12. Gli incantesimi di base

Nonostante l'interfaccia grafica di Linux sia piuttosto sofisticata ed efficace, la vera potenza di Linux si rivela quando usate la sua interfaccia "vera", cioè quella testuale che trovate ad esempio nelle sue console o nelle finestre di terminale. Quando l'interfaccia grafica non parte, risulta lenta o annaspa, l'interfaccia testuale scatta veloce e risolve ogni problema. Inoltre l'interfaccia di testo è l'unica disponibile nelle installazioni su macchine molto modeste.

Insomma, qualunque cosa facciate, prima o poi vi troverete ad aver bisogno dell'interfaccia non grafica di Linux. Lo so bene, perché troppe volte, durante la stesura di questo libro, non sapevo che pesci pigliare o l'interfaccia grafica si rifiutava di fare quello che le chiedevo; tutto si risolveva aprendo una console e digitando gli arcani comandi dell'interfaccia testuale. Certo, bisogna tribolare per impararli, mentre l'interfaccia grafica "punta e clicca" è molto più intuitiva, ma se volete il potere, dovete guadagnarvelo.

Solitamente i libri su Linux si dilungano in chilometriche spiegazioni su tutte le possibili varianti di tutti i comandi disponibili nell'interfaccia testuale. Questo no. Tutti gli approfondimenti dei singoli comandi sono disponibili nella documentazione allegata alla vostra copia di Linux; qui vi presento soltanto una selezione dei comandi indispensabili per iniziare a lavorare.

Scorciatoie

Nonostante le dicerie che affermano il contrario, agli utenti Linux non piace usare la tastiera più dello stretto necessario. Non sono innamorati della riga di comando e non si eccitano a digitare comandi arcani e lunghissimi come **find / −type f \(−perm −04000 −o −perm −02000 \) > suid−sgid.txt**. Infatti esistono varie tecniche per risparmiare sul numero di caratteri da digitare.

Completamento automatico

Quando digitate un nome di file o di directory, è sufficiente scriverne le prime lettere e poi premere il tasto Tab. Se non ci sono ambiguità, cioè se nella directory corrente c'è un solo file o una sola sottodirectory il cui nome contiene le lettere che avete digitato, Linux scrive automaticamente il resto.

Ad esempio, immaginate di essere in una directory contenente tre programmi dai nomi chilometrici:

- *so−5_2−ga−bin−linux−it.bin*
- *soa−5_2−ga−bin−linux−it.bin*
- *sop−5_2−ga−bin−linux−it.bin*

Come vedrete nei prossimi capitoli, non sono nomi inventati: la maggior parte dei nomi di file in Linux è di questa lunghezza. Questo però non vuol dire che dovete intrecciarvi le dita nel tentativo di scriverli giusti. Grazie al completamento automatico, per avviare il programma prescelto vi basta digitare l'inizio del suo nome (in alcune configurazioni di Linux, dovrete precederlo con "./", per i motivi spiegati nella sezione *Problemi di path* più avanti).

Infatti **so−** è quanto basta per risolvere le ambiguità (digitare **so** senza trattino non basterebbe, perché nell'elencazione ci sono tre file che iniziano con *so*) e consentire a Linux di capire a quale file vi state riferendo. Premendo il tasto Tab, il nome completo verrà scritto da Linux per voi.

Comodo, vero? Questo trucchetto vale un po' dappertutto in Linux, sia per i nomi di file, sia per i nomi di directory. Ad esempio, per digitare il percorso **/usr/bin/office52/program/setup**, basta digitare **/u** e il tasto Tab, **b** e di nuovo Tab, **off** e ancora Tab, **p** e Tab, e infine **se** e Tab. In altre parole:

Voi scrivete	**Linux completa con...**
/u[TAB]	/usr/
b[TAB]	/usr/bin/
off[TAB]	/usr/bin/office52/
p[TAB]	/usr/bin/office52/program/
se[TAB]	/usr/bin/office52/program/setup
(14 battute)	(31 battute)

Esiste anche il completamento automatico parziale: ad esempio, supponete che una directory contenga tre soli file i cui nomi cominciano tutti per *configurazion* (ad esempio *configurazione_stampante*, *configurazioni_schermi*, *configurazione_account*). Digitando **cat** (un comando di visualizzazione) e il tasto Tab, la parte comune del nome *(configurazion*) verrà scritta automaticamente da Linux.

Se ci sono anche altri file con nomi completamente diversi, basta digitare l'inizio del nome che vi interessa: ad esempio, se nella directory appena citata c'è anche un file di nome *impostazione_internet*, basta digitare la C minuscola seguita dal tasto Tab per dire a Linux "mi interessa uno dei file che comincia per C minuscola, completa quello che puoi", ottenendo così il completamento automatico della parte comune (*configurazion*). Confusi? È più facile da mettere in pratica che da descrivere.

Anche molti programmi supportano il completamento automatico nei loro parametri. Se usate un programma che richiede un file come parametro (che so, *zip*), potete digitare **zip** seguito dalle prime lettere del nome del file e poi premere Tab per lasciare che Linux completi il nome del file.

Spesso è difficile determinare quante lettere occorre digitare per non essere ambigui e consentire il completamento automatico. In casi come questi potete digitare il tasto Tab due volte: ottenete l'elenco dei file e delle directory che corrispondono alle lettere immesse fino a quel punto.

Questo tipo di scorciatoia non è appannaggio esclusivo dell'interfaccia testuale di Linux. Funziona anche in buona parte dei programmi dell'interfaccia grafica. È inutile tediarvi elencando dove funziona e dove no: ricordatevi di provarlo ogni volta che dovete digitare un percorso o un nome di file, e imparerete presto quali (pochi) punti di Linux non accettano la scorciatoia.

Concisione o crudele ermetismo?

Nelle prossime pagine noterete che i nomi dei comandi di Linux sono incredibilmente corti. Molti sono addirittura di due lettere (**mv, ls, cd** e simili). Anche questo dimostra che i linuxiani non amano digitare più del necessario.

Tuttavia molti trovano che questa concisione sia un po' eccessiva e finisca per rendere poco comprensibili i comandi. Quando è stato creato Linux c'era carestia di vocali? Sarebbe stato davvero così faticoso digitare, che so, **move** al posto di **mv** (il comando per spostare i file)? In fin dei conti, con *move* se uno sa un minimo d'inglese il senso del comando è chiarissimo: *move* infatti vuol dire "sposta".

La scelta di essere così concisi non deriva da un innato sadismo dei linuxiani. Linux, infatti, non fa altro che imitare fedelmente l'organizzazione e i comandi del sistema operativo UNIX dal quale deriva. UNIX, a sua volta, è così conciso perché ai tempi in cui fu concepito non si usavano i monitor: il computer "visualizzava" comandi e risposte su una specie di stampante.

Sì, lo so che sembra improbabile a chi si avvicina adesso all'informatica, ma il monitor grafico a colori che diamo oggi per scontato era un costosissimo privilegio di pochi eletti negli anni Sessanta e Settanta: la prossima volta che guardate *Guerre stellari* (il primo della serie, intendo), fate attenzione alla presentazione degli schemi della Morte Nera. Quello era il massimo livello della grafica computerizzata dell'epoca, altro che i giochi di ombreggiatura tridimensionale che trovate sulla Playstation. Di conseguenza, quasi tutti coloro che avevano accesso agli enormi computer di quegli anni, e quei pazzi che usavano i primi rudimentali personal computer, comunicavano col cervellone tramite stampante.

Sempre in tema di fantascienza, se vi siete mai chiesti perché diavolo il computer di *Spazio: 1999* rispondeva principalmente tramite striscioline di carta invece di usare uno schermo, adesso sapete perché. Allora i computer erano fatti così, e la fantascienza estrapolava le tecnologie più moderne del momento. Nel Centro di Comando di base Alpha c'erano tanti teleschermi per la comunicazione video, ma un solo monitor collegato al computer, e lo si usava solo per le grandi occasioni.

Insomma, stampare *mv* al posto di *move* consumava il 50% in meno di inchiostro e richiedeva metà tempo. Moltiplicate questo risparmio per cento o mille volte e vedrete che comincia ad essere significativo, soprattutto se siete voi a pagare carta e inchiostro.

In un certo senso, questa estrema brevità dei comandi Linux è linguisticamente democratica. Molti altri sistemi operativi a interfaccia testuale, come ad esempio il DOS, usano parole inglesi come comandi: *copy, rename, format, delete*, eccetera. Questo ovviamente li rende più facili per chi mastica l'inglese, ma chi parla altre lingue è svantaggiato.

Con comandi come *ls, cp, dd*, Linux invece è difficile anche se sapete l'inglese: solo alcuni dei suoi comandi sono basati su parole di questa lingua. Per cui gli anglofoni non partono avvantaggiati.

I comandi più recenti

Per richiamare sulla riga di comando gli ultimi comandi immessi, premete ripetutamente Ctrl−P o il tasto freccia verso l'alto. Questo vi evita di dover ridigitare sequenze di comandi lunghe e ripetitive e vi consente di riesaminare i comandi immessi, per vedere esattamente che cosa è stato fatto quando qualcosa va storto.

Caratteri jolly

Se avete dimestichezza con il DOS, saprete cosa sono i *caratteri jolly.* Sono dei caratteri speciali che possono rappresentare uno o più caratteri (donde il nome) quando dovete specificare un nome di file in un comando.

Ad esempio, supponete di avere nella directory corrente due file di nome *pippo1* e *pippo2* e di volerli cancellare. Il comando da usare è **rm**, descritto in dettaglio tra poco. Potreste dare due comandi:

rm pippo1
rm pippo2

Ma potreste essere più efficienti dando il comando:

rm pi*

Se i file da cancellare fossero più numerosi, il vantaggio di questo metodo sarebbe ancora più evidente. L'asterisco, insomma, sta per "qualsiasi carattere o serie di caratteri". Se digitate **rm ***, tutti i file contenuti nella directory corrente verranno cancellati, qualunque sia il loro nome. Notate la differenza rispetto al DOS, dove "qualsiasi file" si specifica digitando "*.*": in Linux, invece, è sufficiente l'asterisco singolo per indicare anche file che hanno un'estensione.

C'è anche un altro carattere jolly importante, ed è il punto interrogativo. Si comporta in modo simile all'asterisco, con una sola differenza da ricordare: mentre l'asterisco rappresenta qualsiasi numero di caratteri, il punto interrogativo rappresenta *un solo* carattere.

Ad esempio, se la directory corrente contiene tre file, *papero*, *paperino* e *paperina*, il comando **rm paper*** significa "cancella tutti i file che cominciano per *paper*", e quindi cancella tutti e tre i file. Se invece digitate **rm paper?**, ordinate a Linux di cancellare tutti i file che cominciano per *paper* e proseguono con un solo carattere. Di conseguenza, cancellate soltanto *papero* (l'unico file che corrisponde alla specifica), mentre *paperino* e *paperina* si salvano.

Taglia e incolla con il mouse

In Linux, il mouse funziona anche nelle finestre di terminale e nelle console e consente di selezionare, copiare e incollare porzioni di testo. Windows fa le stesse cose nelle finestre di MS−DOS. La differenza è il modo in cui si comandano queste funzioni nei due sistemi operativi.

In Windows, per poter selezionare del testo la finestra di MS−DOS non può occupare tutto lo schermo, ma va ridotta in modo che compaiano i pulsanti per attivare la modalità di selezione, di copia e di incollaggio. Poi trascinate il puntatore del mouse sulla porzione di testo desiderata e premete Invio (o selezionate il pulsante Copia) per mettere la selezione nella memoria temporanea della Clipboard. Infine, quando volete incollare altrove nella finestra di MS−DOS il testo selezionato, cliccate sul pulsante Incolla.

In Linux la cosa è un po' più semplice, ma richiede del tempo per dimenticare le abitudini acquisite con Windows. Infatti non occorre selezionare pulsanti per attivare le varie modalità: basta trascinare il mouse, senza preamboli, sopra la porzione di testo che vi interessa. Rilasciando il pulsante del mouse, il testo viene automaticamente copiato nella memoria temporanea. Cliccando con il terzo pulsante (reale o emulato), la porzione di testo copiata viene incollata.

Le buone notizie non sono finite. Questo sistema rapido di selezione, copia e incollaggio funziona non solo nelle finestre di terminale, ma anche nelle console, e addirittura da una console all'altra. Fra l'altro, un doppio clic su una parola seleziona automaticamente tutta la parola. Se invece trascinate il mouse sopra l'ultima parola di una riga, possono succedere due cose: se trascinate fino alla fine della parola, copiate la parola; se invece trascinate appena oltre la fine della parola, copiate la parola più l'"a capo" che c'è a fine riga.

Questo è molto comodo quando occorre venire in soccorso di un Linux che fa le bizze. Infatti mentre in Windows è in genere il mouse il primo a paralizzarsi in caso di avaria mentre la tastiera continua a funzionare, in Linux capita più spesso che sia la tastiera a dare i numeri e il mouse rimanga operativo. Selezionando le lettere giuste per comporre i comandi e gli "a capo", si può emulare la tastiera quanto basta per tirarsi fuori dai guai.

Fermo o sparo!

Quelli che presento qui non sono comandi in senso stretto, ma vi conviene conoscerli lo stesso perché vi permetteranno di tirarvi fuori dalle situazioni difficili.

- **All'avvio, i messaggi spariscono dallo schermo troppo in fretta.** Di solito questo non è un problema e di certo non siete obbligati a leggere tutti i messaggi che compaiono ad ogni accensione, ma se Linux ha dei problemi di avvio, i primi indizi delle cause di questi problemi si annidano fra questi messaggi. Se avete bisogno di tornare

indietro o scorrerli durante l'avvio, magari per leggerli esattamente a qualcuno che vi sta aiutando, usate la combinazione di tasti **Maiusc−PgSu** per rivedere i messaggi scomparsi dal video e **Maiusc−PgGiù** per farli scorrere in avanti. Questo trucchetto funziona sia con i messaggi di avvio, sia con quelli di chiusura, a patto di non cambiare console. Più in generale, funziona in tutte le schermate delle console.

- **Nonostante il trucchetto sopra indicato, i messaggi scorrono troppo veloci perché vengono sostituiti da quelli nuovi.** Usate **Ctrl−S** per sospendere l'arrivo di nuovi messaggi. Quando avete finito, digitate **Ctrl−Q** per riavviare il flusso di messaggi.
- **Non riuscite a uscire dall'interfaccia grafica**. Anche questo può capitare, ma non è un problema grave come in Windows, perché l'interfaccia grafica è per Linux soltanto un programma come gli altri: il sistema operativo vero e proprio non è influenzato da eventuali magagne dell'interfaccia. Di conseguenza, in caso di blocco dell'interfaccia potete tranquillamente chiuderla brutalmente digitando **Ctrl−Alt−Backspace**. Il sistema operativo sottostante continuerà a funzionare (le applicazioni non grafiche non verranno neppure interrotte); se avete impostato Linux per avviare automaticamente l'interfaccia grafica, dopo una breve attesa l'interfaccia si riavvierà.
- **Non riuscite a chiudere Linux.** Aprite una console e digitate **Ctrl−Alt−Canc**. Non è necessario fare login: basta aprire la console. Fate attenzione: in Windows, questa digitazione richiama un menu che consente di interrompere brutalmente un qualsiasi programma e, se ripetuta, chiude ancora più brutalmente Windows (è l'equivalente di un reset o di un calo di tensione), mentre in Linux le cose funzionano diversamente: basta un solo Ctrl−Alt−Canc per imporre direttamente la chiusura del sistema operativo. Inoltre la chiusura di Linux è ordinata: è come digitare **shutdown −r now** oppure scegliere *Fine sessione* dal Pannello e *Riavvia il computer* dalla schermata grafica di login. In entrambi i casi, comunque, perdete i dati non salvati.
  Alcune installazioni di Linux sono configurate in modo da disabilitare la possibilità di spegnere il sistema digitando **Ctrl−Alt−Canc** per motivi di sicurezza. La faccenda è spiegata in dettaglio nel Capitolo 18.

Conto fino a tre e poi riavvio

A proposito del comando **shutdown**: ci sono alcune opzioni interessanti che vale la pena di conoscere. Innanzi tutto, il parametro **−r** ordina a Linux di spegnersi e riavviarsi; in alternativa, potete dare il parametro **−h**, che si limita a spegnere il computer senza riavviarlo.

Al posto di **now**, invece, potete specificare un'ora (nel formato *ore:minuti*) oppure un tempo (espresso in minuti e preceduto dal segno "+").

Ad esempio:

- **shutdown −h 23:55** avvia la chiusura di Linux cinque minuti prima di mezzanotte e spegne il computer.
- **shutdown −r +5** avvia la chiusura di Linux tra cinque minuti e successivamente riavvia il computer.

Avete presente i classici film d'azione dove l'eroe innesca la superbomba per distruggere tutto e poi ovviamente succede un contrattempo per cui il tempo impostato sul timer non gli basta e deve tornare indietro a fermarlo? Be', il comando **shutdown** può incastrarvi allo stesso modo. Capita spesso di dare il comando di chiusura del sistema in modo che si spenga da solo entro cinque minuti per poi rendersi conto che c'è ancora qualcosa di importante da fare prima della chiusura e che non c'è tempo di aspettare il riavvio.

Per casi come questo c'è l'opzione **−c**. In sostanza, se date il comando **shutdown −c** e poi digitate **Ctrl−C** prima dello scadere del tempo, otterrete (dopo una lunga, angosciante pausa) la risposta *Shutdown cancelled.* Attenzione: dato che **shutdown** disattiva quasi immediatamente ogni ulteriore possibilità di fare login, in pratica potete dare il comando di cancellazione soltanto dalla console aperta dalla quale avete dato il comando di chiusura.

Ampersand, chi era costui?

Cosa c'entra con l'informatica Hans Christian Ampersand, famoso autore danese di fiabe celebri come *La piccola fiammiferaia*? Niente, assolutamente niente. *Ampersand* (non *Andersen*) è il nome del simbolo "&". Tutti lo chiamano "E commerciale", ma non è il suo nome ufficiale.

Tutto questo preambolo di dubbio umorismo serve a introdurre una funzione molto comoda delle finestre di terminale dell'interfaccia grafica: la possibilità di lanciare un programma digitandone il nome al prompt. Esempio: aprite una finestra di terminale e vi trovate nella necessità di aprire un editor di testi per modificare un file. Vi basta digitare **kedit** per avviare l'editor.

Sistema comodo, per carità, ma con un difetto: finché non chiudete il programma lanciato in questo modo, la finestra di terminale è inutilizzabile. Il prompt infatti ricompare solo alla chiusura del programma. Magari volete lanciare un'elaborazione molto lunga e intanto fare qualcos'altro, ma non potete. Dovreste aprire una seconda finestra di terminale.

Ecco dove entra in scena il nostro ampersand. Per ovviare a questo problema, infatti, basta digitare l'ampersand dopo il nome del programma (ad esempio **kedit &**). In questo modo, il programma viene lanciato separatamente dalla finestra di terminale, che quindi ritorna al prompt ed è utilizzabile immediatamente senza dover attendere la fine del programma lanciato.

La sibillina riga di comando

Se siete stati allevati nell'era della grafica computerizzata, probabilmente ritenete che l'interfaccia testuale sia rozza e primitiva. In realtà è un sistema molto economico per fornire tante informazioni in poco spazio e senza richiedere ingenti potenze di calcolo. L'unico suo svantaggio è che richiede un po' di uso del cervello da parte dell'utente per interpretarne il significato. È per motivi come questi che Linux non piace a tutti. Molta gente preferisce la pappa pronta e qualcuno che gliela imbocchi.

Ecco come decifrare le informazioni presenti nella riga di comando, o per essere più precisi, nel *prompt dei comandi*.

Considerate ad esempio questo prompt:

```
[root@deepspace9 /root]#
```

Quel *root@deepspace9*, che somiglia tanto a un indirizzo di email, è proprio un indirizzo di e−mail: indica chi siete (*root*) e su che computer vi trovate (*deepspace9*). Tuttavia funziona soltanto se impostate correttamente il software di smistamento locale della posta. Non confondetelo con il vostro normale indirizzo e−mail su Internet.

Se vi sembra stupido specificare il nome del computer di fronte al quale siete seduti perché sapete benissimo dove siete, ricordate che le console possono essere attivate anche durante una connessione via Internet a un computer remoto: il vostro computer diventa un terminale della macchina remota. In questo caso, il prompt vi indicherà il nome della macchina a cui siete collegati per ricordarvi che state dando comandi a *quel* computer e non a quello davanti al quale siete seduti.

Il prompt prosegue con la specificazione della directory corrente (in questo caso */root*) e si conclude con un carattere # (abitualmente chiamato "cancelletto"). Il carattere finale è importante, perché vi ricorda che poteri avete in quel momento: se è il cancelletto, avete i superpoteri di *root*; se è il dollaro ($), avete i privilegi di un utente normale.

Fare login in una console

C'è una differenza notevole fra usare una finestra di terminale e usare una console. Se aprite una finestra di terminale, Linux presume che vogliate lavorare con l'identità dell'utente che in quel momento sta lavorando nell'interfaccia grafica di Linux, per cui non vi chiede né nome né password.

Se fate login da una console, invece, vi viene chiesto esplicitamente chi siete (alla richiesta *login:* digitate il vostro nome utente) e la vostra password; quindi potete accedere con identità diverse a seconda delle necessità, e lo potete fare anche simultaneamente. Ad esempio, tipicamente si entra in una console come *root* per la manutenzione e si lavora nell'interfaccia grafica come utente normale.

Ecco un tipico esempio di login:

```
deepspace9 login: root
Password: [non visualizzata]
Last login: Thu Aug 31 14:19:51 on tty3
[root@deepspace9 /root]#
```

Una cosa interessante del login da console è che vi viene detto in che giorno e a che ora, e da che terminale o console, l'utente ha effettuato l'ultimo login. Questo è molto utile in termini di sicurezza: se notate che l'ora indicata non corrisponde al vostro ultimo accesso, gatta ci cova.

Per terminare una sessione in una console, digitate il comando **logout** o **exit**. Vi conviene imparare a usare **exit**, dato che questo comando funziona sia per le console, sia per le finestre di terminale.

Problemi di path

A differenza di altri sistemi operativi (tipo il DOS), molte installazioni di Linux non includono automaticamente nel *path* la directory corrente, specialmente se state lavorando come *root*. In altre parole, se accedete al computer come *root* e digitate il nome di un comando o di un programma contenuto nella directory corrente, può darsi che Linux non lo lanci perché non lo trova.

Dietro questo comportamento apparentemente stravagante ci sono considerazioni di sicurezza molto importanti sulle quali non mi dilungo. L'importante, per ora, è sapere come scavalcare questa limitazione se si presenta: basta digitare "./" (punto e slash) prima del nome del comando o programma da lanciare dalla directory corrente.

Quando serve aiuto: man

Molto presto le brevi e superficiali spiegazioni di questo capitolo non vi basteranno più. Se volete saperne di più su un comando o una qualsiasi funzione di Linux, la prima risorsa da consultare è la cosiddetta *pagina man* corrispondente.

"Man" non è, come potreste pensare, l'inglese di *uomo*, per cui i linuxiani usano delle misteriose "pagine uomo"; è la contrazione di *manual*, nel senso di "manuale, guida". In altre parole, *pagina man* significa semplicemente "pagina del manuale di Linux".

E allora perché non chiamarla semplicemente "pagina del manuale"? Perché il comando che si usa per consultare le pagine del manuale è **man**. E siccome gli utenti di Linux non sono quei mostri di memoria che si dice in giro, consultano molto, molto spesso le pagine del manuale online di Linux. Usano così spesso il comando **man** seguito dall'argomento su cui vogliono informazioni che parlare di "pagine man" è diventata una consuetudine, anche per distinguere il manuale online da quelli cartacei.

Ad esempio, se volete saperne di più sul comando **cat**, digitate **man cat.**

Figura 12−1. Il risultato del comando *man cat*.

Potete anche chiedere informazioni sul comando *man* digitando **man man**. Anche i più importanti file del sistema operativo hanno una pagina man apposita: ad esempio, **man modules.conf** spiega tutti i parametri che potete includere nel file */etc/modules.conf*.

Quando usate il comando *man*, potete usare i tasti PgSu e PgGiù per far scorrere il testo una schermata alla volta e i tasti freccia verticali per farlo scorrere una riga alla volta. Una cosa che probabilmente troverete comoda è, nelle finestre di terminale, la possibilità di "allungare" verticalmente la finestra (cosa impossibile con le finestre DOS di Windows) per visualizzare più testo: potete anche allargarla orizzontalmente se ridimensionate la finestra prima di avviare il comando *man*.

Per uscire da *man* basta digitare **q**.

Molte di queste pagine man sono in inglese, ma è possibile che ne sia stata preparata una versione italiana non presente nella vostra distribuzione di Linux. Nel Capitolo 22 (*Risorse*) trovate l'indirizzo Internet da consultare per ottenere la versione italiana più recente delle pagine man. Nella pagina man del comando *man*, inoltre, trovate le istruzioni su come installare le pagine man italiane al posto di quelle inglesi.

Panico: Linux grafico non parte

L'avete fatta grossa: siete entrati come *root*, avete modificato malamente qualche file di configurazione del sistema e adesso l'interfaccia grafica non funziona più o addirittura gran parte di Linux è completamente inservibile.

Non è uno scenario improbabile: è capitato a tutti almeno una volta, e se non vi è ancora capitato, vi capiterà. Sto facendo il menagramo? No, per carità: semplicemente conosco la natura umana. Se siete interessati a Linux, siete interessati anche a smanettarlo, e questo prima o poi vi porterà a commettere qualche errore. Ecco come rimediare.

Riavviate il computer, facendo reset se necessario, e all'avvio, quando compare la schermata grafica in cui scegliete fra Linux e DOS (cioè Windows), premete Ctrl−X. Compare la richiesta *boot*, alla quale rispondete digitando **linux 3**. Questo fa partire Linux senza avviare l'interfaccia grafica. Se conoscete i comandi descritti in questo capitolo, potete così accedere ai file che avete modificato e ripararli, ad esempio usando un editor di testi come **pico**, che si richiama digitandone il nome sulla riga di comando.

Una volta risolto il problema, potete avviare l'interfaccia grafica digitando **startx**. Quando la avviate in questo modo, non vi chiede di fare login: l'avete già fatto nella console, e l'interfaccia grafica si avvia usando il nome e la password che avete dato al login e acquisisce i privilegi che ha l'utente con il quale avete fatto login. In altre parole, se ad esempio avete

avviato Linux come *root*, digitando **startx** l'interfaccia grafica parte a nome di *root* e con i suoi privilegi.

Lanciare l'interfaccia grafica in questo modo ha anche un altro effetto interessante. Quando la chiudete, non compare la schermata che vi propone di riavviare il server X: tornate direttamente alla riga di comando, senza troppi salamelecchi. Questo vi consente di entrare e uscire rapidamente dall'interfaccia grafica, e in effetti è il metodo usato dai veri linuxiani, che non avviano automaticamente la grafica ma lo fanno solo al momento del bisogno.

Ancora panico: avete dimenticato la password di root

Come è possibile dimenticarsi una cosa così fondamentale come la password di root? Vi assicuro, per esperienza personale, che è facilissimo. Tipicamente capita durante una reinstallazione frettolosa o al ritorno dalle vacanze. Basta passare un po' di tempo lontano dal computer e le password scivolano via dalla memoria con disinvoltura imbarazzante.

Il rimedio è semplice: leggete il Capitolo 18 (*Sicurezza*), che spiega come scavalcare la password di root (ebbene sì, si può).

Trasformazione!

Per cambiare identità, ad esempio per passare dall'utente *cesira* all'utente *mario*, non occorre eseguire tutta la procedura di chiusura e di accesso: si usa il comando **su** seguito dal nome dell'utente.

Ad esempio, supponete di aver fatto login come *cesira*, che è il nome di login di un utente normale non privilegiato. La finestra di terminale o la console vi presenterà un prompt di questo tipo:

```
[cesira@deepspace9 cesira] $
```

Se ora digitate **su** e premete Invio, vi viene chiesta la password di *root* (se *su* non è seguito dal nome di un utente, Linux presume che vogliate diventare *root*). Immettendo la password, diventate *root* a tutti gli effetti e con tutti i suoi poteri, e il prompt cambia di conseguenza:

```
[root@deepspace9 cesira] #
```

Notate due dettagli importanti: la directory corrente non cambia (siete ancora nella home directory dell'utente *cesira*), e al posto del simbolo di dollaro c'è il cancelletto, che vi ricorda che siete *root*. Questo è importante per motivi di sicurezza: è un promemoria per evitarvi di lasciare aperto un accesso *root*.

Potete usare il comando *su* anche per cambiare da un utente normale a un altro utente normale. Ad esempio, se l'utente *cesira* vuole assumere l'identità di *dilbert*, digita **su dilbert** e immette la password dell'utente *dilbert***.** È necessario conoscere la password dell'utente per evitare che un utente faccia disastri spacciandosi per qualcun altro o sbirci nel lavoro di un altro utente.

Se siete *root*, invece, siete onnipotenti. Non vi verrà chiesta la password dell'utente di cui volete assumere l'identità: la assumerete, punto e basta.

Per tornare all'utente originale, digitate **exit**. Mi raccomando, se sovrapponete più sessioni di **su** (ad esempio diventate prima *root*, poi usate ancora **su** per diventare utente normale), digitate un numero di **exit** corrispondente al numero di sessioni di **su**. Non digitatene troppi, altrimenti chiuderete la finestra di terminale o, se siete in una console, dovrete rifare login.

Dove siete?

Anche se il prompt contiene il nome della directory corrente, capita abbastanza spesso di non sapere esattamente dove si trovi questa directory. Ad esempio, in Linux ci sono due directory *sbin*: una è */usr/sbin* e l'altra è */sbin*. In entrambi i casi, il prompt visualizza soltanto *sbin*.

Come sapere in quale delle due directory vi trovate? Basta usare il comando **pwd**. Digitandolo, Linux vi risponderà con il percorso completo che porta alla directory corrente. Il comando prende il nome dalle iniziali di *print working directory*, cioè "stampa la directory di lavoro".

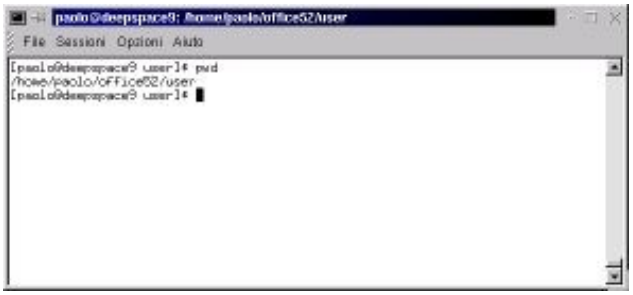

Figura 12−2. Usare il comando *pwd* per sapere dove siete.

Se siete stati attenti, avrete notato che questo problema di orientamento esiste soltanto per le console. Nelle finestre di terminale, infatti, il percorso corrente è indicato chiaro e tondo nella barra del titolo.

Trovare un file

La struttura di directory di Linux è obiettivamente piuttosto complessa, ed è facile dimenticarsi dove si trova un determinato file. Per trovarlo rapidamente dovunque sia nel computer, potete usare due comandi: **find** e **locate**.

Primo metodo: find

Usate il comando **find** seguito dalla directory dalla quale volete iniziare la ricerca, dalla specificazione *−name* e dal nome del file da ricercare o una sua approssimazione che usi i caratteri jolly (l'asterisco e il punto interrogativo). Troverete che *find* è decisamente più veloce della versione grafica di questo comando, cioè *kfind*.

Per esempio, supponete di dover trovare il programma *Xconfigurator*. Si tratta di un file importante, che serve per configurare l'interfaccia grafica, per cui può capitare spesso di averne bisogno. Per trovarlo potete digitare:

find / −name Xconfigurator

Dopo una breve attesa, otterrete la risposta:

```
/usr/X11R6/bin/Xconfigurator
/usr/X11R6/share/Xconfigurator
```

In altre parole, nel computer ci sono due file di nome *Xconfigurator*: uno si trova in */usr/X11R6/bin/* e l'altro si trova in */usr/X11R6/share/*.

C'è un problema: specificando il carattere *slash* dopo il comando *find*, la ricerca esplora l'intera struttura di directory

partendo dal livello più alto (la root directory). Ovviamente un'esplorazione così approfondita può richiedere un bel po' di tempo: se una ricerca dura troppo a lungo, potete interromperla digitando **Ctrl−C**.

Se avete un'idea di dove si trovi il file, potete specificare un percorso al posto dello slash: in questo modo *find* limita la ricerca alle directory e sottodirectory di quel percorso, e il tempo necessario si riduce massicciamente. Ad esempio:

find /usr/ −name Xconfigurator

ricerca il file *Xconfigurator* soltanto nella directory */usr/* e in qualsiasi sua sottodirectory.

Se omettete il percorso, *find* cerca soltanto nella directory corrente e nelle sue sottodirectory. Alcune configurazioni di Linux non "vedono" il contenuto della directory corrente per motivi di sicurezza e quindi è possibile che dobbiate digitare "./" per includere la directory corrente nell'area ricerca di *find*.

Fin qui tutto bene, a patto di sapere *esattamente* come si chiama il file, il che significa ricordarsi anche se è scritto in maiuscolo o in minuscolo. Per fortuna *find* accetta anche i caratteri jolly, per cui potete specificarli al posto del nome del file. Ad esempio:

find /usr/ −name Xconf*

trova tutti i file che sono presenti nella directory */usr/* o in una sua sottodirectory e il cui nome inizia per *Xconf.* Specificando un punto interrogativo, invece, *find* trova qualsiasi file il cui nome ha un qualsiasi carattere (uno solo) al posto del punto interrogativo. Ad esempio:

find /usr/ −name ?configurator

trova qualsiasi file, nella directory */usr/* e nelle sue sottodirectory, il cui nome inizia con qualsiasi lettera o cifra e prosegue con *configurator*. Questo è un buon sistema da usare quando non ricordate se il nome del file contiene delle lettere maiuscole.

Ma si può fare di meglio. Infatti se specificate *iname* al posto di *name*, il comando *find* non farà distinzione tra maiuscole e minuscole, per cui:

find /usr/ −iname xconfigurator

troverà il file *Xconfigurator* anche se il suo nome ha l'iniziale maiuscola.

Secondo metodo: locate

Se la ricerca è a largo raggio, ad esempio perché include tutta la struttura di directory del computer, ci possono volere alcuni minuti prima che *find* finisca di spazzolarsi tutti i dischi a caccia del file che gli avete chiesto. In casi come questo si usa il comando **locate** seguito dal nome del file.

Ad esempio, per trovare il solito file *Xconfigurator* digitate **locate Xconfigurator**.

Questo comando è velocissimo perché invece di leggersi tutti i dischi legge un database contenente i nomi e le ubicazioni di tutti i file. Tuttavia ha un difetto: il database deve essere aggiornato, altrimenti non rispecchia la situazione reale e allora addio ricerca. Per aggiornare il database si usa il comando **updatedb** (bisogna essere *root* per eseguirlo), che richiede un po' di tempo per l'esecuzione.

Morale della favola: se state cercando file che sono presenti sul disco da tempo, usate *locate*, perché li troverà senza dover prima eseguire l'aggiornamento del database con *updatedb*. Se invece dovete trovare file creati di recente, o prima dell'ultimo aggiornamento del database, usate *find*.

Cambiare la password

Può capitare che la password di un utente venga scoperta da qualcuno. In genere questo avviene per sbadataggine dell'utente, tipicamente perché non centra il tasto Invio alla richiesta di login e quindi la sua password viene digitata in chiaro sullo schermo e dietro di lui c'è qualcuno che sbircia. Anche se l'utente non commette errori, è abbastanza facile carpire la sua password guardando la tastiera mentre la digita, soprattutto se la password è corta.

Come regola generale, insomma, cambiate la vostra password ogni volta che avete il dubbio che qualcuno l'abbia vista, e comunque cambiatela lo stesso periodicamente. Il comando da digitare è **passwd**.

Alla risposta *Changing password for*, seguita dal nome dell'utente e dalla richiesta *(current) UNIX password*, rispondete con la password attuale. Alla richiesta *New UNIX password*, rispondete con la nuova password. Infine, alla richiesta *Retype new UNIX password*, rispondete digitando di nuovo la password. Se Linux risponde con *passwd: all authentication tokens updated successfully*, ce l'avete fatta.

Cambiare la password può essere meno facile di quel che sembra, soprattutto nel caso degli utenti normali. Linux è molto schizzinoso in fatto di password. Ad esempio, rifiuterà la nuova password:

- se è troppo simile a quella vecchia;
- se è troppo corta;
- se è basata su una parola del dizionario interno di Linux, che considera anche le parole scritte a rovescio;
- se coincide con il nome dell'utente;
- se gli gira (giuro). Per essere più precisi, Linux memorizza in un file le password già usate e adopera delle complesse regole per decidere se la password nuova che proponete è troppo simile ad una password già usata.

Il superutente *root* non è soggetto a queste limitazioni, peraltro dettate dal buonsenso informatico, e può scegliersi la password che vuole. Non solo: non gli viene chiesta neppure quella vecchia. Per questo motivo *non dovete mai lasciare aperta una sessione di root e allontanarvi dal computer*. Chiunque può passare di lì e digitare semplicemente **passwd**, scegliersi una nuova password e diventare padrone del vostro computer, chiudendovi fuori. È come farsi rubare le chiavi di casa, insomma.

Trovate maggiori dettagli su come scegliersi una buona password nel Capitolo 18 (*Sicurezza*).

Il superutente *root*, essendo amministratore, può inoltre cambiare la password agli altri utenti: il comando è **passwd** seguito dal nome dell'utente. Non gli viene chiesta la vecchia password: questo significa che l'utente normale può cambiarsi la password quante volte gli pare, ma l'amministratore di sistema può a sua volta cambiargliela senza aver bisogno di chiedergli quella attuale. Mettiamo bene in chiaro chi comanda, insomma.

Andare in giro per il computer

Inevitabilmente prima o poi avrete bisogno di passare da una directory a un'altra, soprattutto per evitare di digitare ogni volta il percorso completo che porta a un file. Cambiare directory è semplice: si digita **cd** seguito dal nome della directory di destinazione, preceduto se necessario dal suo percorso.

Ad esempio, supponiamo che siate nella directory */home/cesira*. Se volete andare nella directory */home/cesira/Desktop*, cioè in una directory che sta sotto quella corrente, basta digitare **cd Desktop/**.

Per salire di un livello, ad esempio per passare dalla directory */home/cesira/Desktop/* alla directory */home/cesira/*, vi basta digitare **cd ..** (attenzione: è "cd" seguito da uno spazio e da due punti; non confondetevi con il comando **CD** del DOS, che accetta anche la forma senza spazio).

Se invece volete andare nella directory */usr/bin/*, cioè in una directory situata lungo un altro ramo della struttura di directory, dovete digitare anche il percorso: *cd /usr/bin/*.

Altra finezza: dovunque siate, se digitate **cd** e basta venite teletrasportati direttamente alla vostra home directory.

Mi raccomando, non pensate di poter scorrazzare liberamente per il disco come fate in Windows. Se state lavorando come utente non privilegiato e cercate di passare a una directory per la quale non avete diritto d'accesso, Linux vi risponderà con un bel *Permesso negato*. Soltanto *root* ha il diritto di esplorare tutto il computer.

Elencare i file di una directory

In Linux ci sono due comandi di elencazione dei file: uno è **dir**, che vi sarà familiare se avete usato il DOS o la finestra DOS di Windows. L'altro, più diffuso, è **ls**, ed è questo il comando che vi presento qui.

Per elencare il contenuto della directory corrente è sufficiente digitare **ls**. Se volete includere anche i file nascosti (cioè quelli che iniziano con il punto), aggiungete il parametro **−a**.

Se volete elencare i file di una directory diversa da quella corrente, digitate **ls** seguito dal percorso della directory che vi interessa. Ad esempio, **ls /tmp** elenca tutti i file (non nascosti) contenuti nella directory */tmp*. Se volete elencare soltanto i file che hanno un nome specifico, indicate il nome alla fine del comando, come in **ls /tmp/foto*** per elencare tutti i file contenuti nella directory */tmp/* il cui nome inizia per *foto*.

Questo formato del comando, però, elenca soltanto i nomi dei file. Se volete maggiori dettagli, aggiungete il parametro **−l**, che elenca anche la data di ultimo aggiornamento dei file e lo spazio complessivo occupato dai file elencati.

Fra l'altro, il comando **ls −l**, se non specificate un nome di file da elencare, elenca tutti i file presenti nella directory e vi dice quanto spazio (espresso in kilobyte) occupano nella riga che inizia con *totale*.

Un'altra opzione molto comoda è **−−color** (sì, con due trattini): attiva la visualizzazione a colori dei nomi dei file e delle directory. Le directory sono elencate in blu, i file in bianco, i file di testo in bianco (in nero nelle finestre di terminale) e i file non di testo (binari) sono elencati in fucsia. Di solito quest'opzione è attivata automaticamente.

Come per tutti gli altri comandi di Linux, anche qui valgono i permessi. Un utente normale non può elencare i file contenuti nella home directory di un altro utente normale: solo *root* vede tutto e può tutto.

Copiare un file

Il comando per *copiare* un file o una directory, cioè scriverne una copia in un altro posto (o con un altro nome) senza però cancellare l'originale, è **cp**. Specificate prima il nome del file di partenza e poi quello di destinazione.

Supponiamo che l'utente *cesira* abbia nella propria home directory (*/home/cesira/*) il file *foto_fidanzato.jpg* e ne voglia creare una copia di nome *copia_foto_fidanzato.jpg* (si vede che è un fidanzato che le piace proprio tanto): in tal caso il comando diventa:

cp foto_fidanzato.jpg copia_foto_fidanzato.jpg.

Se il file di partenza e/o quello di destinazione non sono nella directory corrente, si specifica il loro percorso. Considerate l'esempio precedente, ma supponete che l'utente *cesira* voglia creare una copia del suo file nella directory */tmp*. Il comando diventa allora:

cp foto_fidanzato.jpg /tmp/foto_fidanzato.jpg

Potete usare i caratteri jolly per specificare un insieme di file, e potete cambiare a vostro piacimento il nome del file di destinazione. Al posto del nome di destinazione potete anche specificare un nome di directory: in questo caso i file verranno copiati dentro la directory che avete specificato.

Una delle stranezze di Linux rispetto al mondo Windows è la gestione delle date dei file. Infatti in Windows, quando copiate un file, la sua data di ultima modifica rimane invariata. In Linux no: la copia prende la data e l'ora in cui è stata creata. Questo può scombussolare non poco, soprattutto se state creando delle copie di sicurezza e volete sapere qual è la più recente. Se volete che Linux mantenga uguali le date nel file di partenza e nel file di destinazione, specificate il parametro **−p**.

Ad esempio, se l'utente *cesira* vuole creare una copia del file *foto_fidanzato.jpg* e chiamarla *copia_foto_fidanzato.jpg* nella medesima directory, ma vuole che la data della copia rimanga uguale a quella dell'originale, deve digitare **cp −p foto_fidanzato.jpg copia_foto_fidanzato.jpg**.

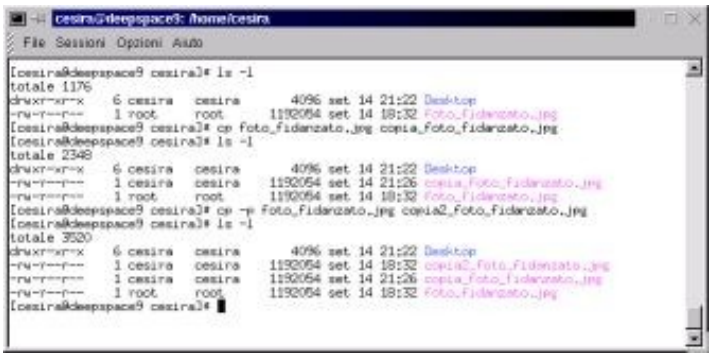

Figura 12−3. Copia di file con e senza il parametro *−p* che preserva le date.

Spostare un file

Se sapete copiare un file, sapete quasi tutto quello che occorre sapere per spostarlo da un punto a un altro del computer. La struttura del comando **mv** che si usa per spostare i file è infatti molto simile a quella del comando **cp**.

Ad esempio, per spostare il file *pippo.txt* dalla directory corrente alla sottodirectory */Desktop* si usa questo comando:

mv pippo.txt Desktop/

Come in **cp**, potete specificare, se necessario, il percorso dell'attuale ubicazione del file da spostare. Anche qui valgono le solite limitazioni derivanti dai permessi: se siete un utente normale, non potete spostare file da e verso directory sulle quali non avete diritti di scrittura.

Con **mv** potete spostare sia i file, sia le directory. A differenza di **cp**, **mv** non cambia le date dei file.

Rinominare un file

Per cambiare il nome a un file o a una directory non c'è un comando apposito: si usa ancora il comando **mv**, specificando il nome vecchio e il nome nuovo. Ad esempio:

mv foto_fidanzato.jpg foto_ex_fidanzato.jpg

rinomina il file *foto_fidanzato.jpg* assegnandogli il nome *foto_ex_fidanzato.jpg*.

Cancellare un file

C'è poco da raccontare in proposito: il comando è **rm** seguito dal nome del file da cancellare. Come al solito, non potete cancellare file per i quali non avete i permessi appropriati, e potete specificare il percorso se il file non si trova nella directory corrente.

Una particolarità degna di nota è che **rm**, quando vi chiede di confermare o meno la richiesta di cancellare un file, accetta sia la *Y* di *Yes*, sia la *S* di *Sì*. Inoltre se specificate **−r** dopo **rm**, potete usare questo comando per cancellare una directory e tutte le sue sottodirectory e i file contenuti.

Se dovete eseguire cancellazioni di massa, inoltre, vi conviene usare il parametro **−f**, che evita la richiesta di conferma per ogni singolo file e procede direttamente alla cancellazione. Assicuratevi di sapere esattamente quello che state facendo!

Visualizzare un file di testo semplice

Questa è senz'altro un'attività molto frequente, visto il numero di file di questo tipo presenti in qualsiasi installazione di Linux, per cui il sistema operativo è dotato di più di un comando per gestirla. I due principali sono **cat** e **less**. Cominciamo da **cat**, che equivale al comando **type** sotto DOS: **cat** seguito dal nome del file visualizza il contenuto del file. Se il file è troppo lungo per essere contenuto nella schermata, la visualizzazione scorre fino alla fine del file.

Ovviamente questo non è molto utile se vi interessa leggere tutto il file: va bene soltanto se dovete visualizzare rapidamente un file molto breve. Se volete sfogliare per bene un file più grande, usate **less**, che ha una miriade di opzioni (leggete bene la sua pagina man) ma ha un uso di base molto semplice: digitate **less** seguito dal nome del file e usate i tasti **PgSu** e **PgGiù** per far scorrere il file. Quando avete finito, digitate **q** per terminare.

Gestire le directory

Per creare una directory si usa il comando **mkdir** seguito dal nome della directory da creare; per cancellare una directory esistente, si usa invece **rmdir** seguito dal nome della directory da cancellare. Attenzione, però: la directory da cancellare deve essere vuota. Se non lo è, usate il comando **rm** con il parametro **−r** seguito dal nome della directory.

Per cambiare nome a una directory, invece, si usa il comando **mv** che avete già visto.

Spazio libero

Per sapere quanto spazio libero avete sul disco rigido, digitate **df**. Per essere precisi, questo comando elenca sia lo spazio libero sia quello utilizzato, e lo fa non soltanto per il disco rigido, ma per tutti i filesystem visibili a Linux. Ad esempio, in Figura 12−4 la risposta del comando **df** elenca la situazione di ben tre partizioni: */dev/hda1*, che è la partizione Windows; */dev/hda2*, che è la partizione Linux e risiede sullo stesso disco rigido che ospita la partizione Windows; e */dev/hdb1*, che

è la partizione unica di un secondo disco rigido.

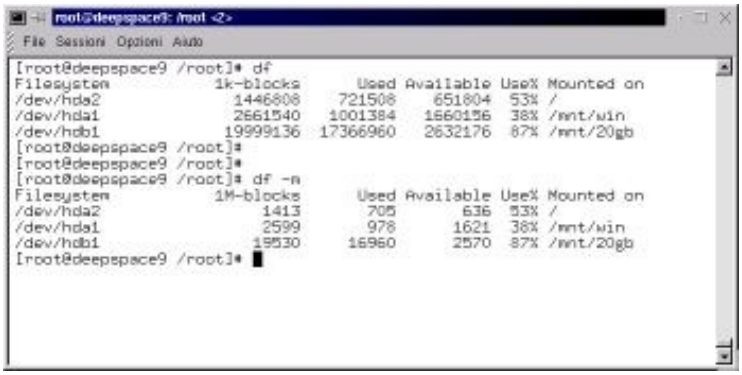

Figura 12−4. Spazio libero su disco, in kilobyte e in megabyte.

Nel suo formato standard, **df** elenca il numero di blocchi da 1024 byte. Tuttavia, coi tempi che corrono, è probabile che troviate più comprensibile il formato che usa i megabyte come unità di misura: come mostrato in Figura 12−4, il comando per ottenere questo formato è **df −m**.

Tenere a bada i programmi

Uno dei motivi principali per cui si ricorre alla console è l'eliminazione dei programmi che sono andati in tilt. Linux è stabile, ma non è detto che lo siano tutte le applicazioni che gli girano sotto. Quello che sto per descrivere è molto simile a quello che succede in Windows premendo Ctrl−Alt−Canc: compare una finestra che elenca le applicazioni attive e consente di sceglierle e terminarle individualmente, in modo da riportare Windows al funzionamento normale.

Quando il computer sembra rallentare o bloccarsi, il primo passo per tornare alla normalità è scoprire quale applicazione sta intasando la macchina. Il comando per scoprirlo è **top**, che elenca i programmi (tecnicamente si chiamano *processi*) in esecuzione. Lo stesso comando visualizza anche il consumo di memoria e di swapfile e varie altre statistiche di utilizzo. Come noterete, offre molti più dettagli della schermata corrispondente in Windows.

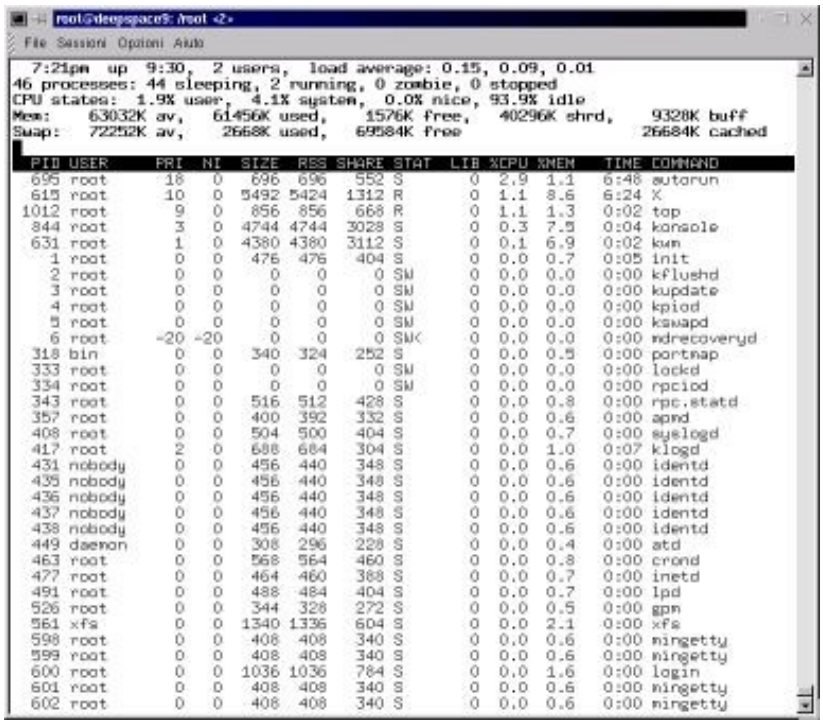

Figura 12−5. Il comando *top* elenca la miriade di processi attivi.

In caso di applicazioni in crisi, la colonna più importante è quella dell'utilizzo del processore (*%CPU*), seguita dalla colonna che indica il consumo di memoria. Un programma che ha un consumo smodato di queste due risorse mentre la macchina è apparentemente inattiva è molto probabilmente bloccato ed è quindi candidato al patibolo digitale.

Per interrompere brutalmente ma non troppo l'esecuzione di un programma, dandogli una piccola possibilità di salvare i file che sta manipolando, si usa il comando **kill** seguito dal nome del programma (desunto dalla colonna *Command* della schermata di **top**). Se **kill** da solo non riesce a interrompere il programma, digitate **kill −9** al posto del normale **kill**. Questo fermerà immediatamente il programma recalcitrante, lasciando cocci dappertutto.

Per motivi abbastanza evidenti, un utente normale non può cancellare l'esecuzione di programmi che ha lanciato un altro utente. Il compito di poliziotto della CPU è riservato all'utente *root*.

Proprietari dei file

Ciascun file ha uno o più proprietari (*owner*) i cui nomi si scoprono usando il comando **ls −l**. Ad esempio, un'elencazione di questo tipo:

```
−rw−r−−r−− 1 cesira cesira 1155894 set 16 00:45 foto_fidanzato.jpg
```

indica che il proprietario è l'utente *cesira*: il nome è ripetuto due volte perché la prima indica l'utente proprietario, il secondo indica il *gruppo* di utenti che ne condivide la proprietà. È un'informazione importante, perché il proprietario di un file ha sempre dei diritti (o *permessi*, come vedremo tra poco) maggiori rispetto agli altri utenti. Ad esempio, un file può essere modificabile soltanto dal suo proprietario e/o dal gruppo specificato e gli altri lo possono leggere e basta. Ovviamente *root* può sempre fare tutto quel che gli pare.

L'indicazione del proprietario di ciascun file ha anche lo scopo di facilitare l'amministrazione del sistema quando ci sono tanti utenti. Chi ha creato quell'enorme file che ha intasato il disco rigido? È facile scoprirlo: il file è firmato con il nome

del suo creatore.

Capita spesso di dover cambiare il proprietario di un file. Ad esempio, un file creato da *root* può servire a un utente non privilegiato che lo deve modificare, ma se il proprietario del file rimane *root*, l'utente non privilegiato non può modificarlo. Allora *root* cambia il proprietario del file e ne assegna la proprietà all'utente non privilegiato, che può così lavorarci senza ulteriori problemi.

Un altro caso tipico è quando *root* installa un programma e desidera che l'utente normale possa eseguirlo: *root* può assegnarne la proprietà all'utente normale.

Ovviamente questo meccanismo ha delle limitazioni per motivi di sicurezza. Un utente comune, non privilegiato, può cambiare soltanto le proprietà dei propri file ma non quelle degli altri. Certamente non può prendere un file che appartiene a *root* e intestarselo, né può prendere uno dei propri file e intestarlo a *root*.

L'utente normale può assegnare la proprietà dei propri file soltanto ad utenti normali. È grazie a questa protezione che i file di sistema di Linux sono protetti dagli errori (e dalle incursioni intenzionali) degli utenti comuni: sono tutti intestati a *root*.

Per cambiare proprietario a un file si usa il comando **chown**, seguito dal nome del nuovo proprietario e dal nome del file. Tornando all'esempio della foto del fidanzato di Cesira, *root* può assegnare il file a un altro utente (diciamo *dilbert*) digitando **chown dilbert foto_fidanzato.jpg**. Così facendo, le proprietà del file diventeranno le seguenti:

```
-rw-r--r-- 1 dilbert cesira 1155894 set 16 00:45 foto_fidanzato.jpg
```

Lo stesso comando può cambiare anche il gruppo che è proprietario del file: basta specificare, oltre al nome dell'utente, anche il nome del gruppo, separandoli con il segno di "due punti". Ad esempio, **chown dilbert:utenti_sfigati foto_fidanzato.jpg** assegna la proprietà del file all'utente *dilbert* e al gruppo *utenti_sfigati*.

Se volete modificare soltanto il gruppo proprietario, potete usare il comando **chgrp**. Ad esempio, **chgrp utenti_sfigati foto_fidanzato.jpg** mantiene l'utente proprietario originale ma cambia il gruppo proprietario in *utenti_sfigati*.

Permessi

In Linux, file e directory sono contrassegnati da *permessi* che definiscono chi può leggerli, chi può modificarli e chi può eseguirli. Quando elencate i file con il comando **ls**, o con kfm nell'interfaccia grafica, avrete notato quella strana serie di lettere apparentemente senza senso, tipo *drwxr-xr-x*. Non è marziano: è una forma molto, molto concisa per indicare la natura e i permessi di un file.

Il primo carattere definisce il tipo di file: ad esempio, *d* indica che si tratta di una directory (in Linux, anche le directory sono considerate come file); il trattino indica che si tratta di un file normale; *l* segnala che il file è un link. Ci sono anche altre possibilità, ma sorvoliamo.

I nove caratteri rimanenti sono suddivisi idealmente in blocchi di tre: il primo blocco indica i permessi che ha l'utente proprietario del file; il secondo indica quelli che ha il gruppo al quale appartiene l'utente; il terzo indica quelli che hanno tutti gli altri utenti.

I caratteri di ciascun blocco indicano il tipo di permesso concesso agli utenti definiti da quel blocco: *r* indica il permesso di lettura (dall'inglese *read*); *w* indica il permesso di scrittura (da *write*); *x* segnala il permesso di esecuzione. Il trattino indica "nessun permesso".

Complicato, vero? Chiarisco con qualche esempio.

Immaginate di avere un file che **ls −l** elenca in questo modo:

```
−rw−r−−r−− 1 cesira cesira 1155894 set 16 00:45 foto_fidanzato.jpg
```

Il primo carattere (trattino) indica che si tratta di un file normale, e fin qui niente di difficile. La prima terna di caratteri (*rw−*) specifica i permessi concessi all'utente proprietario (*cesira*): dato che sono presenti la R e la W, vuol dire che il file può essere letto e scritto dall'utente.

La seconda terna di caratteri (*r−−*) specifica invece cosa è permesso fare agli utenti del gruppo (che in questo caso è omonimo di *cesira*): siccome è specificata soltanto la R, gli utenti del gruppo possono soltanto leggere il file ma non modificarlo.

L'ultima terna di caratteri è uguale alla seconda, e indica i permessi concessi agli utenti che non fanno parte del gruppo: in questo caso l'unico permesso concesso è la lettura.

Non è un sistema molto intuitivo, lo so. L'unica cosa che vi posso consigliare è esercitarvi a decifrare i permessi elencati da **ls −l** il più possibile.

Dopo cotanta premessa, finalmente vengo al sodo. Il comando per modificare i permessi è **chmod**. La sua sintassi è complicata quanto il meccanismo che ho appena descritto, ma cercherò di semplificarne la presentazione. Non è un comando che userete in continuazione, per cui non vergognatevi se vi trovate a consultare più volte queste pagine.

Ecco la sequenza delle parti che compongono il comando:

- La parola **chmod**.
- Una lettera che specifica per chi volete modificare i permessi: se volete cambiarli per l'utente, la lettera è **u**; se volete modificarli per il gruppo, è **g**; se la modifica deve avere effetto sugli altri utenti, è **o**. Se volete applicare la modifica contemporaneamente a utente, gruppo e altri utenti, la lettera è **a**.
- Un segno "+" o "−", a seconda che vogliate aggiungere o togliere un permesso.
- Una lettera che specifica il tipo di permesso che volete aggiungere o togliere: *r* per la lettura, *w* per la scrittura, *x* per l'esecuzione. Ce ne sono anche altre, ma non è necessario approfondirle qui.
- Il file a cui devono applicarsi tutte queste modifiche.

Faccio qualche esempio su un ipotetico file *pippo.txt*:

- Per togliere all'utente proprietario il diritto di modificare il file (e quindi concedere questo diritto solo a *root*), digitate **chmod u−w pippo.txt**. Notate la mancanza di spazi fra la U e il parametro **−w**.
- Per ridare il diritto di modifica all'utente proprietario, il comando diventa **chmod u+w pippo.txt**.
- Per togliere agli altri utenti il diritto di lettura, per cui solo l'utente proprietario e gli utenti del gruppo proprietario possono leggere il file, digitate **chmod o−r pippo.txt**.
- Per togliere a tutti quanti il diritto di lettura, per cui soltanto *root* può avere accesso al file, digitate **chmod a−r pippo.txt**.

Esiste anche un'altra sintassi del comando **chmod** che si basa su codici numerici, ma ve la risparmio. È in effetti molto efficiente se sapete a memoria i codici numerici che si applicano ai casi più ricorrenti, ma per carità, in un libro introduttivo come questo sarebbero fuori posto. Per queste cose c'è la relativa pagina man.

Per gli amanti della grafica

La storia dei permessi e dei proprietari è piuttosto complicata, vero? Se pensate che una versione grafica di questi comandi possa rendervi le cose un po' più chiare, nell'interfaccia grafica, e specificamente in kfm, potete cliccare con il pulsante destro sul nome di un file. Otterrete un menu: scegliendo *Proprietà* e la scheda *Permessi* potrete fare graficamente quasi tutte le cose descritte qui.

Caro vecchio DOS: Mtools

Se avete dimestichezza con il DOS, sia come sistema operativo autonomo, sia come finestra all'interno di Windows, vi farà piacere sapere che Linux capisce buona parte dei comandi DOS usati per accedere a dischi e dischetti formattati dal DOS. Basta prefissarli con la lettera M, e non è necessario eseguire prima il mount. Collettivamente, questi pseudo−comandi DOS sono conosciuti come *mtools*.

Ad esempio:

- **mdir a:** elenca i file contenuti nel floppy inserito nel drive che in DOS/Windows sarebbe etichettato dalla lettera A:;
- **mcopy a:pippo.jpg** copia il file *pippo.jpg* dal dischetto in A: alla directory corrente;
- **mformat a:** formatta in formato DOS il dischetto contenuto nel drive A:;
- **mren a:pippo.jpg pluto.jpg** rinomina il file *pippo.jpg* sul dischetto in A: e gli assegna il nuovo nome *pluto.jpg*;
- **mdel a:pippo.jpg** cancella il file *pippo.jpg* dal dischetto in A:.

L'elenco completo dei comandi DOS "emulati" da Linux è nella pagina man *mtools*.

Il funzionamento di questi comandi DOS in Linux richiede un minimo di configurazione del file */etc/mtools.conf*. Di norma, questo file è già impostato in modo da consentire l'accesso ai drive per floppy (A: e B:). Per attivare anche l'accesso alla partizione Windows del computer, aggiungete una riga con questa istruzione:

drive c: file="/dev/hda1"

Lo stesso sistema si può applicare ad altre partizioni o altri dischi rigidi eventualmente presenti nel computer: ad esempio, un secondo disco rigido formattato da Windows può essere reso visibile agli mtools sostituendo "c:" con "d:" e "/dev/hda1" con "/dev/hdb1", e così via.

Sostenete *Da Windows a Linux*!

Questo libro è distribuito **gratuitamente**, ma le **donazioni** sono sempre ben accette, sia tramite **PayPal,** sia tramite il collaudato sistema della **banconota in una busta**. Se volete dettagli e istruzioni su come procedere, le trovate presso **http://www.attivissimo.net/donazioni/donazioni.htm**.

Grazie!