallel Port**, **Serial Port**, o USB, a seconda del tipo di connessione. Si noti che, se avete una stampante di tipo seriale, sarà necessario installare il pacchetto `cups-serial`. Le stampanti di tipo USB devono essere collegate al computer e accese.

Figura 11-5. Installazione di una nuova stampante, passo 2

Sono disponibili molti tipi di connessione:

Ethernet

Per stampanti che sono direttamente connesse a una rete locale.

LPD/LPR

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

Samba

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

Questi tipi di connessione non verranno documentati in questa sede, infatti la loro configurazione è piuttosto semplice se avete raccolto in precedenza le informazioni necessarie.

11.1.2.3. Inserimento della marca della stampante

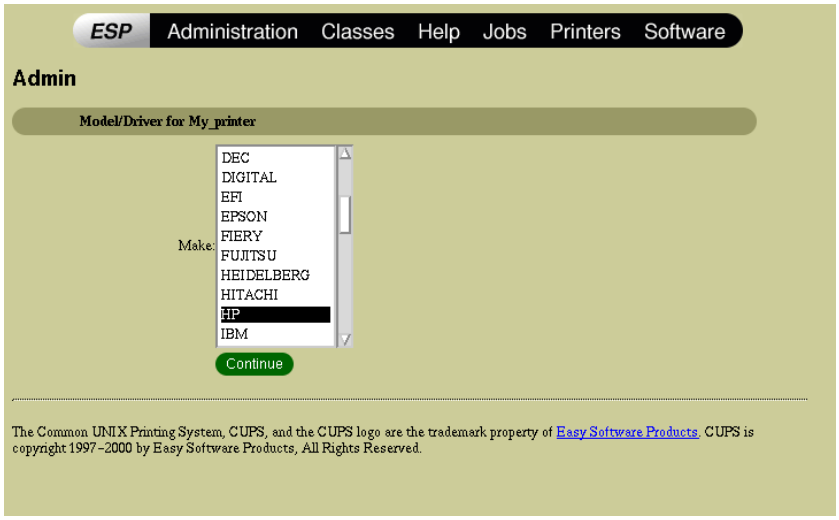


Figura 11-6. Installazione di una nuova stampante, passo 3

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

11.1.2.4. Scelta del modello della stampante

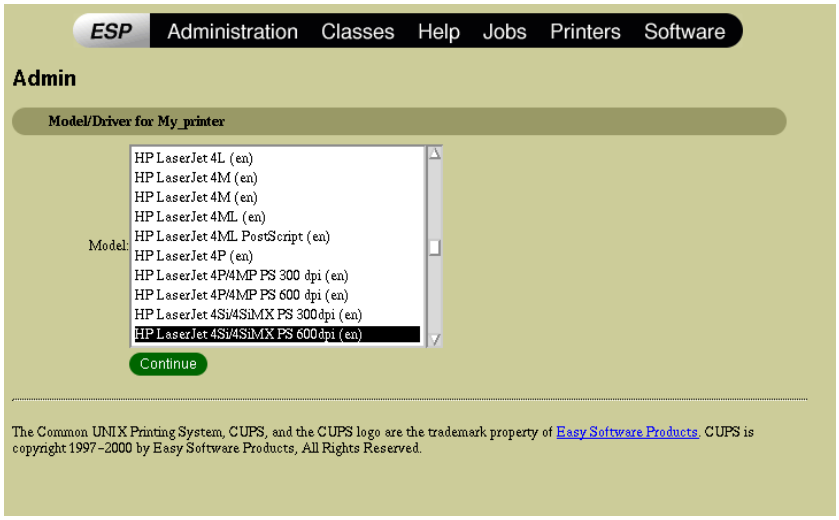


Figura 11-7. Installazione di una nuova stampante, passo 4

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

Se tutto è andato bene, dovreste poter vedere la vostra nuova stampante nella pagina **Printers**.

11.1.2.5. Configurazione finale e test della stampante

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

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

11.1.3. Una nota riguardo la sicurezza

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

```
#BrowseInterval 30
```

con

```
BrowseInterval 0
```

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

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

```
service cups restart
```

come `root`.

11.1.4. Gestione della coda di stampa

Questa caratteristica è particolarmente utile per stampanti caratterizzate da un elevato carico di lavoro, ma potreste comunque averne bisogno in maniera occasionale, ad esempio per annullare un comando di stampa di 10.000 pagine dato per errore. Quando inviate un lavoro alla stampante potete controllare se ne avete altri in attesa e, se siete l'amministratore della macchina da cui dipende la stampante, anche quelli di tutti gli utenti: è sufficiente collegarsi alla pagina relativa alla stampante in questione (Figura 11-8).

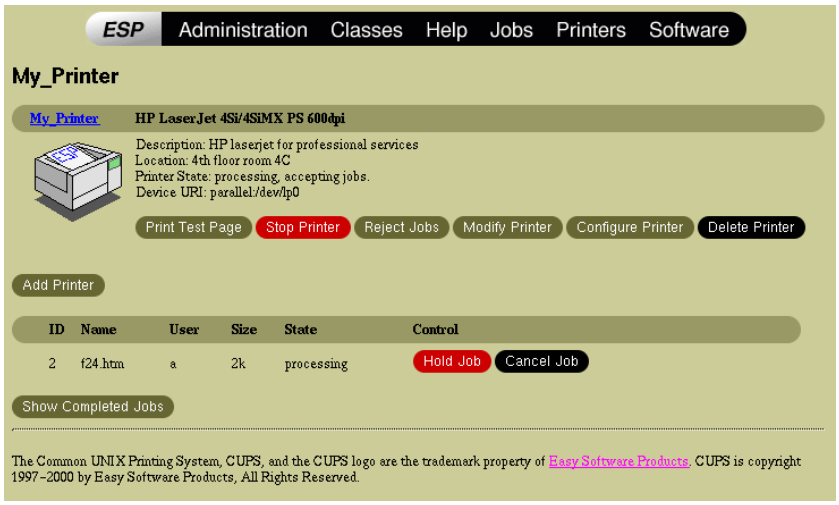


Figura 11-8. La pagina relativa allo stato della stampante

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

- **Hold Job** ("sospendi lavoro"): per mettere il lavoro in una lista d'attesa, verrà mandato in stampa soltanto quando tornerete alla pagina relativa e cliccherete sul pulsante verde **Release Job** ("riprendi lavoro").
- **Cancel Job** ("annulla lavoro"): per cancellare questo lavoro in modo definitivo, depennandolo dalla coda di stampa.

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

Se siete interessati a funzionalità avanzate per quanto riguarda la gestione della coda di stampa, potete utilizzare il *Gestore stampa* del *Centro di controllo KDE*.

11.2. La stampa di documenti

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

11.2.1. Semplice stampa di un file

Supponiamo che abbiate salvato sul vostro disco rigido un'immagine scaricata da un sito web, e che desideriate stamparla. Per prima cosa lanciate *XPP* dal menu **Applicazioni+Publishing→X Printing Panel**, vedrete comparire la finestra principale (Figura 11-9).

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

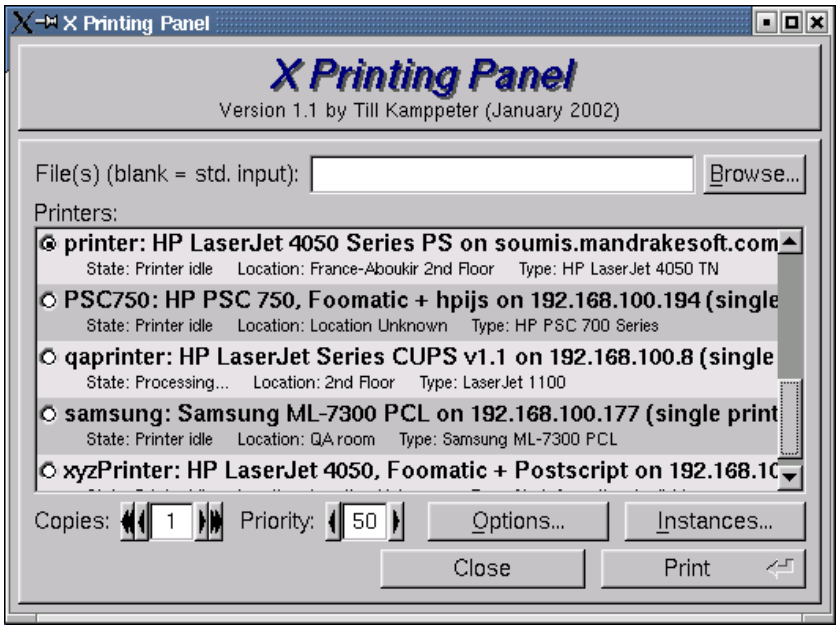


Figura 11-9. La finestra principale di `XPP`

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

Per scegliere il file da stampare, cliccate sul pulsante **Browse...**, comparirà una finestra di dialogo (Figura 11-10) grazie alla quale potrete navigare all'interno del vostro disco rigido e scegliere il file che desiderate stampare. Notate che, se sapete già dove si trova tale file, potete anche digitare il percorso completo nel campo di immissione testo accanto al pulsante.

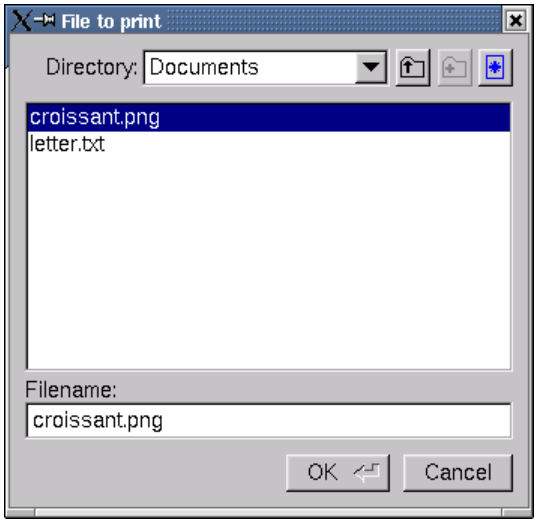


Figura 11-10. Scelta di un file con `XPP`

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

Se desiderate usare *XPP* come programma (filtro) di stampa per documenti mandati in stampa da altre applicazioni (come *Galeon*, o uno dei molti altri che lo consentono), tutto quello che dovete fare è cambiare il comando di stampa: in genere nella finestra di dialogo relativa alle opzioni di stampa è presente un campo di testo che contiene la stringa **lpr**, sostituitela con xpp, e questo è tutto.

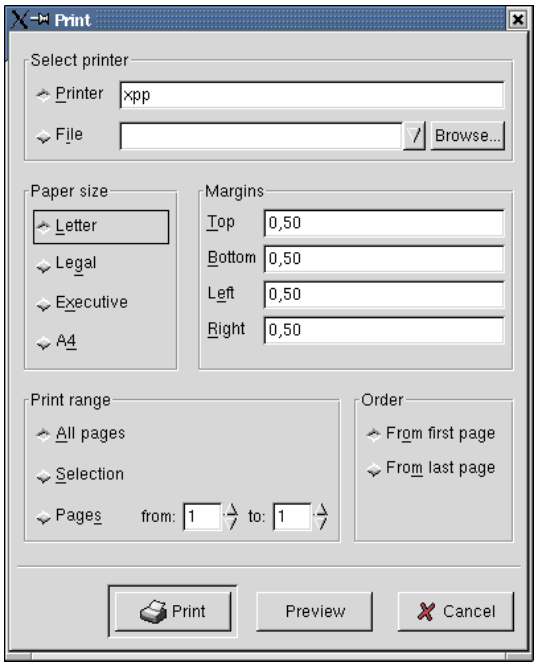


Figura 11-11. La finestra di stampa di `netscape`

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

11.2.2. Configurazione avanzata

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

In secondo luogo, grazie al pulsante **Options...** potete accedere a una finestra di dialogo che presenta diverse schede di configurazione dei parametri di stampa (Figura 11-12).

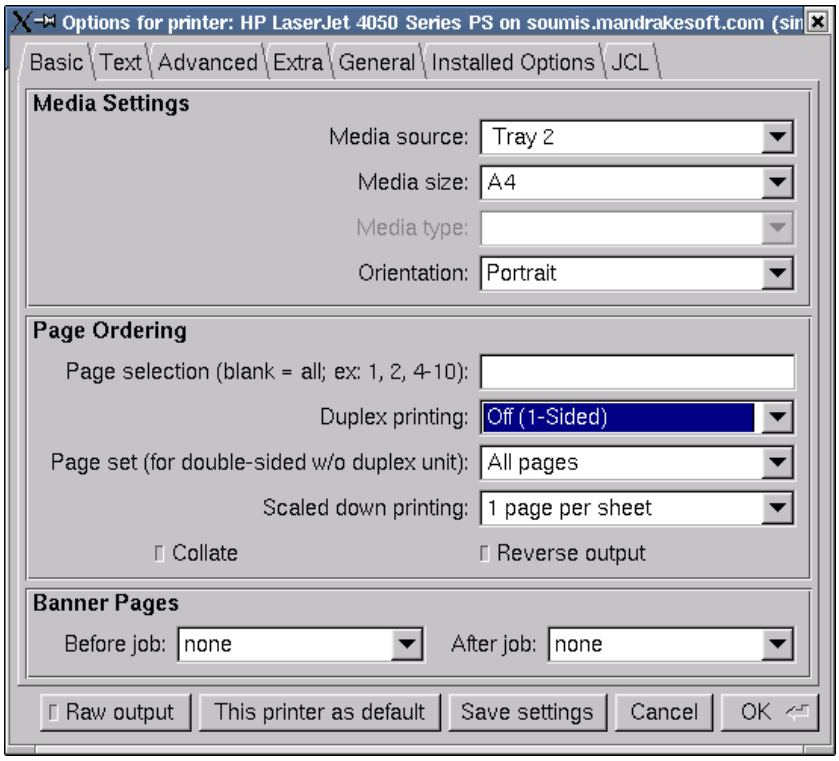


Figura 11-12. La scheda delle opzioni base di `XPP`

La prima scheda è divisa in due parti:

Media Settings

In questa sezione potete scegliere i valori di tre parametri, per quanto, a seconda delle caratteristiche della stampante, non è certo che siano tutti o anche solo in parte applicabili. Il primo consiste nella scelta tra il vassoio per la carta e l'alimentazione manuale. Quindi abbiamo le dimensioni della carta, e infine l' orientamento della stampa.

Page Ordering

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

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

La scheda **Text** (Figura 11-13) offre opzioni ancora più raffinate per cambiare il modo in cui vengono stampati i file di testo.

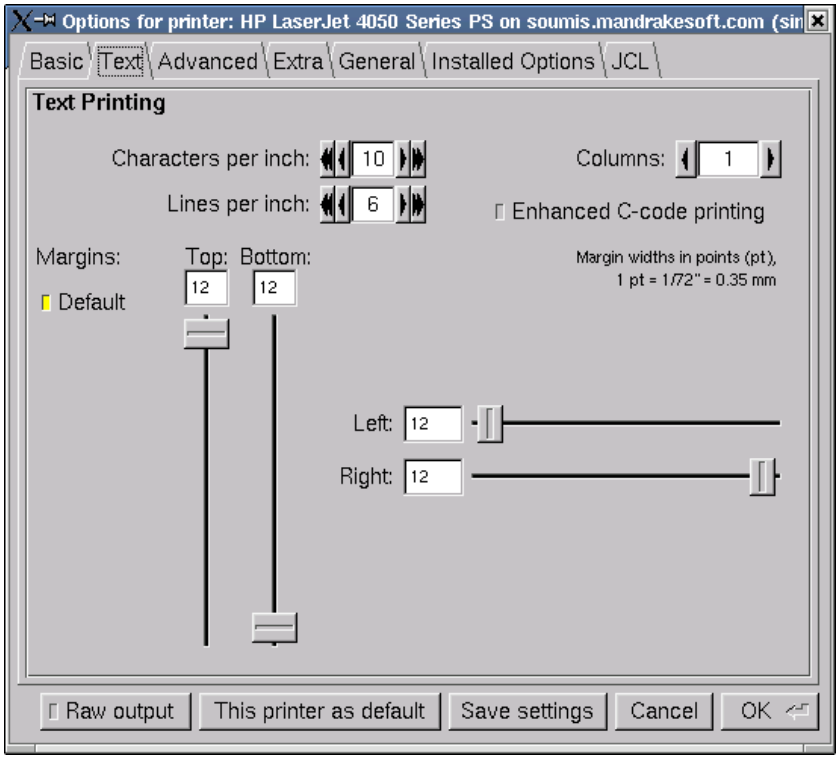


Figura 11-13. La scheda relativa alle opzioni per i file di testo di `XPP`

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

La scheda **Advanced** (Figura 11-14) offre opzioni finalizzate a modificare l'aspetto della pagina.

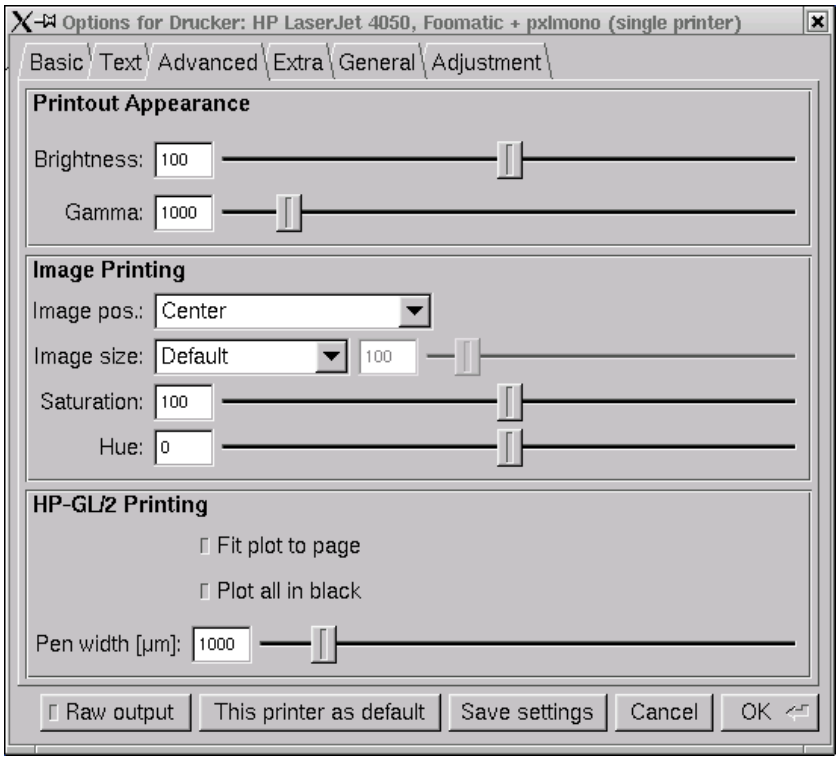


Figura 11-14. La scheda relativa alle opzioni avanzate di `XPP`

Questa scheda è divisa in tre parti:

Printout Appearance

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

Image Printing

I primi due parametri cambiano la posizione relativa e le dimensioni di un'immagine in rapporto alla pagina; gli ultimi due modificano il modo in cui vengono corretti i colori.

HP-GL/2 Printing

Tre parametri:

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

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

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

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

Capitolo 12. msec – gli strumenti di Mandrake per la sicurezza

12.1. Introduzione a msec

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

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

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

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

12.2. Impostazione del livello di sicurezza

MSEC è un RPM di base. Questo significa che la sua installazione avviene automaticamente durante l'installazione di un sistema **Mandrake Linux**.

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

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

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

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

12.2.1. Livello 0

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

- Il mio computer è connesso a Internet?
- Il mio computer è connesso in rete con altri computer?
- Questo computer sarà usato da altre persone?
- Sul mio computer ci sono file riservati ai quali non devono accedere altre persone?
- È possibile che io danneggi il sistema a causa della mia incompleta conoscenza di *GNU/Linux*?

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

12.2.2. Livello 1

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

12.2.3. Livello 2

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

12.2.4. Livello 3

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

Dal punto di vista dell'utente, il sistema in questo caso è un po' più chiuso, e saranno necessarie delle nozioni di base su *GNU/Linux* per compiere alcune operazioni particolari. La sicurezza di questo livello è paragonabile a quella offerta da una distribuzione **Red Hat** standard o da una qualsiasi **Mandrake Linux** precedente.

12.2.5. Livello 4

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

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

12.2.6. Livello 5

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

12.3. Caratteristiche dei livelli di sicurezza

Quella che segue è la descrizione delle varie caratteristiche di sicurezza che ogni livello introduce nel sistema. Queste caratteristiche possono essere di due tipi: di rinforzo del sistema, nel qual caso vengono apportate modifiche al sistema stesso, o di controllo periodico, per le quali ciò non avviene.

12.3.1. Rinforzo del sistema

Le caratteristiche del primo tipo, infatti, modificano i permessi, i proprietari e i gruppi dei file e delle directory del sistema, in base al livello di sicurezza impostato.

Esse modificano anche i file di configurazione per adeguarli al livello selezionato, e riavviano eventuali programmi che ne abbiano bisogno per poter rendere attive le modifiche effettuate. Le funzioni di rinforzo del sistema si riattivano ogni ora, e tutte le modifiche vengono registrate in *syslog*.

Caratteristica \ Livello	**0**	**1**	**2**	**3**	**4**	**5**
umask per gli utenti	002	002	002	002	077	077
umask per `root`	002	002	002	002	002	077

Caratteristica \ Livello	**0**	**1**	**2**	**3**	**4**	**5**
accesso senza password	sì					
autorizzati a connettersi al server *X*	tutti	locale	locale	nessuno	nessuno	nessuno
il server *X* riceve connessioni da	tutti	tutti	tutti	tutti	locale	locale
timeout della shell	nessuno	nessuno	nessuno	nessuno	3600	900
`su` solo da membri del gruppo wheel						sì
dimensione della cronologia della shell					10	10
`sulogin` in runlevel 1					sì	sì
proibisci elenco utenti nei display manager					sì	sì
ignora echo ICMP					sì	sì
ignora messaggi di errore ICMP simulati					sì	sì
login di root diretto	sì	sì	sì	sì		
abilita libsafe					sì	sì
proibisci agli utenti `at` e `crontab`					sì	sì
scadenza password (in giorni)					60	30
proibisci *autologin*				sì	sì	sì
consenti "issue" (disegno per login in modo testo)	tutti	tutti	tutti	locale	locale	nessuno
`.` in `$PATH`	sì	sì				
avvisi nel file `/var/log/security.log`		sì	sì	sì	sì	sì
avvisi direttamente su *tty*			sì	sì	sì	sì
avvisi in `syslog`			sì	sì	sì	sì
avvisi inviati via e-mail a `root`			sì	sì	sì	sì
registra tutti gli eventi del sistema anche su `/dev/tty12`				sì	sì	sì
solo `root` può usare `ctrl-alt-del`					sì	sì
i servizi sconosciuti sono disabilitati					sì	sì
accetta connessioni da	tutti	tutti	tutti	tutti	locale	nessuno

12.3.1.1. umask per gli utenti

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

12.3.1.2. umask per root

La stessa cosa, ma per `root`.

12.3.1.3. accesso senza password

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

12.3.1.4. autorizzati a connettersi al server X

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

12.3.1.5. Il server X riceve connessioni da...

1. `tutti`: il server *X* riceve connessioni via TCP.
2. `locale`: il server *X* riceve connessioni da socket *UNIX*.

12.3.1.6. Timeout della shell

Se l'utente è inattivo per un certo numero di secondi, la shell termina.

12.3.1.7. su solo da membri del gruppo wheel

Solo i membri del gruppo wheel sono autorizzati a usare `su` per assumere l'identità di `root`.

12.3.1.8. Dimensione della cronologia della shell

È il numero di comandi che rimangono memorizzati al termine di una sessione della shell.

12.3.1.9. sulogin in runlevel 1

`sulogin` serve a proteggere l'accesso all'initlevel di un singolo utente (è necessario digitare la password di `root` per ottenere l'accesso).

12.3.1.10. Proibisci elenco utenti nei display manager

Impedisce che sia mostrato l'elenco degli utenti presenti sul sistema in *KDM* e `GDM`.

12.3.1.11. Ignora echo ICMP

Non risponde alle richieste di ping.

12.3.1.12. Ignora messaggi di errore ICMP simulati

Evita di gestire i messaggi di errore ICMP simulati.

12.3.1.13. Login di root diretto

Consente il login di root senza usare il programma `su`.

12.3.1.14. Abilita libsafe

Abilita la protezione libsafe (solo se è installato il pacchetto libsafe).

12.3.1.15. Proibisci agli utenti at e crontab

Gli utenti non possono usare i programmi `at` e `crontab`, mentre `root` può usarli. Se volete darne l'autorizzazione solo ad alcuni utenti, aggiungeteli rispettivamente in `/etc/at.allow` e `/etc/cron.allow`.

12.3.1.16. Scadenza password

Le password scadono dopo un certo numero di giorni; gli utenti devono cambiare la propria password prima della data di scadenza se non vogliono che i loro account siano disattivati.

12.3.1.17. Proibisci autologin

Non è possibile usare il programma *autologin*.

12.3.1.18. Consenti "issue"

Se impostato su `nessuno`, non sarà mostrato alcun disegno in nessun login in modo testo. Con `locale`, viene mostrato il disegno solo sulla console locale. Se impostato su `tutti`, il disegno viene mostrato in tutti i tipi di login.

12.3.1.19. . in $PATH

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

12.3.1.20. Avvisi nel file security.log

Tutti gli avvisi generati dalla verifica quotidiana vengono registrati nel file di nome `/var/log/security.log`.

12.3.1.21. Avvisi direttamente su tty

Tutti gli avvisi generati dalla verifica quotidiana vengono mostrati direttamente sulla console dell'utente `root`.

12.3.1.22. Avvisi in syslog

Gli avvisi generati dalla verifica quotidiana vengono indirizzati verso il servizio *syslog*.

12.3.1.23. Avvisi inviati via email a root

Gli avvisi generati dalla verifica quotidiana vengono anche inviati via email a `root`.

12.3.1.24. I servizi sconosciuti sono disabilitati

Durante l'installazione di un pacchetto, i servizi vengono aggiunti tramite `chkconfig --add <servizio>` **solo** se il nome del servizio è presente nel file `/etc/security/msec/server`.

12.3.1.25. Accetta connessioni da...

1. `tutti`: tutti i computer possono connettersi alle porte aperte.
2. `locale`: solo *localhost* può connettersi alle porte aperte.
3. `nessuno`: nessun computer può connettersi alle porte aperte.

Questa protezione è garantita dal pacchetto *tcp wrapper*. Se volete consentire l'accesso a un servizio quando questo non è consentito a nessuno, usate il file `/etc/hosts.allow`. Ad esempio, se volete consentire connessioni *ssh* provenienti da qualsiasi luogo, aggiungete la seguente linea in `/etc/hosts.allow`:

```
sshd: ALL
```

Caratteristica \ Livello	0	1	2	3	4	5

12.3.2. Controlli periodici

Se il livello di sicurezza è superiore a 0, i controlli vengono effettuati ogni notte.

Caratteristica \ Livello	0	1	2	3	4	5
controllo generale di sicurezza			sì	sì	sì	sì
controllo dei file *suid* `root`			sì	sì	sì	sì
controllo MD5 dei file *suid* `root`			sì	sì	sì	sì
controllo file scrivibili				sì	sì	sì
controllo permessi				sì	sì	sì
controllo file gruppo *suid*				sì	sì	sì
controllo file senza proprietario				sì	sì	sì
controllo promiscuità				sì	sì	sì
controllo porte in ascolto				sì	sì	sì
controllo integrità file `passwd`				sì	sì	sì
controllo integrità file `shadow`				sì	sì	sì
controllo integrità dal database RPM				sì	sì	sì

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

- controllo file *suid* `root`
- controllo MD5 dei file *suid* `root`
- controllo file scrivibili
- controllo file *suid*
- controllo file senza proprietario
- controllo porte in ascolto
- controllo integrità dal database RPM

12.3.2.1. Controllo generale di sicurezza

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

12.3.2.2. Controllo dei file suid root

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

12.3.2.3. Controllo MD5 dei file suid root

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

12.3.2.4. Controllo file scrivibili

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

12.3.2.5. Controllo permessi

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

12.3.2.6. controllo file sgid

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

12.3.2.7. Controllo file senza proprietario

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

12.3.2.8. Controllo promiscuità

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

12.3.2.9. Controllo porte in ascolto

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

12.3.2.10. Controllo integrità file passwd

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

12.3.2.11. Controllo integrità file shadow

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

12.3.2.12. Controllo integrità dal database RPM

Controlla che nessun file dei pacchetti installati sia stato modificato, e verifica che nessun pacchetto sia stato installato, rimosso o aggiornato dall'ultima volta che è stata effettuata la verifica.

12.4. Personalizzazione

12.4.1. Modifiche al filesystem

Le modifiche apportate al filesystem vengono memorizzate in `/var/lib/msec/perm.<level>`. Questo file può essere modificato tramite `/etc/security/msec/perm.local`, usando lo stesso formato del file in `/var/lib/msec/perm.*`

12.4.2. Modifiche ai file di configurazione

Le modifiche ai file di configurazione possono essere impostate tramite `/etc/security/msec/level.local`. Fate riferimento alla pagina di manuale di `mseclib` per ulteriori dettagli.

12.4.3. Modifiche ai controlli

La configurazione dei controlli si trova in `/var/lib/msec/security.conf`. Queste impostazioni possono essere scavalcate usando `/etc/security/msec/security.conf` con la stessa sintassi. Ad esempio, la variabile `MAIL_USER` può essere usata per cambiare l'indirizzo email al quale viene inviato il resoconto quotidiano.

Capitolo 13. La compilazione e l'installazione di software libero

Spesso i neofiti si domandano come si fa a installare del software libero direttamente dai sorgenti. Compilare personalmente i programmi è davvero facile, perché molti dei passi da seguire sono gli stessi a prescindere dal software specifico che desiderate installare. Lo scopo di questo capitolo è quello di guidare il principiante un passo alla volta, spiegandogli il significato di ogni mossa. Si presume che il lettore abbia una conoscenza minima di *UNIX* (dei comandi `ls` o `mkdir`, ad esempio).

Questo capitolo costituisce soltanto una guida, non un manuale di riferimento; il che spiega perché al termine vengono proposti numerosi riferimenti, in maniera da poter soddisfare le domande che restano senza risposta. Questa guida molto probabilmente è suscettibile di miglioramenti, dunque ogni osservazione o correzione in merito al contenuto sarà accolta con piacere.

13.1. Introduzione

Quello che distingue il software libero, detto anche "software liberamente distribuibile", dal software proprietario è la possibilità di accedere ai suoi sorgenti[1]. Questo comporta anche che il software libero viene spesso distribuito in forma di archivi dei sorgenti. Gli utenti dei programmi liberamente distribuibili, dunque, devono compilare i sorgenti di persona prima di poter usare il software, e questo può sembrare piuttosto inusuale ai principianti.

Esistono versioni compilate di molti programmi che appartengono al mondo del software libero: l'utente che va di fretta non deve far altro che installare questi binari precompilati. Una parte di questi programmi, tuttavia, non viene distribuita secondo questa modalità, oppure si tratta di versioni iniziali che ancora non vengono rilasciate in forma binaria; se usate un sistema operativo poco diffuso, inoltre, oppure una piattaforma hardware piuttosto rara, una gran parte del software libero non sarà ancora stata compilata per il vostro sistema. È importante sapere che compilare di persona il software da installare vi permette di abilitare soltanto le opzioni che vi interessano, o di estendere la funzionalità del programma aggiungendo estensioni di vario tipo, in maniera da ottenere un software che sia perfettamente adeguato ai vostri bisogni.

13.1.1. Prerequisiti

Per compilare software avete bisogno di:

- un computer con un sistema operativo funzionante;
- una conoscenza di base del sistema operativo che utilizzate;
- una certa quantità di spazio sul vostro disco rigido;
- un compilatore (in genere per il linguaggio *C*) e un programma di archiviazione (`tar`);
- un po' di cibo (in casi difficili la compilazione può impiegare un tempo considerevole). Un vero hacker mangia pizza - non caviale;
- qualcosa da bere (per la stessa ragione). Un vero hacker beve cola - per la caffeina;
- il numero di telefono del vostro amico smanettone che ricompila il kernel tutte le settimane;
- soprattutto pazienza, e in buona quantità!

Compilare dei sorgenti in genere non presenta grandi problemi, ma se non lo avete mai fatto prima il più piccolo intralcio può bloccarvi all'istante. Lo scopo di questo capitolo è quello di mostrarvi come potete evitare situazioni di questo tipo.

1. Questo non è del tutto esatto, in quanto anche alcune case che producono software proprietario mettono a disposizione i sorgenti. Ma, a differenza di quello che succede con il software libero, l'utente finale non ha il permesso di utilizzarli come vorrebbe.

13.1.2. Compilazione

13.1.2.1. Principi di base

Per tradurre un codice sorgente in un file binario è necessario effettuare una ***compilazione***; in genere i sorgenti sono scritti in *C* o *C++*, che sono i linguaggi più diffusi nella comunità del software libero (in particolare per *UNIX*). Una parte dei programmi liberamente distribuibili è scritta in linguaggi che non richiedono compilazione (il *perl*, ad esempio, o il linguaggio della *shell*), ma necessitano comunque di una configurazione.

La compilazione di programmi scritti in *C* viene effettuata per mezzo di un compilatore *C*, ovviamente, che di solito è `gcc`, il compilatore liberamente distribuibile creato dal progetto GNU (potrete saperne di più visitando il relativo sito web (`http://www.gnu.org/`)). La compilazione di un intero pacchetto software è un'operazione complessa, che si basa sulla compilazione in sequenza dei diversi file sorgenti (per una serie di motivi, è più semplice per il programmatore distribuire parti diverse del proprio lavoro in file separati). Al fine di rendere l'intero processo più semplice, queste operazioni ripetitive vengono gestite da un programma chiamato `make`.

13.1.2.2. Le quattro fasi della compilazione

Per capire come funziona la compilazione, così da poter risolvere eventuali problemi, dovete conoscere le quattro fasi da cui è composta. Lo scopo è quello di tradurre poco a poco un testo scritto in un linguaggio che è comprensibile per un essere umano ben allenato (e cioè il linguaggio *C*) in un linguaggio comprensibile da una macchina (o da un essere umano **molto** ben allenato, e anche in tal caso non sarebbero in molti a riuscirci). `gcc` esegue quattro programmi uno dopo l'altro, ciascuno dei quali gestisce una fase della compilazione:

1. `cpp`: Il primo passo consiste nel rimpiazzare le direttive (del ***preprocessore***) con delle istruzioni in *C* vere e proprie. Questo significa, di norma, inserire un header (`#include`) o definire una macro (`#define`). Al termine di questa fase viene generato un puro codice *C*.
2. *cc1*: Questa fase consiste nella conversione del codice *C* in ***linguaggio assembler***. Il codice generato dipende dall'architettura su cui deve girare il programma.
3. `as`: Questo passo consiste nella produzione di codice ***oggetto*** (o codice binario) a partire dal linguaggio assembler. Al termine di questa fase viene generato un file con estensione `.o`.
4. *ld*: Come ultimo passo (***linkage***) vengono collegati tutti i file oggetto (`.o`) e le librerie relative, e viene prodotto un file eseguibile.

13.1.3. Struttura di una distribuzione

Una distribuzione di software libero strutturata in maniera corretta presenta sempre la medesima organizzazione dei file:

- Un file `INSTALL`, che descrive la procedura di installazione.
- Un file `README`, che contiene informazioni generali riguardanti il programma (breve descrizione, autore, URL del sito dove lo si può scaricare, documentazione relativa al programma, collegamenti utili, etc.). Se il file `INSTALL` non è presente, di solito una breve descrizione della procedura di installazione è contenuta nel file `README`.
- Un file `COPYING`, che contiene la licenza o descrive le condizioni secondo le quali può essere distribuito il software. A volte al suo posto si trova un file `LICENSE`.
- Un file `CONTRIB` o `CREDITS`, che contiene una lista delle persone che hanno contribuito al progetto (partecipazione diretta, suggerimenti utili, programmi di terze parti, etc.).
- Un file `CHANGES` (o, meno frequentemente, un file `NEWS`), che contiene una lista degli ultimi miglioramenti e dei bug eliminati.
- Un file `Makefile` (si veda la sezione *make*, pag. 89), che permette di compilare il software (è un file di cui ha bisogno `make`). Spesso questo file all'inizio non esiste e viene generato durante il processo di configurazione.
- Molto spesso, un file `configure` o `Imakefile`, che permette di generare un nuovo file `Makefile`,
- Una directory che contiene i sorgenti, e dove viene in genere collocato il file binario alla fine della compilazione. Spesso viene chiamata `src`.

- Una directory che contiene la documentazione relativa al programma (normalmente in formato `man` o *Texinfo*), il cui nome spesso è `doc`.
- A volte, una directory che contiene dati specifici del software (si tratta, tipicamente, di file di configurazione, esempi di dati prodotti, o file che costituiscono risorse di vario tipo).

13.2. Decompressione

13.2.1. Archivi `tar.gz`

Il formato standard[2] per la compressione nei sistemi *UNIX* è il formato `gzip`, sviluppato dal progetto GNU, che viene considerato uno dei migliori strumenti per la compressione generica dei file.

`gzip` viene spesso associato a un programma che si chiama `tar`. `tar` è il superstite di tempi antidiluviani, quando gli operatori di computer archiviavano i loro dati su nastri. Oggi floppy disk e CD-ROM hanno rimpiazzato i nastri, ma `tar` viene ancora impiegato per creare archivi. Tutti i file presenti in una directory, ad esempio, possono essere riuniti in un unico file; quest'ultimo può essere facilmente compresso con `gzip`.

Questo è il motivo per cui gran parte del software libero è disponibile sotto forma di archivi `tar`, compressi con `gzip`. La loro estensione, perciò, è `.tar.gz` (o anche `.tgz` per brevità).

13.2.2. Uso di `GNU Tar`

Per decomprimere un archivio di questo tipo, è possibile usare `gzip` e poi `tar`. Ma la versione GNU di `tar` (`gtar`) permette di usare `gzip` ***al volo***, e di decomprimere un archivio quasi senza accorgersene (e senza il bisogno di spazio aggiuntivo su disco).

L'uso di `tar` segue questa sintassi:

```
tar <opzioni> <file .tar.gz> [file]
```

L'opzione `<file>` non è necessaria: se omessa, l'operazione richiesta verrà effettuata sull'intero archivio. Quindi non è necessario specificare questo argomento per estrarre il contenuto di un archivio `.tar.gz`.

Ad esempio:

```
$ tar xvfz guile-1.3.tar.gz
-rw-r--r-- 442/1002      10555 1998-10-20 07:31 guile-1.3/Makefile.in
-rw-rw-rw- 442/1002       6668 1998-10-20 06:59 guile-1.3/README
-rw-rw-rw- 442/1002       2283 1998-02-01 22:05 guile-1.3/AUTHORS
-rw-rw-rw- 442/1002      17989 1997-05-27 00:36 guile-1.3/COPYING
-rw-rw-rw- 442/1002      28545 1998-10-20 07:05 guile-1.3/ChangeLog
-rw-rw-rw- 442/1002       9364 1997-10-25 08:34 guile-1.3/INSTALL
-rw-rw-rw- 442/1002       1223 1998-10-20 06:34 guile-1.3/Makefile.am
-rw-rw-rw- 442/1002      98432 1998-10-20 07:30 guile-1.3/NEWS
-rw-rw-rw- 442/1002       1388 1998-10-20 06:19 guile-1.3/THANKS
-rw-rw-rw- 442/1002       1151 1998-08-16 21:45 guile-1.3/TODO
...
```

Tra le opzioni di `tar`:

- `v` rende `tar` prolisso. Questo significa che mostrerà sullo schermo tutti i file che trova nell'archivio. Se questa opzione viene omessa, l'operazione verrà eseguita in modo silenzioso.
- `f` è un'opzione indispensabile: se non è indicata, `tar` cercherà di usare un nastro invece di un file archivio (e cioè il dispositivo `/dev/rmt0`).
- `z` vi permette di usare `tar` con un archivio "gzippato" (cioè compresso con `gzip`, deve presentare un'estensione `.gz` o `.tgz`). Se vi dimenticate di usare questa opzione con questo tipo di archivio, `tar` mostrerà un messaggio d'errore. Viceversa, questa opzione non dev'essere impiegata con un archivio non compresso.

2. Un nuovo programma, chiamato `bzip2`, più efficiente per quel che riguarda i file di testo (e più esigente in materia di potenza di calcolo), viene usato sempre più di frequente. Si veda la sezione successiva *bzip2*, pag. 86 che se ne occupa specificamente.

tar vi consente di eseguire un certo numero di azioni su un archivio (estrazione, lettura, creazione, aggiunta di file...). Un'opzione stabilisce quale azione dev'essere compiuta dal programma:

- x: vi permette di estrarre file dall'archivio.
- t: elenca il contenuto dell'archivio.
- c: vi permette di creare un archivio. Potete usarla per fare una copia di backup dei vostri file personali, ad esempio.
- r: consente di aggiungere file alla fine di un archivio esistente. Non può essere usata con un archivio compresso.

13.2.3. bzip2

Un nuovo formato di compressione, chiamato bzip2, ha cominciato a rimpiazzare gzip nell'uso comune. bzip2 produce archivi più piccoli rispetto a gzip, ma non è ancora diventato uno standard. Le estensioni .tar.bz2 hanno iniziato a diffondersi soltanto in tempi recenti.

bzip2 viene usato anche per mezzo del comando tar, proprio come gzip. L'unico accorgimento da adottare è la sostituzione della lettera z con la j. Ad esempio:

```
$ tar xvjf foo.tar.bz2
```

Alcune distribuzioni invece usano, o hanno usato, l'opzione I:

```
$ tar xvfI foo.tar.bz2
```

Un altro metodo (che sembrerebbe essere più compatibile, ma richiede più tempo per essere digitato!):

```
$ tar --use-compress-program=bzip2 -xvf foo.tar.bz2
```

bzip2 deve essere installato e incluso nella vostra variabile d'ambiente PATH prima di lanciare tar.

13.2.4. Fatelo e basta!

13.2.4.1. Il modo più semplice

Adesso che siete pronti a decomprimere l'archivio, non dimenticate di eseguire questa operazione in veste di amministratori del sistema (root). Sarà necessario compiere azioni che non sono permesse a un utente normale, e anche se parte di quello che farete è alla portata di quest'ultimo, è più semplice essere root per tutto il tempo.

Il primo passo è quello di spostarsi nella directory /usr/local/src e copiarvi l'archivio. In tal modo, dovreste riuscire sempre a ritrovare l'archivio di partenza se il programma installato dovesse venir rimosso. Se non avete molto spazio a disposizione sul vostro disco rigido, salvate l'archivio su un floppy disk dopo aver installato il programma. Nulla vi vieta di cancellarlo, ma accertatevi di poterlo comunque reperire sul web nel caso doveste averne bisogno.

La decompressione di un archivio tar, in genere, dovrebbe portare alla creazione di una nuova directory (potete accertarvi che ciò avvenga grazie all'opzione t, prima di decomprimere l'archivio). Una volta creata, spostatevi in tale directory, in modo da esser pronti per continuare.

13.2.4.2. Il modo più sicuro

I sistemi *UNIX* (di cui *GNU/Linux* e *freebsd* costituiscono validi esempi) sono sistemi sicuri. Questo significa che gli utenti normali non possono né compiere operazioni che possano mettere a rischio il sistema (la formattazione di un disco rigido, ad esempio), né modificare i file di altri utenti. Tale condizione comporta anche un'immunizzazione del sistema per quanto riguarda i virus.

root, d'altra parte, può fare tutto - anche lanciare un programma malevolo. Avere a disposizione il codice sorgente vi permette di esaminarlo per escludere la presenza di codice malevolo (virus o cavalli di Troia). È meglio prestare la massima attenzione a questo problema[3].

3. Un proverbio del mondo BSD dice: "Non fidarti mai di un pacchetto di cui non hai i sorgenti."

Il nostro suggerimento, pertanto, è di creare un utente dedicato a compiti amministrativi (`free` o `admin`, ad esempio) usando il comando `adduser`. Questo utente deve avere il permesso di scrivere nelle seguenti directory: `/usr/local/src`, `/usr/local/bin` e `/usr/local/lib`, come pure tutta la sotto-gerarchia di `/usr/man` (è anche possibile che debba avere il permesso di copiare file altrove). Vi raccomandiamo di rendere questo utente proprietario delle directory necessarie, oppure di creare un gruppo per lui e di rendere tali directory accessibili in scrittura per il gruppo.

Dopo aver preso queste precauzioni, potete seguire le istruzioni della sezione *Il modo più semplice*, pag. 86.

13.3. Configurazione

Una conseguenza puramente tecnica del fatto di avere a disposizione i sorgenti creati dagli autori è il ***porting*** del software. Il software libero sviluppato per un sistema *UNIX* può essere utilizzato su tutti i sistemi *UNIX*, che siano aperti o proprietari, una volta effettuate alcune modifiche. Questo richiede una configurazione del software prima di procedere alla compilazione.

Esistono diversi sistemi di configurazione. Siete obbligati a usare quello scelto dall'autore del programma, e, talvolta, ne serve più d'uno. In genere potete:

- usare *AutoConf* (consultate la sezione *AutoConf*, pag. 87) se esiste un file chiamato `configure` nella directory principale della distribuzione;
- usare *imake* (consultate la sezione *imake*, pag. 89) se un file di nome `Imakefile` è presente nella directory principale della distribuzione;
- eseguire uno ***script*** dalla shell (ad esempio `install.sh`) seguendo le istruzioni contenute nel file `INSTALL` (o nel file `README`).

13.3.1. AutoConf

13.3.1.1. La teoria

AutoConf viene usato per configurare il software in maniera corretta. Crea i file richiesti dal processo di compilazione (`Makefile` ad esempio), e talvolta cambia i file sorgenti direttamente (ad esempio usando un file `config.h.in`).

Il principio di funzionamento di *AutoConf* è semplice:

- Il programmatore autore del software sa quali test sono richiesti per configurarlo (ad esempio: "quale versione di questa ***libreria*** state usando?"). Perciò li raccoglie in un file chiamato `configure.in`, seguendo una sintassi ben precisa.
- Quindi esegue *AutoConf*, che genera uno script di configurazione chiamato `configure` dal file `configure.in`. Questo script esegue i test richiesti quando il programma viene configurato.
- L'utente finale esegue lo script, e *AutoConf* configura tutto ciò che è necessario per la compilazione.

13.3.1.2. Un esempio pratico

Un esempio dell'uso di *AutoConf*:

```
$ ./configure
loading cache ./config.cache
checking for gcc... gcc
checking whether the C compiler (gcc  ) works... yes
checking whether the C compiler (gcc  ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for main in -lX11... yes
checking for main in -lXpm... yes
checking for main in -lguile... yes
checking for main in -lm... yes
checking for main in -lncurses... yes
checking how to run the C preprocessor... gcc -E
checking for X... libraries /usr/X11R6/lib, headers /usr/X11R6/include
```

```
checking for ANSI C header files... yes
checking for unistd.h... yes
checking for working const... yes
updating cache ./config.cache
creating ./config.status
creating lib/Makefile
creating src/Makefile
creating Makefile
```

È possibile aggiungere alcune opzioni per mezzo della linea di comando, in modo da avere un controllo maggiore su quello che viene generato da `configure`. Ad esempio:

```
$ ./configure --with-gcc --prefix=/opt/GNU
```

oppure (con *bash*):

```
$ export CC=`which gcc`
$ export CFLAGS=-O2
$ ./configure --with-gcc
```

o ancora:

```
$ CC=gcc CFLAGS=-O2 ./configure
```

13.3.1.3. Cosa fare... se non funziona?

Se `configure` si blocca, in genere comparirà un errore del tipo `configure: error: Cannot find library guile`; la maggior parte degli errori dello script `configure` hanno questo aspetto.

Questo errore significa che lo script `configure` non è riuscito a trovare una libreria (nell'esempio citato si tratta della libreria `guile`). Il metodo seguito da `configure` è quello di compilare un breve programma di prova che usa tale libreria. Se la compilazione di tale programma fallisce, ciò significa che non sarà possibile compilare il software in questione. Di conseguenza, venite avvertiti dell'errore.

- Scoprite la causa dell'errore esaminando la parte finale del file `config.log`, che contiene un registro di tutti i passi seguiti nel corso della configurazione. Il compilatore *C* è sufficientemente chiaro nei suoi messaggi d'errore, e questo vi aiuterà a risolvere il problema.
- Controllate che la sunnominata libreria sia installata correttamente: se non lo è, installatela (dai sorgenti, o usando un file binario precompilato) e lanciate di nuovo `configure`. Un metodo efficace per controllare è quello di cercare il file che contiene i simboli della libreria, che è sempre `lib<name>.so`. Ad esempio,

  ```
  $ find / -name 'libguile*'
  ```

  o anche:

  ```
  $ locate libguile
  ```

- Controllate che il compilatore possa accedervi, il che significa che deve trovarsi in una delle seguenti directory: `/usr/lib` , `/lib` , `/usr/X11R6/lib` (oppure in una di quelle specificate dalla variabile d'ambiente `LD_LIBRARY_PATH`, trattata nel punto 5.b di *Cosa fare... se non funziona?*, pag. 91). Controllate che questo file sia una libreria digitando il comando `file libguile.so`.
- Controllate che i file header che corrispondono a tale libreria siano installati nel posto giusto (in genere `/usr/include`, oppure `/usr/local/include` o anche `/usr/X11R6/include`). Se non sapete quali sono gli header di cui avete bisogno, controllate che sia stata installata la versione per sviluppatori della libreria richiesta (ad esempio, `gtk+-devel` invece di `libgtk`). La versione per sviluppatori di una libreria fornisce i file "include" necessari per compilare software che faccia uso della suddetta libreria.
- Accertatevi di avere spazio sufficiente sul disco rigido (lo script `configure` ha bisogno di una certa quantità di spazio per dei file temporanei). Usate il comando `df -k` per mostrare lo spazio disponibile sulle partizioni del vostro sistema, e prestate attenzione a quelle piene o quasi piene.

Se non capite il messaggio di errore contenuto nel file `config.log`, non esitate a chiedere aiuto alla comunità del software libero (consultate la sezione *Supporto tecnico*, pag. 95).

Controllate inoltre se `configure` risponde sempre `No` a ragione, oppure se risponde `No` anche quando siete sicuri della presenza di una data libreria. Sarebbe davvero strano, ad esempio, che la libreria `curses` non fosse presente sul vostro sistema. In tal caso, la variabile `LD_LIBRARY_PATH` probabilmente è sbagliata!

13.3.2. imake

imake vi permette di configurare un programma creando un file `Makefile` sulla base di semplici regole. Queste regole determinano quali file debbano essere compilati per costruire il file binario, e *imake* genera il `Makefile` corrispondente. Queste regole sono state specificate in un file chiamato `Imakefile`.

La caratteristica più interessante di *imake* è il fatto che fa uso di informazioni dipendenti dalla particolare ***installazione*** (ovvero dalla particolare architettura hardware). È molto comodo per applicazioni che usano *X Window System*. Ma *imake* viene usato anche per molte altre applicazioni.

Il modo più semplice per usare *imake* è quello di spostarsi nella directory principale dell'archivio decompresso, e lanciare poi lo script `xmkmf`, che richiama il programma `imake`:

```
$ xmkmf -a
imake -DUseInstalled -I/usr/X11R6/lib/X11/config
make Makefiles
```

Se la vostra installazione non funziona correttamente, ricompilate e installate *X11R6*!

13.3.3. Vari script da shell

Leggete i file `INSTALL` o `README` per avere più informazioni. Di solito dovrete lanciare un file del tipo `install.sh` o `configure.sh`. A questo punto, lo script di installazione può essere non interattivo (e stabilire autonomamente ciò di cui ha bisogno), oppure chiedervi informazioni sul sistema (percorsi, ad esempio).

Se non riuscite a determinare quale sia il file da eseguire, potete digitare `./` (in una shell *bash*), e poi premere due volte il tasto **TAB** (tasto di tabulazione). Nella sua configurazione predefinita, *bash* completa automaticamente la linea di comando inserendo il nome di un file eseguibile contenuto nella directory corrente (e quindi un eventuale script di configurazione). Se è presente più di un file eseguibile, vi verrà presentata una lista: non dovete far altro che scegliere il file giusto.

Un caso particolare è quello della installazione di moduli *perl* (ma non solo). L'installazione di tali moduli viene effettuata lanciando in esecuzione uno script di configurazione, che è scritto in *perl*. Il comando da eseguire in genere è:

```
$ perl Makefile.PL
```

13.3.4. Alternative

Alcune distribuzioni di software libero sono organizzate in modo non ottimale, soprattutto durante le prime fasi di sviluppo (ma l'utente è avvertito!). A volte richiedono di cambiare "a mano" alcuni file di configurazione. Normalmente questi file sono un file `Makefile` (consultate la sezione *make*, pag. 89) e un file `config.h` (questo nome è del tutto convenzionale).

Vi sconsigliamo queste modifiche, fatta eccezione per quegli utenti che sanno quello che fanno. Queste operazioni richiedono una buona conoscenza e una certa determinazione per riuscire, ma la pratica le rende comunque banali.

13.4. Compilazione

Adesso che il software è stato configurato in maniera corretta, tutto quello che resta da fare è compilarlo. Questa fase in genere è facile, e non presenta seri problemi.

13.4.1. make

Lo strumento preferito dalla comunità del software libero per compilare sorgenti è `make`. Presenta due caratteristiche particolarmente interessanti:

- lo sviluppatore risparmia tempo, perché `make` permette di gestire la compilazione del proprio progetto in maniera assai efficiente;
- l'utente finale può compilare e installare il software con pochi comandi dalla shell, anche se non ha nessuna conoscenza preliminare di come si sviluppa un programma.

Le azioni che devono essere eseguite per ottenere una versione compilata dei sorgenti sono conservate in un file chiamato normalmente `Makefile` o `GNUMakefile`. Quando `make` viene invocato, infatti, legge questo file – se esiste – nella directory corrente. Se non esiste, è sempre possibile specificare un altro file usando l'opzione `-f` del comando `make`.

13.4.2. Regole

`make` opera in accordo con un sistema di ***dipendenze***. Pertanto compilare un file binario (***target***) richiede il completamento di una serie di passi ("dipendenze"). Ad esempio, se vogliamo creare l'ipotetico file binario `glloq`, i file oggetto `main.o` e `init.o`, file intermedi del processo di compilazione, devono essere compilati e collegati (*linked*). Questi file oggetto sono anch'essi dei risultati della compilazione, e le loro dipendenze sono i file sorgenti.

Questo testo costituisce un'introduzione minima allo scopo di sopravvivere nel mondo spietato di `make`. Se volete saperne di più, vi consigliamo di recarvi sul sito web di **APRIL** (`http://www.april.org/groupes/doc/`), dove troverete una documentazione su `make` molto più dettagliata. Per una documentazione veramente esaustiva, fate riferimento a **Managing Projects with Make**, seconda edizione, **O'Reilly**, di Andrew Oram e Steve Talbott.

13.4.3. Attenti, pronti, via!

In genere l'uso di `make` segue un certo numero di convenzioni. Ad esempio:

- `make` senza argomenti si limita a eseguire la compilazione del programma, senza installarlo.
- `make install` compila il programma (ma non sempre) e provvede a installare i file necessari nelle locazioni appropriate nella struttura gerarchica del filesystem. Alcuni file non sempre vengono installati correttamente (`man`, `info`), devono pertanto essere copiati a mano dall'utente stesso. Talvolta `make install` dev'essere eseguito di nuovo nelle sottodirectory; normalmente questo è necessario con moduli sviluppati da terze parti.
- `make clean` cancella tutti i file temporanei e, nella maggior parte dei casi, anche i file eseguibili creati dal processo di compilazione.

Il primo passo è quello di compilare il programma, e quindi di digitare (esempio del tutto immaginario):

```
$ make
gcc -c glloq.c -o glloq.o
gcc -c init.c -o init.o
gcc -c main.c -o main.o
gcc -lgtk -lgdk -lglib -lXext -lX11 -lm glloq.o init.o main.o -o glloq
```

Perfetto, il file binario è stato compilato in maniera corretta. Adesso siamo pronti per la fase successiva, che prevede l'installazione dei file della distribuzione (file binari, file di dati, etc.). Consultate la sezione *Installazione*, pag. 94.

13.4.4. Spiegazioni

Se la vostra curiosità vi ha spinto a dare un'occhiata al file `Makefile`, avrete scoperto che contiene dei comandi noti (`rm`, `mv`, `cp`, etc.), ma anche delle stringhe di caratteri dall'aspetto inusuale, qualcosa come `$(CFLAGS)`.

Si tratta di ***variabili*** , cioè stringhe che sono normalmente definite all'inizio del file `Makefile` e vengono successivamente sostituite dal valore con il quale sono associate. Il loro uso semplifica molto le cose quando si vogliono usare le stesse opzioni di compilazione più volte nella stessa riga.

Ad esempio, per stampare su video la stringa " `foo` " usando il comando `make all`:

```
TEST = foo
all:
        echo $(TEST)
```

Nella maggior parte dei casi sono presenti le seguenti variabili:

1. `CC`: indica il compilatore. In genere è `cc`, che nella maggioranza dei sistemi aperti è sinonimo di `gcc`. Se in dubbio, inserite qui `gcc`.

2. LD: si tratta del programma usato per portare a termine l'ultima fase del processo di compilazione (consultate la sezione *Le quattro fasi della compilazione*, pag. 84). Come opzione predefinita, è il comando ld.
3. CFLAGS: sono argomenti supplementari che vengono forniti al compilatore durante le prime fasi di compilazione. Fra questi:
   - -I<path>: indica al compilatore dove trovare alcuni header supplementari (-I/usr/X11R6/include, ad esempio, consente di includere gli header che si trovano nella directory /usr/X11R6/include).
   - -D<symbol>: definisce un simbolo addizionale, utile per programmi la cui compilazione dipende dai simboli definiti (ad esempio: usa il file string.h se è definito HAVE_STRING_H).

   Spesso ci sono righe di compilazione come questa:

   ```
   $(CC) $(CFLAGS) -c foo.c -o foo.o
   ```

4. LDFLAGS (o LFLAGS): sono argomenti usati durante l'ultima fase del processo di compilazione. Fra di essi:
   - -L<path>: specifica un percorso supplementare dove cercare librerie (ad esempio: -L/usr/X11R6/lib).
   - -l<library>: specifica una libreria supplementare da usare durante l'ultima fase del processo di compilazione.

13.4.5. Cosa fare... se non funziona?

Non lasciatevi prendere dal panico, può capitare a tutti. Fra le cause di errore più comuni:

1. glloq.c:16: decl.h: No such file or directory

   Il compilatore non è riuscito a trovare l'header corrispondente. La fase di configurazione del software, tuttavia, avrebbe dovuto prevenire questo errore. Ecco come risolvere il problema:

   - controllate che l'header in questione esista realmente sul disco in una delle seguenti directory: /usr/include, /usr/local/include, /usr/X11R6/include, oppure una delle loro sottodirectory. Se così non fosse, cercatelo su tutto il disco rigido (con i comandi find o locate) e, se ancora non riusciste a trovarlo, accertatevi di aver installato la libreria corrispondente. Troverete esempi relativi ai comandi find e locate nelle loro rispettive pagine di manuale.
   - controllate che l'header in questione sia accessibile in lettura: digitate less <percorso>/<file>.h per fare una prova.
   - se si trova in una directory come /usr/local/include o /usr/X11R6/include, può essere necessario aggiungere un nuovo argomento al compilatore: aprite il file Makefile corrispondente (fate attenzione e aprite il file giusto, nella directory dove si è interrotto il processo di compilazione [4]) con il vostro editor di testi preferito (*Emacs*, *Vi*, etc.). Cercate la riga incriminata, e aggiungete la stringa -I<percorso> – dove <percorso> è il percorso che indica dove può essere trovato l'header in questione – subito dopo il punto in cui viene invocato il compilatore (gcc, o $(CC), a volte). Se non sapete dove aggiungere questa opzione, inseritela all'inizio del file, dopo CFLAGS=<qualcosa> o dopo CC=<qualcosa>.
   - lanciate make ancora una volta e, se si interrompe di nuovo, controllate che questa opzione (si veda il punto precedente) sia stata aggiunta durante la compilazione sulla riga in questione.
   - se ancora non funziona, chiedete aiuto al vostro guru locale, oppure rivolgetevi alla comunità del software libero per risolvere il vostro problema (consultate la sezione *Supporto tecnico*, pag. 95).

2. glloq.c:28: 'struct foo' undeclared (first use this function)

   Le strutture sono particolari tipi di dato usati da tutti i programmi. Un gran numero di strutture vengono definite dal sistema negli header: il messaggio significa che il problema è sicuramente causato da un header mancante o usato in modo inappropriato. La procedura corretta per risolvere il problema è la seguente:

4. Analizzate il messaggio di errore restituito da make. Le ultime righe, in genere, contengono il nome di una directory (sono messaggi simili a questo: make[1]: Leaving directory '/home/benj/Project/foo'). Scegliete la riga con il numero più alto e spostatevi nella directory da essa indicata: per essere sicuri che sia quella giusta, lanciate di nuovo make per vedere se ottenete lo stesso messaggio di errore.

- Controllate se la struttura in questione è definita nel programma o dal sistema. Un metodo possibile è l'impiego del comando `grep` per controllare se la struttura è stata definita in uno degli header.

  Ad esempio, quando siete nella directory principale della distribuzione digitate:

  ```
  $ find . -name '*.h'| xargs grep 'struct foo' | less
  ```

  È possibile che sullo schermo compaiano un gran numero di righe di testo (una per ogni volta che è definita una funzione che fa uso di questo tipo di struttura, ad esempio). Se esiste, trovate la riga dove è definita la struttura esaminando il file header che avete individuato usando il comando `grep`.

  La definizione di una struttura è:

  ```
  struct foo {
        <contenuto della struttura>
  };
  ```

  Controllate che corrisponda con quanto avete trovato: se le cose stanno così, questo significa che l'header non è incluso nel file `.c` difettoso. Ci sono due soluzioni:

  - aggiungete la riga `#include "<filename>.h"` all'inizio del file `.c` difettoso;
  - oppure fate un copia-incolla della definizione della struttura all'inizio di questo file (non è proprio elegante, ma per lo meno dovrebbe funzionare).

- Se invece non corrisponde, fate la stessa cosa con i file header di sistema (che in genere si trovano nelle directory `/usr/include`, `/usr/X11R6/include`, o `/usr/local/include`). Ma stavolta usate la riga `#include <<filename>.h>`.
- Se questa struttura risulta ancora non esistente, cercate di scoprire in quale libreria (ovvero un set di funzioni riunito in un unico pacchetto) dovrebbe essere definita: controllate nel file `INSTALL` o `README` quali sono le librerie usate dal programma, e il numero di versione richiesto. Se la versione di cui ha bisogno il programma non è quella installata sul vostro sistema, aggiornate la libreria in questione.
- Se, malgrado tutto, ancora non funziona, accertatevi che il programma funzioni correttamente con la vostra architettura (alcuni programmi non sono stati ancora portati su tutti i sistemi *UNIX*). Controllate anche di aver correttamente configurato il programma (quando avete lanciato `configure`, ad esempio) per la vostra architettura.

3. `parse error`

   Si tratta di un problema piuttosto complicato da risolvere, in quanto spesso l'errore appare relativamente a una certa riga, ma dopo che il compilatore l'ha incontrato. Talvolta si tratta semplicemente di un tipo di dati non definito. Se incontrate un messaggio d'errore come:

   ```
   main.c:1: parse error before 'glloq_t
   main.c:1: warning: data definition has no type or storage class
   ```

   allora il problema sta nel fatto che il tipo `glloq_t` non è stato definito. La soluzione è più o meno la stessa che per il problema precedente.

   potrebbe esserci un `parse error` nelle vecchie librerie `curses`.

4. `no space left on device`

   Questo è un problema che può essere risolto facilmente: non è rimasto spazio sufficiente sul disco rigido per generare un file binario a partire dai sorgenti. La soluzione consiste nel liberare una quantità di spazio sufficiente nella partizione che contiene la directory di installazione (cancellate file temporanei o directory di sorgenti, disinstallate i programmi che non utilizzate). Se avete decompresso quest'ultima in `/tmp`, fatelo piuttosto in `/usr/local/src`, in modo da evitare di riempire inutilmente la partizione `/tmp`. Controllate, inoltre, se vi sono file `core` sul vostro disco. In caso affermativo, cancellateli, o fateli cancellare se appartengono a un altro utente.

5. `/usr/bin/ld: cannot open -lglloq: No such file or directory`

Questo significa, come avrete capito, che il programma `ld` (usato da `gcc` durante l'ultima fase del processo di compilazione) non riesce a trovare una libreria. Per includere una libreria, `ld` cerca un file il cui nome si trova negli argomenti del tipo `-l<library>`. Questo file è `lib<library>.so`, e se `ld` non riesce a trovarlo mostra un messaggio d'errore. Per risolvere il problema, seguite i passi descritti qui di seguito:

a. Controllate che il file sia presente sul disco rigido usando il comando `locate`. Le librerie grafiche si trovano, in genere, in `/usr/X11R6/lib`. Ad esempio:

```
$ locate libglloq
```

Se questa ricerca non è fruttuosa, potete effettuarne un'altra usando il comando `find` (ad esempio: `find /usr -name libglloq.so*`). Se proprio non riuscite a trovare la libreria, vuol dire che non è presente sul vostro sistema e dunque dovrete installarla.

b. Una volta localizzata la libreria, accertatevi che `ld` possa accedervi: il file `/etc/ld.so.conf` specifica dove potrà trovare queste librerie: aggiungete il percorso della libreria incriminata al termine del file (è possibile che si debba riavviare il computer perché questa modifica venga recepita dal sistema). Potete anche aggiungere la stessa directory modificando il contenuto della variabile d'ambiente `LD_LIBRARY_PATH`: se la directory da aggiungere è `/usr/X11R6/lib`, ad esempio, digitate:

```
export \ LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/X11R6/lib
```

(se la vostra shell è *bash*).

c. Se ancora non funziona, accertatevi che il formato della libreria sia quello di un file eseguibile (cioè ELF) usando il comando `file`. Se si tratta di un collegamento (detto anche link) simbolico, controllate che il collegamento sia valido e che non punti a un file inesistente (ad esempio con il comando `nm libglloq.so`). I permessi potrebbero essere errati, ad esempio se usate un account che non sia `root`, oppure se la libreria è protetta in lettura.

6. `glloq.c(.text+0x34): undefined reference to 'glloq_init'`

Il problema riguarda un simbolo che non è stato risolto durante l'ultima fase del processo di compilazione. Si tratta, in genere, di problemi dovuti alle librerie. Le cause possono essere molteplici:

- la prima cosa da fare è sapere se il simbolo **dovrebbe** trovarsi in una libreria. Ad esempio, se si tratta di un simbolo che comincia con `gtk`, appartiene alla libreria `gtk`. Se il nome della libreria non è immediatamente identificabile (`frobnicate_foobar`), potete elencare i simboli di una libreria usando il comando `nm`. Ad esempio,

```
$ nm libglloq.so
0000000000109df0 d glloq_message_func
000000000010a984 b glloq_msg
0000000000008a58 t glloq_nearest_pow
0000000000109dd8 d glloq_free_list
0000000000109cf8 d glloq_mem_chunk
```

Aggiungendo l'opzione `-o` al comando `nm` è possibile stampare il nome della libreria su ogni riga, il che facilita la ricerca. Supponiamo di dover trovare il simbolo `bulgroz_max`, una soluzione brutale ma efficace è effettuare una ricerca come:

```
$ nm /usr/lib/lib*.so | grep bulgroz_max
$ nm /usr/X11R6/lib/lib*.so | grep bulgroz_max
$ nm /usr/local/lib/lib*.so | grep bulgroz_max
/usr/local/lib/libfrobnicate.so:000000000004d848 T bulgroz_max
```

Perfetto! Il simbolo `bulgroz_max` viene definito nella libreria `frobnicate` (la lettera maiuscola `T` si trova davanti al suo nome). A questo punto non dovete fare altro che aggiungere `-lfrobnicate` nella riga di compilazione modificando il file `Makefile`: inserite questa stringa al termine della riga in cui vengono definite le variabili `LDFLAGS` o `LFGLAGS` (oppure `CC`, nel peggiore dei casi), o nella riga che corrisponde alla creazione del file binario.

- la compilazione viene effettuata con una versione della libreria che non è quella richiesta dal software. Leggete il file `README` o `INSTALL` della distribuzione per sapere qual è la versione che dev'essere usata.

- non tutti i file oggetto della distribuzione sono linkati correttamente. Il file dove viene definita questa funzione è assente. Digitate `nm -o *.o` per sapere di quale file si tratta e aggiunge il file `.o` corrispondente sulla riga di compilazione se è assente.
- la funzione o variabile causa del problema forse non esiste. Cercate di cancellarla: modificate il file sorgente in questione (il suo nome viene indicato all'inizio del messaggio di errore). Si tratta di una soluzione disperata, che avrà come sicura conseguenza un funzionamento "anarchico" del software (***segfault*** all'avvio, etc.).

7. `Segmentation fault (core dumped)`

   Talvolta il compilatore si pianta in modo deplorevole, e produce questo messaggio d'errore. Non possiamo darvi altro consiglio se non quello di installare una versione più recente.

8. spazio esaurito su `/tmp`

   La compilazione necessita di uno spazio di lavoro temporaneo durante le sue diverse fasi: se questo spazio non è disponibile, non viene portata a termine con successo. È quindi necessario ripulire la partizione, ma state attenti ad alcuni programmi in esecuzione (il server *X*, delle pipe, etc.) che potrebbero bloccarsi se vengono cancellati alcuni dei loro file temporanei: dovete sapere quello che state facendo! Se `/tmp` fa parte di una partizione che contiene altro (ad esempio la directory radice), cercate e cancellate eventuali file di tipo `core`.

9. `make/configure` in ricorsione infinita

   Molto spesso si tratta di un problema relativo al tempo sul vostro sistema. `make`, infatti, ha bisogno di conoscere la data fornita dall'orologio di sistema e la data dei file che controlla. Confronta le date e usa il risultato per sapere se il file binario è più recente rispetto alla dipendenza.

   Alcuni problemi di data possono indurre `make` a compilare se stesso in maniera ricorsiva, senza interruzioni (o a compilare e compilare di nuovo un sottoalbero in ricorsione infinita). In tal caso, l'uso del comando `touch` (il cui scopo qui è di aggiornare i file in questione all'ora corrente) in genere risolve questo problema.

   Ad esempio:

   ```
   $ touch *
   ```

   O anche (più rozzo, ma efficace):

   ```
   $ find . | xargs touch
   ```

13.5. Installazione

13.5.1. Con make

Adesso che la compilazione è terminata, dovete copiare i file che sono stati generati in un luogo appropriato (normalmente in una delle sottodirectory di `/usr/local`).

`make` di solito può eseguire questo compito. Il ***target*** `install` è un target particolare, riservato a questo scopo specifico. Il comando `make install`, quindi, permette di installare i file richiesti.

La procedura da seguire, in genere, è descritta nel file `INSTALL` o nel `README`. Ma a volte lo sviluppatore si è dimenticato di pensarci: in tal caso dovrete installare ogni file personalmente.

Pertanto copiate:

- i file eseguibili (programmi) nella directory `/usr/local/bin`
- le librerie (i file `lib*.so`) nella directory `/usr/local/lib`
- gli header (i file `*.h`) nella directory `/usr/local/include` (attenzione a non cancellare quelli originali)
- i file di dati in genere finiscono in `/usr/local/share`. Se non conoscete la procedura di installazione, potete cercare di lanciare in esecuzione i programmi senza aver copiato i file di dati, e metterli

nel posto giusto quando vi viene richiesto (ad esempio con un messaggio d'errore come `Cannot open /usr/local/share/glloq/data.db`).

- per la documentazione le regole da seguire sono di poco differenti:
  - i file `man` in genere vengono collocati in una delle sottodirectory di `/usr/local/man`. Di solito questi file sono in formato `troff` (o `groff`) e la loro estensione è una cifra. Il loro nome è il nome di un comando (ad esempio `echo.1`). Se la cifra è `n`, copiate il file in `/usr/local/man/man<n>`;
  - i file `info` vanno copiati nella directory `/usr/info` oppure in `/usr/local/info`.

Avete finito! Congratulazioni! Adesso siete pronti per compilare un intero sistema operativo!

13.5.2. Problemi

Se avete appena installato del software libero, *GNU tar* ad esempio, e se, quando lo eseguite, viene lanciato un altro programma, oppure non si comporta esattamente come faceva quando lo avete provato direttamente dalla directory `src`, si tratta di un problema di `PATH`, per cui i programmi vengono trovati in una directory che precede quella in cui avete installato del nuovo software. Controllate eseguendo il comando `type -a <program>`.

La soluzione consiste nel collocare la directory di installazione in una posizione più alta nella variabile `PATH`, e/o di cancellare/rinominare i file che vengono eseguiti al posto di quelli richiesti, e/o di rinominare i vostri nuovi programmi (come `gtar` in questo esempio) in maniera tale che non vi sia più nessuna confusione.

Potete anche impostare un alias, se la vostra shell lo consente (ad esempio, potreste fare in modo che `tar` significhi `/usr/local/bin/gtar`).

13.6. Supporto

13.6.1. Documentazione

Esistono diverse fonti di documentazione:

- *HOWTO*, brevi documenti su punti specifici; in genere abbastanza lontani da quanto ci serve adesso, ma a volte utili. Cercateli sul vostro disco in `/usr/doc/HOWTO` (ma non sono sempre lì, a volte la posizione è diversa: controllate con il comando `locate HOWTO`);
- le pagine di manuale. Digitate `man <comando>` per ottenere documentazione in merito al comando `<comando>`;
- testi specifici sull'argomento. Molti grandi editori hanno cominciato a pubblicare libri che riguardano i sistemi aperti (in particolare su *GNU/Linux*). Spesso risultano molto utili se siete agli inizi e non capite tutti i termini usati in questo manuale.

13.6.2. Supporto tecnico

Se avete comprato una distribuzione **Mandrake Linux** "ufficiale" potete rivolgervi al personale di supporto tecnico per avere informazioni sul vostro sistema. Probabilmente questo personale è impegnato in troppe altre cose per poter aiutare tutti gli utenti nell'installazione di software supplementare, ma alcuni di loro potrebbero offrirvi un numero `x` di giorni in cui potete chiedere aiuto in merito all'installazione. Forse potranno dedicare un po' del loro tempo per aiutarvi a risolvere problemi di compilazione.

Potete anche fare affidamento sulla comunità del software libero per essere aiutati:

- nei ***gruppi di discussione*** (su *Usenet*) comp.os.linux.* (`news:comp.os.linux.*`) e it.comp.os.linux.* (`news:it.comp.os.linux.*`) viene data risposta a molte domande che riguardano *GNU/Linux*. I gruppi di discussione che appartengono alla gerarchia comp.os.bsd.* (`news:comp.os.bsd.*`) sono dedicati ai sistemi BSD. Potrebbero esserci altri gruppi di discussione che riguardano altri sistemi *UNIX*. Ricordatevi di leggerli per un certo periodo prima di formulare la vostra richiesta;

- nella comunità del software libero esistono numerose associazioni o gruppi di appassionati che offrono un supporto volontario. Il modo migliore per trovare quelli più vicini al luogo in cui vivete è di consultare le liste di collegamenti sui siti web dedicati, o di leggere i gruppi di discussione che vi abbiamo suggerito prima per un certo periodo di tempo;
- alcuni ***canali*** IRC offrono un'assistenza in tempo reale (ma alla cieca) da parte di alcuni ***guru***. Recatevi, ad esempio, sul canale `#linux`, disponibile su quasi tutta la rete IRC, oppure `#linuxhelp` su *IRCNET*;
- come ultima risorsa, chiedete allo sviluppatore del software (se quest'ultimo ha riportato il suo nome e il suo indirizzo ***e-mail*** in un file della distribuzione) se siete sicuri di aver trovato un bug; questo potrebbe essere dovuto alla vostra particolare architettura, ma si suppone che il software libero sia facilmente portabile, dopo tutto.

13.6.3. Come reperire il software libero

Esistono molti riferimenti che vi possono aiutare a trovare del software libero:

- il grande sito FTP sunsite (`ftp://sunsite.unc.edu`) o uno dei suoi mirror;
- i siti web elencati qui di seguito raccolgono un gran numero di programmi liberamente distribuibili che possono essere usati sotto *UNIX* (ma su di essi è anche possibile trovare del software proprietario):
  - FreshMeat (`http://www.freshmeat.net/`) è, probabilmente, il sito più completo;
  - Linux France (`http://www.linux-france.org/`) contiene un gran numero di link relativi a programmi che girano sotto *GNU/Linux*. La maggior parte di essi, ovviamente, funziona anche con altre piattaforme *UNIX* aperte;
  - la pagina del software sul sito del progetto GNU (`http://www.gnu.org/software/`), per una lista esaustiva di tutto il software GNU. Naturalmente tutti questi programmi sono software libero, e la maggior parte è distribuita secondo la licenza GPL;

- potete anche effettuare un'indagine con un motore di ricerca come Altavista (`http://www.altavista.com/`) o Lycos (`http://ftpsearch.lycos.com/`), facendo una richiesta simile alla seguente: `+software +download` oppure `"download software"`.

13.7. Ringraziamenti

- Revisione, correzione e commenti sgradevoli (in ordine alfabetico): Sébastien Blondeel, Mathieu Bois, Xavier Renaut e Kamel Sehil.
- ***Beta-testing***: Laurent Bassaler.
- Traduzione inglese: Fanny Drieu. Edizione inglese: Hoyt Duff

Capitolo 14. Compilazione e installazione di nuovi kernel

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

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

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

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

14.1. Dove trovare i sorgenti del kernel

I siti principali dai quali scaricare i sorgenti del kernel sono due:

1. **Il kernel ufficiale di Mandrake Linux.** Nella directory `SRPMS` di uno qualsiasi dei mirror (`http://www.MandrakeLinux.com/en/cookerdevel.php3`) che ospitano `Cooker` (la versione in via di sviluppo di **Mandrake Linux**) troverete questi pacchetti:

kernel-2.4.??.?mdk-?-?mdk.src.rpm

I sorgenti usati per compilare il kernel incluso nella distribuzione. Si tratta di una versione notevolmente modificata al fine di introdurre ulteriori funzionalità.

kernel-linus2.4-2.4.??-?mdk.src.rpm

I sorgenti del kernel standard, nella forma in cui vengono distribuiti dal curatore del kernel *GNU/Linux*.

Se scegliete questo metodo (e noi vi incoraggiamo a farlo!), non dovete far altro che scaricare l'RPM che contiene i sorgenti, installarlo (come `root`) e passare a *Configurazione del kernel*, pag. 98.

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

   Su tutti questi server FTP i sorgenti si trovano nella directory `/pub/linux/kernel`. Dopo di che, spostatevi nella directory che ospita la serie che più vi interessa, senza dubbio si tratterà della `v2.4`. Non c'è nulla che vi impedisca di provare i kernel della serie 2.5, ma ricordate che questi sono in fase sperimentale. Il file che contiene i sorgenti del kernel si chiama `linux-<kernel.version>.tar.bz2`, ad esempio `linux-2.4.8.tar.bz2`.

   Sono anche disponibili delle patch da applicare ai sorgenti del kernel, in maniera da effettuare degli aggiornamenti incrementali: così, se avete già i sorgenti della versione 2.4.8 del kernel e volete aggiornarli alla 2.4.10, non è necessario scaricare il voluminoso archivio relativo a quest'ultima versione, ma potete semplicemente procurarvi e applicare le ***patch*** `patch-2.4.9.bz2` e `patch-2.4.10.bz2`. Come regola generale, questo è il metodo più conveniente, visto che i sorgenti al momento occupano più di 23 Mb.

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

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

```
$ cd /usr/src
$ mv linux linux.old
$ tar xjf /path/to/linux-2.4.8.tar.bz2
```

Il comando `mv linux linux.old` è necessario perché potrebbero essere presenti i sorgenti di un'altra versione del kernel. Con questo comando sarete sicuri di mantenere questa versione meno recente, evitando di cancellarla scrivendoci sopra quella nuova. Una volta decompresso l'archivio, avrete a vostra disposizione una directory `linux` contenente i sorgenti del nuovo kernel.

E adesso, le patch: supponiamo, ad esempio, che vogliate ***applicare una patch*** per passare dalla versione 2.4.8 alla 2.4.10, e che abbiate già scaricato i file necessari allo scopo: spostatevi nella nuova directory `linux`, poi applicate le patch:

```
$ cd linux
$ bzcat /path/to/patch-2.4.9.bz2 | patch -p1
$ bzcat /path/to/patch-2.4.10.bz2 | patch -p1
$ cd ..
```

In genere, quando ci si sposta da una versione 2.4.x a una versione 2.4.y è necessario applicare tutte le patch numerate 2.4.x+1, 2.4.x+2, ..., 2.4.y in questo preciso ordine. Per tornare dalla 2.4.y alla 2.4.x, ripetete esattamente la stessa procedura, ma applicando le patch in ordine inverso e utilizzando l'opzione `-R` del comando `patch` (`R` sta per *Reverse*). Così, per tornare dal kernel 2.4.10 alla versione 2.4.8, dovreste digitare:

```
$ bzcat /path/to/patch-2.4.10.bz2 | patch -p1 -R
$ bzcat /path/to/patch-2.4.9.bz2 | patch -p1 -R
```

Se volete accertarvi che una patch verrà applicata correttamente prima di applicarla realmente, aggiungete l'opzione `--dry-try` dopo il comando `patch`.

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

```
$ mv linux linux-2.4.10
$ ln -s linux-2.4.10 linux
```

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

```
$ cd linux
```

14.3. Configurazione del kernel

Per cominciare, spostatevi nella directory `/usr/src/linux`.

Per prima cosa un piccolo trucco: se volete, potete personalizzare la versione del kernel che state per compilare. La versione del kernel è determinata dalle prime quattro righe del `Makefile`:

```
$ head -4 Makefile
VERSION = 2
PATCHLEVEL = 4
SUBLEVEL = 4
EXTRAVERSION =
```

Più oltre nel `Makefile`, noterete che la versione del kernel è compilata come:

```
KERNELRELEASE=$(VERSION).$(PATCHLEVEL).$(SUBLEVEL)$(EXTRAVERSION)
```

Tutto quello che dovete fare è modificare una di queste variabili per cambiare la versione. È preferibile cambiare solo EXTRAVERSION. Supponiamo che la impostiate come `-foo`, ad esempio, in tal caso la versione del vostro nuovo kernel sarà 2.4.10-foo. Non esitate a cambiare questo campo definendo nuove versioni ogni volta

che ricompilate un nuovo kernel. In tal modo potrete sperimentare opzioni diverse conservando le versioni precedenti.

Adesso passiamo alla configurazione, potete scegliere fra:

- `make xconfig` per usare un'interfaccia grafica,
- `make menuconfig` per usare un'interfaccia basata sulla libreria `ncurses`, oppure
- `make config` per usare l'interfaccia più rudimentale, riga per riga, sezione per sezione.
- `make oldconfig` stessa interfaccia dell'opzione precedente, ma basato sulla configurazione precedente. Si veda *Stoccaggio e riutilizzo dei file di configurazione del kernel*, pag. 99.

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

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

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

Dato che le opzioni possibili sono diverse centinaia, non è possibile descriverle tutte in questa sede. Se siete arrivati a questo punto del capitolo, inoltre, probabilmente sapete già quello che state facendo; per cui vi lasceremo provare i vari programmi di configurazione e attivare/disattivare le opzioni che riterrete opportune. Ecco alcuni consigli, comunque, per evitare di ritrovarsi con un kernel inutilizzabile:

1. A meno che non usiate un ramdisk iniziale, non dovete **mai** compilare i driver necessari per effettuare il mount del vostro filesystem radice (driver relativi sia all'hardware sia al filesystem) come moduli! E se usate un ramdisk iniziale, rispondete `Y` alla domanda riguardo il supporto ext2fs, dato che questo è il filesystem usato per i ramdisk. Sarà inoltre necessario il supporto `initrd`.
2. Se sul vostro sistema sono presenti delle schede di rete, compilate i loro driver come moduli. In questo modo potrete stabilire quale scheda sarà la prima, quale la seconda, e così via, inserendo degli alias adeguati nel file `/etc/modules.conf`. Se compilate i driver all'interno del kernel, l'ordine in cui essi verranno caricati dipenderà dall'ordine di linkaggio, che potrebbe non corrispondere all'ordine desiderato.
3. E infine: se non sapete lo scopo di un'opzione, leggete il testo di aiuto corrispondente! Se neanche questo è sufficiente a chiarirvi le idee, lasciatela così com'è. Se state usando le opzioni `config` e `oldconfig` premete il tasto **?** per accedere all'aiuto in linea.

Potete anche consultare il file `/usr/src/linux/Documentation/Configure.help` che contiene un testo di aiuto in merito a ciascuna opzione, secondo l'ordine in cui compaiono. Nella sua intestazione, inoltre, troverete dei collegamenti che vi porteranno a numerose traduzioni.

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

14.4. Stoccaggio e riutilizzo dei file di configurazione del kernel

La configurazione del kernel viene salvata nel file `/usr/src/linux/.config`. È caldamente consigliato farne una copia di backup, ad esempio nella directory personale dell'utente `root`. In tal modo non solo potrete riutilizzarla in seguito, ma avrete anche la possibilità di salvare configurazioni per la compilazione di kernel diversi, dato che per far questo è sufficiente dare nomi diversi ai file di configurazione.

Una possibilità, a riguardo, è quella di dare nomi ai file di configurazione seguendo la numerazione delle versioni del kernel. Supponendo che abbiate modificato la vostra versione del kernel come mostrato in *Configurazione del kernel*, pag. 98, potete digitare:

```
$ cp .config /root/config-2.4.10-foo
```

Se decidete di aggiornare il kernel alla versione 2.4.12 (ad esempio), potrete riutilizzare questo file, poiché le differenze tra le opzioni di configurazione di questi due kernel saranno molto piccole. Quindi potete ricorrere direttamente alla copia di backup:

```
$ cp /root/config-2.4.10-foo .config
```

Ma copiare il file precedente non significa che il kernel sia già pronto per essere compilato. Dovrete lanciare di nuovo `make menuconfig` (o qualunque altro sia il programma di configurazione che usate di solito) perché alcuni file necessari per il buon esito della compilazione sono creati e/o modificati da questi programmi.

A parte il fatto che ripercorrere di nuovo tutto il percorso tra i menu è un compito alquanto tedioso, è possibile che così facendo vi sfugga qualche interessante nuova opzione. Potete sfuggire a questa eventualità digitando `make oldconfig`. Questo metodo presenta due vantaggi:

1. è veloce,
2. se è disponibile una nuova opzione di configurazione del kernel che non è presente nel vostro file di configurazione, il programma si fermerà e attenderà che abbiate effettuato una scelta.

Dopo aver copiato il file `.config` nella directory home di `root`, come suggerito poco sopra, eseguite `make mrproper`. In questo modo sarete sicuri che non è rimasto nulla della vecchia configurazione e otterrete un kernel "pulito".

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

14.5. Compilazione del kernel e dei moduli, installazione dei moduli

Piccola precisazione prima di cominciare: se state per ricompilare un kernel con esattamente lo stesso numero di versione di quello già presente sul vostro sistema, per prima cosa è necessario cancellare i vecchi moduli. Ad esempio, se volete ricompilare la versione 2.4.10 del kernel, dovete prima cancellare la directory `/lib/modules/2.4.10`.

La compilazione del kernel e dei moduli, e poi l'installazione dei moduli, è compiuta poche righe di istruzioni:

```
make dep
make clean bzImage modules
make modules_install install
```

Una rapida spiegazione riguardo quanto segue il comando `make`: `dep`, `bzImage`, etc., come pure `oldconfig` e altri termini che abbia usato in precedenza, sono detti **bersagli**. Se specificate più bersagli per `make`, come nell'esempio che abbiamo appena visto, questi verranno eseguiti secondo l'ordine in cui sono riportati. Ma se uno di essi non viene portato a termine con successo, `make` non andrà oltre [1] .

Diamo uno sguardo ai vari bersagli e vediamo cosa fanno:

- `dep`: calcola le dipendenze fra i diversi file sorgenti. È necessario eseguire questa operazione ogni volta che viene modificata la configurazione, altrimenti alcuni file potrebbero non essere compilati e l'intero processo si bloccherebbe.
- `bzImage`: si occupa della compilazione del kernel. Si noti che questo bersaglio, come pure `zImage`, è valido soltanto per i processori *`Intel`*. La differenza tra `bzImage` e `zImage` consiste nel fatto che il primo porterà alla generazione di un kernel che verrà caricato nella memoria alta. Questo bersaglio, inoltre, genera il file `System.map` relativo al kernel. Vedremo successivamente a cosa serve questo file.
- `modules`: come suggerisce il nome stesso, questo bersaglio è responsabile della generazione dei moduli per il kernel che avete appena compilato. Se avete deciso di non far uso dei moduli, questo bersaglio non farà niente.
- `modules_install`: provvede a installare i moduli. Come opzione predefinita, i moduli verranno installati nella directory `/lib/modules/<kernel-version>`. Questo bersaglio, inoltre, calcola le dipendenze dei moduli (a differenza della serie 2.2.x).

1. In questo caso specifico, se `make` non ha buon esito significa che c'è un bug nel kernel... Se dovesse verificarsi questa eventualità, per favore inviateci un messaggio in cui descrivete il problema!

- `install`: quest'ultimo bersaglio provvederà infine a copiare il kernel e i moduli nei luoghi appropriati, e a modificare la configurazione del bootloader in modo che il nuovo kernel sia disponibile al momento del boot. Non usatelo se preferite eseguire un'installazione manuale, come descritto in *Installazione del nuovo kernel*, pag. 101.

A questo punto, ogni componente è stata compilata e correttamente installata, tutto è pronto per un test! Riavviate il computer e scegliete il nuovo kernel tra le voci del menu di avvio sistema. Si noti che il vecchio kernel resta disponibile, in maniera tale che possiate utilizzarlo in caso di problemi con quello nuovo. Potete anche scegliere di installare il kernel e cambiare il menu di avvio manualmente: questo è l'argomento della prossima sezione.

14.6. Installazione del nuovo kernel

Il kernel compilato si trova nel file `arch/i386/boot/bzImage` (oppure `zImage` se avete scelto di lanciare `make zImage`). La directory standard in cui vengono installati i kernel è `/boot`. Dovete anche copiarvi il file `System.map` per essere sicuri che alcuni programmi (`top`, ad esempio) funzioneranno in modo corretto. Di nuovo, per amor di precisione e per identificare in maniera chiara i vostri kernel, è preferibile dare nomi basati sul numero di versione. I comandi da digitare sono i seguenti:

```
$ cp arch/i386/boot/bzImage /boot/vmlinux-2.4.10-foo
$ cp System.map /boot/System.map-2.4.10-foo
```

Adesso dovete comunicare al boot loader che sul vostro sistema è stato installato un nuovo kernel. Ci sono due possibilità: *grub* o *LILO*. Si noti che al momento il boot loader standard utilizzato da **Mandrake Linux** è *LILO*.

14.6.1. Aggiornamento di grub

Com'è ovvio, per prima cosa vogliamo essere sicuri di potere effettuare il boot usando il kernel attualmente in uso! Il metodo più semplice per aggiornare *grub* è usare *drakboot* (consultate il capitolo Modifica della configurazione di avvio del *Manuale dell'utente*). In alternativa, potete modificare a mano il file di configurazione secondo le istruzioni che troverete qui di seguito.

Il file che dovete modificare è `/boot/grub/menu.lst`. Questo è l'aspetto tipico di un file `menu.lst` subito dopo aver installato una distribuzione **Mandrake Linux** e prima di qualsiasi modifica:

```
timeout 5
color black/cyan yellow/cyan
i18n (hd0,4)/boot/grub/messages
keytable (hd0,4)/boot/fr-latin1.klt
default 0

title linux
kernel (hd0,4)/boot/vmlinuz root=/dev/hda5

title failsafe
kernel (hd0,4)/boot/vmlinuz root=/dev/hda5  failsafe

title Windows
root (hd0,0)
makeactive
chainloader  +1

title floppy
root (fd0)
chainloader +1
```

Questo file si compone di due parti: l'intestazione, che contiene le opzioni più comuni (le prime cinque righe), e le diverse sezioni (o entrate), ciascuna delle quali corrisponde a un diverso kernel di *GNU/Linux* oppure a un altro OS da voi specificato. `timeout 5` determina la quantità di tempo (in secondi) che avete a disposizione prima che *grub* lanci il sistema operativo (o il kernel) predefinito. L'immagine predefinita è determinata dalla direttiva `default 0`, che in questo caso specifica che la sezione predefinita è la prima. La direttiva `keytable`, se presente, stabilisce dove si trova la mappa di tastiera scelta dall'utente: nell'esempio in questione si tratta di una tastiera francese; se tale direttiva non è presente, come opzione predefinita verrà usata una tastiera qwerty. Tutte le sigle `hd(x,y)` che vedete si riferiscono alla partizione numero `y` sul disco numero `x` secondo quanto vede il *BIOS*.

Quindi incontriamo la parte contenente le sezioni. Nel nostro esempio sono state definite quattro sezioni: `linux`, `failsafe`, `windows`, e `floppy`.

- La riga di opzioni per la sezione `linux` comunica a *grub* qual è l'immagine di boot che vogliamo sia caricata (`kernel hd(0,4)/boot/vmlinuz`, ovvero il kernel `vmlinuz` che si trova nella directory `/boot/` nella quarta partizione del primo disco rigido). Seguono le opzioni da passare al kernel specificato: in questo caso `root=/dev/hda5` informa il kernel del fatto che il filesystem radice si trova nella partizione `/dev/hda5`. L'opzione `/dev/hda5` corrisponde perfettamente alla stringa `hd(0,4)` usata da *grub*, ma nulla vieta che il kernel si trovi su di una partizione diversa rispetto al filesystem radice.
- La sezione `failsafe` assomiglia molto alla precedente, fatta eccezione per il fatto che verrà comunicato un argomento al kernel (`failsafe`) per ordinargli di entrare in modalità "single user" (monoutente) o "rescue" (salvataggio).
- La sezione `Windows` semplicemente ordina a *grub* di caricare il settore di boot della prima partizione, che probabilmente contiene un settore di boot di *Windows*.
- L'ultima sezione, `floppy`, non fa altro che eseguire il boot a partire da un floppy disk inserito nel primo lettore, qualunque sia il sistema operativo installato su di esso. Potrebbe trattarsi di un dischetto di boot di *Windows*, o anche di un kernel *GNU/Linux* su dischetto.

A seconda del livello di sicurezza che usate sul vostro sistema, alcune delle voci qui descritte potrebbero non essere presenti nel vostro file di configurazione.

Adesso arriviamo al nocciolo della questione: dobbiamo aggiungere un'altra sezione per comunicare a *grub* che abbiamo installato un nuovo kernel. In questo esempio verrà inserita prima delle altre entrate, ma niente vi impedisce di collocarla in un'altra posizione<:

```
title foo
kernel (hd0,4)/boot/vmlinux-2.4.10-foo root=/dev/hda5
```

Non dimenticate di modificare questa entrata in base alla vostra effettiva configurazione hardware! Il filesystem radice di *GNU/Linux* qui si trova in `/dev/hda5`, ma è possibile (e probabile) che sulla vostra macchina risieda altrove.

E con questo abbiamo finito. A differenza di *LILO*, come vedremo tra poco, non dobbiamo fare nient'altro: è sufficiente riavviare il computer e la nuova entrata sarà ben visibile. Selezionatela e il vostro nuovo kernel verrà usato per il boot.

Se avete compilato il kernel abilitando l'opzione relativa al framebuffer, probabilmente vorrete usarlo: in tal caso, dovete istruire il kernel riguardo la risoluzione di partenza. La lista di modi video è disponibile nel file `/usr/src/linux/Documentation/fb/vesafb.txt` (soltanto nell'eventualità che si tratti del framebuffer VESA! Altrimenti, fate riferimento al file corrispondente). Per il modo 800x600 a 32 bit[2], il numero della modalità video è `0x315`, perciò dovete aggiungere la direttiva:

```
vga=0x315
```

in maniera tale che la sezione diventi più o meno così:

```
title foo
kernel (hd0,4)/boot/vmlinux-2.4.10-foo root=/dev/hda5 vga=0x315
```

Per avere ulteriori informazioni, per favore consultate le pagine info relative a *grub* (`info grub`).

2. 8 bit significa 2^8 colori, cioè 256; 16 bit significa 2^{16} colori, cioè 64k, o 65536, colori; alla risoluzione di 24 bit, come pure di 32, i colori sono codificati utilizzando 24 bit,s cioè 2^{24} colori possibili, in altre parole 16M, o un po' più di 16 milioni di colori.

14.6.2. Aggiornamento di lilo

Il modo più semplice di aggiornare *LILO* è usare *drakboot* (si veda il capitolo Modifica della configurazione di avvio nel *Manuale dell'utente*). In alternativa, potete modificare a mano il file di configurazione secondo le istruzioni che troverete qui di seguito.

Il file di configurazione di *LILO* è `/etc/lilo.conf`. Quello che segue è un tipico esempio di `lilo.conf` subito dopo aver installato il pacchetto *LILO* e prima di ogni modifica:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/fr-latin1.klt
lba32
prompt
timeout=50
message=/boot/message
image=/boot/vmlinuz-2.4.8-17mdk
        label=linux
        root=/dev/hda1
        read-only
other=/dev/hda2
        label=dos
        table=/dev/hda
```

Un file di configurazione `lilo.conf` consiste di una sezione principale seguita da una sezione per ogni sistema operativo (o kernel) dal quale si può effettuare il boot. Nell'esempio visto in precedenza, la sezione principale consiste delle seguenti istruzioni:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
vga=normal
default=linux
keytable=/boot/fr-latin1.klt
lba32
prompt
timeout=50
message=/boot/message
```

La direttiva `boot=` comunica a *LILO* dove deve installare il suo settore di boot; nel nostro esempio si tratta del MBR (*Master Boot Record*) del primo disco rigido IDE. Se desiderate creare un floppy disk basato su *LILO*, non dovete far altro che sostituire `/dev/hda` con `/dev/fd0`. La direttiva `prompt` impone a *LILO* di mostrare il menu al momento del boot: dato che è stato specificato un tempo di attesa, *LILO* procederà all'avvio del sistema usando l'immagine predefinita al momento in cui questo è esaurito (cioè dopo 5 secondi: `timeout=50`). Se cancellate la direttiva `timeout=`, *LILO* attenderà finché non avrete digitato qualcosa.

Segue poi una sezione `linux`:

```
image=/boot/vmlinuz-2.4.8-17mdk
        label=linux
        root=/dev/hda1
        read-only
```

Una sezione destinata a effettuare il boot di un kernel *GNU/Linux* comincia sempre con la direttiva `image=`, seguita dal percorso completo di un kernel *GNU/Linux* valido. Come ogni altra sezione, contiene un'etichetta `label=` che funge da identificatore univoco. La direttiva `root=` comunica a *LILO* qual è la partizione che ospita il filesystem `root` per questo sistema *GNU/Linux*. Ovviamente potrebbe essere diverso per il vostro sistema... La direttiva `read-only` ordina a *LILO* di montare questo filesystem `root` in modalità di sola lettura al momento dell'avvio del sistema: se questa direttiva non è presente, il fatto vi verrà segnalato con un messaggio di errore.

Subito dopo viene la sezione relativa a *Windows*:

```
other=/dev/hda2
        label=dos
        table=/dev/hda
```

Una sezione che comincia con `other=`, infatti, viene usata da *LILO* per avviare qualsiasi sistema operativo diverso da *GNU/Linux*: l'argomento passato a questa direttiva è il luogo in cui si trova il settore di boot di questo sistema operativo, nel caso specifico si tratta di un sistema *Windows*. Per trovare il settore di boot,

situato all'inizio della partizione che ospita questo secondo sistema operativo, *GNU/Linux* ha anche bisogno di sapere dove si trova la tabella delle partizioni, che gli permetterà di individuare la partizione in questione: questa informazione la ottiene grazie alla direttiva `table=`. La direttiva `label=`, esattamente come in una sezione `linux`, serve per identificare la sezione.

Adesso è il momento di aggiungere una sezione che provvede ad avviare il sistema usando il nuovo kernel. Potete inserirla ovunque dopo la sezione principale, a patto che non venga inserita all'interno di un'altra sezione. Ecco quale sarà il suo aspetto:

```
image=/boot/vmlinux-2.4.10-foo
        label=foo
        root=/dev/hda1
        read-only
```

Non dimenticate di modificarla in base alla configurazione del vostro sistema! Non per caso che abbiamo immaginato una situazione diversa rispetto a quella che abbiamo usato prima per *grub*...

Se avete compilato il vostro kernel abilitando l'opzione relativa al framebuffer, fate riferimento al paragrafo corrispondente, visto in precedenza, per *grub*; la differenza adesso è che l'opzione si trova su una propria riga:

```
vga=0x315
```

Ecco l'aspetto che il nostro `lilo.conf` dovrebbe avere dopo tutte le modifiche apportate, compreso l'inserimento di qualche commento supplementare (tutte le righe che cominciano con un #), che verrà ignorato da *LILO*:

```
#
# Sezione principale
#
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
# Al momento del boot vogliamo la VGA normale. Il framebuffer, se
# lo usiamo, cambierà automaticamente la risoluzione:
vga=normal
# Il nostro messaggio di boot...
message=/boot/message
# L'immagine predefinita, indichiamo il nostro nuovo kernel:
default=foo
# Mostra il prompt...
prompt
# ... e aspetta 5 secondi
timeout=50
#
# Il nostro nuovo kernel: immagine predefinita
#
image=/boot/vmlinux-2.4.10-foo
        label=foo
        root=/dev/hda1
        read-only
# Se impieghiamo il framebuffer VESA:
        vga=0x315
#
# Il kernel originale
#
image=/boot/vmlinuz-2.4.8-17mdk
        label=linux
        root=/dev/hda1
        read-only
#
# La sezione Windows
#
other=/dev/hda2
        label=dos
        table=/dev/hda
```

Non dimenticate di adattare il file in modo che corrisponda alla vostra effettiva configurazione! Il filesystem `root` di *GNU/Linux* qui si trova in `/dev/hda1`, ma certo potrebbe risiedere altrove sul vostro sistema; lo stesso vale per il sistema operativo *Windows*.

Adesso che il file di configurazione è stato modificato in maniera appropriata, dovete dire a *LILO* di cambiare il settore di boot:

```
$ lilo
Added foo *
Added linux
Added dos
$
```

In questo modo avete la possibilità di compilare tutti i kernel che volete, e di aggiungere le sezioni necessarie per utilizzarli. Adesso non vi resta che riavviare il sistema per provare il nuovo kernel.

Capitolo 15. Risoluzione dei problemi più frequenti

15.1. Introduzione

Questo capitolo introdurrà alcuni principi guida fondamentali per la risoluzione dei problemi più comuni: cosa fare quando tutto sembra andare per il verso sbagliato o, ancora meglio, cosa fare per essere **preparati** quando qualcosa va per il verso sbagliato, e come rimediare.

Quante volte vi è capitato di sentirvi stupidi per non aver fatto una copia di backup di quel piccolo file di configurazione che avete modificato fino a comprometterne le funzioni? Quante volte avete perso le preferenze di tutte le vostre applicazioni dopo aver installato del software poco rispettoso verso i suoi "vicini", oppure per aver cancellato per errore dei file di configurazione? Quante volte il vostro computer ha smesso di effettuare il boot dopo aver installato quel kernel altamente sperimentale? A me è successo molte volte, in effetti più volte di quanto mi piacerebbe ammettere **:-)**

Alcune persone ricompilano il kernel, o modificano i file di configurazione, ogni giorno della settimana, ogni settimana del mese, ogni mese dell'anno. Molto probabilmente non appartenete a questo gruppo, ma, credetemi, un giorno dovrete fare qualcosa del genere, per cui è meglio supporre che compiti come questi non siano rari nella vita di ogni giorno di un sistema *GNU/Linux*. Ciascuna di queste operazioni, tuttavia, può essere effettuata senza problemi di sorta usando un pizzico di buon senso e seguendo alcune procedure e linee guida che vi spiegheremo in questo capitolo. Queste vi saranno d'aiuto quando **quei** momenti capiteranno.

Per cui, passiamo ai principi basilari necessari per essere pronti...

15.2. Creazione di un disco di boot

Nel caso il vostro sistema non possa più essere avviato normalmente per una delle ragioni che abbiamo descritto sopra, la prima cosa di cui avrete bisogno sarà un disco di boot. Un disco di boot vi permetterà di avviare il sistema e di risolvere in pochi minuti il problema che ha reso il vostro sistema instabile o non avviabile.

Sotto **Mandrake Linux** potete scegliere fra due modi diversi di creare un disco di boot: usando la linea di comando, o un programma a interfaccia grafica. Per creare un disco di boot è necessario, nel primo caso, essere `root`; se, invece, lanciate il programma a interfaccia grafica come utente normale sarà il programma stesso a chiedervi la password di `root`.

15.2.1. Creazione di un disco di boot usando la linea di comando

Una volta all'interno di una console, digitate `su` e la relativa password per diventare `root`, e poi:

```
[root@localhost]# mkbootdisk --device /dev/fd0 `uname -r`
```

quindi premete il tasto **Invio**. Vedrete comparire un messaggio più o meno come questo:

```
Insert a disk in /dev/fd0. Any information on the disk will be lost.
Press <Enter> to continue or ^C to abort:
```

Vediamo di spiegare questo esempio. `mkbootdisk` ha bisogno di due parametri fondamentali: uno è `--device [nome-del-dispositivo]`, che comunica a `mkbootdisk` il nome del dispositivo sul quale vogliamo scrivere il disco di boot. Nel nostro esempio abbiamo indicato `/dev/fd0`, che è il primo lettore di floppy del sistema. Nel 99,99% dei casi dovrebbe funzionare senza problemi, se così non fosse dovete soltanto specificare il dispositivo corretto.

L'altro parametro indispensabile è `[versione-del-kernel]`, che specifica a `mkbootdisk` il kernel da copiare sul disco di boot. Nel nostro esempio abbiamo digitato `` `uname -r` ``, che come risultato dà il nome del kernel attualmente in uso. In base al nostro esempio, quindi, otterremo la creazione di un dischetto di boot nel primo drive floppy (`/dev/fd0`) contenente il kernel corrente.

Notate che, se scegliete il kernel attualmente in uso, il risultato sarà un disco di boot che lo riproduce esattamente, compresi i moduli e altre caratteristiche particolari di quel kernel. Se non volete includere tutti i moduli nel vostro disco di boot, oppure se volete aggiungerne qualcuno (il modulo per il supporto di una unità a nastro, ad esempio), vi consigliamo il nostro strumento a interfaccia grafica, *drakfloppy*.

15.2.2. Usare drakfloppy per creare un disco di avvio

drakfloppy è uno strumento basato su GUI che vi consente di creare dischi di boot altamente personalizzati. Per usare *drakfloppy* potete usare il menu principale del pannello: spostatevi nella sezione **Configurazione/Boot e Init** e selezionate la voce corrispondente. A meno che non siate già `root`, vi verrà chiesta la password relativa, come potete vedere in Figura 15-1.

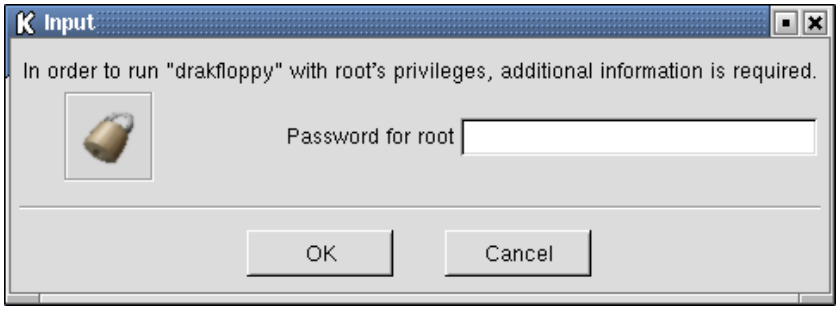


Figura 15-1. Inserite la password di `root`

Dopo di che comparirà sullo schermo la finestra principale di *drakfloppy* (Figura 15-2). Se desiderate generare un disco di boot "predefinito" (cioè identico a quello creato da linea di comando come abbiamo visto nell'esempio precedente), non dovete far altro che inserire un floppy nel lettore, selezionare il lettore appropriato dal menu a discesa e cliccare sul pulsante **Crea il disco**.

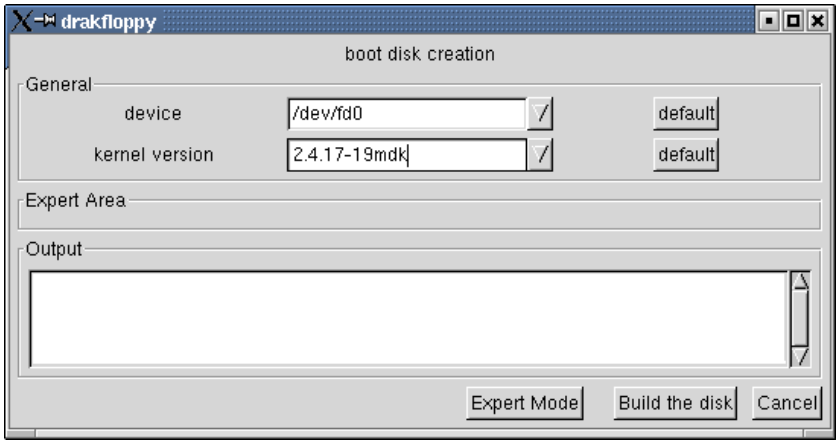


Figura 15-2. La finestra principale di `drakfloppy`

Se, invece, volete personalizzare il vostro disco di boot, dovrete cliccare sul pulsante **Modo esperto**: la finestra di *drakfloppy* diventerà quella che potete vedere in Figura 15-3.

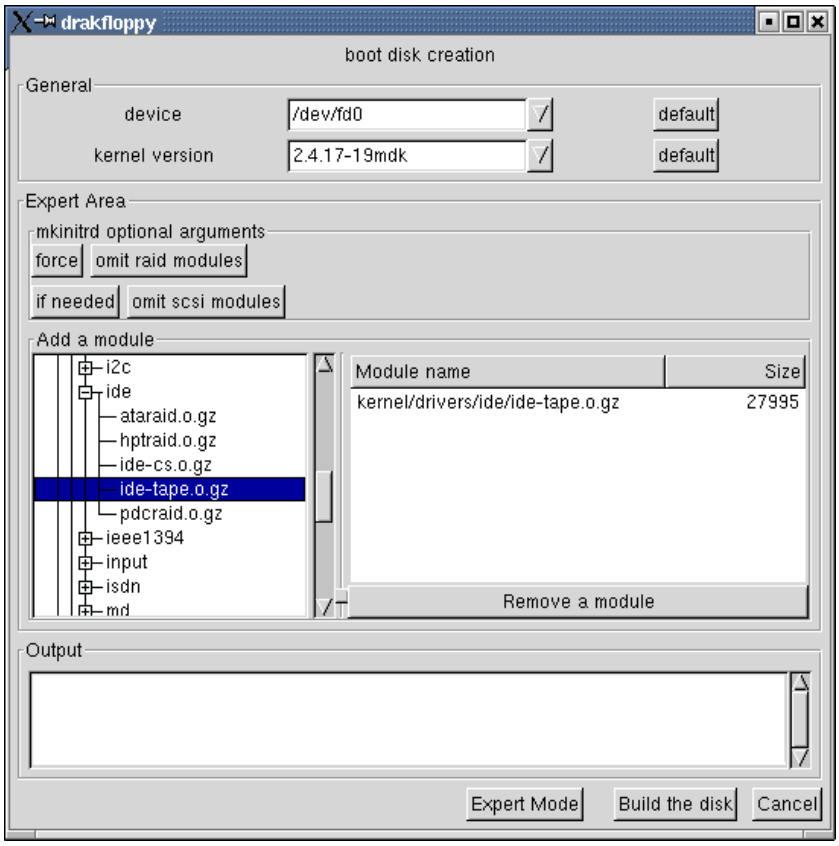


Figura 15-3. Creazione di un disco di boot personalizzato

L'**Area per esperti** è divisa in due sezioni: una che contiene dei pulsanti con delle opzioni per `mkinitrd` e un'altra, **Aggiungi un modulo**, che contiene una lista dei moduli organizzata secondo una struttura ad albero. Il significato dei pulsanti, nella prima, risulta piuttosto intuitivo. In questo esempio vogliamo includere il modulo per unità a nastri di tipo IDE e caricarlo preventivamente. Quando avrete terminato la personalizzazione del disco di avvio, cliccate sul pulsante **Crea il disco** per crearlo.

15.2.3. Test del disco di boot

A questo punto, è un'ottima idea provare il vostro disco di boot per essere sicuri che **funziona regolarmente**. Poche cose possono essere più imbarazzanti dello scoprire che, in caso di necessità, il dischetto di boot non può avviare il sistema a causa di errori di lettura. Se il dischetto è in grado di effettuare il boot normalmente, allora...

15.2.4. Avete finito!

Congratulazioni! Vi siete procurati lo strumento più importante per le operazioni di ripristino di un sistema danneggiato: un disco di boot **:-)** Adesso passiamo a qualche considerazione essenziale riguardo il secondo strumento in ordine di importanza: le copie di backup.

15.3. I backup

15.3.1. Perché fare copie di backup?

Effettuare copie di backup del vostro sistema è il **solo** metodo che consente di ripararlo nel caso sia rimasto gravemente danneggiato per qualche motivo (se, ad esempio, avete involontariamente cancellato alcuni file essenziali per il sistema, o se qualcuno è riuscito a penetrare nel vostro sistema e ne ha intenzionalmente compromesso il funzionamento, etc.). Per essere più sicuri, dovreste anche effettuare copie di salvataggio dei

vostri dati personali (file audio, immagini, documenti di testo, messaggi di posta elettronica, rubrica degli indirizzi, etc.).

Dovreste effettuare le copie di backup usando dispositivi di supporto adatti allo scopo (quanto più resistenti e duraturi possibile), e conservarli in un luogo sicuro, possibilmente lontano da quello che è il vostro luogo di lavoro abituale. Potete anche fare due copie di backup, una da tenere nelle vicinanze del computer e l'altra da conservare in un altro luogo. In breve, dovete essere sicuri di poter ripristinare tali backup se volete che tutto ciò abbia un senso.

15.3.2. Preparazione del sistema

Probabilmente tutto quello che vi serve è già installato sul sistema; anche il disco di boot dovrebbe trovarsi sempre a portata di mano (ne avete **creato** uno, vero?). Volendo, è possibile effettuare delle copie di backup usando il solo comando `tar` e un programma di compressione come `gzip` o `bzip2`. Potete vedere un esempio in *Esempio di backup usando TAR*, pag. 114.

In alternativa, potete usare alcuni programmi espressamente concepiti per questo scopo, quali *Taper*, *Time Navigator*, *Arkeia*, etc.

15.3.3. Di cosa fare il backup?

Ebbene, questa probabilmente è la domanda più difficile che ogni amministratore di sistema si pone quando arriva il momento di effettuare il backup. La risposta dipende da un certo numero di variabili: desiderate fare una copia di salvataggio soltanto dei vostri dati personali, dei file di configurazione o dell'intero sistema? Quanto tempo e/o spazio sarà necessario? Avete intenzione di effettuare il ripristino del backup sulla stessa macchina (o su una sulla quale è stata installata la stessa versione del sistema operativo) oppure su una completamente diversa?

Dato che questa guida si propone di aiutarvi a risolvere i problemi più comuni, ci concentreremo su un metodo di backup che ci consenta di riportare rapidamente il nostro sistema nello stato in cui si trovava prima che si verificasse quel terribile evento che lo ha reso inutilizzabile.

Come regola generale, sarà necessario fare il backup delle seguenti directory: `/etc`, `/home`, `/root` e `/var`. Se fate un backup completo del contenuto di queste directory avrete messo al sicuro non solo i vostri file di configurazione, ma anche i vostri dati personali (quelli che si trovano nella vostra directory home nel ramo `/home`). Notate che copiare tutte queste directory potrebbe richiedere un tempo piuttosto **lungo**, ma si tratta probabilmente del metodo più sicuro.

Uno schema più sofisticato consiste nel fare il backup soltanto dei file di configurazione che sono stati modificati, ignorando quelli che non sono cambiati. Questo approccio richiede una "pianificazione" maggiore, ma permette backup più veloci (e anche un ripristino più veloce) e che possono essere spostati da una macchina (o versione del sistema) all'altra più "semplicemente".

Come passo successivo, vi proponiamo una lista dei file cui dovreste prestare particolare attenzione. Notate che questa lista non è esaustiva, soprattutto se apportate molte modifiche alla configurazione del sistema[1].

Nella directory `/etc`:

`/etc/lilo.conf`

Contiene la configurazione del boot loader *LILO*. Se usate *grub* invece di *LILO*, i file di cui fare il backup si trovano nella directory `/boot/grub`.

`/etc/fstab`

Contiene la configurazione delle partizioni del disco rigido e i relativi punti di mount.

`/etc/modules.conf`

Specifica i moduli da caricare e i loro parametri in base all'hardware presente nel vostro sistema. Potrebbe rivelarsi di scarsa utilità nel caso intendiate effettuare il ripristino su una macchina **molto** diversa, ma potreste comunque ricavarne delle indicazioni utili.

1. Ma se siete in grado di personalizzare a fondo il vostro sistema, probabilmente non avreste comunque bisogno di questa lista.

`/etc/isapnp.conf`

Contiene le impostazioni di `ISAPnP` se lo avete utilizzato per configurare le vostre schede ISA Plug & Play.

`/etc/X11/XF86Config`

Il file che raccoglie le opzioni di configurazione di *X*, il motore grafico di *GNU/Linux* e di tutti i suoi window manager e ambienti grafici.

`/etc/cups`

Il file di configurazione di *cups*, il sistema di stampa predefinito di **Mandrake Linux**.

`/etc/printcap`

Se non usate *cups*, preferendogli il sistema di stampa `lpr`, allora dovete fare il backup di questo file invece di `cups` per mettere in salvo le impostazioni della vostra stampante.

`/etc/bashrc`

Le opzioni di configurazione della shell *bash* per tutti gli utenti.

`/etc/profile`

Configura l'ambiente di sistema e alcuni programmi che vengono eseguiti al momento dell'avvio del sistema.

`/etc/crontab`

La configurazione delle operazioni eseguite da cron a intervalli regolari, per la manutenzione del sistema ad esempio.

`/etc/rc.d/*`

La configurazione dei vari runlevel del sistema. In genere un backup di questi file non è necessario, a meno che non abbiate aggiunto un runlevel personalizzato o ne abbiate modificato uno di quelli predefiniti.

`/etc/inittab`

Specifica il runlevel predefinito in cui verrà avviato il sistema.

`/etc/ssh`

Contiene le impostazioni di `ssh`. Se effettuate degli accessi remoti è **molto** importante che questo file venga conservato.

Se sul vostra sistema è normalmente in esecuzione un server web, un server FTP o altri servizi, ricordatevi di fare una copia di backup anche dei file di configurazione relativi.

Nella directory `/root` e in tutte le home directory degli utenti (`/home/nome_dell_utente`), le directory che seguono:

`~/.gnome/*`

Impostazioni dell'ambiente grafico *GNOME*.

`~/.kde/*`

Impostazioni dell'ambiente grafico *KDE*.

`~/.netscape/*`

Impostazioni dei programmi Netscape: i preferiti del Navigatore, i filtri di posta di Messenger, etc.

`~/nsmail/*`

Contiene **tutti** i vostri messaggi di posta elettronica e dei gruppi di discussione. Certo non volete rischiare di **perderli**, giusto?

`~/Mail/*`

Se usate *kmail* questa directory contiene **tutti** i vostri messaggi di posta elettronica. Certo non volete rischiare di **perderli**, giusto?

`~/.ssh/*`

Contiene le impostazioni personalizzate per l'uso di `ssh`. Se utilizzate `ssh`, fatene una copia di backup...

Inoltre non dimenticate di controllare i file che seguono:

`~/.bash_profile`

Contiene le variabili d'ambiente, gli alias e altre impostazioni della shell *bash*.

`~/.bashrc`

Altre impostazioni di *bash*.

`~/.cshrc`

Contiene le variabili d'ambiente, gli alias e altre impostazioni della shell *CSH*.

`~/.tcshrc`

Contiene le variabili d'ambiente, gli alias e altre impostazioni della shell *tcsh*.

Notate che non possiamo menzionare ogni file di configurazione che può essere presente sul vostro sistema perché questo richiederebbe un intero libro. Inoltre i programmi utilizzati variano da utente a utente: se non utilizzate *netscape*, ad esempio, non è necessario fare il backup dei file e delle directory relativi a *netscape*, se non usate `ssh` non è necessario preoccuparsi dei suoi file di configurazione, e così via.

Riassumendo, fate un backup di tutti i file di configurazione dei programmi che usate abitualmente, e di tutti i file di configurazione che avete modificato. Inoltre fate copie di backup di tutti i vostri dati personali, e di quelli di tutti gli utenti del sistema. Non ve ne pentirete, credetemi.

15.3.4. Cosa usare per il backup?

L'altra domanda fondamentale: la risposta dipende dalla quantità di dati che devono essere copiati nel backup, dal tempo che volete dedicare a questa operazione, dalla facilità di accesso ai supporti del backup, e un'altra lunga lista di fattori da prendere in considerazione.

Come indicazione generale, vi servono supporti che siano come minimo abbastanza capienti da accogliere i dati di cui volete fare il backup, e sufficientemente veloci perché l'intero processo non duri un'eternità.

15.3.5. Supporti di backup

Per aiutarvi nella scelta vi proponiamo una breve descrizione dei supporti di backup comunemente disponibili. Noterete una grande variabilità in termini di capacità, affidabilità e velocità. L'elenco non segue nessun ordine particolare, solo quello in cui ci sono venuti in mente. Notate che il vostro software di backup potrebbe non essere in grado di sfruttarli tutti.

L'elenco che segue non costituisce certo una rassegna esaustiva di tutti i supporti per backup esistenti, inoltre quanto scritto qui sotto potrebbe cambiare in futuro. Alcune informazioni, come la durata media prevista, sono tratte dai siti web dei produttori e/o da esperienze personali e comuni. Molti punti di vista, inoltre, come giudizi sul prezzo o sulla velocità, possono essere piuttosto **personali**.

Floppy Disk

La sua capacità arriva fino a 1.44 Mb[2]. Sono facili da portare con sé, ma per le necessità odierne lo spazio può risultare del tutto insufficiente. Il modo migliore per trasportare piccoli file. Lenti. Economici. Esiste un lettore di dischetti in praticamente tutti i computer. La durata media prevista è di 4 o 5 anni.

Floppy Disk LS120

La sua capacità è di 120 Mb. Le dimensioni sono identiche rispetto a quelle di un floppy normale, ma la capacità è almeno dieci volte maggiore. Non altrettanto economici. Necessita di un lettore di floppy speciale, ma quest'ultimo può leggere e scrivere anche i dischetti normali. Potrebbe costituire un buon sostituto per questi ultimi, ma la sua velocità è inferiore rispetto a quella dei drive *ZIP*. Accessibili in lettura e scrittura. La durata media prevista è più o meno la stessa dei dischi *ZIP*.

Dischi *ZIP*

La sua capacità arriva fino a 250 Mb. Malgrado non siano sottili come i floppy, sono comunque facilmente trasportabili, e molto più rispondenti alle necessità odierne. Le caratteristiche generali sono piuttosto equilibrate, per quanto possano risultare ancora un po' costosi. Accessibili in lettura e scrittura. La durata media prevista è di dieci anni per i dischi da 100 Mb, forse qualcosa in più per quelli da 250 Mb.

CD-R

Negli ultimi tempi la sua capacità arriva fino a 700 Mb, per quanto lo standard sia di 650 Mb. Un supporto molto economico e affidabile. Oggi molti sostengono che la capacità è ormai insufficiente, ma 650 Mb ci sembrano una quantità ragionevole. La sua caratteristica più attraente è data dal fatto che quasi ogni computer sulla faccia della terra dispone di un lettore CD-ROM, per cui possono essere letti praticamente dappertutto. Vengono scritti una volta per tutte, ma in seguito possono essere letti quante volte desiderate (o potete, ad esser precisi...). La durata media prevista è di 20 anni, forse di più se conservati in un posto sicuro e letti non troppo di frequente **:-)**

CD-RW

Valgono le stesse considerazioni fatte per i CD-R, con la differenza che possono essere formattati e riscritti circa mille volte. Un supporto ragionevolmente economico e affidabile. La durata media prevista è di 15 anni, forse di più se conservati in un posto sicuro e letti non troppo di frequente.

DVD registrabili/riscrivibili

Si tratta di una delle ultimissime entrate nel campo dei supporti di backup. La sua capacità è di 4.7 Gb per i DVD registrabili su un unico lato. I masterizzatori di DVD sono ancora piuttosto costosi, ma questo fatto è almeno in parte compensato dalla possibilità di archiviare ben 4.7 Gb di dati su un singolo disco. La durata media prevista è di 15 anni, forse di più se conservati in un posto sicuro e letti non troppo di frequente.

Nastri magnetici

La loro capacità va dai 120 Mb (c'è qualcuno che ha ancora dei nastri così vecchi?) fino a diversi gigabyte. Sono supporti costosi e non molto affidabili (ehi, dopo tutto sono **davvero** nastri magnetici). Malgrado ciò, la loro grande capacità ne fa il supporto ideale (almeno per il momento) per il backup di server e altri sistemi che ospitano una grande quantità di dati usando un unico supporto, o pochi. Lo svantaggio maggiore è dato dal fatto che l'accesso al nastro è di tipo sequenziale, il che costituisce un grave rallentamento per le operazioni di backup/ripristino; tuttavia le unità a nastro di tipo SCSI sono sufficientemente veloci per i bisogni attuali, e dopo tutto offrono **realmente** molti gigabyte di spazio per archiviare i vostri file. Lettura/scrittura. La durata media prevista raggiunge i 30 anni per i nastri costruiti secondo le tecnologie più recenti.

Dischi rigidi

Forse vi chiederete se l'autore di questo capitolo ha alzato il gomito una volta di troppo: ebbene, posso rassicurarvi in merito, non è successo niente del genere[3]. Il fatto è che, visto il calo costante dei loro prezzi, i dischi rigidi dovrebbero essere considerati seriamente anche come un possibile supporto per i vostri backup. Sono piuttosto economici, offrono un sacco di spazio (fino a 100 Gb al momento in cui sto scrivendo), e sono molto affidabili e veloci, i più veloci tra tutti i supporti descritti in questo elenco.

2. In effetti possono essere formattati fino a 1.92 Mb usando *SuperFormat* e il vostro lettore standard, ma questa è un'altra storia...
3. N.d.T.: Anche perché l'autore di questo capitolo è astemio **:-)**

Se possedete un computer portatile questa possibilità potrebbe esservi preclusa[4], ma sui vostri sistemi da scrivania aggiungere un disco rigido in più per scopi di backup potrebbe essere un'ottima scelta. In effetti, potreste anche evitare di aggiungere un nuovo disco rigido ed effettuare i backup su quello che avete, ma questa potrebbe rivelarsi una pessima idea poiché non vi mette al riparo nel caso di un guasto del disco rigido.

Altri supporti rimovibili

Sono disponibili sul mercato anche altri dispositivi basati su supporti rimovibili (l'*ORB* di Castlewood, e il *JAZ* di IOMEGA, ad esempio) che offrono un buon rapporto prezzo/prestazioni e sono adatti per effettuare backup. Alcuni sono stati persino propagandati come "sostituti per i dischi rigidi" (il JAZ, ad esempio), ma quando usati come dischi rigidi la loro durata potrebbe risentirne a causa delle limitazioni nella loro progettazione: in effetti **non sono** dischi rigidi. I vostri risultati possono variare sensibilmente, comunque, quel che dovete fare è scegliere saggiamente (usando un pizzico di buon senso) in base ai vostri bisogni, e ... buona fortuna!

Directory remote

Ebbene, forse queste non possono essere considerate dei "supporti" veri e propri, ma vi accenneremo rapidamente in quanto si tratta di una buona scelta se disponete di spazio sufficiente e di un collegamento veloce.

Se il vostro ISP (il fornitore di servizi Internet) mette a vostra disposizione una certa quantità di spazio sul proprio server potreste utilizzarlo per copiarvi i vostri file oltre alle pagine web. Sul web troverete molte offerte di servizi di stoccaggio dati remoto. Se, invece, avete una rete di cui fanno parte due o più macchine, potete effettuare i vostri backup su una macchina "remota" della vostra rete (diversa da quella che necessita di un backup, naturalmente...).

Tenete presente che effettuare dei backup "remoti" può rappresentare una falla nella sicurezza del vostro sistema, per cui non copiate il vostro archivio top secret e neppure i vostri file più importanti su backup remoti. Vi ricordiamo, inoltre, che in caso di blocco irreversibile del sistema probabilmente non potrete nemmeno connettervi a un sito remoto per recuperare i vostri file...

In conclusione, ricordate che potete anche usare più supporti diversi contemporaneamente seguendo una vostra strategia di backup: nastri e CD-R, dischi rigidi e nastri, dischi rigidi e CD-R, etc.

15.3.6. Quando effettuare il backup?

Ci sono molti modi di organizzare un piano per effettuare dei backup a intervalli regolari. In questa sezione ve ne proporremo alcuni. Ricordate che non sono obbligatori, non sono i migliori, né, tanto meno, gli unici possibili. Sono delle indicazioni generali che potrebbero tornarvi utili per costruire una vostra strategia di backup.

È possibile scegliere fra molte strategie diverse, in gran parte dipendono dal tipo di supporto che utilizzate, da quanto spesso i vostri dati cambiano, e da quanto importanti questi dati sono per voi o per la vostra azienda. Secondo una di queste strategie, ad esempio, è necessario effettuare un backup completo ogni fine settimana, e un backup incrementale (dei soli file che sono stati modificati) ogni giorno; è inoltre previsto un backup completo ogni mese, da conservare in due copie diverse in due luoghi diversi. Questa strategia può dimostrarsi utile per un'azienda, ma non per l'utente di un singolo computer. Per i vostri dati personali potete seguire uno schema di questo tipo: effettuare una copia settimanale dei vostri file sul disco rigido, e ogni mese copiare questi backup su CD-R o nastro.

4. Se il vostro portatile è relativamente recente potrebbe offrire lo spazio per l'installazione di un secondo disco rigido. Inoltre usando la porta USB o la porta parallela è possibile collegare dei dischi esterni.

15.3.7. Esempio di backup usando TAR

Adesso vi proponiamo un piccolo script di backup che usa `tar` per effettuare un backup completo della vostra directory home.

È necessario avere i permessi di lettura sui file e di lettura ed esecuzione sulle directory di cui si vuole fare il backup, altrimenti l'operazione non avrà successo.

```
#!/bin/bash

# Questo script crea un backup compresso della vostra directory home
# in un file chiamato backup.tar.gz o backup.tar.bz2, a seconda dello
# schema di compressione usato.

BACKUP_DIRS=$HOME

# Rimuovete il commento della riga seguente se volete dei backup
# compressi con GZip
#tar cvzf backup.tar.gz $BACKUP_DIRS

# Con questo comando effettuiamo il backup usando BZip...
tar cvjf backup.tar.bz2 $BACKUP_DIRS
```

Come potete vedere, questo è uno script di backup **molto** semplice: si limita a fare un backup della vostra directory home e copia l'archivio compresso in quella stessa directory. Vediamo di migliorarlo un po'...

```
#!/bin/bash

# Questo script crea un backup compresso di tutte le directory
# specificate e copia il file che ne risulta in una directory
# di nostra scelta.

BACKUP_DIRS="$HOME /etc /etc/rc.d"
BACKUP_FILENAME=`date '+%b%d%Y'`
BACKUP_DEST_DIR=$HOME

# Rimuovete il commento della riga seguente se volete dei backup
# compressi con GZip
#tar cvzf $BACKUP_DEST_DIR/$BACKUP_FILENAME.tar.gz $BACKUP_DIRS

# Con questo comando effettuiamo il backup usando BZip...
# Marcate la riga seguente come commento se volete dei backup
# compressi con GZip
tar cvjf $BACKUP_DEST_DIR/$BACKUP_FILENAME.tar.bz2 $BACKUP_DIRS
```

Come potete vedere da quest'ultimo esempio, abbiamo aggiunto alcune directory e abbiamo usato un metodo per assegnare un nome al file in maniera tale che questo includa la data del backup.

Naturalmente in seguito potete spostare il file `tar.bz2` o `tar.gz` prodotto dallo script su qualsiasi supporto desideriate. Potete anche effettuare direttamente il backup su un supporto di vostra scelta, purché vi accertiate di averlo montato su una directory e di aver cambiato la variabile `BACKUP_DEST_DIR` dello script in modo che punti a tale directory. Vi incoraggiamo a migliorare ancora questo script e a renderlo più flessibile.

Per il ripristino di backup prodotti in questo modo, per favore consultate *Esempio di ripristino usando TAR*, pag. 115.

15.4. Il ripristino

Il ripristino di un backup dipende dal programma, dal supporto e dalla pianificazione che avete utilizzato per effettuarlo. Non cercheremo di trattare tutti i possibili casi in questa sede, ci limiteremo a ricordarvi di prestare attenzione al fatto che, se volete ripristinare i vostri file di configurazione e i vostri dati secondo la situazione preesistente, dovete accertarvi di effettuare il ripristino di file e/o directory esattamente nelle locazioni dove si trovavano al momento del backup.

15.4.1. Esempio di ripristino usando TAR

Ecco, quindi, un piccolo script per il ripristino del backup che abbiamo fatto con `tar` usando lo script visto in *Esempio di backup usando TAR*, pag. 114.

È necessario avere i diritti di scrittura dei file e delle directory di cui si vuole fare il ripristino, altrimenti l'operazione non avrà successo.

```
#!/bin/bash

# Questo script estrae il contenuto di un backup che si trova nella
# directory specificata, copiando i file nella loro sede originaria.

BACKUP_SOURCE_DIR=$HOME
RESTORE_FILENAME=$1

# Rimuovete il commento della riga seguente se volete effettuare il
# ripristino di backup compressi con GZip
#tar xvzf $BACKUP_SOURCE_DIR/$RESTORE_FILENAME

# Con questo comando effettuiamo il ripristino usando BZip...
tar xvjf $BACKUP_SOURCE_DIR/$RESTORE_FILENAME
```

Come potete vedere, questo script è piuttosto semplice: tutto quello che dobbiamo fare è digitare come argomento il nome del backup che intendiamo ripristinare (solo il nome del file, non il percorso completo) e lo script si occuperà di estrarre i file e ricopiarli nello stesso luogo in cui si trovavano originariamente.

15.4.2. Creazione di un CD-ROM di ripristino

Esiste un solo metodo per prepararsi all'eventualità di un "disastro totale", ed è quello che consiste nell'effettuare un backup "completo" del vostro sistema. Applicazioni come *`mkCDrec`* sono in grado di mettervi in condizioni di ripartire nel giro di pochi minuti.

Se siete gli orgogliosi proprietari di un Mandrake Linux - Edizione PowerPack Deluxe, troverete questo programma nel CD-ROM "contrib". In caso contrario, potete procurarvelo, insieme alla documentazione completa, visitando il sito web di mkCDrec (`http://mkcdrec.ota.be`).

`mkCDrec` vi permette di effettuare il backup su più CD-ROM, di "clonare" il vostro disco rigido (ovvero di copiare tutto il contenuto di un disco o una partizione su un altro disco con caratteristiche simili – come minimo le stesse dimensioni), e altro ancora.

Per effettuare il ripristino di un intero sistema usando *`mkCDrec`* dovete riavviare il computer usando il primo CD-ROM della serie e seguire le istruzioni che compariranno sullo schermo.

15.5. Il mio sistema si blocca durante la fase di boot

Può succedere che il vostro sistema si pianti mentre sta effettuando il boot. In tal caso, non lasciatevi prendere dal panico, continuate a leggere.

15.5.1. Il sistema si blocca durante il boot

Se il vostro sistema inizia a effettuare il boot senza problemi, ma si blocca quando compare il messaggio `Rebuilding RPM database` o `Finding module dependencies`, dovete solo premere **CTRL**+**C**: in tal modo il sistema salterà queste operazioni e continuerà fino al termine del boot. Una volta terminato, digitate `rpm --rebuilddb` come `root` se il sistema si è bloccato nella fase `Rebuilding RPM database`; se, invece, si trattava della fase `Finding module dependencies`, molto probabilmente questo significa che avete effettuato un aggiornamento del kernel, ma questo non è stato eseguito in maniera appropriata. Controllate che i file nelle directory `/boot` e `/lib/modules` corrispondano alla versione del kernel attualmente in uso (il loro nome deve contenere il numero della versione corrente). Se non corrispondono, consultate il capitolo *Compilazione e installazione di nuovi kernel*, pag. 97 per sapere come rimediare a questa situazione.

Se il processo di boot si blocca quando compare il messaggio `RAMDISK: Compressed image found at block 0`, significa che il file `initrd` non è più valido. Potete provare a effettuare il boot usando un'altra delle immagini

descritte in `lilo.conf`, oppure effettuare il boot con un sistema di emergenza (usando il dischetto di boot o un'altra immagine tra quelle disponibili) e modificare o cancellare la sezione `initrd=` nel file `/etc/lilo.conf`.

15.5.2. Il controllo del filesystem al momento del boot è fallito

Se, per un motivo qualsiasi, non avete potuto uscire dal sistema nella maniera corretta, al momento del boot successivo il sistema eseguirà un controllo automatico del filesystem. Può succedere che questo test non abbia buon esito e si interrompa, nel qual caso il sistema vi metterà a disposizione una console. Digitate `e2fsck -py [dispositivo]`, dove `[dispositivo]` è il nome della partizione di cui è fallito il controllo del filesystem. L'opzione `-p` dice al comando `e2fsck` di fare tutte le riparazioni necessarie senza interpellarci; l'opzione `-y`, invece, risponde automaticamente `yes` a ogni eventuale domanda. Quando la fase di test e riparazione è finita, premete i tasti **CTRL+D** per lasciare la console di emergenza: il sistema si riavvierà.

Se questo errore compare con una certa regolarità, alcuni blocchi del vostro disco rigido potrebbero essere rovinati. Digitate `e2fsck -c [dispositivo]` per stabilire se è questo il caso. Questo comando provvederà a individuare e contrassegnare automaticamente eventuali blocchi rovinati, in maniera tale da impedire al filesystem di scrivere dati su di essi. `e2fsck`, come abbiamo visto, controlla il filesystem in modalità automatica soltanto se quest'ultimo non è stato smontato correttamente durante l'ultima chiusura del sistema, oppure se se è stato raggiunto il `maximal mount count` (il numero di volte che un filesystem può essere montato prima che il sistema faccia un controllo di routine). Per obbligare il programma a fare un controllo usate l'opzione `-f`.

È consigliabile effettuare il processo di individuazione di eventuali blocchi rovinati soltanto su partizioni non montate. Inoltre può richiedere una quantità di tempo spaventosa: ricordate che, per quanto si tratti di un'operazione necessaria, avrete il tempo di bere più di un caffé prima che venga portata a termine.

15.6. Re-installazione del bootloader

Può capitare di fare uno sbaglio e di cancellare il MBR (Master Boot Record), oppure il responsabile è qualche programma impazzito, o ancora è qualche virus che gira sotto *Windows* (che avete ancora accanto a *GNU/Linux*) a farlo. Potreste quindi pensare che ormai è impossibile effettuare il boot del sistema, giusto? **sbagliato!** Come vedremo, ci sono molti modi per recuperare il boot loader.

15.6.1. Uso di un disco di boot

Questo fatto non potrà certo sorprendervi: per recuperare il boot loader **avrete bisogno** di un disco di boot. Senza di esso potreste incontrare seri problemi[5]. Avete creato un dischetto di boot, vero?

Per prima cosa, inserite il dischetto nel lettore e riavviate il computer. Il passo successivo dipende in gran parte dal boot loader da voi utilizzato: *LILO* o *grub*. Qualunque sia il boot loader, comunque, tutti i comandi che dovrete eseguire vanno lanciati come `root`.

15.6.1.1. Usando LILO

Se usate *LILO*, è sufficiente digitare quanto segue dalla linea di comando: `/sbin/lilo`. In questo modo *LILO* verrà reinstallato nel settore di boot del vostro disco rigido e il problema sarà risolto.

15.6.1.2. Usando GRUB

Se usate *grub*, le cose sono un po' diverse rispetto a *LILO*... ma non abbiate timore, siamo qui per aiutarvi.

L'esempio che segue presuppone che voi abbiate installato *grub* nel MBR del primo disco IDE, e che il file `stage1` si trovi nella directory `/boot/grub/`.

5. A meno che non abbiate fatto un backup dell'MBR, ne riparleremo in seguito...

Per prima cosa, richiamate la shell di *grub* digitando `grub`. Quindi digitate questo comando: `root (hd0,0)`; in questo modo comunicherete a *grub* che i file di cui ha bisogno si trovano nella prima partizione (il secondo `0`) del primo disco rigido (`hd0`). Poi digitate quanto segue: `setup (hd0)`; con questo comando installerete *grub* nel MBR del primo disco rigido. Ecco fatto!

Potete anche provare a usare il comando `grub-install /dev/hda` per installare *grub* sul MBR del primo disco rigido, ma il metodo descritto in precedenza è preferibile.

15.6.1.3. Fatto!

Questo è tutto quello che dovete sapere riguardo la reinstallazione del boot loader.

15.6.2. Riparazione di un Super-Block danneggiato

Quanto segue è applicabile soltanto ai filesystem ext2 ed ext3. Se state utilizzando un altro filesystem, per favore consultate la documentazione relativa per avere informazioni su questo argomento.

Il super-block è il primo blocco di ogni partizione ext2fs. Contiene dati importanti riguardo il filesystem stesso, come le dimensioni, lo spazio libero rimasto, etc. Ha funzioni simili a quelle della FAT (*File Allocation Table*) delle partizioni *DOS* e *Windows*. Una partizione il cui super-block è danneggiato non può essere montata. Per fortuna il filesystem ext2fs conserva diverse copie del super-block sparse su tutta la partizione.

Avviate il vostro sistema con il disco di boot che avete creato in precedenza (ne **avete** creato uno, vero?). Le copie di backup si trovano, in genere, all'inizio di ogni blocco di 8 Kb (8192 byte): la copia di backup più vicina al super-block, quindi, si trova in corrispondenza del byte numero 8193. Per effettuare il ripristino del super-block usando questa copia, digitate `e2fsck -b 8193 /dev/hda4`; ricordatevi di cambiare `hda4` in maniera che corrisponda al nome della vostra partizione danneggiata! Se anche quel blocco risulta danneggiato, provate quello successivo al byte numero 16384, e così via, finché non ne troverete uno intatto. Riavviate il vostro computer per attivare i cambiamenti.

15.7. I Runlevel

15.7.1. Breve descrizione dei runlevel

Un runlevel consiste in una configurazione del software di sistema tale da permettere l'esistenza di alcuni processi selezionati soltanto. Per ogni runlevel, i processi ammissibili sono specificati nel file `/etc/inittab`. Ci sono otto runlevel predefiniti: `0`, `1`, `2`, `3`, `4`, `5`, `6`, `S`. Volendo, potete anche creare un vostro runlevel personalizzato. Per una descrizione più dettagliata riguardo i runlevel consultate il capitolo *I file di avvio del sistema: init sysv*, pag. 59.

15.7.2. Che cosa possono fare per me i runlevel?

Effettuare il boot del sistema in un runlevel diverso da quello usuale può aiutarvi a risolvere alcuni problemi. Supponiamo, ad esempio, che abbiate apportato una modifica alla configurazione di *X* tale da renderlo non più utilizzabile, e che al momento del boot venga avviato come opzione predefinita: se questo è il caso, potete riavviare il sistema in modalità console, correggere l'errore e ripartire normalmente con *X*. Vediamo come possiamo fare.

Come opzione predefinita, *GNU/Linux* effettua il boot nel runlevel `3` (la console) oppure nel runlevel `5` (*X*). Il runlevel predefinito è specificato nel file `/etc/inittab`. Cercate una riga come `id:3:initdefault:` (se il vostro sistema effettua il boot nella console) o `id:5:initdefault:` (se il vostro sistema lancia automaticamente *X*).

Se volete effettuare il boot in un runlevel diverso da quello indicato in `/etc/inittab`, dovete specificarlo quando compare il prompt del boot. Se usate *LILO*, digitate `linux init 3` per avviare il sistema nella console, o `linux init 5` per lanciare una sessione X Window. Se usate *grub*, invece, premete due volte il tasto **E**, aggiungete `init 3` per effettuare il boot usando la console o `init 5` per avviare X Window, quindi premete il tasto **Invio** e poi **B** per effettuare il boot.

15.8. Ripristino di file cancellati

In questa sezione discuteremo di alcuni metodi per recuperare file e directory cancellati. Ricordate che gli strumenti che eseguono questo tipo di operazione non hanno poteri magici, e che sono limitati dalla quantità di tempo che è trascorsa dal momento in cui avete cancellato il file che adesso volete recuperare.

Potreste chiedervi: "Ora che ho involontariamente cancellato questo file, come posso fare per recuperarlo?". Non abbiate timore, sono a vostra disposizione alcuni programmi di utilità progettati specificamente per il filesystem ext2 di *GNU/Linux* che vi permettono di recuperare file e directory cancellati per errore. Dovete sapere, tuttavia, che questi programmi non possono recuperare file cancellati mesi fa perché quasi sicuramente sono stati sovrascritti nell'uso normale del disco rigido (lo spazio occupato dai file cancellati viene marcato come "libero" dal filesystem). Il modo **migliore** per difendersi dalla cancellazione accidentale (o meno) dei vostri dati è effettuare regolarmente dei backup secondo quanto vi abbiamo spiegato in precedenza.

Per favore ricordate che non esistono (ancora) strumenti per recuperare file cancellati su filesystem ReiserFS. Visitate frequentemente la pagina home di ReiserFS (`http://www.namesys.com/`) per le ultime novità e aggiornamenti in merito a tale filesystem.

E adesso diamo uno sguardo a questi strumenti di recupero file cancellati. Uno di questi è *Recover*. Si tratta di uno strumento "interattivo". Se siete gli orgogliosi proprietari di un Mandrake Linux - Edizione PowerPack Deluxe, troverete questo programma nel CD-ROM "contribs". Altrimenti potrete reperirlo sul sito web RPMFind (`http://www.rpmfind.net`). Fate una rapida ricerca e scaricate il file RPM appropriato, dopo di che installatelo. A questo punto è sufficiente digitare `recover [opzioni]` e rispondere alle domande che vi porrà. Queste riguardano l'impostazione di un periodo di tempo all'interno del quale cercare file e directory, in maniera da ridurre i tempi di ricerca[6].

Dopo che il programma avrà terminato la fase di ricerca, vi chiederà dove volete salvare i file e le directory recuperate. Indicate una directory a vostra scelta, i vostri dati vi verranno copiati. Notate che non sarà possibile recuperare i nomi dei file, ma soltanto il loro contenuto; in seguito potrete ispezionarli o cercare di rinominarli finché non avrete individuato quello desiderato. Sempre meglio che nulla **:-)**

Esistono anche dei mini-*HOWTO* che trattano proprio del recupero di file cancellati in ext2, consultate quelli intitolati Ext2fs-Undeletion (`http://www.linuxdoc.org/HOWTO/mini/Ext2fs-Undeletion.html`) (traduzione italiana: Ext2fs-Undeletion (`http://ildp.linux.it/HOWTO/mini/Ext2fs-Undeletion.html`)) e undeletion of whole directory structures (`http://www.linuxdoc.org/HOWTO/mini/Ext2fs-Undeletion-Dir-Struct/index.html`).

15.9. Recupero di un sistema bloccato

Quando il vostro computer "si pianta", il sistema non risponde più ai comandi e i dispositivi di input, come la tastiera e il mouse, sembrano essere bloccati. Questa è la possibilità peggiore, e può essere il sintomo di un problema grave nella configurazione del sistema, nel software o nell'hardware. Vi mostreremo come affrontare questa fastidiosa situazione.

Se il sistema si blocca, la vostra principale preoccupazione dovrebbe essere di chiudere la sessione di lavoro nella maniera corretta. Supponiamo che stiate lavorando sotto *X*, se le cose stanno così provate a eseguire quanto segue esattamente in quest'ordine:

- Provate a "uccidere" il server *X* premendo contemporaneamente i tasti **ALT+CTRL+BACKSPACE**.

6. Potete anche cercare **tutti** i file cancellati, ma questa ricerca richiederà più tempo...

- Passate a un'altra console premendo i tasti **ALT**+**CTRL**+**F2**. Se questo tentativo ha successo, effettuate il login come `root` e digitate il comando: `kill -15 $(pidof X)`, o il comando `kill -9 $(pidof X)` se il primo sembra non produrre nessun effetto. Per vedere se *X* è ancora in esecuzione lanciate `top`.
- Se il vostro computer è connesso a una rete locale, potete provare a connettervi al vostro sistema usando `ssh` da un'altra macchina della rete. Vi consigliamo di accedere al sistema come utente normale e digitare `su` per diventare `root`.
- Se il sistema non risponde positivamente a nessuno di questi tentativi, sarete costretti a ricorrere alla sequenza "SysRq" ("System Request"). La sequenza "SysRq" richiede che l'utente prema tre tasti contemporaneamente: il tasto **ALT** di sinistra, il tasto **SysRq** (etichettato come **StampaSchermo** o **PrintScreen** sulle tastiere più vecchie, e **R Sist** su quelle più recenti) e il tasto di una lettera.
  - **Left ALT**+**SysRq**+**r** fa funzionare la tastiera in modalità "raw": adesso possiamo provare di nuovo a premere **ALT**+ **CTRL**+ **BACKSPACE** per "uccidere" *X*. Se ancora non funziona, continuate a leggere.
  - **Left ALT**+**SysRq**+**s** tenta di scrivere su disco tutti i dati non salvati (ovvero tentata di effettuare un "sync" del disco).
  - **Left ALT**+**SysRq**+**e** invia un segnale di "fine operazioni" a tutti i processi, con l'eccezione di `init`.
  - **Left ALT**+**SysRq+i** invia un segnali di "fine operazioni" a tutti i processi, fatta eccezione per `init`.
  - **Left ALT**+**SysRq**+**u** tenta di montare nuovamente tutti i filesystem già montati in modalità di sola lettura. Questo passo elimina il *dirty flag* ed evita il controllo automatico del filesystem al momento del riavvio.
  - **Left ALT**+**SysRq**+**b** riavvia il sistema. Otterreste lo stesso risultato premendo il pulsante *reset* sulla vostra macchina.

Ricordate che questa è una sequenza, in altre parole dovete premere una combinazione di tasti dopo l'altra secondo l'ordine corretto: **R**aw, **S**ync, t**E**rm, k**I**ll, **U**mount, re**B**oot[7]. Se volete saperne di più su questa caratteristica consultate il file `/usr/src/linux/Documentation/sysrq.txt`.

- Se niente di quanto abbiamo descritto sopra risolve il problema, incrociate le dita e premete il pulsante "reset" della vostra macchina. Se siete fortunati, *GNU/Linux* si limiterà a effettuare un controllo del filesystem al momento del riavvio.

È essenziale che in qualche modo riusciate a individuare la causa dei blocchi di sistema, perché questi possono causare seri danni al filesystem. Potreste anche prendere in considerazione la possibilità di usare `ReiserFS`, un filesystem di tipo *journaling* che fa parte della distribuzione **Mandrake Linux** a partire dalla versione 7.0, che riesce a gestire eventuali malfunzionamenti in maniera da minimizzare il rischio di perdita dei dati. Notate, comunque, che sostituire ext2fs con `ReiserFS` richiede una nuova formattazione delle vostre partizioni.

15.10. Come terminare applicazioni fuori controllo

Ecco un'operazione che non richiede troppo sforzo. Non dovreste averne bisogno, in effetti, ma nel caso che ciò avvenga... Esistono molti modi per terminare l'esecuzione di un programma: potete individuare il PID del programma impazzito e usare il comando `kill` per eliminarlo, oppure potete usare `xkill` o un altro strumento grafico, come i programmi che mostrano l'albero dei processi.

15.10.1. Dalla console

La prima cosa da fare per terminare un programma bloccato o malfunzionante è di individuare il suo PID (per `process ID`). Per fare questo, e supponendo che sia *netscape* il programma in questione, digitate quanto segue sulla linea di comando: `ps aux | grep netscape`. Otterrete le seguenti informazioni:

```
pippo      3505  7.7 23.1 24816 15076 pts/2   Z    21:29   0:02 /usr/lib/netscape
```

Il programma ci dice, fra le altre cose, che *netscape* è stato lanciato dall'utente `pippo` e che il suo PID è `3505`.

Adesso che sappiamo il PID del programma malfunzionante, possiamo eseguire il comando `kill` per terminarlo. Pertanto digitiamo quanto segue: `kill -9 3505`, e abbiamo finito! *netscape* verrà terminato. Notate che questa procedura va usata **soltanto** quando il programma non risponde più ai vostri comandi. **Evitate** di usarla come metodo standard per uscire dalle applicazioni.

Tutto quello che abbiamo fatto è stato inviare il segnale `KILL` al processo numero `3505`. Il comando `kill` accetta altri segnali oltre a `KILL`, in maniera tale da permettervi un notevole controllo dei vostri processi. Per ulteriori informazioni, consultate la pagina di manuale relativa e il capitolo *Controllo dei processi*, pag. 33.

15.10.2. Usando XKILL

Mandrake Linux comprende un programma "kill" basato su interfaccia grafica: `xkill`. è molto utile per terminare applicazioni grafiche bloccate o malfunzionanti con un solo clic. Se usate *KDE*, troverete la sua icona sul vostro desktop: dovete soltanto cliccare su di essa e, dopo che il puntatore del mouse sarà diventato un teschio con sotto delle tibie incrociate, spostatelo sopra la finestra dell'applicazione che volete terminare e cliccateci sopra. Ecco fatto! Se invece usate *GNOME*, dovete lanciare `xkill` usando il menu di sistema (**Applicazioni/Monitoring**), la modalità d'uso è la stessa.

Se usate *KDE* potete anche lanciare `xkill` per mezzo della seguente scorciatoia da tastiera: **CTRL+ALT+ESC**

15.10.3. Usando altri strumenti basati su GUI

Un'altra alternativa è l'uso di uno dei vari programmi basati su GUI il cui scopo è quello di tenere sotto controllo lo stato del sistema: *KPM*, *KSySGuard*, e *GTOP*, per citarne solo alcuni. Questi programmi vi permettono di selezionare i processi in esecuzione per mezzo del mouse, e di inviare loro uno dei segnali possibili, compreso quello di terminazione del processo.

15.11. Strumenti di risoluzione dei problemi specifici di Mandrake Linux

Ogni programma di amministrazione del sistema (*Control Center*, *RpmDrake*, *drakgw*, *draknet*, *mandrakeupdate*, *drakbackup*, etc.) è, potenzialmente, uno strumento utile per risolvere eventuali problemi. Potete usare questi programmi per ripristinare la configurazione, aggiungere o rimuovere software, aggiornare il vostro sistema con le versioni più recenti e aggiornate del software distribuito da **MandrakeSoft**, etc.

15.12. Considerazioni finali

Bene, come avete potuto constatare ci sono molti modi per rimettere in piedi il sistema dopo un'emergenza oltre alla completa reinstallazione dello stesso [8] Certo, per applicare alcune delle tecniche descritte in questo capitolo è necessaria una certa esperienza, ma siamo sicuri che non vi ci vorrà molto per acquisirla. Speriamo, comunque, che non siate costretti a diventare dei maestri in quest'ambito ... anche se certo non guasta averne una conoscenza di base. Speriamo che quanto illustrato in questo capitolo, e i relativi esempi, vi saranno utili al momento opportuno: buona fortuna nel risolvere un'emergenza!

8. Il modo normale per risolvere i problemi di certi altri sistemi operativi...

Appendice A. La Licenza Pubblica Generica GNU

Questa è una traduzione italiana non ufficiale della Licenza Pubblica Generica GNU (GNU General Public License), applicabile alla maggior parte dei programmi che sono contenuti nelle distribuzioni **Mandrake Linux**. Non è pubblicata dalla Free Software Foundation e non ha valore legale nell'esprimere i termini di distribuzione del software che usa la licenza GPL. Solo la versione originale in inglese della licenza ha valore legale. Ad ogni modo, speriamo che questa traduzione aiuti le persone di lingua italiana a capire meglio il significato della licenza GPL.

Traduzione originale curata da gruppo Pluto, da ILS e dal gruppo italiano di traduzione GNU. Ultimo aggiornamento: 19 aprile 2000.

A.1. Premessa

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 gli utenti. Questa licenza si applica alla maggioranza dei programmi della Free Software Foundation (alcuni altri programmi della Free Software Foundation sono invece coperti dalla Licenza Pubblica Generica Minore GNU, *GNU Lesser General Public License*, LGPL) e ad ogni altro programma i cui autori abbiano deciso di usare questa licenza. 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 egli stesso 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 vostri diritti in due modi:

1. coprendo il software con un copyright;
2. offrendovi una licenza che dia il permesso legale di copiare, distribuire e modificare il software.

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 i destinatari sappiano che ciò che loro possiedono non è l'originale, in modo che eventuali problemi introdotti da altri non influiscano 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 di sua proprietà. Per prevenire questa evenienza, abbiamo chiarito che ogni brevetto deve essere concesso in licenza a chiunque per un utilizzo libero, oppure non essere concesso in alcun modo.

Seguono gli esatti termini e condizioni per la copia, la distribuzione e la modifica.

A.2. Termini e condizioni per la copia, la distribuzione e la modifica

1. Questa licenza si applica a ogni programma o altra opera che contenga una nota da parte del detentore del copyright che dica che tale opera può essere distribuita nei 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, riprodotta letteralmente oppure modificata e/o tradotta in un'altra lingua (da qui in avanti, la traduzione è sotto tutti gli aspetti considerata una "modifica").

   Attività diverse dalla copia, distribuzione e modifica non sono coperte da questa licenza e non rientrano nelle sue finalità. 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 stesso (indipendentemente dal fatto che sia stato creato eseguendo il programma). In base alla natura del programma il suo output può essere o meno coperto da questa Licenza.

2. È lecito copiare e distribuire copie letterali del codice sorgente del programma così come viene ricevuto, con qualsiasi mezzo, a condizione che venga riprodotta chiaramente su ogni copia una appropriata nota di copyright e di assenza di garanzia; che si mantengano intatti tutti i riferimenti a questa Licenza e all'assenza di ogni garanzia; che si dia a ogni altro destinatario del programma una copia di questa Licenza insieme al programma.

   È possibile richiedere un pagamento per il trasferimento fisico di una copia del programma, è anche possibile a propria discrezione richiedere un pagamento in cambio di una copertura assicurativa.

3. È lecito modificare la propria copia o copie del programma, o parte di esso, creando perciò un'opera basata sul programma, e copiare e distribuire tali modifiche o tale opera secondo i termini della precedente sezione 1, a patto che siano soddisfatte tutte le condizioni che seguono:

   a. Bisogna indicare chiaramente nei file che si tratta di copie modificate e la data di ogni modifica.

   b. Bisogna fare in modo che ogni opera distribuita o pubblicata, che in parte o nella sua totalità derivi dal programma o da parti di esso, sia concessa in licenza gratuita nella sua interezza ad ogni terza parte, secondo i termini di questa licenza.

   c. Se normalmente il programma modificato legge comandi interattivamente quando viene eseguito, bisogna fare in modo che all'inizio dell'esecuzione interattiva usuale esso stampi un messaggio contenente una appropriata nota di copyright e di assenza di garanzia (oppure che specifichi il tipo di garanzia che si offre). Il messaggio deve inoltre specificare che chiunque può ridistribuire il programma alle condizioni qui descritte e deve indicare come reperire questa licenza. Se però lo stesso programma originale è interattivo ma normalmente non stampa un simile 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 possano 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'intera opera, 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 derivate dal programma o che lo contengano.

   Inoltre, la semplice aggregazione di un'opera non derivata dal programma con il programma stesso o con un'opera da esso derivata, su di un mezzo di memorizzazione o di distribuzione, non è sufficiente a includere l'opera non derivata nell'ambito di questa licenza.

4. È lecito copiare e distribuire il programma (o un'opera basata su di esso, come espresso nella sezione 2) sotto forma di codice oggetto o eseguibile secondo i termini delle precedenti sezioni 1 e 2, a patto che si applichi una delle seguenti condizioni:

a. Il programma sia corredato dal codice sorgente completo, in una forma leggibile da calcolatore, e tale sorgente sia fornito secondo le regole delle precedenti sezioni 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 delle precedenti sezioni 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 punto 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 il destinatario.

5. Non è lecito copiare, modificare, sublicenziare, o distribuire il programma in modi diversi da quelli espressamente previsti da questa licenza. Ogni altro tipo di tentativo di copiare, modificare, sublicenziare o distribuire il programma non è autorizzato, e farà terminare automaticamente i diritti garantiti da questa Licenza. In ogni caso, qualsiasi soggetto che abbia ricevuto copie o diritti, coperti da questa licenza, da parte di soggetti che abbiano violato la licenza come qui indicato, non vedranno invalidata la loro licenza, purché si comportino conformemente ad essa.

6. Il destinatario non è obbligato ad accettare questa licenza, poiché non l'ha firmata. D'altra parte nessun altro documento garantisce il permesso di modificare o distribuire il programma o le opere derivate 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 opere basate su di esso.

7. Ogni volta che il programma o un'opera basata su di esso vengono ridistribuiti, il ricevente 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 al ricevente nel suo esercizio dei diritti qui garantiti. Chi distribuisce programmi coperti da questa licenza non è comunque tenuto a imporre il rispetto di questa Licenza a terzi.

8. Se, come conseguenza del giudizio di un tribunale, o di una imputazione per la violazione di un brevetto o per ogni altra ragione (non limitatamente a questioni di brevetti), vengono imposte condizioni che contraddicono le condizioni di questa licenza, che queste condizioni siano dettate dalla corte, da accordi tra le parti o altro, queste condizioni non esimono nessuno dall'osservazione di questa licenza. Se non è possibile distribuire un prodotto in un modo che soddisfi simultaneamente gli obblighi dettati da questa licenza e altri obblighi pertinenti, il prodotto non può essere affatto distribuito. Per esempio, se un brevetto non permettesse a tutti quelli che lo ricevono di ridistribuire il programma senza obbligare al pagamento di diritti, allora l'unico modo per soddisfare contemporaneamente il brevetto e questa licenza è di non distribuire affatto il Programma.

   Se una qualunque parte di questa sezione è ritenuta non valida o non applicabile in una qualunque circostanza, deve comunque essere applicata l'idea espressa dalla sezione stessa; in ogni altra circostanza invece deve essere applicata questa sezione nella sua interezza.

   Non è nelle finalità di questa sezione 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 questa sezione 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 il licenziatario non può imporre la scelta del sistema di distribuzione.

Questo paragrafo serve a rendere il più chiaro possibile ciò che crediamo sia una conseguenza del resto di questa Licenza.

9. Se in alcuni paesi la distribuzione o l'uso del programma sono limitati da brevetto o dall'uso di interfacce coperte da copyright, il detentore del copyright originale che pone il programma sotto questa licenza può aggiungere limiti geografici espliciti alla distribuzione, per escludere questi paesi dalla distribuzione stessa, in modo che il programma possa essere distribuito solo nei paesi non esclusi da questa regola. In questo caso i limiti geografici sono inclusi in questa licenza e ne fanno parte a tutti gli effetti.
10. All'occorrenza la Free Software Foundation può pubblicare revisioni o nuove versioni di questa Licenza Pubblica Generica. Tali nuove versioni saranno simili a questa nello spirito, ma potranno differire nei dettagli al fine di affrontare nuovi problemi e nuove situazioni.

    Ad ogni versione viene assegnato un numero identificativo. Se il programma asserisce di essere coperto da una particolare versione di questa licenza e "da ogni versione successiva", il destinatario può scegliere se seguire le condizioni della versione specificata o di una qualsiasi versione successiva pubblicata dalla Free Software Foundation. Se il programma non specifica quale versione di questa licenza deve applicarsi, il destinatario può scegliere una qualsiasi versione tra quelle pubblicate dalla Free Software Foundation.

11. Se si desidera incorporare parti del programma in altri programmi liberi le cui condizioni di distribuzione differiscano da queste, è possibile scrivere all'autore del programma per chiederne l'autorizzazione. Per il software il cui copyright è detenuto dalla Free Software Foundation, si scriva alla Free Software Foundation; talvolta facciamo eccezioni alle regole di questa Licenza. La nostra decisione sarà guidata da due finalità: preservare la libertà di tutti i prodotti derivati dal nostro software libero e promuovere la condivisione e il riutilizzo del software in generale.

NESSUNA GARANZIA

12. POICHÉ IL PROGRAMMA È CONCESSO IN USO GRATUITAMENTE, NON C'È ALCUNA 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 DEL DESTINATARIO. SE IL PROGRAMMA DOVESSE RIVELARSI DIFETTOSO, IL DESTINATARIO SI ASSUME IL COSTO DI OGNI MANUTENZIONE, RIPARAZIONE O CORREZIONE NECESSARIA.
13. NÉ IL DETENTORE DEL COPYRIGHT NÉ ALTRE PARTI CHE POSSONO MODIFICARE O RIDISTRIBUIRE IL PROGRAMMA COME PERMESSO IN QUESTA LICENZA SONO RESPONSABILI PER DANNI NEI CONFRONTI DEL DESTINATARIO, 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 DAL DESTINATARIO 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

Appendice B. GNU Free Documentation License

B.1. GNU Free Documentation License

Versione 1.1, marzo 2000

0. PREMESSA

Lo scopo di questa licenza è di rendere un manuale, un testo o altri documenti scritti "liberi", nel senso di assicurare a tutti la libertà effettiva di copiarli e ridistribuirli, con o senza modifiche, per fini di lucro e non. 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 tutte le opere derivate dal documento originale devono essere ugualmente libere. Essa è di complemento alla GNU General Public License, 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 essere accompagnato da manuali che consentano di esercitare le identiche libertà che il programma stesso consente. Ma questa licenza non è limitata alla documentazione del software; può essere utilizzata per qualsiasi testo, indipendentemente dall'argomento trattato o dall'avvenuta pubblicazione cartacea. Consigliamo questa licenza principalmente per opere che abbiano fini didattici o per manuali di consultazione.

1. APPLICABILITÀ E DEFINIZIONI

Questa licenza è applicabile a qualsiasi manuale o altra opera che contenga una nota, scritta dal detentore del copyright, in cui si dica che l'opera può essere distribuita nei termini di questa licenza. Con "documento", in seguito, ci si riferisce a qualsiasi manuale od opera contenente tale nota. Qualsiasi persona del pubblico è un destinatario della licenza, e viene indicato con "voi".

Viene indicata come "versione modificata" del documento qualsiasi opera contenente il documento stesso o parte di esso, sia esso riprodotto alla lettera oppure modificato e/o tradotto in un'altra lingua.

Una "sezione secondaria" è un'appendice o una premessa del documento che riguarda esclusivamente il rapporto dell'editore o dell'autore del documento con l'argomento generale del documento stesso (o con argomenti affini), e non contiene nulla che possa far parte dell'argomento generale (per esempio, se il documento è in parte un manuale di matematica, una sezione secondaria non può contenere spiegazioni di matematica). Il suddetto rapporto può basarsi su relazioni storiche con l'argomento o con argomenti affini, oppure su posizioni legali, commerciali, filosofiche, etiche o politiche pertinenti.

Le "sezioni non modificabili" sono particolari sezioni secondarie i cui titoli sono inclusi nella lista delle sezioni non modificabili, contenuta nella nota che indica che il documento è pubblicato sotto questa licenza.

I "testi di copertina" sono dei brevi brani di testo che sono elencati come testi della prima di copertina o testi dell'ultima di copertina nella nota che indica che il documento è pubblicato sotto questa licenza.

Una copia "trasparente" del documento indica una copia leggibile da un calcolatore, che sia in un formato le cui specifiche sono disponibili pubblicamente, i cui contenuti possano essere visti e modificati direttamente e semplicemente con generici editor di testi, con generici programmi di grafica (per immagini composte da pixel) o con programmi di disegno facilmente reperibili (per i disegni), e che sia pronta per l'impaginazione o per la conversione automatica in diversi formati adatti all'impaginazione. Una copia che sia in un formato diversamente trasparente, la cui struttura sia stata progettata per intralciare o scoraggiare modifiche future da parte dei lettori, non è trasparente. Una copia non trasparente è detta "opaca".

Esempi di formati adatti per copie trasparenti sono l'ASCII puro senza marcatori, il formato di input per Texinfo, il formato di input per LaTeX, SGML o XML basati su una DTD disponibile al pubblico, e semplice HTML conforme agli standard e progettato per essere modificato manualmente. I formati opachi comprendono PostScript, PDF, formati proprietari che possono essere letti e modificati solo con elaboratori di testi proprietari, SGML o XML per i quali non siano pubblicamente disponibili la DTD e/o gli strumenti per l'elaborazione, e HTML generato automaticamente da alcuni elaboratori di testo per soli scopi di output.

Con "pagina del titolo", in un libro stampato, indichiamo la pagina del titolo stessa più le pagine seguenti necessarie a contenere, in modo leggibile, il materiale che questa licenza richiede che compaia nella pagina del titolo. Per opere in formati che non contemplino una vera e propria pagina del titolo, con "pagina del titolo" si intende il testo in prossimità della più evidente occorrenza del titolo dell'opera, precedente l'inizio del corpo del testo.

2. COPIE ALLA LETTERA

Potete riprodurre e distribuire il documento con l'ausilio di qualsiasi mezzo, per fini di lucro e non, a condizione che tutte le copie contengano questa licenza, le note sul copyright e l'avviso che questa licenza si applica al documento, e che non aggiungiate altre condizioni al di fuori di quelle di questa stessa licenza. Non potete usare misure tecniche per impedire o controllare la lettura o la ulteriore riproduzione delle copie che producete o distribuite. Potete però ricevere compensi in cambio delle copie prodotte. Se distribuite un numero di copie sufficientemente elevato dovete seguire anche le condizioni della sezione 3.

Potete anche prestare copie, con le stesse condizioni sopra menzionate, e potete mostrare le copie in pubblico.

3. COPIE IN QUANTITÀ

Se pubblicate a mezzo stampa più di 100 copie del documento, e la nota della licenza del documento richiede la presenza dei testi di copertina, dovete racchiudere le copie in copertine che riportino, in modo chiaro e leggibile, tutti i testi di copertina indicati: i testi della prima di copertina in prima di copertina, e i testi dell'ultima di copertina in ultima di copertina. Entrambe le copertine devono inoltre identificare voi, in modo chiaro e leggibile, come editore che pubblica quelle copie. La prima di copertina deve presentare il titolo completo con tutte le parole che lo compongono egualmente visibili ed evidenti. Potete aggiungere altro materiale alle copertine. La riproduzione con modifiche limitate alle sole copertine, purché queste modifiche preservino il titolo del documento e rispettino le condizioni viste in precedenza, può essere considerata una copia alla lettera sotto tutti gli altri aspetti.

Se i testi richiesti per qualche copertina sono troppo lunghi per essere riprodotti sulla relativa copertina in modo leggibile, dovete metterne la prima parte (finché la copertina ne può ragionevolmente contenere) sulla copertina vera e propria, e far continuare il testo nelle pagine adiacenti.

Se pubblicate o distribuite copie opache del documento in numero superiore a 100, dovete anche accludere a ogni copia opaca una copia trasparente leggibile da un calcolatore, oppure indicare all'interno o insieme ad ogni copia opaca l'indirizzo di una rete informatica pubblicamente accessibile contenente una copia trasparente completa del documento, priva di materiale aggiuntivo, che possa essere scaricata dal generico pubblico con accesso alla rete anonimamente e gratuitamente usando i protocolli di rete pubblici standard. Se adottate quest'ultima opzione dovete disporre le appropriate misure, al momento di iniziare la distribuzione in quantità di copie opache, affinché la copia trasparente rimanga accessibile all'indirizzo indicato per almeno un anno dopo l'ultima distribuzione di una copia opaca (direttamente o attraverso vostri agenti o rivenditori) di quella edizione al pubblico.

È caldamente consigliato, benché non obbligatorio, contattare gli autori del documento con largo anticipo, prima di distribuirne un numero considerevole di copie, in modo da permettere loro di fornirvi una versione aggiornata del documento.

4. MODIFICHE

Potete copiare e distribuire una versione modificata del documento rispettando le condizioni delle precedenti sezioni 2 e 3, purché pubblichiate la versione modificata sotto questa stessa identica licenza, con la versione modificata nel ruolo di "documento", così da permetterne la distribuzione e la modifica a chiunque ne possieda una copia. Inoltre, nella versione modificata dovete:

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, quando esistenti, devono essere elencate nella sezione "Storia" del documento). Potete usare lo stesso titolo di una versione precedente se l'editore di quella versione ve ne ha dato il permesso.

B. Elencare nella pagina del titolo, come autori, una o più persone o gruppi responsabili delle modifiche nella versione modificata, insieme ad almeno cinque fra i principali autori del documento originale (o tutti gli autori principali, se questi sono meno di cinque).

C. Indicare 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, vicino alle altre note di copyright, un'appropriata nota di copyright per le modifiche da voi effettuate.

F. Includere, immediatamente dopo le note di copyright, un avviso di licenza che dia il permesso pubblico di usare la versione modificata nei termini di questa licenza, nella forma mostrata nell'addendum alla fine di questo testo.

G. Mantenere in questo avviso di licenza l'intera lista di sezioni non modificabili e testi di copertina indicati nell'avviso di licenza del documento.

H. Includere una copia non modificata di questa licenza.

I. Conservare la sezione intitolata "Storia" e il suo titolo, e aggiungere a questa una voce che riporti almeno il titolo, l'anno, i nuovi autori e l'editore della versione modificata come figurano nella pagina del titolo. Se non ci sono sezioni intitolate "Storia" nel documento, createne una che riporti il titolo, l'anno, gli autori e l'editore del documento come figurano nella pagina del titolo, quindi aggiungete una voce che descriva la versione modificata come appena detto.

J. Conservare l'indirizzo in rete, se esistente, indicato nel documento per consentire l'accesso pubblico a una copia trasparente del documento stesso, e allo stesso modo gli indirizzi in rete indicati nel documento per le precedenti versioni su cui esso si basava. Questi possono essere collocati nella sezione "Storia". Potete omettere l'indirizzo di un'opera se essa è stata pubblicata almeno quattro anni prima del documento stesso, o se l'editore della versione a cui l'indirizzo si riferisce ve ne dà il permesso.

K. Conservare il titolo di qualsiasi sezione intitolata "Riconoscimenti" o "Dediche", e mantenere inalterati all'interno della sezione stessa tutto il contenuto e il tono di ciascuno dei riconoscimenti e/o dediche ai collaboratori ivi contenuti.

L. Conservare inalterate tutte le sezioni non modificabili del documento, sia nei testi che nei titoli. I numeri delle sezioni o elementi equivalenti non sono considerati parte dei titoli delle sezioni.

M. Cancellare qualsiasi sezione intitolata "Certificazioni". Una sezione di questo tipo non può essere inclusa nella versione modificata.

N. Non cambiare il titolo di sezioni esistenti in "Certificazioni" o in titoli che siano già utilizzati per sezioni non modificabili.

Se la versione modificata comprende nuove sezioni di primaria importanza, o appendici classificabili come sezioni secondarie che non contengono materiale copiato dal documento originale, avete la facoltà di dichiarare non modificabili tutte queste sezioni o solo alcune di esse. Per fare ciò, aggiungete i loro titoli nella lista delle sezioni non modificabili nella nota di licenza della versione modificata. Questi titoli devono essere diversi da quelli di qualsiasi altra sezione.

Potete aggiungere una sezione intitolata "Certificazioni", a patto che essa non contenga altro che certificazioni conferite alla vostra versione modificata da parte di vari soggetti - ad esempio, indicazioni di avvenuta revisione o di approvazione del testo da parte di una organizzazione come definizione ufficiale di uno standard.

Potete aggiungere alla lista dei testi di copertina della versione modificata un testo lungo fino a cinque parole come testo della prima di copertina, e un testo lungo non più di 25 parole come testo dell'ultima di copertina. Possono essere aggiunti da (o indirettamente da parte di) un unico soggetto un solo testo di prima di copertina e un solo testo di ultima di copertina. Se il documento include già un testo di copertina per la stessa copertina, precedentemente aggiunto da voi o indirettamente da parte dello stesso soggetto per conto del quale voi operate, non ne potete aggiungere un altro; potete però sostituire il vecchio testo, previo consenso esplicito da parte del precedente editore che lo aveva aggiunto.

Gli autori e gli editori del documento non concedono il permesso, con questa licenza, di usare i loro nomi per pubblicizzare una qualsiasi versione modificata o per indicare o implicare per essa una certificazione.

5. UNIONE DI DOCUMENTI

Potete unire il documento con altri documenti pubblicati sotto questa stessa licenza, secondo i termini definiti nella precedente sezione 4 per quanto concerne le versioni modificate, a condizione che includiate nell'unione tutte le sezioni non modificabili di tutti i documenti originali, inalterate, e le elenchiate tutte come sezioni non modificabili del documento risultante dall'unione, nella relativa nota di licenza.

Nel documento risultante è sufficiente che sia presente una sola copia di questa licenza, ed eventuali sezioni non modificabili identicamente ripetute possono essere sostituite da una singola copia. Se ci sono più sezioni non modificabili aventi lo stesso titolo ma contenuti differenti, rendete unico il titolo di ciascuna di queste sezioni aggiungendovi alla fine, fra parentesi, il nome dell'autore o dell'editore originali della sezione, se noti, o altrimenti un numero distintivo. Apportate gli stessi cambiamenti ai titoli delle sezioni nell'elenco delle sezioni non modificabili nella nota di licenza del documento risultante.

Durante l'operazione dovete unire le varie sezioni intitolate "Storia" dei vari documenti originali in un'unica sezione intitolata "Storia"; allo stesso modo dovete unire tutte le sezioni intitolate "Riconoscimenti" e tutte le sezioni intitolate "Dediche". Dovete eliminare tutte le sezioni intitolate "Certificazioni".

6. RACCOLTE DI DOCUMENTI

Potete organizzare una raccolta che consista del documento e di altri documenti pubblicati sotto questa licenza, e sostituire le singole copie di questa licenza contenute nei vari documenti con una sola copia inclusa nella raccolta, a condizione che rispettiate, per tutti gli altri aspetti e per ciascun documento, le regole imposte da questa licenza per le copie alla lettera.

Potete estrapolare un singolo documento da una simile raccolta e distribuirlo individualmente sotto questa licenza, a condizione che inseriate una copia di questa licenza nel documento stesso e che rispettiate questa licenza sotto tutti gli altri aspetti per quanto concerne le copie alla lettera di quel documento.

7. RACCOLTE CON OPERE INDIPENDENTI

Una raccolta costituita dal documento o sue derivazioni più altri documenti od opere separati e indipendenti, all'interno di o a formare un archivio o un supporto per la distribuzione, non è considerata nella sua interezza come versione modificata del documento, a condizione che non venga rivendicato alcun copyright per l'intera raccolta. Una simile raccolta viene detta "aggregato", e questa licenza non si applica alle altre opere raccolte in essa insieme al documento per il solo fatto di essere raccolte insieme ad esso, se non sono esse stesse opere derivanti dal documento.

Se le condizioni riguardanti il testo di copertina indicate nella sezione 3 sono applicabili a queste copie del documento, allora, se il documento costituisce meno di un quarto dell'intero aggregato, i testi di copertina del documento possono essere scritti su copertine che delimitano solo il documento all'interno dell'aggregato. Altrimenti devono apparire nella copertina dell'intero aggregato.

8. TRADUZIONI

La traduzione è considerata un tipo di modifica, e di conseguenza potete distribuire traduzioni del documento rispettando i termini della sezione 4. La sostituzione delle sezioni non modificabili con le rispettive traduzioni richiede un permesso speciale da parte dei detentori dei corrispondenti copyright, ma potete includere traduzioni di tutte le sezioni non modificabili, o solo di parte di esse, in aggiunta alle versioni originali delle stesse sezioni. Potete includere una traduzione della presente licenza a condizione che includiate anche la versione originale in inglese della licenza stessa. In caso di discordanza fra la traduzione e l'originale inglese di questa licenza sarà considerata valida la versione originale inglese.

9. RISOLUZIONE DELLA LICENZA

Non potete riprodurre, modificare, sublicenziare o distribuire il documento al di fuori dei termini espressamente previsti da questa licenza. Ogni altro tentativo di riprodurre, modificare, sublicenziare o distribuire il documento non è autorizzato, e farà terminare automaticamente i diritti che questa licenza vi garantisce. Comunque, per quanto riguarda eventuali soggetti che abbiano ricevuto copie o diritti da voi sotto questa licenza, le loro licenze continueranno a essere valide, a condizione che i predetti soggetti continuino a rispettare in pieno i termini delle licenze stesse.

10. REVISIONI FUTURE DI QUESTA LICENZA

La Free Software Foundation può pubblicare, di tanto in tanto, nuove versioni aggiornate della GNU Free Documentation License. Tali nuove versioni saranno simili nello spirito alla presente versione, ma potrebbero differire in alcuni dettagli per affrontare nuove problematiche e nuovi argomenti. Si veda http://www.gnu.org/copyleft (`http://www.gnu.org/copyleft/`).

A ogni versione della licenza viene assegnato un numero che la contraddistingue. Se nel documento è specificato che ad esso si applica un particolare numero di versione della licenza "o qualsiasi versione successiva", potete scegliere se seguire i termini e le condizioni della particolare versione specificata o di una qualsiasi versione successiva che sia stata pubblicata (in forma definitiva) dalla Free Software Foundation. Se nel documento non è specificato alcun particolare numero di versione di questa licenza, potete scegliere una qualsiasi versione fra tutte quelle che sono state pubblicate (in forma definitiva) dalla Free Software Foundation.

B.2. Come applicare questa licenza ai vostri documenti

Per applicare questa licenza a un documento che avete scritto, includete una copia della licenza stessa nel documento e aggiungete le seguenti note di copyright e di licenza immediatamente dopo la pagina del titolo:

> Copyright (c) ANNO VOSTRO NOME. è consentita la riproduzione, la distribuzione e/o la modifica di questo documento secondo i termini della GNU Free Documentation License, versione 1.1 o qualsiasi versione successiva pubblicata dalla Free Software Foundation, considerando le sezioni non modificabili ELENCARNE I TITOLI, i testi della prima di copertina ELENCO, e i testi dell'ultima di copertina ELENCO. Una copia della licenza è acclusa nella sezione intitolata "GNU Free Documentation License".

Se non ci sono sezioni non modificabili, scrivete "senza sezioni non modificabili", invece di dire quali sono non modificabili. Se non ci sono testi della prima di copertina, scrivete "nessun testo della prima di copertina", invece di "i testi della prima di copertina ELENCO"; allo stesso modo per i testi dell'ultima di copertina.

Se il vostro documento contiene esempi non banali di codice sorgente di programmi, vi consigliamo di pubblicare parallelamente anche gli esempi, applicandovi una licenza per il software libero di vostra scelta, come ad esempio la GNU General Public License, al fine di permetterne l'uso come software libero.

Glossario

account

Su un sistema `Unix`, un account (o `login`) è la combinazione di un nome, una directory personale, una password e una `shell` che consentono a un utente di connettersi al sistema.

al volo

Si dice che qualcosa viene eseguito "al volo" (ingl. *on the fly*) quando questo accade in un'unica operazione, senza passaggi intermedi.

alias

Il meccanismo usato in una `shell` per sostituire una stringa con un'altra prima di eseguire il comando. Potete vedere tutti gli alias definiti nella sessione corrente digitando `alias` al prompt.

ambiente

È il contesto di esecuzione di un processo. Include tutte le informazioni di cui ha bisogno il sistema operativo per gestire il processo, e quanto serve al processore per eseguire il processo in maniera corretta.
Si veda anche: processo.

APM

Advanced Power Management. Una caratteristica propria di alcuni `BIOS` che permette alla macchina di entrare in standby dopo un certo periodo di inattività. Sui portatili, APM ha anche il compito di verificare la carica residua della batteria e, se questa caratteristica è supportata, di stimare il periodo di funzionamento che questa permette.

arp

Address Resolution Protocol. Il protocollo Internet usato per rimappare dinamicamente un indirizzo Internet su indirizzi fisici (hardware) che appartengono a una rete locale. Il suo uso è limitato a reti che supportano il broadcasting hardware.

ASCII

American Standard Code for Information Interchange. La codifica standard impiegata per memorizzare caratteri, inclusi i caratteri di controllo, su un computer. La prima metà del set di caratteri in molte codifiche a 8-bit (come l'ISO 8859-1, il set di caratteri predefinito di Linux) è costituita proprio dall'ASCII.
Si veda anche: ISO 8859.

assembler

È il linguaggio di programmazione più vicino al metodo di funzionamento interno di un computer, pertanto è definito come linguaggio di programmazione di "basso livello". Il principale vantaggio dell'assembler è la velocità, in quanto i programmi in assembler sono scritti sotto forma di istruzioni del processore, di conseguenza sono poche o comunque semplici le traduzioni richieste al momento di generare i file eseguibili. Il suo grande svantaggio è dato dal fatto che è un linguaggio strettamente dipendente dal tipo di processore (o di architettura). La scrittura di programmi complessi in assembler, inoltre, è un processo lungo e difficile. In conclusione, si tratta del linguaggio di programmazione che garantisce la maggior velocità per l'esecuzione dei programmi, ma non certo per la loro scrittura, e non è portabile fra architetture hardware diverse.

ATAPI

AT Attachment Packet Interface. Un'estensione delle specifiche ATA (*Advanced Technology Attachment*, più comunemente note come IDE, *Integrated Drive Electronics*) che mette a disposizione comandi supplementari per controllare lettori CD-ROM e unità a nastro magnetico. I controller IDE che sono equipaggiati con tale estensione sono noti anche come controller EIDE (*Enhanced IDE*).

ATM

Questo acronimo significa *Asynchronous Transfer Mode*. Una rete ATM suddivide i dati in pacchetti di dimensioni standard (53 byte: 48 per i dati e 5 per l'intestazione) che possono essere trasmessi in modo efficiente da punto a punto. L'ATM è una tecnologia di rete a pacchetti progettata per reti ottiche ad alta velocità (multi-megabit).

atomico

Un insieme di operazioni si dice atomico quando queste vengono eseguite una di seguito all'altra, e non possono essere sospese o interrotte.

attraversamento

Per una directory di un sistema `Unix`, significa che l'utente ha il permesso di accedere a tale directory e forse anche alle directory sotto di essa. Perché questo sia possibile, l'utente deve avere anche il permesso di esecuzione sulla directory.

background
Nell'ambito di una *shell*, un processo viene eseguito in background ("in secondo piano") se potete digitare comandi mentre il processo in questione è in esecuzione.
Si veda anche: job, foreground.

backup
Il salvataggio di dati importanti su di un supporto sicuro, archiviato in un luogo sicuro. I backup dovrebbero essere effettuati con regolarità, soprattutto nel caso di informazioni e file di configurazione d'importanza critica (le prime directory di cui bisogna fare un backup sono `/etc`, `/home` e `/usr/local`). Molte persone usano strumenti tradizionali come il programma `tar`, usato con `gzip` o `bzip2`, per effettuare il backup di directory e file. Potete usare questi strumenti, o altri come `dump` e `restore`, come pure molte applicazioni di backup disponibili per Linux, liberamente distribuibili o commerciali.

batch
È un metodo di elaborazione per cui il processore riceve una serie di compiti, e li esegue uno dopo l'altro finché non ha portato a termine anche l'ultimo; da quel momento è pronto per un'altra serie di processi.

beep
Il piccolo rumore emesso dall'altoparlante interno del computer per avvertirvi di qualche ambiguità quando usate il completamento automatico della linea di comando (nel caso, ad esempio, che ci sia più di una soluzione possibile per il completamento automatico). Altri programmi potrebbero emettere un beep per informarvi di qualche particolare situazione.

beta testing
È il nome dato al processo di prova della versione preliminare (detta versione "beta") di un programma. In genere i programmi vengono rilasciati in versioni alfa e beta proprio per essere sottoposti a varie prove prima di arrivare alla versione finale.

bit
Sta per *BInary digiT*. Una singola cifra che può avere valore 0 o 1, in quanto il calcolo viene effettuato in base due.

boot
La procedura d'avvio che si verifica subito dopo l'accensione di un computer, quando le periferiche vengono riconosciute una dopo l'altra e il sistema operativo viene caricato in memoria.

bootloader
È un programma che provvede ad avviare un sistema operativo. Molti bootloader offrono la possibilità di caricare più di un sistema operativo grazie a un menu di avvio che vi permette di scegliere il sistema desiderato. Bootloader come *grub* sono molto diffusi proprio grazie a questa caratteristica, e sono molto utili nel caso di computer in cui risiedano due o più sistemi operativi.

BSD
Berkeley Software Distribution. Una variante di *Unix* sviluppata presso il dipartimento di informatica dell'Università di Berkeley. Questa versione è sempre stata considerata più avanzata sul piano tecnico rispetto alle altre, e ha introdotto molte innovazioni nel mondo informatico in generale e, in particolare, per quanto riguarda *Unix*.

buffer
Una piccola parte della memoria di dimensioni fisse, che può essere associata a un file in modalità a blocchi, a una tabella di sistema, a un processo, e così via. La consistenza di tutti i buffer viene mantenuta per mezzo della buffer cache.
Si veda anche: buffer cache.

buffer cache
Una parte essenziale del kernel di un sistema operativo, ha il compito di tenere aggiornati tutti i buffer, ridimensionando la cache quando necessario, svuotando i buffer non più necessari, e altro ancora.
Si veda anche: buffer.

bug
Comportamento illogico o incoerente di un programma in una situazione particolare, oppure un comportamento che non rientra nei casi previsti dalla documentazione del programma o dagli standard cui quest'ultimo dovrebbe essere conforme. Nuove funzionalità spesso introducono nuovi bug in un programma. Il termine, dal punto di vista storico, deriva dai tempi (antichi!) delle schede perforate: un insetto (ingl. *bug*) era scivolato in un buco di una scheda perforata e, di conseguenza, il programma non

si era comportato correttamente. Dopo aver scoperto la causa, Ada Lovelace esclamò "It's a bug!", e da allora la definizione si è diffusa ed è diventata universale.

byte
Otto bit consecutivi, interpretati come un numero tra 0 e 255 in base due.
Si veda anche: bit.

canali IRC
Sono i "luoghi" all'interno dei server IRC dove potete chiacchierare con altre persone. I canali sono creati nei server IRC, e gli utenti entrano in questi canali per comunicare con altre persone. I messaggi scritti su un canale sono visibili soltanto alle persone connesse a quel canale. Due o più utenti possono creare un canale "private" per non essere disturbati da altri utenti. I nomi dei canali cominciano con un `#`.

carattere nullo
È il carattere (o byte) corrispondente al numero 0, e viene utilizzato per indicare la fine di una stringa.

case
Senza equivalente in italiano, in relazione alle stringhe di caratteri il *case* è la differenza fra lettere minuscole e lettere maiuscole.

CHAP
Challenge-Handshake Authentication Protocol. Protocollo usato dagli ISP nella fase di autenticazione dei loro client. Secondo tale schema viene inviato un valore al client (la macchina che si collega), quest'ultimo calcola un *hash* sulla base di tale valore e lo invia al server, infine il server confronta l' *hash* con quello che lui stesso ha calcolato. Differisce dal PAP anche per il fatto che la connessione viene ri-autenticata periodicamente dopo l'autenticazione iniziale.
Si veda anche: PAP.

CIFS
Common Internet FileSystem. Il predecessore del filesystem SMB, usato sui sistemi *DOS*.

client
Programma o computer che in maniera intermittente e temporanea si connette a un altro programma o computer per comunicargli comandi o chiedere informazioni. È uno dei componenti di un **sistema client/server**.

codice oggetto
È il codice generato dal processo di compilazione, che deve essere collegato ad altri codici oggetto e librerie per formare un file eseguibile. Il codice oggetto è comprensibile dalla macchina.
Si veda anche: compilazione, linkage.

compilazione
Si tratta del processo di traduzione del codice sorgente, comprensibile per un essere umano (beh, con un po' di allenamento) e scritto in un qualsiasi linguaggio di programmazione (il *C*, ad esempio), in un file binario comprensibile dall'elaboratore.

completamento automatico
La capacità di una *shell* di espandere automaticamente una sotto-stringa in un nome di file, nel nome di un utente o in altro ancora, a condizione che vi sia una corrispondenza.

compressione
È il metodo usato per ridurre le dimensioni dei file o diminuire il numero di caratteri che devono essere inviati su una connessione di rete. Tra i programmi di compressione dei file citiamo *compress* , `zip` , `gzip` , e `bzip2` .

configurabile via temi
Un'applicazione grafica è configurabile via temi se è in grado di cambiare il suo aspetto in tempo reale. Anche molti window manager sono configurabili via temi.

console
È il nome dato a quello che una volta era chiamato terminale. Si trattava di una macchina utente (uno schermo più una tastiera) connessa a un computer centrale molto potente (mainframe). Sui *PC* un terminale fisico è costituito da schermo e tastiera.
Si veda anche: console virtuali.

console virtuali
Conservano il nome dato a quelli che una volta erano chiamati terminali. Nei sistemi *GNU/Linux* avete a vostra disposizione delle "console virtuali", così chiamate perché vi permettono di usare un solo schermo e una sola tastiera per molte sessioni indipendenti. Come opzione predefinita, sono disponibili in tutto sei console virtuali, cui si può accedere premendo le combinazioni di tasti da **ALT-F1** a **ALT-F6**. Esiste una settima console, anche questa caratteristica predefinita del sistema, **ALT-F7**, che vi consente di accedere a una sessione X in esecuzione. Se vi trovate già in X, invece, potete accedere alle console di testo premendo le combinazioni di tasti che vanno da **CTRL-ALT-F1** a **CTRL-ALT-F6**.
Si veda anche: console.

cookie
File temporanei scritti sul disco rigido locale da un web server remoto. Permettono al server di ricordare le preferenze dell'utente quando quest'ultimo si connette nuovamente.

datagramma
Un datagramma è un pacchetto di dati e intestazioni discreto, contenente indirizzi, che costituisce l'unità di trasmissione di base attraverso una rete IP. Un possibile sinonimo è "pacchetto".

denominazione
Una parola usata comunemente nel campo dell'informatica per indicare un metodo di identificazione degli oggetti. Avrete sentito spesso parlare di "convenzioni di denominazione" per file, funzioni nei programmi, e così via.

desktop
Se state usando il sistema X Window, il desktop (o "scrivania") è la rappresentazione sullo schermo dell'ambiente grafico in cui lavorate, all'interno del quale sono visualizzate le finestre e le icone. È chiamato anche background (o "sfondo"), e in genere è visualizzato come un colore pieno, una sfumatura di due colori o anche un'immagine.
Si veda anche: desktop virtuali.

desktop virtuali
Nel sistema X Window il window manager può mettere a vostra disposizione più di un desktop. Questa utile caratteristica vi permette di organizzare le vostre finestre, evitando il problema del sovraffollamento della scrivania. A tutti gli effetti, è come avere più schermi diversi. Le modalità di spostamento da un desktop virtuale a un altro dipendono dal window manager che state usando.
Si veda anche: desktop, window manager.

DHCP
Dynamic Host Configuration Protocol. Un protocollo progettato per far sì che le macchine presenti in una rete locale ottengano dinamicamente un indirizzo IP da un server DHCP.

dipendenze
Nella compilazione di un programma, sono le fasi che è necessario portare a termine per poter proseguire con l'operazione.
Nella gestione dei pacchetti, invece, sono i pacchetti necessari per il corretto funzionamento di un altro pacchetto.

directory
Parte della struttura del filesystem. All'interno di una directory possono trovarsi file o altre directory. A volte, all'interno di una directory è presente una serie di sotto-directory: spesso ci si riferisce a questa struttura gerarchica con il termine "albero delle directory", all'interno del quale le sotto-directory costituiscono i rami e i file le foglie. Per vedere il contenuto di un'altra directory, dovrete spostarvi in essa o elencarlo da dove vi trovate. Le directory sono soggette alle stesse limitazioni dei file, per quanto i permessi abbiano un significato differente. Le directory speciali '.' e '..' si riferiscono, rispettivamente, alla directory stessa e a quella immediatamente superiore.

directory home
Spesso abbreviata con "home", è il nome della directory personale di un dato utente.
Si veda anche: account.

directory radice
Si riferisce alla directory di livello più alto in un filesystem. Questa non ha una directory superiore, pertanto il simbolo '..' fa riferimento a se stessa. La directory root viene indicata con il simbolo '/'.

disco di avvio
Un floppy disk in grado di avviare il computer, contenente i programmi necessari per caricare il sistema operativo dal disco rigido (ma talvolta all'interno di un disco di avvio si trova tutto quanto serve per avviare un sistema operativo).

distribuzione
È il termine usato per indicare il sistema *GNU/Linux* progettato e assemblato da uno specifico venditore. Una distribuzione è costituita dal kernel e dai programmi essenziali di Linux, ma anche da programmi di installazione, programmi di terze parti e, a volte, anche del software proprietario.

DLCI
Il DLCI è il *Data Link Connection Identifier*, serve a identificare una connessione virtuale da punto a punto per mezzo di una rete Frame Relay. I DLCI normalmente vengono assegnati dal gestore della rete Frame Relay.

DMA
Direct Memory Access. Una caratteristica dell'architettura *PC* che permette a una periferica di leggere o scrivere dalla memoria centrale senza alcun aiuto da parte del processore. Le periferiche PCI usano la tecnica del bus mastering e non hanno bisogno del DMA.

DNS
Domain Name System. Sistema dei nomi di dominio: è il meccanismo distribuito nome/indirizzo utilizzato su Internet. Questo meccanismo permette di far corrispondere un nome di dominio a un indirizzo IP, ed è grazie ad esso che potete cercare e visitare un sito senza che sia necessario conoscerne l'indirizzo IP. Il DNS permette anche l'operazione contraria, cioè ricavare il nome di una macchina dal suo indirizzo IP.

DPMS
Display Power Management System. Protocollo usato da tutti i monitor moderni al fine di gestire le funzionalità di risparmio energetico. I monitor che supportano questa caratteristica sono comunemente noti come "monitor verdi".

echo
È il fenomeno per cui i caratteri digitati nel campo di testo del nome utente, ad esempio, vengono mostrati "tali e quali", invece di visualizzare "*" per ciascuno di loro.

editor
È il termine più comune per indicare i programmi utilizzati per modificare file di testo (da qui anche l'espressione "editor di testi"). Gli editor più famosi su *GNU/Linux* sono l'editor GNU *Emacs* e l'editor Unix *Vi*.

ELF
Executable and Linking Format. È il formato per i file eseguibili usato oggi da quasi tutte le distribuzioni *GNU/Linux*.

email
Sta per *Electronic Mail*, ovvero "posta elettronica". Si tratta dello scambio di messaggi in formato elettronico fra persone che si trovano sulla stessa rete. In modo simile alla posta tradizionale (detta anche *snail mail*, "posta lumaca"), per funzionare la posta elettronica necessita di un destinatario e di un mittente. Il mittente deve avere un indirizzo come "mittente@dominio.del.mittente" e il destinatario deve avere un indirizzo simile ("destinatario@dominio.del.destinatario"). La posta elettronica è un metodo di comunicazione estremamente rapido, in genere occorrono solo pochi minuti per raggiungere chiunque, non importa dove si trovi nel mondo. Per scrivere messaggi di posta elettronica è necessario usare un client email come *pine* o `mutt`, programmi con interfaccia testuale, oppure client dotati di una GUI, come *kmail*.

escape
Nel contesto della shell, è l'azione di circondare qualche stringa di testo fra doppi apici in maniera da impedire alla shell di interpretare tale stringa. Ad esempio, quando avete la necessità di usare degli spazi in qualche linea di comando e inviare con una pipe i risultati a qualche altro comando, dovete racchiudere il primo comando fra doppi apici, altrimenti la shell non lo interpreterà in maniera corretta e non funzionerà come desiderato.

espressione regolare
Un potente strumento utilizzato per cercare e confrontare stringhe di testo. Permette di specificare modelli ai quali devono conformarsi le stringhe. Molti programmi di utilità sotto *Unix* fanno uso di espressioni regolari: `sed`, *awk*, `grep`, *perl* e altri ancora.

ext2
Abbreviazione per "Extended 2 filesystem" . È il filesystem nativo di *GNU/Linux* , e possiede tutte le caratteristiche di un qualsiasi filesystem *Unix* : supporto per file speciali (dispositivi a caratteri, link simbolici, etc.), permessi e proprietà relativi a file e directory, e così via.

FAQ
Frequently Asked Questions. Documento che contiene una serie di domande e risposte riguardo un argomento specifico. Storicamente le FAQ sono comparse nei gruppi di discussione su Usenet, ma questo tipo di documento adesso è comune su molti siti web , e persino dei prodotti commerciali hanno le loro FAQ . In genere costituiscono ottime fonti d'informazione.

FAT
File Allocation Table. Il filesystem usato da *DOS* e *Windows*.

FDDI
Fiber Distributed Digital Interface. Uno strato fisico di rete ad alta velocità, che usa le fibre ottiche per la comunicazione. Usato unicamente su reti molto estese, soprattutto a causa del suo costo.

FHS
Filesystem Hierarchy Standard. Un documento che contiene le linee guida per una coerente organizzazione della struttura gerarchica del filesystem su sistemi *Unix*. La distribuzione **Mandrake Linux** è quasi del tutto conforme a questo standard.

FIFO
First In, First Out. Una struttura dati o un buffer hardware dai quali gli oggetti vengono estratti nello stesso ordine in cui erano stati inseriti. Le pipe di *Unix* costituiscono uno dei più complessi esempi di FIFO.

file in modalità a blocchi
File il cui contenuto è bufferizzato. Tutte le operazioni di lettura/scrittura che riguardano tali file passano attraverso dei buffer, fatto che permette scritture in modo asincrono sull'hardware sottostante e, per quanto riguarda le operazioni di lettura, di non leggere di nuovo ciò che è già presente in un buffer.
Si veda anche: buffer, buffer cache, file in modalità a caratteri.

file in modalità a caratteri
File il cui contenuto non è bufferizzato. Tutte le operazioni di input/output vengono eseguite immediatamente in maniera diretta. Corrispondono ai flussi di dati.
Si veda anche: file in modalità a blocchi.

file nascosto
È un file che non può esser "visto" quando si digita il comando `ls` senza opzioni. I nomi dei file nascosti iniziano con un . (per antica convenzione *Unix*). Tali file vengono usati per archiviare le preferenze personali e le configurazioni dei diversi programmi usati da un utente. Ad esempio, l'elenco degli ultimi comandi eseguiti da *bash* viene salvato nel file `.bash_history`, che è un file nascosto.

filesystem
Il metodo usato per registrare i file su supporti fisici (dischi rigidi, floppy, etc.) in maniera coerente. Esempi di filesystem sono la FAT, l'ext2fs di *GNU/Linux*,l'iso9660 (usato sui CD-ROM), e così via.

filesystem radice
È il filesystem di livello più alto. Questo è il filesystem sul quale *GNU/Linux* monta l'intero albero delle directory. È necessario che il filesystem radice si trovi su una partizione dedicata, in quanto è la base dell'intero sistema. Contiene la directory radice.

finestra
Nell'ambito delle reti, la **finestra** è la massima quantità di dati che la parte ricevente può accettare in un dato momento.

firewall
Una macchina o comunque un dispositivo hardware dedicato che, nella topologia di una rete locale, è l'unico punto di connessione alla rete esterna, e filtra tale connessione, o controlla l'attività di alcune porte, o si accerta che soltanto alcune specifiche interfacce IP possano accedere a queste ultime.

flag
È un indicatore (in genere un solo bit) usato per segnalare (*flag* significa "bandiera") una particolare condizione a un programma. Un filesystem, ad esempio, possiede, fra gli altri, un flag che indica se è

giunto il momento di effettuare un backup con `dump`, così quando il flag è attivo viene effettuata una copia di sicurezza del filesystem, mentre se è inattivo questo non succede.

focus
La possibilità, per una finestra, di ricevere segnali dalla tastiera (come la pressione e il rilascio dei tasti) e clic del mouse, a meno che tali eventi siano intercettati dal window manager.

foreground
Nell'ambito della *shell*, il processo in foreground ("primo piano") è quello attualmente in corso di esecuzione. Siete obbligati ad aspettare che tale processo sia terminato prima di poter digitare di nuovo dei comandi.
Si veda anche: job, background.

Frame Relay
Il *Frame Relay* è una tecnologia di rete idealmente adatta a sostenere un traffico di natura sporadica o caratterizzato da picchi improvvisi di attività. I costi del collegamento in rete sono ridotti quando molti clienti utilizzano il Frame Relay per condividere la stessa ampiezza di banda, facendo affidamento sul fatto che faranno uso della rete in tempi leggermente diversi.

framebuffer
La proiezione della RAM presente su una scheda grafica nella memoria centrale. Questo permette alle applicazioni di accedere alla memoria video senza dover prima dialogare con la scheda grafica. Tutte le workstation grafiche di fascia alta, ad esempio, usano dei framebuffer.

FTP
File Transfer Protocol. È il protocollo standard usato su Internet per trasferire file da una macchina all'altra.

gateway
Anello di collegamento fra due reti IP.

GIF
Graphics Interchange Format . Un formato di file per immagini ampiamente usato sul web . Le immagini GIF possono essere anche compresse e animate. A causa di problemi relativi al copyright, non è una buona idea usare questo formato, ed è preferibile utilizzare, quando possibile, il più avanzato formato PNG.

GNU
GNU's Not Unix. Il progetto GNU è nato per iniziativa di Richard Stallman all'inizio degli anni '80, il suo scopo è lo sviluppo di un sistema operativo libero ("libero" nel senso di "libertà di parola"). Al momento attuale, tutte le varie componenti sono disponibili, con l'eccezione... del kernel. Il kernel del progetto GNU, *Hurd*, non è ancora affidabile al 100%. *Linux* utilizza due componenti, in particolare, del progetto GNU: il suo compilatore *C*, `gcc`, e la licenza GPL.
Si veda anche: GPL.

GPL
General Public License . La licenza del kernel *GNU/Linux* , è l'esatto contrario di tutte le licenze di software proprietario in quanto non impone nessuna limitazione per quanto riguarda la copia, la modifica e la distribuzione del software, a condizione che il codice sorgente sia sempre disponibile. L'unica limitazione, se possiamo chiamarla così, consiste nel fatto che le persone alle quali distribuite il software devono poter beneficiare degli stessi diritti.

groupware
Strumenti di produttività di gruppo, un insieme di prodotti software che consente a un gruppo di persone di lavorare bene insieme.

gruppo proprietario
Nel contesto degli utenti e dei loro file, il gruppo proprietario di un file è il gruppo cui appartiene l'utente che lo ha creato.

GUI
Graphical User Interface . Un'interfaccia tra utente e computer, costituita da menu, pulsanti, icone e così via. La quasi totalità degli utenti preferisce una GUI alla CLI (*Command Line Interface* , "Interfaccia a Linea di Comando") per la maggiore facilità d'uso, malgrado quest'ultima sia più flessibile.

host
Si riferisce a un computer, è il termine usato normalmente per indicare computer connessi a una rete.

HTML
HyperText Markup Language . Il linguaggio usato per creare documenti per il web .

HTTP
HyperText Transfer Protocol . Il protocollo usato per la connessione ai siti web e per il trasferimento di documenti HTML .

icona
Termine che indica una piccola immagine o disegno (normalmente di 16 x 16, 32 x 32, 48 x 48, e a volte 64 x 64 pixel) che rappresenta, in un ambiente grafico, un documento, un file o un programma.

IDE
Integrated Drive Electronics. Il bus di gran lunga più diffuso sui *PC* attuali per la connessione di dischi rigidi. Un bus IDE può ospitare fino a due dispositivi, e la velocità del bus è limitata dal dispositivo che ha la coda di comando più lenta (e non la velocità di trasferimento più lenta!).
Si veda anche: ATAPI.

indirizzo hardware
Si tratta di un numero che identifica in maniera univoca un host in una rete fisica al livello di accesso ai supporti hardware. Possiamo parlare, ad esempio, di **indirizzi Ethernet** e **indirizzi AX.25**.

indirizzo IP
È un indirizzo numerico, composto da quattro elementi, che identifica il vostro computer su Internet. Gli indirizzi IP sono strutturati secondo un metodo gerarchico, con livelli superiori e domini nazionali, domini, sotto-domini e l'indirizzo individuale di ciascuna macchina. Un indirizzo IP ha l'aspetto di qualcosa tipo 192.168.0.1. L'indirizzo individuale di una macchina può appartenere a due tipi diversi: statico o dinamico. Gli indirizzi IP di tipo statico sono indirizzi permanenti, che non cambiano mai. Gli indirizzi IP di tipo dinamico cambiano con ogni nuova connessione alla rete. Gli utenti di connessioni via modem e via cavo in genere hanno indirizzi IP di tipo dinamico, mentre gli utenti di connessioni DSL e altre connessioni ad alta velocità in genere dispongono di indirizzi IP statici.

inode
Elemento che punta al contenuto di un file su un filesystem della famiglia *Unix*. Un inode è identificato in maniera univoca da un numero, e contiene meta-informazioni riguardo il file cui fa riferimento, quali le sue date di accesso, il suo tipo, le sue dimensioni, ma **non il suo nome!**

Internet
La rete di dimensioni smisurate che connette computer di tutto il mondo.

IP masquerading
Si riferisce all'uso di un firewall per nascondere il vero indirizzo IP del vostro computer. Grazie al firewall, ogni connessione alla rete esterna erediterà l'indirizzo IP di quest'ultimo. Questo è utile nelle situazioni in cui è disponibile una connessione a Internet veloce con un unico indirizzo IP, ma desiderate usare più di un computer con indirizzi IP interni già assegnati per connettervi a Internet.

IRC
Internet Relay Chat . Uno dei pochi standard di Internet per conversazioni in tempo reale. Permette la creazione di canali, discussioni private, e anche lo scambio di file. Inoltre è progettato in maniera tale da permettere la connessione dei server l'uno con l'altro, il che spiega perché oggi esistono numerose reti IRC : **Undernet** , **DALnet** , **EFnet** tanto per nominarne alcune.

ISA
Industry Standard Architecture. Il primissimo bus usato sui personal computer, che viene pian piano abbandonato a favore del bus PCI. Alcuni produttori di hardware, tuttavia, continuano a usarlo: accade ancora spesso, purtroppo, di trovare delle schede SCSI di tipo ISA fornite a corredo di periferiche come scanner, masterizzatori, etc.

ISDN
Integrated Services Digital Network. Un insieme di standard di comunicazione aventi come scopo la trasmissione di voce, servizi di rete e video per mezzo di un singolo cavo o fibra ottica. È stato progettato per rimpiazzare, nel lungo periodo, gli attuali sistemi di comunicazione telefonica.

ISO
International Standards Organisation. Un gruppo di società, consulenti, università e altre componenti che ha come scopo l'elaborazione di standard applicabili in vari ambiti, incluso quello dell'informatica. I

documenti che descrivono tali standard sono caratterizzati da un numero: lo standard numero 9660, ad esempio, descrive il filesystem utilizzato sui CD-ROM.

ISO 8859
Lo standard ISO 8859 comprende un certo numero di estensioni a 8-bit al set di caratteri ASCII. Riveste particolare importanza l'ISO 8859-1, l' "Alfabeto Latino N. 1", che ha conosciuto un gran numero di implementazioni ed è spesso visto come lo standard che rimpiazzerà l'ASCII.
L'ISO 8859-1 supporta le seguenti lingue: Afrikaans, Basco, Catalano, Danese, Olandese, Inglese, Faroese, Finnico, Francese, Galiziano, Tedesco, Islandese, Irlandese, Italiano, Norvegese, Portoghese, Scozzese, Spagnolo e Svedese.
Notate che i caratteri dell'ISO 8859-1 costituiscono anche i primi 256 caratteri dell'ISO 10646 (Unicode). Questo standard, tuttavia, non ha il simbolo dell'EURO e non copre in maniera completa Finnico e Francese. L'ISO 8859-15 è una modifica dell'ISO 8859-1 che soddisfa tali necessità.
Si veda anche: ASCII.

ISP
Internet Service Provider. Una società che vende l'accesso a Internet ai suoi clienti, accesso che può avvenire per mezzo delle normali linee telefoniche o su linee dedicate.

job
Nell'ambito di una `shell`, un job è un processo che viene eseguito in background. Potete avere più di un job nella stessa shell, e controllarne l'esecuzione.
Si veda anche: foreground, background.

jolly
I caratteri '*' e '?' vengono usati come caratteri jolly e possono rappresentare qualsiasi cosa. L'asterisco ('*') rappresenta qualsiasi numero di caratteri, incluso nessun carattere. Il punto interrogativo ('?') rappresenta esattamente un unico carattere. I caratteri jolly sono usati spesso nelle espressioni regolari.

JPEG
Joint Photographic Experts Group. Un altro formato di file per immagini molto comune. Il formato JPEG è adatto soprattutto alla compressione di immagini fotografiche, e non funziona molto bene con i disegni.

kernel
È la parte più interna e importante del sistema operativo. Il kernel è responsabile dell'assegnazione delle risorse e della separazione degli ambienti relativi ai vari processi. Gestisce tutte le operazioni a basso livello che permettono ai programmi di interagire direttamente con l'hardware del vostro computer, gestisce la cache del buffer, e così via.

kill ring
Sotto `Emacs`, è l'insieme di aree di testo tagliate o copiate dal momento in cui si è utilizzato l'editor (buffer di copia/cancellazione); queste possono essere richiamate per essere inserite nuovamente, e sono organizzate sotto forma di anello.

LAN
Local Area Network. Nome generico che indica una rete di macchine connesse tra loro a breve distanza, ad esempio all'interno di uno stesso edificio.

lanciare
L'azione di richiamare, o avviare, un programma.

LDP
Linux Documentation Project. Un'organizzazione senza scopo di lucro che cura della documentazione relativa a `GNU/Linux`. I suoi documenti più conosciuti sono gli `HOWTO`, ma si occupa anche dell'organizzazione e dell'aggiornamento di numerose FAQ, e persino di un certo numero di libri.

libreria
Si riferisce a una collezione di procedure e funzioni in forma binaria che i programmatori possono usare nei loro programmi (purché la licenza della libreria glielo consenta). Il programma cui spetta il compito di caricare le librerie condivise al momento dell'esecuzione è chiamato *dynamic linker*.

linea di comando
Quella fornita da una shell, che permette all'utente di digitare comandi direttamente. È anche l'oggetto di un'eterna guerra tra i suoi sostenitori e i suoi detrattori **:-)**

link
Il riferimento a un inode in una directory, che pertanto assegna un nome (di file) all'inode. Fra gli inodi che non hanno un link, e quindi nemmeno un nome, possiamo citare, ad esempio: pipe anonime (quelle usate dalla shell), socket (ovvero connessioni di rete), dispositivi di rete, e così via.

link simbolico
File speciali, che non contengono nulla a parte una stringa che fa riferimento a un altro file. Ogni accesso a essi è equivalente a un accesso al file il cui nome è rappresentato da detta stringa; quest'ultimo può esistere o no, e il suo percorso può essere dato in termini relativi o assoluti.

linkage
L'ultima fase di un processo di compilazione, consiste nel collegare insieme tutti i file oggetto per produrre un file eseguibile; in questa stessa fase, tutti i simboli non risolti vengono identificati con librerie dinamiche (a meno che non sia stato richiesto un linkage di tipo statico, nel qual caso il codice di questi simboli verrà incluso nel file eseguibile).

Linux
Si riferisce a un sistema operativo simile a *Unix* che può girare su computer di vario tipo, il cui uso e adattamento è libero e gratuito per chiunque. Linux (il kernel) è stato scritto da Linus Torvalds.

livelli di sicurezza
Una caratteristica esclusiva di **Mandrake Linux** che vi permette di impostare diversi livelli di restrizioni a seconda di quanto vogliate rendere sicuro il vostro sistema. Esistono 6 livelli predefiniti, che vanno da 0 a 5, dove 5 indica il livello più alto di sicurezza. Potete anche definire un vostro personale livello di sicurezza.

login
Il nome di accesso di un utente su un sistema *Unix*, e l'atto del connettersi al sistema.

lookup table
È una tabella che stabilisce una corrispondenza fra codici (o etichette) e il loro reale significato. Spesso si tratta di un file di dati usato da un programma per ottenere ulteriori informazioni riguardo un oggetto particolare.
harddrake, ad esempio, usa una simile tabella per sapere cosa significa il codice assegnato da un produttore a un dispositivo hardware. Questa è una riga appartenente a tale tabella, che ci fornisce informazioni in merito all'oggetto `CTL0001`

```
CTL0001 sound sb Creative Labs SB16 \ HAS_OPL3|HAS_MPU401|HAS_DMA16|HAS_JOYSTICK
```

loopback
Interfaccia di rete virtuale di una macchina con se stessa, che permette ai programmi in esecuzione di fare a meno di prendere in considerazione il caso speciale per cui due entità di rete sono, di fatto, la stessa macchina.

major
Numero specifico alla classe del dispositivo (numero primario).

MBR
Master Boot Record. Nome dato al primo settore di un disco rigido in grado di effettuare il boot. L'MBR contiene il codice utilizzato per caricare il sistema operativo in memoria o un bootloader (come *LILO*), e la tabella delle partizioni del disco rigido cui appartiene il settore che lo ospita.

menu a discesa
È un menu che si presenta come "arrotolato", con un bottone in uno dei suoi angoli: quando cliccate su quel bottone, il menu "si srotola" mostrandovi l'intero contenuto.

meta-espansione
Nell'ambito della *shell*, la possibilità di raggruppare un certo numero di nomi di file grazie a un modello di meta-espansione.
Si veda anche: modello di meta-espansione.

MIME
Multipurpose Internet Mail Extensions . Una stringa con il formato `tipo/sotto-tipo` che descrive il contenuto di un file allegato a un messaggio di posta elettronica. Questo permette ai client di posta elettronica che supportano lo standard MIME di definire azioni in base al tipo del file.

minor
Numero che identifica un particolare dispositivo fisico all'interno di una classe (numero secondario).

modalità comandi
In `Vi` (o uno dei suoi cloni), è lo stato del programma in cui la pressione di un tasto (questo riguarda soprattutto le lettere) non avrà come conseguenza il suo inserimento nel file che viene modificato, bensì l'esecuzione di un'azione specifica per quel tasto (a meno che il programma non offra la possibilità di modificare i comandi e voi abbiate personalizzato la vostra configurazione). Se ne può uscire digitando uno dei comandi "torna alla modalità inserimento" : **i** , **I** , **a** , **A** , **s** , **S** , **o** , **O** , **c** , **C** , ...

modalità inserimento
In `Vi` (o uno dei suoi cloni), è lo stato del programma in cui premendo un tasto sarà inserito il carattere corrispondente nel file che viene modificato (con l'eccezione di alcuni casi particolari come il completamento automatico di un'abbreviazione, la giustificazione a destra alla fine della riga, etc.). È possibile uscirne premendo il tasto **Esc** (o **Ctrl-[**).

modalità lettura-scrittura
Per un file, significa che può essere scritto. Potete leggere il suo contenuto e modificarlo.
Si veda anche: modalità sola lettura.

modalità prolissa
Per quanto riguarda i comandi, la modalità prolissa significa che il comando comunica allo standard output (o allo standard error) tutte le azioni che compie e il risultato di queste azioni. A volte i comandi possono definire un "livello di prolissità", in altre parole può essere controllata la quantità di informazioni che il comando trasmette al momento del suo utilizzo.

modalità sola lettura
Per un file, significa che non può essere scritto. Potete leggere il suo contenuto, ma non potete modificarlo.
Si veda anche: modalità lettura-scrittura.

modello di meta-espansione
Una stringa composta di caratteri normali e caratteri speciali. I caratteri speciali (meta-caratteri) sono interpretati ed espansi dalla `shell`.

monoutente
Termine usato per descrivere la condizione di un sistema operativo, o persino il sistema operativo stesso, che permette l'accesso al sistema e il suo uso soltanto a un utente per volta.

montare, montato
Un dispositivo è detto montato quando è connesso e riconosciuto dal filesystem `GNU/Linux` . Quando montate un dispositivo potete esplorarne i contenuti. Questo termine è almeno in parte obsoleto, in quanto grazie al programma "supermount" incluso in **Mandrake Linux** , gli utenti non sono più costretti a montare manualmente i supporti rimovibili.
Si veda anche: punto di mount.

MPEG
Moving Pictures Experts Group. Un comitato dell' ISO che definisce gli standard per la compressione di audio e video. MPEG è anche il nome dei loro algoritmi.

MSS
The Maximum Segment Size (**MSS**) is the largest quantity of data that can be transmitted at one time. If you want to prevent local fragmentation MSS would equal MTU-IP header.

MTU
La **MTU** (*Maximum Transmission Unit* , "unità massima di trasmissione") è il parametro che determina le dimensioni massime dei datagrammi che possono essere trasmessi da un'interfaccia IP senza che si renda necessario spezzarli in unità più piccole. La MTU dovrebbe essere più grande del più grande datagramma che intendete trasmettere intero. Notate che questo impedisce la frammentazione solo a livello locale, qualche altro collegamento lungo il percorso potrebbe avere una MTU più piccola, nel qual caso la frammentazione si verificherebbe in prossimità di quest'ultimo. Valori tipici sono 1500 byte per un'interfaccia ethernet, e 576 per un'interfaccia SLIP.

multitasking
La capacità di un sistema operativo di condividere il tempo di elaborazione della CPU fra più processi contemporaneamente in esecuzione. A basso livello, questa caratteristica è conosciuta anche come multiprogrammazione. Il passaggio da un processo a un altro richiede che tutto il contesto attuale del processo

venga salvato e ricaricato quando è di nuovo il turno di quel processo. Questa operazione è nota come *context switch* ("passaggio da un contesto all'altro") e, su macchine Intel, si verifica 100 volte al secondo: dunque è sufficientemente veloce da dare a un utente l'impressione che il sistema operativo stia eseguendo più applicazioni nello stesso momento. Esistono due tipi di multitasking: il *preemptive multitasking* ("multitasking con prelazione"), in cui il sistema operativo ha la responsabilità di sottrarre la CPU a un processo per passarla a un altro; e il *cooperative multitasking* ("multitasking cooperativo"), in cui è il processo stesso che decide quand'è il momento di restituire la CPU. La prima variante, ovviamente, è la scelta migliore, in quanto nessun programma può assorbire l'intero tempo di elaborazione della CPU e bloccare così altri processi; *GNU/Linux* è caratterizzato da un multitasking di questo tipo. Il metodo in base al quale viene scelto il processo da eseguire dipende da diversi parametri, ed è chiamato *scheduling* ("pianificazione").

multiutente
È usato per descrivere un sistema operativo che consente l'accesso e l'uso del sistema a più utenti nello stesso momento, ciascuno dei quali è messo in grado di svolgere la propria attività in maniera indipendente dagli altri utenti. Per offrire un supporto multiutente è indispensabile un sistema operativo caratterizzato dal multitasking. *GNU/Linux* è un sistema operativo multitasking e multiutente, come qualsiasi altro sistema *Unix*.

NCP
NetWare Core Protocol: protocollo definito dalla **Novell** per accedere a file e servizi di stampa tramite Novell NetWare.

newsgroup, gruppi di discussione
(Ingl. *newsgroups*) si riferisce ad aree dedicate alla discussione, a informazioni e a notizie varie cui si può accedere con un programma client appropriato per leggere e scrivere messaggi in relazione all'argomento del gruppo. Il gruppo di discussione `alt.os.linux.mandrake`, ad esempio, è un gruppo alternativo (alt) che tratta del sistema operativo (os) *GNU/Linux*, in particolare di **Mandrake Linux** (mandrake). I nomi dei gruppi di discussione sono strutturati in questo modo per rendere più facile la ricerca di un argomento specifico.

NFS
Network FileSystem. Un filesystem di rete creato dalla **Sun Microsystems**, avente come obiettivo la condivisione di file all'interno di una rete in modo del tutto trasparente.

NIC
Network Interface Controller. Adattatore installato in un computer per ottenere una connessione fisica a una rete, tipicamente una scheda *Ethernet*.

NIS
Network Information System. Il NIS era conosciuto anche come *Yellow Pages*, ma la **British Telecom** possiede un copyright su questo nome. Il NIS è un protocollo definito dalla **Sun Microsystems**, avente come scopo la condivisione di informazioni all'interno di un **dominio** NIS, che può comprendere un'intera LAN, parte di una LAN o anche più LAN. Può esportare database di password, database di servizi, informazioni sui gruppi e altro ancora.

nome utente
È un nome (o, più in generale, una parola) che identifica un utente in un sistema. Ad ogni nome utente corrisponde un'unica e univoca UID (*user ID*)
Si veda anche: login.

open source
Si riferisce al nome dato al codice liberamente distribuibile di un programma reso disponibile per la comunità di programmatori e pubblico in un senso più generale. Secondo la teoria che sta alla base di questo concetto, se si permette al codice sorgente di essere usato e modificato da un ampio gruppo di programmatori il risultato finale sarà, in ultima analisi, un prodotto migliore per tutti. Tra i più diffusi programmi open source citiamo *Apache*, *sendmail* e *GNU/Linux*.

pagina di manuale
Un breve documento contenente la definizione di un comando e il suo uso, da consultarsi per mezzo del comando `man nome_del_comando`. È la prima cosa che si dovrebbe (imparare a) leggere quando si sente pronunciare il nome di un comando sconosciuto **:-)**

PAP
Password Authentication Protocol. Protocollo usato da molti ISP nella fase di autenticazione del client. Secondo questo metodo, il client (e quindi il computer dell'utente) invia una coppia identificatore/password non criptata al server.
Si veda anche: CHAP.

password (parola d'ordine)
È una parola segreta, o la combinazione di parole o lettere, usata per garantire la sicurezza di particolari operazioni. Le password sono usate insieme ai nomi di login degli utenti su sistemi operativi multiutente, siti web, siti FTP, e così via. Dovrebbero essere frasi o combinazioni alfa-numeriche difficili da indovinare, e non dovrebbero mai essere basate su semplici parole reperibili in un dizionario. Le password assicurano che nessun altro possa accedere a un computer o a un sito utilizzando il vostro account.

password ombra
Un metodo di gestione delle password sui sistemi `Unix` (ingl. "shadow password") in cui il file che contiene le password criptate non è più accessibile in lettura, mentre lo è quando si usa il sistema di gestione delle password normale.

patch, applicare una patch
File che contiene una serie di correzioni da applicare a un codice sorgente al fine di aggiungere nuove caratteristiche, di rimuovere difetti, o di modificarlo secondo i desideri e i bisogni dell'autore. L'azione, ovvero l'applicazione di una patch (ingl. *to patch*), consiste nell'applicare tali correzioni direttamente all'archivio del codice sorgente.

PCI
Peripheral Components Interconnect. Un bus creato dalla **Intel** che ha raggiunto lo status di standard per le architetture odierne, e che viene usato anche su altre architetture. È il successore del bus ISA, e offre numerosi servizi: identificazione del dispositivo, informazioni sulla configurazione, condivisione di IRQ, bus mastering e altri ancora.

PCMCIA
Personal Computer Memory Card International Association. Chiamato sempre più spesso "PC Card" per motivi di semplicità, è lo standard relativo alle schede esterne inserite in un computer portatile: modem, dischi rigidi, espansioni di memoria, schede `Ethernet` e altro ancora. L'acronimo originario viene talvolta letto, a fini umoristici, come *People Cannot Memorize Computer Industry Acronyms*...

percorso
Si riferisce alla collocazione di file e directory nel filesystem. I diversi componenti di un percorso sono separati dal carattere '/'. Esistono due tipi di percorso sui sistemi `GNU/Linux`: il percorso **relativo** corrisponde alla posizione di un file o una directory in relazione alla directory corrente; il percorso **assoluto** è la posizione di un file o una directory in relazione alla directory radice.

pipe
Uno speciale tipo di file `Unix`. Un programma scrive dati in una pipe (lett. "tubo"), e un altro programma legge i dati dall'altra estremità. Le pipe `Unix` sono di tipo FIFO (*First In First Out*), in modo che i dati vengano letti esattamente nello stesso ordine in cui sono stati immessi. Sono usate molto spesso nella shell.
Si veda anche: pipe con nome.

pipe con nome
Una pipe `Unix` a cui è assegnato un nome nel filesystem, a differenza delle pipe usate nelle shell.
Si veda anche: link, pipe.

pixmap
Acronimo di "pixel map" ("mappa basata su pixel"). È un altro modo per riferirsi alle immagini di tipo bitmap, composte da pixel.

plugin
Programma aggiuntivo usato per mostrare o riprodurre i contenuti multimediali di un documento web . Nel caso in cui il vostro navigatore non sia ancora in grado di mostrare o riprodurre un particolare tipo di informazione, non dovrebbe essere troppo difficile reperire il relativo plugin sul web .

PNG
Portable Network Graphics. Formato di file per immagini creato principalmente per essere usato sul web, è stato progettato per essere un sostituto non vincolato da brevetti del formato GIF, rispetto al quale offre alcune caratteristiche supplementari.

PnP
Plug'N'Play. Nato come un'espansione del bus ISA mirata ad aggiungere informazioni di configurazione per i dispositivi, è diventato un termine dall'uso più generico che riunisce tutti i dispositivi in grado di comunicare i loro parametri di configurazione. In quanto tali, tutti i dispositivi PCI sono Plug'n'Play.

POP
Post Office Protocol. Il protocollo usato comunemente per scaricare la posta elettronica da un ISP.

porting
Si riferisce al metodo di traduzione di un programma in maniera tale che possa essere usato in un sistema operativo diverso da quello per il quale era stato scritto, o che possa essere usato in sistemi "simili". Per poter eseguire un programma nativo di *Windows* sotto *GNU/Linux* (in modo nativo), ad esempio, è necessario che prima venga "portato" su *GNU/Linux*.

PPP
Point to Point Protocol. È il protocollo usato per trasmettere dati usando linee seriali. Viene usato comunemente per inviare pacchetti IP su Internet, ma può essere usato anche con altri protocolli, quale ad esempio il protocollo IPX di Novell.

precedenza
Determina l'ordine di valutazione degli operandi in una espressione. Ad esempio: nel caso di `4 + 3 * 2` il risultato è 10, in quanto la moltiplicazione ha la precedenza rispetto all'addizione. Se volete che prima venga calcolata l'addizione, dovete aggiungere delle parentesi così: `(4 + 3) * 2`, e il risultato finale sarà 14, poiché le parentesi hanno la precedenza rispetto all'addizione e alla moltiplicazione, pertanto le operazioni che si svolgono fra le parentesi vengono calcolate prima di ogni altra.

preprocessori
Sono direttive di compilazione che vengono sostituite dal compilatore con codice nel linguaggio di programmazione del file sorgente. Esempi di preprocessori in *C* sono `#include`, `#define`, etc.

processo
In ambito *Unix*, un processo è costituito dall'istanza di un programma in esecuzione, insieme con il suo ambiente.

prompt
È la stringa di caratteri che precede il cursore di una *shell*. Quando è visibile è possibile digitare dei comandi.

proprietario
Nel contesto degli utenti e dei loro file, il proprietario di un file è l'utente che lo ha creato.

protocollo
I protocolli organizzano la comunicazione fra macchine differenti all'interno di una rete, usando mezzi hardware o software. Definiscono il formato dei dati trasferiti, se una macchina ne controlla un'altra, etc. Tra i protocolli più diffusi citiamo HTTP, FTP, TCP, e UDP.

proxy
Una macchina che si interpone fra una rete e Internet, il cui ruolo è quello di accelerare i trasferimenti di dati per i protocolli più diffusi (HTTP e FTP, ad esempio). Conserva una cache dei dati ottenuti in seguito alle ultime richieste, il che evita la necessità di dover richiedere il trasferimento di un file presente nella cache se un'altra macchina chiede lo stesso file. I proxy sono molto utili su reti che dispongono di una scarsa ampiezza di banda (ovvero: le connessioni via modem). A volte il proxy è l'unica macchina in grado di accedere all'esterno.

punto di mount
È la directory tramite la quale una partizione o un dispositivo di altro tipo è connesso al filesystem *GNU/Linux*. Il vostro CD-ROM, ad esempio, è montato sulla directory `/mnt/cdrom`: per esplorare il contenuto di ogni CD che verrà inserito dovrete partire da tale directory.

quota
È un metodo per limitare l'uso del disco da parte degli utenti. Gli amministratori possono imporre limiti alle dimensioni delle directory home degli utenti impostando dei valori di quota su specifici filesystem.

RAID
Redundant Array of Independent Disks. Un progetto nato presso il dipartimento di scienze informatiche dell'Università di Berkeley, il cui scopo è distribuire l'archiviazione dei dati su di un insieme di dischi

rigidi usando diversi schemi. In un primo momento, questa caratteristica è stata implementata usando dei drive floppy, che è il motivo per cui l'acronimo originario significava *Redundant Array of Inexpensive Disks*.

RAM
Random Access Memory. Termine utilizzato per indicare la memoria principale di un computer.

RFC
Request For Comments. Gli RFC sono i documenti ufficiali che descrivono gli standard di Internet. Documentano tutti i protocolli, il loro uso, i loro requisiti, e così via. Quando volete sapere come funziona un certo protocollo, procuratevi l'RFC corrispondente.

root
È il super-utente di un qualsiasi sistema *Unix*. Tipicamente root (ovvero l'amministratore del sistema) è la persona responsabile per la manutenzione e la supervisione del sistema, inoltre ha anche l'accesso completo a qualunque cosa si trovi sul sistema.

route
Si riferisce al percorso che esiste tra due particolari macchine all'interno di una rete.

RPM
Redhat Package Manager. Un formato sviluppato dalla **Red Hat** per creare pacchetti software, viene usato in molte distribuzioni *GNU/Linux*, inclusa la **Mandrake Linux**.

run level
È una configurazione del software di sistema che permette solo ad alcuni specifici processi di esistere. I processi consentiti sono definiti, per ciascun run level, nel file `/etc/inittab`. I run level definiti sono otto: 0, 1, 2, 3, 4, 5, 6, S; il passaggio dall'uno all'altro può essere effettuato soltanto da un utente in possesso dei privilegi di amministratore, eseguendo i comandi `init` e `telinit`.

schermo intero
Questo termine viene usato nel caso di programmi la cui finestra occupa l'intera area visibile dello schermo.

script
Gli script della *shell* sono sequenze di comandi che devono essere eseguiti come se fossero stati digitati uno dopo l'altro. Gli script della *shell* sono pressappoco l'equivalente *Unix* dei file batch del *DOS*.

SCSI
Small Computers System Interface. Un bus caratterizzato da un'alta velocità di trasferimento dati, progettato per consentire la connessione di periferiche di vario tipo. A differenza del bus IDE, il bus SCSI non è limitato dalla velocità massima alla quale le periferiche accettano i comandi. Solo le macchine di fascia alta integrano un bus SCSI direttamente sulla scheda madre, i *PC* in genere necessitano di schede aggiuntive.

server
Programma o computer che offre una caratteristica o un servizio e attende la connessione dei **client** per eseguire i loro ordini o dar loro le informazioni che chiedono. Uno dei componenti di un **sistema client/server**.

shell
La *shell* è l'interfaccia fondamentale verso il kernel del sistema operativo, e offre la linea di comando nella quale gli utenti possono digitare comandi per eseguire programmi e software di sistema. Tutte le shell offrono un linguaggio script che può essere usato per rendere automatiche certe operazioni o semplificare operazioni complesse eseguite frequentemente. Gli script della *shell* sono simili ai file batch del sistema operativo *DOS*, ma sono molto più potenti. Tra le shell più comuni citiamo *bash*, *sh*, e *tcsh*.

sistema client/server
Sistema o protocollo basato su di un **server** e uno o più **client**.

sistema operativo
È l'interfaccia tra le applicazioni e l'hardware. Il compito principale di un qualsiasi sistema operativo è quello di gestire tutte le risorse specifiche della macchina. In un sistema *GNU/Linux* questo compito è svolto dal kernel e dai moduli caricabili. Tra gli altri sistemi operativi più conosciuti citiamo *AmigaOS*, *MacOS*, *FreeBSD*, *OS/2*, *Unix* e *Windows*.

SMB
Server Message Block. Protocollo usato dalle macchine `Windows` (`9x` o `NT`) per condividere file e stampanti all'interno di una rete.
Si veda anche: CIFS.

SMTP
Simple Mail Transfer Protocol. Si tratta del più diffuso protocollo per trasmettere messaggi di posta elettronica. Viene utilizzato dai programmi che si occupano del trasferimento della posta, come `sendmail` e `postfix`, detti anche "server SMTP ".

socket
Tipo di file corrispondente a una qualsiasi connessione di rete.

soft link
Si veda "link simbolico".

specifico dell'installazione
Significa che le informazioni usate da programmi come `imake` e `make` per compilare dei file sorgenti dipendono dal sistema installato, dall'architettura del computer, dalle librerie installate nel sistema, e così via.

standard error
Il descrittore di file numero 2, aperto da ogni processo, convenzionalmente usato per stampare messaggi di errore, e associato normalmente allo schermo del terminale.
Si veda anche: standard input, standard output.

standard input
Il descrittore di file numero 0, aperto da ogni processo, convenzionalmente usato come il descrittore di file dal quale il processo riceve i dati.
Si veda anche: standard error, standard output.

standard output
Il descrittore di file numero 1, aperto da ogni processo, convenzionalmente usato come il descrittore di file nel quale il processo stampa il suo output.
Si veda anche: standard error, standard input.

streamer
Si riferisce a un dispositivo che riceve "flussi" (ingl. *streams*, un invio di dati non interrotti o divisi in frammenti più piccoli) di caratteri come input. Un tipico esempio di streamer è un'unità a nastro magnetico.

SVGA
Super Video Graphics Array. Standard di visualizzazione grafica progettato dalla VESA per l'architettura `PC`. La risoluzione è di 800 x 600 x 16 colori.

switch
Gli switch (lett. "interruttori") sono usati per cambiare il comportamento di alcuni programmi, e vengono detti anche opzioni o argomenti da linea di comando. Per stabilire se un programma ha degli switch opzionali che possono essere usati, leggete le relative pagine `man` o provate a passare l'argomento `--help` al programma (digitate cioè `nome_programma --help`).

target
Letteralmente "bersaglio", è l'oggetto della compilazione, ovvero il file binario che verrà generato dal compilatore.

TCP
Transmission Control Protocol. È il più diffuso e affidabile protocollo che usa l' IP per trasmettere pacchetti di rete. Il TCP aggiunge i controlli necessari all' IP in modo da avere la certezza che i pacchetti sono stati consegnati. A differenza dell' UDP, il TCP lavora in modalità di connessione, il che significa che per rendere possibile uno scambio di dati fra due macchine, deve prima essere stabilita una connessione fra di esse.

telnet
Crea una connessione a un host remoto e consente l'accesso a quella macchina, a condizione che si disponga di un account. Telnet è il metodo più usato per gli accessi a distanza, tuttavia ci sono alternative migliori e più sicure, come `ssh`.

URL
Uniform Resource Locator. Una stringa in un formato speciale usata per identificare una risorsa su Internet in maniera univoca. Questa risorsa può essere un file, un server o altro ancora. La sintassi per una URL è `protocollo://nome.del.server[:porta]/percorso/della/risorsa`. Quando viene inserito soltanto il nome di una macchina e il protocollo è `http://`, come opzione predefinita viene generalmente trasmesso il file `index.html` eventualmente presente sul server.

valori discreti
Valori che non sono continui. In altre parole, esiste una "spaziatura" fra due valori consecutivi.

variabili
Sono stringhe usate nei file `Makefile` che verranno rimpiazzate dal valore loro assegnato ogni volta che appaiono. In genere si trovano all'inizio del `Makefile`. Vengono usate per semplificare la gestione di `Makefile` e dell'albero dei file sorgenti.
In termini più generali, nella programmazione le variabili sono parole che fanno riferimento ad altre entità (numeri, stringhe, tabelle, etc.) suscettibili di variazione durante l'esecuzione del programma.

variabili d'ambiente
Una parte dell'ambiente di un processo. Le variabili d'ambiente sono direttamente visibili dalla *`shell`*.
Si veda anche: processo.

VESA
Video Electronics Standards Association. Un'associazione che si occupa di standard in relazione all'architettura *`PC`*. Dobbiamo a essa lo standard SVGA, ad esempio.

visualizzatore a pagine
Programma che visualizza un file di testo una schermata alla volta (ingl. *pager*), in modo da facilitare gli spostamenti avanti e indietro nel file e la ricerca di stringhe. Fra i programmi di questo genere vi consigliamo `less`.

WAN
Wide Area Network. Questa rete, per quanto simile a una LAN, connette computer che non sono fisicamente collegati agli stessi cavi e sono separati da un distanza più grande.

window manager
Il programma responsabile per il *look and feel* di un ambiente grafico, che si occupa delle barre delle finestre, delle cornici, dei pulsanti, dei menu predefiniti e di alcune scorciatoie da tastiera. Senza di esso, sarebbe molto difficile, se non impossibile, avere dei desktop virtuali, ridimensionare le finestre in tempo reale, spostarle sullo schermo, e così via.

Indice