McGraw Hill

Guida completa

Il punto di riferimento per ogni esigenza

Stephen Coffin

UNIX System V Release 4

LAGUNA COMPLETA

La guida completa

Stephen Coffin

UNIX System V

Release 4

McGraw-Hill Libri Italia srl

Milano - New York - St. Louis - San Francisco - Oklahoma City - Auckland - Bogotà - Caracas - Hamburg - Lisboa - London - Madrid - Montreal - New Delhi - Paris - San Juan São Paulo - Singapore - Sydney - Tokyo - Toronto

Titolo originale: *UNIX System V Release 4 - The Complete Reference*

Realizzazione editoriale:
EDIGEO srl, via del Lauro 3, 20121 Milano
Traduzione: Roberto Pierini
Copertina: Achilli & Piazza e Associati
Composizione e stampa: Legoprint s.r.l., stabilimento di Trento

ISBN 88-386-0181-X

1ª edizione giugno 1991
3ª ristampa novembre 1994

Printed in Italy
4567890LGPLGP90654

Indice

Introduzione

Benevenuti nel grande mondo del sistema operativo UNIX. Se avete esperienza di altri sistemi operativi, resterete sorpresi dalla vastità, diversità e ricchezza dell'ambiente offerto dal sistema UNIX. Se non avete precedente esperienza, entrerete facilmente nel mondo di UNIX senza preconcetti.

Il sistema UNIX offre un ambiente talmente complesso che neanche un libro di queste dimensioni può coprire *completamente* tutte le sue caratteristiche e funzionalità. Non scoraggiatevi, il sistema UNIX e questo libro sono organizzati in maniera che i principianti possano iniziare con semplici operazioni e progredire fino a conoscere completamente il sistema. D'altra parte, se siete interessati ad approfondire l'argomento, oppure se volete sperimentare a fondo uno dei migliori sistemi operativi esistenti, troverete nel sistema UNIX abbondanti motivi di interesse. Neanche gli esperti conoscono tutto il sistema. In ogni caso, procuratevi un accesso a un sistema UNIX e fatene uso; in breve tempo l'uso del sistema UNIX vi apparirà naturale e tornerete con riluttanza ad altri ambienti.

Il libro

Guida Completa UNIX System V Release 4 provvede a una dettagliata copertura della più recente release del sistema operativo UNIX. Se avete accesso a un microcomputer basato su Intel 80386 o 80486, una workstation RISC ad alte prestazioni, un Apple Macintosh con AUX, potrete utilizzare questa guida con profitto.

Questo libro parte dai concetti e dalle idee di base del sistema operativo UNIX. I principianti senza alcuna esperienza di computer possono imparare a usare il sistema UNIX partendo dall'inizio, con i Capitoli 1, 3 e 4.

Il libro procede con i comandi a livello utente necessari per utilizzare il sistema UNIX. Potete usare il vostro sistema UNIX come un sistema completamente autonomo o potete connetterlo a una rete locale (Local Area Network) ed entrare in una più larga comunità di utenti.

Questo libro è stato scritto dopo un'ampia esperienza acquisita con la release standard della versione SVR4 di AT&T per macchine 80386 e 80486. Questa release è conosciuta come *porting base* per sistemi SVR4, poiché tutti gli sviluppatori di varianti SVR4 partono da questa base per l'adattamento al loro specifico hardware. Il materiale in questo libro è mirato a macchine 80386 e 80486, e in larga parte è corretto anche per qualsiasi altra macchina SVR4.

Dopo che avrete terminato questo libro dovreste essere in grado di usare il sistema UNIX su qualsiasi computer. Infine, il libro dovrebbe diventare una guida per rispondere ai problemi quotidiani che incontrerete nell'uso del sistema UNIX.

Organizzazione del libro

È conveniente leggere questo libro davanti a un terminale di un sistema UNIX, in modo da provare gli esempi e verificare i risultati. Potete personalizzare il sistema secondo le vostre necessità e sperimentare i comandi e le funzioni come vi suggerisce il vostro interesse. Potrete trovare alcune differenze nei risultati degli esempi sul vostro sistema, ma non preoccupatevi: qualsiasi sistema ha delle piccole varianti che non influenzano i principi di funzionamento.

Ogni capitolo inizia con la parte più semplice e di uso più generale dell'argomento del capitolo; potete diventare un utente competente di sistema UNIX leggendo solo la prima parte di ogni capitolo. Tuttavia, se volete maggiori informazioni, o se avete necessità di documentarvi ulteriormente in materia, vi sarà di aiuto la seconda parte dei capitoli. Ciascun capitolo termina con il paragrafo "Approfondimenti" che introduce aspetti più complessi o particolareggiati dell'argomento del capitolo. Mano a mano che la vostra conoscenza ed esperienza crescono, potete ritornare alla seconda parte di ciascun capitolo per acquisire informazioni addizionali che potevano non esservi necessarie per una prima conoscenza.

Ricordate che il sistema UNIX è un ambiente molto vasto, esistono infatti numerosi volumi che sviluppano gli argomenti di uno solo dei paragrafi "Approfondimenti". In generale, la prima metà del libro (fino al Capitolo 11) tratta argomenti tradizionali per gli utenti di sistemi UNIX, inclusi i più importanti strumenti e programmi di utilità; nella seconda metà del libro ciascun capitolo inizia con argomenti di interesse generale, quindi passa a trattare informazioni relative all'amministrazione e al funzionamento di un sistema UNIX. I capitoli finali trattano argomenti di sottosistemi speciali che possono non essere disponibili su tutti i sistemi UNIX.

Questo libro non può descrivere dettagliatamente tutti i comandi e neppure tutte le opzioni possibili per ciascun comando; consultate lo *UNIX User's Manual* per integrare le informazioni qui presentate con materiale più dettagliato e di più esatto riferimento.

Il Capitolo 1 contiene un'introduzione al sistema UNIX, una breve storia del sistema e una descrizione delle nuove caratteristiche introdotte con SVR4. Il Capitolo 2 ha una funzione didattica; insegna alcuni fondamenti del sistema, incluse le procedure per il login, leggere la posta e poche altre funzioni comuni. I Capitoli 3, 4, 7, 8 coprono argomenti considerati parte del "classico" sistema UNIX: file system, l'interprete dei comandi (shell) e alcuni dei più comuni comandi. Questo materiale costituisce il necessario bagaglio di conoscenze fondamentali per usare quotidianamente il sistema UNIX. I Capitoli 5 e 6 trattano gli editor standard di sistemi UNIX. Anche se conoscete programmi di word processing, la conoscenza dei concetti di base degli editor UNIX è importante per comprendere molti di questi comandi. Il Capitolo 9 introduce alla struttura della documentazione ufficiale del sistema UNIX. Neanche un libro completo come questo può sostituire la documentazione per cui è molto importante conoscere come utilizzarla. Il Capitolo 10 tratta degli strumenti di elaborazione numerica, utili a molti utenti. Il Capitolo 11 introduce i concetti fondamentali di multitasking in sistema UNIX e tratta gli strumenti per l'esecuzione di più applicazioni simultaneamente.

Il Capitolo 12 fornisce un'introduzione alla gestione del sistema (*system administration*): queste informazioni sono d'importanza critica se volete controllare il sistema UNIX; molte delle conoscenze realmente necessarie per gestire il sistema sono trattate nei capitoli successivi; gli ultimi capitoli introducono altri comandi e concetti chiave del sistema UNIX. Il Capitolo 13 tratta della stampa; descrive strumenti per la stampa di file e spiega anche come installare una stampante sulla vostra macchina. I Capitoli 14 e 15 descrivono gli strumenti classici di comunicazione in sistema UNIX, tra i quali la posta elettronica e il pacchetto **uucp**; questi capitoli trattano l'uso di questi importanti strumenti e il modo di configurarli secondo le vostre necessità.

Il Capitolo 16 tratta i due più potenti shell disponibili nei sistemi SVR4. Il Capitolo 17 tratta il pacchetto per word processing **troff**. Il Capitolo 18 affronta i problemi relativi all'uso dei dischi e nastri magnetici, tra cui il modo di gestire lo spazio di disco a vostra disposizione, il salvataggio e il ripristino dei dati, e l'installazione di nuovi dischi nel sistema. Il Capitolo 19 tratta l'uso dell'MS-DOS nei sistemi UNIX. Il Capitolo 20 descrive argomenti come la temporizzazione e la pianificazione delle attività nel sistema. Il Capitolo 21 tratta i procedimenti di inizializzazione (boot) e fermata (shut down) di una macchina UNIX; queste informazioni sono utili se possedete un sistema proprio, altrimenti aiutano a comprendere le attività di sistema non visibili all'utente. Il Capitolo 22 tratta un argomento che nell'ambiente odierno deve essere considerato importante: la sicurezza del sistema.

I Capitoli 23 e 24 trattano caratteristiche del sistema UNIX relativamente recenti, che possono non essere presenti su tutte le macchine. X Window System, trattato nel Capitolo 23, fornisce un ambiente con mouse e finestre per sistemi UNIX. Il Capitolo 24 descrive strumenti LAN e argomenti relativi alle reti. Il Capitolo 25 descrive il modo di assemblare il vostro sistema UNIX e di configurare il software di sistema per il vostro ambiente. Dovreste consultare subito il Capitolo 25 se state considerando l'acquisto di un computer per usare il sistema UNIX. Infine, il Capitolo 26 considera alcuni elementi addizionali per introdurvi nella comunità degli utenti UNIX e alcuni apprezzamenti extra che vanno oltre lo scopo primario di questo libro.

Quanto trattato nei Capitoli 17, 19, 23, 24 può non essere utile se la vostra macchina non dispone di hardware e software aggiuntivo affrontato da questi capitoli; d'altra parte molte macchine SVR4 dispongono di queste caratteristiche, che sono parti importanti del sistema UNIX.

Convenzioni usate

I concetti e le parole della terminologia del sistema UNIX sono in *corsivo* la prima volta che vengono introdotti, in seguito sono nel normale carattere del testo. Il carattere corsivo viene usato anche per gli argomenti che dovete sostituire con un valore reale; per esempio, *nome-host* dovrà essere sostituito con il nome effettivo del vostro sistema UNIX.

I nomi dei comandi e dei file che compaiono nel testo sono in **neretto**, così pure l'input quando appare nel testo stesso. Lettere maiuscole e minuscole sono caratteri differenti per i sistemi UNIX: abbiate cura di usare il formato corretto per tutti i caratteri che inviate al sistema.

Gli esempi di input e output col sistema UNIX sono abitualmente separati dal testo circostante, come:

```
$ cat LEGGIMI
```

Alcuni esempi più lunghi sono presentati in figure separate. Non preoccupatevi se il risultato sul vostro sistema non corrisponde esattamente: fra le diverse release di sistemi UNIX, e fra le versioni fornite dai diversi venditori, esiste ampia variabilità.

A chi è destinato il libro

Guida Completa UNIX System V Release 4 è destinata ai principianti e agli utenti intermedi di un sistema UNIX *general purpose* multiutente o che possiedono un sistema personale. Questo libro vi aiuterà a orientarvi nell'ambiente di UNIX, vi insegnerà come usare i comandi più importanti e vi metterà in grado di approfondire la vostra conoscenza del sistema UNIX.

Il libro presuppone una precedente esperienza di computer e di sistemi operativi. Tuttavia, anche persone senza esperienza precedente possono utilizzare questo libro per apprendere i fondamenti di sistema UNIX e procedere fino a diventare degli esperti.

Se volete gestire un vostro sistema UNIX personale, questo libro sarà un valido riferimento nell'uso giornaliero del sistema; esso copre tutte le principali aree di gestione e aiuta a comprendere non solo *come* ma anche *perché* il sistema UNIX lavora in un certo modo.

Il libro è orientato alla versione di sistema UNIX System V, Release 4 (nota anche come SVR4 o 5.4); le descrizioni e gli esempi che vi sono inclusi sono presi da questa release. Tuttavia, molte versioni di sistema UNIX oggi disponibili agiscono in maniera molto simile nel livello base e intermedio dei comandi; il libro sarà utile anche nel caso possediate una versione precedente.

Capitolo 1

Descrizione generale di UNIX System V Release 4

Nei venti anni successivi alla sua realizzazione, il sistema operativo UNIX ha subito una notevole evoluzione fino a diventare uno tra gli ambienti di sviluppo più utilizzati e conosciuti. Oggi, un numero sempre crescente di elaboratori (attualmente oltre un milione), dai più piccoli ai più potenti, supporta l'ambiente UNIX per merito della sua grande flessibilità e affidabilità.

Esaminiamo in dettaglio le caratteristiche che hanno contribuito alla sua espansione:

Strumenti software Il sistema UNIX ha introdotto un nuovo modo di programmare: ogni applicazione è l'insieme di componenti distinti, ognuno dei quali svolge un singolo compito nella maniera migliore. In questo modo è possibile sviluppare nuovi programmi utilizzando software già esistente e questo rende il sistema UNIX un ambiente di lavoro particolarmente efficiente.

Portabilità È possibile portare il sistema UNIX su qualsiasi elaboratore di medie e grandi dimensioni. Oggi i personal hanno una capacità di elaborazione che un tempo era riservata solo a elaboratori molto costosi e il sistema UNIX rappresenta il supporto naturale a queste macchine potenti ma economiche. Solo poche modifiche sono state necessarie per adattare il sistema UNIX ai nuovi personal e questo conferma la sua grande portabilità. Ciò non può essere sottovalutato in quanto lo sviluppo software è una attivi-

tà noiosa e costosa. La maggior parte delle applicazioni come i word processor, i database e i sistemi grafici può richiedere molti anni di sviluppo. Per utilizzare le applicazioni anche dopo l'accantonamento dell'hardware su cui sono state sviluppate, senza riscriverle ex-novo per le nuove macchine, è sufficiente conservare lo stesso sistema operativo sui nuovi elaboratori. Il sistema UNIX permette di trasportare senza eccessive difficoltà le applicazioni tra personal computer, minicomputer ed elaboratori di grandi dimensioni, tra vecchie e nuove architetture di sistemi e specialmente tra differenti versioni (release) di sistemi UNIX.

Flessibilità UNIX è un sistema operativo estremamente flessibile e questa peculiarità lo rende molto apprezzato da progettisti hardware e software, che lo ritengono adatto per applicazioni di natura diversa come per esempio l'automazione di fabbrica, i sistemi di commutazione telefonica e i giochi. Un numero sempre crescente di nuove funzioni e comandi inediti viene aggiunto rapidamente al sistema UNIX e questo potenziamento continuo ha convinto la maggior parte dei programmatori a preferire il sistema UNIX a qualsiasi altro sistema operativo come supporto alle proprie applicazioni.

Potenza UNIX è uno dei più semplici e completi sistemi operativi presenti sul mercato in quanto dispone di funzioni e comandi predefiniti che sono assenti in altri sistemi operativi e di una sintassi chiara e concisa che ne semplifica l'impiego.

Multiutenza e concorrenza UNIX è un sistema multitasking time sharing che può facilmente gestire diverse attività in parallelo. In un sistema UNIX monoutente, il programmatore può contemporaneamente scrivere un file, inviarlo su stampante e spedire posta a un altro elaboratore. In un sistema multiutente, UNIX permette a più utenti di svolgere simultaneamente diverse attività, dando a ognuno di essi l'impressione di essere l'unico utilizzatore della macchina.

Eleganza UNIX è sicuramente uno tra i più raffinati sistemi operativi, in quanto, una volta acquistata familiarità con alcuni suoi concetti base, gli utenti operano con semplicità ottenendo soluzioni eleganti. Chi è abituato a lavorare in ambiente UNIX e usa altri sistemi operativi, spesso si meraviglia di come aspetti che sembrano scontati sotto UNIX si rivelano complicati in altri ambienti di programmazione. Per questo motivo, i progettisti spesso ricorrono a UNIX come esempio nello sviluppo di altri sistemi operativi.

Orientamento alle reti Le recenti versioni di UNIX sono strutturate per un uso agevole e funzionale delle reti di trasmissione dati. Gli strumenti di trasmissione dati inclusi nel sistema e l'agevole accettazione di driver a basso

livello di periferiche addizionali sono punti di forza dell'attuale ambiente di rete, dove utenti di workstation personali condividono le risorse centrali come i database e i dispositivi di comunicazione. Le capacità di gestione delle reti da parte di UNIX diventano sempre più importanti via via che aumentano di numero i computer connessi in gruppi di lavoro. Oggi le estensioni delle reti vanno dalle piccole reti locali (LAN) ai collegamenti su scala mondiale (WAN) che consentono la trasmissione di ampi database. Il sistema UNIX, con le sue capacità di multiprocesso (multitasking) e la larga disponibilità di software di comunicazione, rende semplice e agevole la gestione di reti di computer.

Tutte queste considerazioni aiutano a capire perché il sistema UNIX ha acquisito una rapida popolarità negli ultimi anni e perché continuerà a sviluppare le sue potenzialità e a diffondersi ulteriormente tra gli utenti di elaboratori.

1.1 Lodi e critiche del sistema UNIX

UNIX è stato oggetto, più di ogni altro sistema operativo, sia di elogi incondizionati che di critiche feroci. I suoi estimatori ne sottolineano l'eleganza, la potenza e la flessibilità, mentre i suoi detrattori mettono in rilievo la sua scarna sintassi, la strana denominazione dei comandi, la documentazione poco strutturata e una certa complessità. Di solito gli esperti del sistema UNIX si uniscono alla categoria degli estimatori, mentre i principianti si schierano dalla parte dei detrattori.

Originariamente il sistema operativo UNIX venne sviluppato da un gruppo di esperti di computer come uno strumento di sviluppo per uso personale, ignorando così le esigenze dei principianti a vantaggio della velocità e della precisione, ma le innate caratteristiche di semplicità di UNIX hanno rapidamente annullato il divario esistente tra esperto e principiante.

Fortunatamente, le versioni più recenti di UNIX sono state indirizzate anche agli utenti meno abili, senza ridurre per questo la potenza del sistema e altri suoi vantaggi. La versione di UNIX SVR4 permette infatti a quasi tutti gli utenti di sfruttare appieno le potenzialità offerte. I miglioramenti raggiunti con le ultime versioni di UNIX sono da ricercarsi nelle seguenti aree:

Robustezza Il sistema UNIX attuale non necessita di continua assistenza software per rimanere efficiente e garantire elevate prestazioni. Molte operazioni che in versioni precedenti dovevano venire effettuate periodicamente, come per esempio cancellare file di controllo oppure reinizializzare il sistema, sono diventate automatiche o inutili.

Coerenza Nel corso degli anni la sintassi dei comandi UNIX ha raggiunto una uniformità tale da ridurre le differenze e le ambiguità nei diversi co-

mandi. Anche se sono ancora presenti diverse incoerenze, la tendenza è di eliminarle definitivamente.

User agent La maggior parte delle versioni UNIX dispone attualmente di strumenti semplificati (anche sistemi esperti in qualche caso) che agevolano nella configurazione e nella gestione del sistema.

Nuove funzioni Le ultime versioni di UNIX si sono arricchite di nuovi comandi e nuove caratteristiche software. Molte di queste mirano a introdurre nei prodotti AT&T System V alcuni concetti presenti in BSD (Berkeley Software Development) e XENIX. Tra le varie migliorie e le nuove caratteristiche vi è una nuova organizzazione del sistema per la gestione delle reti; è stato potenziato l'ambiente di sviluppo ed è in continuo potenziamento il supporto della condivisione dei processi in sistemi multiprocessori.

Interoperabilità La grande flessibilità del sistema UNIX è stata adesso formalizzata nella *Applications Binary Interface* (ABI) e nella *Applications Programming Interface* (API). Queste definizioni di ambienti standard di sviluppo nel sistema UNIX garantiscono un'evoluzione del sistema in maniera controllata e pianificata. I produttori di software possono confidare che i loro investimenti nello sviluppo di prodotti software manterranno il loro valore mentre il software di sistema UNIX (API) e le macchine che usano il sistema UNIX (ABI) evolvono nel tempo. In un più ampio periodo di tempo, il sistema UNIX si sta muovendo verso una maggiore aderenza allo standard internazionale POSIX per sistemi operativi. Lo standard POSIX prevede un insieme minimo di funzioni che alla fine saranno supportate su scala mondiale dai sistemi compatibili UNIX. Naturalmente continueranno a comparire ulteriori innovazioni, ma gli standard POSIX, API e ABI aiuteranno a garantire una base sicura per utenti e sviluppatori.

Condivisione di sistema operativo Le ultime versioni per microcomputer permettono la condivisione della macchina e dei file tra il sistema UNIX e il sistema MS-DOS, oppure tra l'ambiente Apple Macintosh e il sistema UNIX. Condividere il sistema operativo significa poter sfruttare le capacità di concorrenza dei processi (multitasking) offerte dal sistema UNIX per eseguire l'intero MS-DOS (o Macintosh Finder) come un processo operante in ambiente UNIX. In questo modo, è possibile eseguire tutte le funzioni a bassa priorità (background) del sistema UNIX in parallelo all'esecuzione di programmi in MS-DOS.

Questi e altri miglioramenti apportati al sistema UNIX hanno contribuito a dissolvere molte delle critiche mosse finora nei suoi confronti. Sicuramente UNIX è tuttora un sistema operativo potente e complesso, destinato a diventare d'uso sempre più agevole e affidabile.

A causa della sua storia di sistema nato a uso di esperti, e di un certo alone mistico, intorno al sistema UNIX è sorto un grande numero di controversie e discussioni.

1.2 La filosofia del sistema UNIX

Il sistema UNIX è nato essenzialmente come reazione ai sistemi operativi di grandi dimensioni, complessi e poco flessibili, disponibili alla fine degli anni Sessanta. Il principale obiettivo di progettazione è stato quello di ottenere un sistema operativo di elevata potenzialità, definendo comandi che non cerchino di risolvere tutti i problemi possibili. Viceversa, un aspetto fondamentale tipico delle tendenze meno recenti riguarda infatti la necessità di disporre di comandi che eseguono una singola azione, ma con semplicità e nel modo migliore. Per esempio, un comando che concatena e visualizza i file non deve inviarli anche su stampante. Questi compiti devono essere lasciati ad altri programmi che sono in grado di eseguirli meglio.

La filosofia di suddividere il problema in aspetti semplici e indivisibili ha originato effetti non trascurabili. In primo luogo, è emersa una esigenza prima inesistente: i dati prodotti in uscita da alcuni di questi comandi devono costituire i dati di ingresso di altri comandi. Si definirono quindi i concetti di struttura a pipeline, di file standard input e standard output, concetti questi che esamineremo in dettaglio successivamente. In secondo luogo, poiché lo sviluppo di nuovi comandi e di nuove applicazioni crea meno problemi rispetto ad altri sistemi operativi, il sistema UNIX è in continua evoluzione. Questa rapidità e facilità di aggiornamento e sviluppo del sistema è una diretta conseguenza del modo con cui è stata concepita la struttura dei suoi comandi. Il sistema UNIX definisce infatti un elevato numero di comandi specifici e, anche se non è semplice imparare nomi di funzioni che in altri sistemi operativi vengono conglobate in un unico comando, molti utenti sono concordi nel sottolineare che le difficoltà associate con la grande quantità di comandi sono controbilanciate dalla flessibilità e dalle potenzialità che questo approccio garantisce.

1.3 Vantaggi e svantaggi del sistema UNIX attuale

Il sistema UNIX è ancora oggetto di controversie, anche se le sue attuali caratteristiche di multiprogrammazione e concorrenza sono sempre più richieste nell'ambiente dei minicomputer e personal. In confronto a MS-DOS, per esempio, il sistema UNIX è più esteso e complesso, tuttavia presenta come contropartita molte caratteristiche che non sono disponibili in MS-DOS e nemmeno in MS-OS/2.

La filosofia originale del primo UNIX è andata via via scomparendo nelle versioni più recenti e i nuovi comandi sono ora dotati di un elevato numero di opzioni e controlli. Questa constatazione non deve trarre in inganno, in quanto i nuovi comandi possono essere più difficili da utilizzare, anche se possono adattarsi più facilmente alle situazioni che si presentano. Concludendo, sono stati conservati i concetti chiave di UNIX originale, come per esempio il pipeline, mentre le potenzialità offerte dai comandi sono state espanse per soddisfare le nuove esigenze.

Il sistema UNIX è principalmente orientato a terminali e dispositivi periferici a caratteri. Occorre un particolare software aggiuntivo chiamato *X Window System* per permettere al sistema UNIX di interfacciare terminali grafici "bit-mapped". Questa dipendenza storica dai terminali ASCII ha lo svantaggio che gli strumenti e le procedure di UNIX sono orientati a linee di comando, mentre strumenti con mouse e menu su terminali grafici sono rari, diversamente da quanto avviene coi nuovi personal. D'altra parte, il sistema UNIX permette agli utenti di dialogare con una macchina remota tramite un terminale a basso costo e attraverso normali linee telefoniche. Ne risulta che il sistema è ideale per applicazioni *server*, in cui l'interfaccia utente principale non è associata alla macchina centralizzata UNIX, ma opera direttamente su un terminale intelligente remoto. Si ha a disposizione una moderna architettura per gli ambienti di reti locali. Lo X Window System Standard, un sistema grafico a finestre, ha la capacità di integrarsi in rete: ciò significa che un'applicazione può essere eseguita su una macchina nella rete (abitualmente una veloce ma costosa macchina server) mentre l'interfaccia utente può essere eseguita su terminali a basso costo dislocati presso ciascun utilizzatore. Questo schema può ridurre in maniera significativa il costo di installazione di computer in un ambiente di rete.

Il sistema UNIX va contro la tendenza corrente di rendere i sistemi operativi invisibili all'utilizzatore tramite interfacce di utente che nascondono la vista del sistema operativo all'utente finale. Il Finder del Macintosh, per esempio, è un sistema operativo che dispone di funzioni relativamente semplici. L'interfaccia utente a finestra e menu aiuta a nascondere agli utenti i comandi, il file system e gli strumenti di gestione. Nel sistema UNIX, al contrario, maggiore è la conoscenza delle funzioni interne del sistema operativo, maggiore è la possibilità di sfruttare al meglio il sistema e ricavarne maggiore produttività. Questa situazione deriva principalmente dal fatto che UNIX è nato come un sistema per esperti, di cui la maggior parte degli utenti conosceva la struttura interna. Ora che il sistema UNIX viene largamente adottato a ogni livello, è stato dotato di strumenti come *user agents*, o funzioni di utente, per isolare l'utente dagli aspetti più complessi e meno immediati del sistema operativo. Comunque, anche se è possibile utilizzare il sistema senza avere una conoscenza approfondita delle sue complesse caratteristiche, è senza dubbio vantaggioso sapere come opera il sistema quando lanciate un comando.

1.4 Storia del sistema UNIX

Anche se il sistema UNIX è nato negli AT&T Bell Laboratories, uno dei maggiori istituti di ricerca degli Stati Uniti, la sua storia è pressoché unica in confronto ad altri sistemi operativi, in quanto il suo sviluppo è dovuto alle idee creative di singole persone oltre che alle necessità degli utenti e non deriva direttamente da decisioni burocratiche. Questo è ancora vero oggi, tanto che il sistema UNIX può essere considerato un ambiente molto favorevole per la definizione di nuovi concetti di programmazione.

UNIX è stato creato da un gruppo di persone della AT&T che aveva seguito lo sviluppo del sistema operativo MULTICS, nato al MIT negli anni Sessanta. Quale precursore di sistemi operativi time sharing, MULTICS presentava molti concetti tipici dei sistemi concorrenti odierni, ma sfortunatamente risultava più complesso e intricato del necessario, anche a causa del suo ruolo innovativo. Negli anni Sessanta, la AT&T si ritirava quasi completamente dal progetto MULTICS, lasciando a un gruppo di ricercatori abili le idee su come organizzare un sistema a divisione di tempo. Non potendo più lavorare sul sistema MULTICS, questi ricercatori decisero di creare un moderno sistema operativo. I primi tra questi furono Ken Thompson e Dennis Ritchie, che basarono la prima stesura su un progetto di Rudd Canaday e furono successivamente influenzati dalle idee di altri abili ricercatori, tra i quali J.F. Ossanna e R. Morris. Dopo un periodo di discussioni e riflessioni, riuscirono ad acquistare un PDP-7 della Digital e si misero al lavoro. Come accade per la maggior parte dei progetti migliori, Thompson e Ritchie inventarono inizialmente sul PDP-7 un gioco (Viaggio nello spazio). A seguito di questa esperienza, decisero di concentrare i loro sforzi verso obiettivi più appaganti e poco dopo crearono un nuovo file system che assomigliava molto a quelli odierni. Successivamente potenziarono il sistema definendo un ambiente a processi con scheduling e da ultimo lo completarono nelle parti mancanti con soluzioni rudimentali. La denominazione UNIX è nata in quanto il progetto risultava una semplificazione del sistema MULTICS. Agli inizi degli anni Settanta UNIX veniva supportato dal PDP-7 e a metà degli anni Settanta anche dal classico e diffusissimo PDP-11. Molti dei concetti fondamentali del sistema UNIX attuale erano presenti anche nelle prime versioni, tra cui il sistema di gestione di file, l'ambiente a processi e la struttura delle linee di comando.

La versione originale fu codificata in assembly, ma visto che proprio in quegli anni il gruppo di ricerca stava sviluppando anche il linguaggio C, venne subito riscritta in questo nuovo linguaggio. Infatti nel 1973 il kernel (nucleo) venne ricodificato in C. Oggi, solo poche subroutine di kernel che necessitano di elevate prestazioni risultano ancora scritte in linguaggio assembly. A quegli anni dunque risale il primo tentativo di scrivere un sistema operativo in un linguaggio di alto livello e la portabilità che lo contraddistingue è sicuramente il motivo principale della popolarità che UNIX ha acquisito oggi.

Contemporaneamente vennero definiti gli strumenti per l'elaborazione di testi (conosciuti oggi sotto il nome di **troff**) e il primo vero cliente del sistema UNIX diventò il Bell Laboratories Patent Attorney's Office, che iniziò a usare il programma **troff** nell'autunno del 1971.

UNIX catturò immediatamente l'immaginazione dei ricercatori dei Bell Laboratories e, dopo solo due o tre anni, iniziarono a diffondersi circa una dozzina di sistemi UNIX su elaboratori diversi. Si registrarono nel corso degli anni frequenti miglioramenti del software e la AT&T decise di sviluppare il sistema nei Bell Laboratories. Il programma **troff** fece la sua comparsa proprio in questo periodo, insieme a molte altre innovazioni.

In ogni caso il sistema UNIX ottenne i giusti riconoscimenti nella prima metà degli anni Settanta, con lo sviluppo di elaboratori più potenti del PDP-11, come per esempio il PDP-11/45 e il PDP-11/70. Il sistema UNIX trovò naturale collocazione nell'architettura DEC e venne venduto insieme ai PDP-11. I ricercatori dei Bell Laboratories iniziarono a usare macchine UNIX per elaborare testi, mentre i progettisti della Bell System iniziarono a utilizzare i PDP-11 con UNIX nei sistemi telefonici.

Contemporaneamente, l'AT&T distribuì alle università di tutto il mondo molte copie del sistema UNIX e la generazione di ricercatori negli anni Settanta affinò le proprie capacità lavorando su UNIX. Anche la University of California a Berkeley progettò in quegli anni una propria versione di sistema operativo denominata BSD (Berkeley Software Distribution) che ottenne ampi consensi. Mentre la AT&T rivolse i propri interessi verso il completamento e l'ottimizzazione di UNIX da un punto di vista commerciale, le versioni BSD diventarono predominanti nelle università. La compatibilità esistente tra le versioni AT&T e BSD è opinabile e gli utenti odierni di solito si schierano da una parte o dall'altra. Le versioni BSD mantengono la stessa filosofia di fondo presente nelle versioni AT&T, anche se le migliori innovazioni di un sistema vengono di solito trapiantate anche nell'altro.

Negli anni Settanta la AT&T iniziò a rinominare le versioni del sistema UNIX. Nel passato, a ogni significativo miglioramento del sistema prodotto dal gruppo di ricerca corrispondeva il rilascio di una nuova versione: in particolare, due delle più popolari versioni rilasciate vennero identificate come Sixth Edition e Seventh Edition. Seguendo una riorganizzazione interna del supporto UNIX, la AT&T ne cambiò la numerazione in System III e System V. Effettivamente queste nuove versioni sono dirette discendenti della Seventh Edition e la System V sostituì la System III nella metà degli anni Ottanta. La System IV fu utilizzata internamente nei Bell Laboratories ma, essendo un prodotto di transizione, non ebbe diffusione all'esterno. Ora, nella seconda metà degli anni Ottanta, la AT&T ha standardizzato il nome System V e le più recenti versioni sono denominate System V Release 3 e System V Release 4 e vengono spesso abbreviate rispettivamente in SVR3 e SVR4.

Tra la fine degli anni Settanta e l'inizio degli anni Ottanta, entrambe le versioni BSD e AT&T vennero trasportate su quasi ogni elaboratore fosse

in grado di supportarle. Questa circostanza comportò la necessità di disporre di unità a disco a elevata velocità e di un supporto di gestione della memoria predefinito nella CPU, anche se le versioni sperimentali di UNIX furono adattate a macchine dotate di ROM, ma non di disco rigido. Oggi è possibile acquistare versioni del sistema UNIX per i più potenti supercomputer, per i *mainframe* più utilizzati e per quasi tutti i minielaboratori presenti sul mercato.

Le versioni BSD ebbero un ruolo dominante come piattaforma di sperimentazione dei concetti delle reti, cosicché adesso il meglio nelle reti locali è disponibile per i sistemi UNIX. Il maggior esperimento di rete geografica, ARPANET, fu inizialmente orientato a sistemi UNIX. Il Network File System (NFS), creato da Sun Microsystems, e il Remote File Sharing (RFS), creato da AT&T, sono i prodotti finali dell'evoluzione di quei concetti. Ambedue questi sistemi sono attualmente reti funzionali, e quasi tutti gli ambienti di computer prevedono il supporto delle connessioni a NFS o RFS. Molte architetture LAN basate su PC attinsero liberamente da queste esperienze, tuttavia UNIX detiene ancora una solida priorità nel supporto delle reti. La ricerca in quest'area rimane tuttora molto attiva in ambiente UNIX; ulteriori innovazioni sono immaginabili considerando che i sistemi multiprocessori stanno guadagnando sempre più interesse nell'industria dei computer.

Quando il microelaboratore incrementò le prestazioni in termini di velocità e potenza, diminuendo i costi, entrò nella sfera di azione del sistema UNIX. Le macchine basate su 8088 erano quasi abbastanza potenti da supportare il sistema UNIX, ma solo poche versioni potevano funzionare su questi elaboratori. Il popolare sistema operativo XENIX, in origine una versione ridotta per il PC-IBM che raggiungeva o superava il limite della capacità della macchine, può essere sfruttato appieno su elaboratori dotati di microprocessore 80286 o 80386. Adesso che molte delle caratteristiche di XENIX sono state incluse nella linea principale del sistema UNIX, si prevede che XENIX come linea separata nella storia dei sistemi UNIX sia destinato a scomparire.

Quando i sistemi SVR3 e SVR4 sono diventati disponibili per macchine con architettura PC-AT alla fine degli anni Ottanta, la popolarità di UNIX ha compiuto un grande passo avanti. Le macchine della classe AT 80386 fornirono un ambiente veramente conveniente per il sistema UNIX; attualmente oltre la metà dei sistemi UNIX è installata su questi diffusi computer.

Un altro notevole passo avanti venne compiuto con la crescente ampia accettazione di X Window System come parte del sistema UNIX. X, come viene spesso chiamato in breve, venne realizzato in prototipo come progetto di ricerca al MIT nella metà degli anni Ottanta, venne quindi adottato da un consorzio di produttori di sistemi UNIX, che ne consolidarono la qualità per un prodotto commerciale e ne incrementarono le caratteristiche. Altri sistemi di finestre orientati al mouse vennero introdotti da Sun e Apollo,

ma X ha finito col dominare come standard industriale senza diritti di proprietà (nonproprietary). Oggi molti pacchetti applicativi per sistemi UNIX lavorano in ambiente X.

Recentemente, i Bell Laboratories AT&T hanno sviluppato una nuova versione completa denominata "Tenth Edition" o "Research UNIX System" che, sebbene non commercializzata, è stata largamente distribuita nelle università. Nel tempo, le innovazioni delle versioni di ricerca verranno regolarmente inserite nei prodotti System V.

La fusione delle tre maggiori varianti del sistema UNIX iniziò con le ultime versioni di SVR3 ed è prossima al completamento nella SVR4. Oggi, come risultato di accordi tra AT&T, Sun Microsystems e Microsoft, i sistemi BSD e XENIX si stanno integrando nel prodotto System V. Gran parte del software realizzato per BSD o XENIX potrà essere usato senza modifiche con sistemi SVR4, mentre le varianti scompariranno con l'inclusione delle loro caratteristiche nella versione System V. Il prodotto risultante consentirà anche la compatibilità a livello di codice oggetto tra varianti prodotte per macchine dello stesso tipo.

Mentre l'integrazione di BSD e System V era in corso di realizzazione, diversi altri venditori di systemi UNIX cominciarono a preoccuparsi che il sistema integrato potesse consentire un predominio nel mercato da parte di AT&T e Sun Microsystems. Questi produttori (principalmente IBM, Hewlett-Packard, DEC) formarono un consorzio per sviluppare una variante del sistema al di fuori del controllo di AT&T o Sun. Questo gruppo, chiamato Open Software Foundation (OSF) ha già sviluppato parecchi importanti prodotti del sistema UNIX, il più significativo dei quali è la specificazione dell'interfaccia utente Motif. In risposta, AT&T e un altro gruppo di venditori di sistemi UNIX, hanno costituito un loro grande consorzio, UNIX International, per standardizzare e pubblicizzare le versioni System V negli anni Novanta. Alcune compagnie fanno parte di entrambi questi gruppi. Seguendo la storia dello stimolo e della competizione fra BSD e sistema UNIX, le tensioni competitive fra OSF e UNIX International potrebbero produrre molti positivi risultati in futuro.

Anche Apple Computer ha adattato SVR4 per la sua famiglia di computer Macintosh e incorporato con successo il Finder e la classica presentazione e interfaccia Macintosh nel sistema UNIX, come MS-DOS è stato incorporato nei prodotti Merge e Simultask.

La forza di UNIX nel superare le differenze di hardware viene bene evidenziata dall'apparizione dei *reduced instructions set computers* (RISC) tra la fine degli anni Ottanta e l'inizio dei Novanta. I computer RISC implementano un concetto fondamentalmente differente da quello seguito nei microprocessori allora dominanti, un concetto che rende il progetto dei chip di CPU più facile, più rapido e meno costoso. In un breve periodo di tempo sono stati sviluppati parecchi microprocessori RISC e, a causa delle difficoltà e dei costi per lo sviluppo di un sistema operativo totalmente nuovo per

ogni nuova macchina, virtualmente tutte le macchine RISC hanno adottato il sistema UNIX. Diverse macchine RISC sono state addirittura progettate per fornire le migliori prestazioni al sistema UNIX. Come conseguenza, i migliori e più veloci microcomputer oggi disponibili sono macchine RISC.

Nel corso degli anni Novanta, gli sviluppi più significativi si avranno nelle caratteristiche di *multiprocessing* che consentono architetture di macchine con più chip di CPU centrale. Queste macchine condividono altre risorse come l'alimentazione elettrica, la memoria principale, i dischi e consentono aumentate capacità di elaborazione a un costo minore di quello di macchine aggiuntive. In sistemi multiutente, i numerosi processi possono essere efficientemente distribuiti fra diversi processori, ma anche i sistemi per singolo utente o i server in rete possono ugualmente distribuire il carico addizionale. I sistemi in multiprocesso saranno certo uno dei fattori principali nella vita di SVR4.

Negli anni Novanta probabilmente la variante OSF diventerà una valida alternativa alla versione SVR4; strumenti e idee che avranno successo, originate in uno dei due sistemi, verranno alla fine adottati dall'altro, anche se non immediatamente e neanche nella successiva release del sistema. Solitamente, venditori di software aggiuntivo sono i primi a proporre le caratteristiche di un sistema per un altro, così conviene tenersi informati dei nuovi strumenti e procedure che esistono solo in una certa variante del vostro sistema UNIX.

Lo stesso avviene in altri sistemi operativi; innovazioni in MS-DOS, OS/2, NeXT, ambiente Macintosh, e anche le versioni IBM/6000 del sistema UNIX, sono spesso disponibili come prodotti aggiuntivi per SVR4.

1.5 La versione SVR4

SVR4 è la versione più aggiornata del sistema UNIX AT&T. È stata trasportata sulla maggior parte delle macchine e rappresenta lo standard corrente per la linea AT&T. A seguito di molti cambiamenti effettuati, SVR4 è significativamente migliorata rispetto alle precedenti versioni. Molte delle nuove caratteristiche sono principalmente di interesse per utenti in ambienti di sviluppo e per i gestori del sistema, ma in molti casi alla fine i miglioramenti coinvolgono anche gli utenti finali, risultando in applicazioni migliori, di uso più agevole, più veloci e meno costose. Alcuni dei più importanti potenziamenti vengono descritti nei punti seguenti.

SUPPORTO DELLE RETI

Il più importante potenziamento in SVR4 è l'inclusione di un totale supporto per le reti locali. Sia il sistema AT&T RFS che il sistema a standard industriale NFS sono supportati come reti attuali; sono predisposti strumenti

e agganci per integrare facilmente altre reti in architettura SVR4. Parte di questo supporto consiste in una estesa riorganizzazione del file system per consentire una migliore condivisione dei file fra le macchine in rete, permettendo per la prima volta in System V la gestione di sistemi senza dischi. La gestione delle macchine nella rete è stata notevolmente potenziata, consentendo la gestione remota attraverso la rete. L'aggiunta di un nuovo sistema di monitoraggio delle porte semplifica il trattamento delle richieste di servizi da parte di altri sistemi. Gli strumenti di multiprocesso che saranno aggiunti in seguito consentiranno la distribuzione dei carichi di lavoro tra le macchine in rete.

UNIFICAZIONE

SVR4 ha fatto passi notevoli verso l'unificazione delle varie versioni di sistema UNIX, in particolare delle popolari varianti BSD e XENIX. Ciò consente agli sviluppatori di creare applicazioni per una singola versione di sistema invece di molte differenti versioni separate. Questa sola caratteristica ridurrà notevolmente il costo di sviluppo delle applicazioni per sistemi UNIX e certamente ne farà aumentare la disponibilità e la qualità.

NUOVO AMBIENTE DI SVILUPPO

L'ambiente di sviluppo in linguaggio C è stato grandemente potenziato in SVR4 con un nuovo compilatore C che rispetta le specifiche ANSI ed incrementa notevolmente le prestazioni delle versioni precedenti. Il collegamento dinamico e le librerie condivise consentono agli sviluppatori delle applicazioni di differire molti aspetti dei loro programmi al momento dell'esecuzione invece di includere tutti i sottoprogrammi al momento della compilazione. Per la prima volta sono state documentate tutte le interfacce del kernel e dispositivi periferici, stabilizzando e semplificando agli sviluppatori l'accesso ai servizi di kernel. Uno sviluppo completo del sistema di driver dei dispositivi a flusso di caratteri consente agli sviluppatori di realizzare col sistema operazioni prima impossibili.

INTERNAZIONALIZZAZIONE

L'intero sistema SVR4 è stato adattato per supportare versioni nazionali; la lingua dei messaggi di sistema, le convenzioni per le date, le ore e le monete e molte altre variazioni locali sono state per la prima volta introdotte nel sistema, in modo che gli sviluppatori possano personalizzare i sistemi per un particolare ambiente. Inoltre, tutto il sistema si avvicina molto più di prima al rispetto degli standard internazionali per i sistemi operativi.

X WINDOW SYSTEM

SVR4 include una interfaccia utente grafica (GUI) completa e potente, X Window System; con i suoi standard di interfaccia utente OPEN LOOK e Motif, SVR4 stabilizzerà la visione del sistema UNIX da parte degli utilizzatori, migliorando ancora la qualità degli sviluppi e standardizzando l'ambiente per gli utenti.

COMANDI

Sono stati aggiunti molti nuovi comandi e caratteristiche tradizionali di sistemi UNIX. Sono stati inclusi i due comandi **ksh** e **csh**, aggiunte caratteristiche di *job control* o controllo di esecuzione lavori, prevista la possibilità di configurare diverse console virtuali. Numerosi altri miglioramenti sono stati apportati in tutto il sistema.

FILE SYSTEM VIRTUALE

SVR4 supporta una nuova organizzazione del file system che consente più potenti operazioni sui file. Collegamenti simbolici (symbolic link) consentono collegamenti che superano i confini tra file system, file mappati in memoria potenziano la bufferizzazione e paginazione delle operazioni di I/O del sistema, la gestione della memoria virtuale nel suo insieme è stata potenziata con risultati evidenti. Il Virtual File System (VFS) consente di utilizzare contemporaneamente numerosi tipi di file system; ne risultano molti vantaggi, come l'eliminazione del vecchio limite dei nomi di file a 14 caratteri.

PROCESSI REAL TIME

Le versioni precedenti di sistemi UNIX non erano in grado di rispondere in tempi rapidi e prevedibili alle richieste in tempo reale di applicazioni di controllo, come l'automazione della produzione. In SVR4 un nuovo sistema di scheduling ammette processi in tempo reale e microtemporizzazioni, cosicché il sistema UNIX può coprire oggi il campo completo di applicazioni che vanno dai sistemi multiutenti in time sharing ai processi dedicati di controllo con alte prestazioni.

CONFIGURAZIONE E INSTALLAZIONE SEMPLIFICATE

Un progetto ancora in corso di sviluppo in SVR4 semplificherà l'installazione e la configurazione del sistema. In futuro, sarà sempre più facile installa-

re un nuovo sistema o aggiungere a un sistema esistente nuovi sottosistemi hardware e software. Aggiunte e modifiche a un sistema non dovranno perturbare le parti esistenti, o almeno l'origine di ogni contrasto dovrà essere chiaramente indicata senza alcun disturbo nella macchina. Dovrà essere possibile anche integrare del software senza fermare o reinizializzare il sistema.

SICUREZZA

Infine, la sicurezza del sistema è stata adeguatamente rafforzata; ora il sistema UNIX risponde a molte prescrizioni di sicurezza di enti governativi. In considerazione di possibili intrusioni dolose nei sistemi di computer il rafforzamento della sicurezza è una caratteristica importante che non deve essere sottovalutata.

1.6 Approfondimenti

Un sistema SVR4 richiede una notevole maggior quantità di spazio in memoria centrale e su disco rigido rispetto alla versione SVR3 e altre release precedenti. In compenso, SVR4 mette a disposizione un sistema allo stato dell'arte, nuove caratteristiche di rete e un miglior supporto per gli strumenti di gestione e sviluppo.

PREREQUISITI DEI SISTEMI

Il sistema SVR4 può essere installato su quasi ogni elaboratore che sia dotato di memoria reale in quantità sufficiente e di una memoria di massa veloce, tuttavia le macchine UNIX richiedono un supporto hardware più consistente rispetto a quello richiesto dall'MS-DOS, anche se non di molto superiore a quello necessario per l'OS/2. Generalmente in una macchina SVR4 occorrono almeno 4 megabyte di RAM e almeno 80 megabyte di memoria su un disco rigido veloce.

Un disco rigido veloce è necessario perché più processi vi debbono essere memorizzati temporaneamente, mentre un altro processo utilizza la memoria reale. Questa memorizzazione temporanea è di solito realizzata scaricando da memoria reale su disco i processi non attivi e questa operazione può verificarsi più volte al secondo (swapping). Le versioni del sistema UNIX che utilizzano solo dischi flessibili hanno prestazioni inferiori e sono generalmente considerate molto limitate.

La necessità di disporre di un disco rigido di grande dimensione nasce dall'elevato numero di comandi e funzioni che sono supportati dal sistema UNIX standard. Un sistema completo può contenere più di 40 megabyte di

programmi memorizzati. Esistono più di 2000 comandi di sistema diversi e più di 6000 file e directory in SVR4. Naturalmente è possibile eliminarne una parte quando il sistema viene utilizzato per eseguire una particolare applicazione; il software addizionale di Standard C Development Environment e i vari pacchetti per le reti consumano circa un quarto dello spazio totale su disco. Se non è prevista attività di programmazione, questi file non occorrono, ma è comunque consigliabile disporre di almeno 80 megabyte di spazio di memoria su disco rigido prima di caricare un sistema UNIX SVR4.

La grande quantità di memoria reale richiesta è necessaria per soddisfare alcune caratteristiche a elevate prestazioni offerte dal sistema UNIX. Il livello più basso del sistema operativo, il kernel, che risiede permanentemente in memoria e realizza le connessioni necessarie tra i programmi utente e l'hardware della macchina, occupa da solo più di un megabyte. L'architettura include anche molte memorie temporanee (buffer) che ricoprono la stessa funzione di un disco di memoria nei sistemi operativi monoprocesso. Parte della velocità che si ottiene deriva dalla presenza di queste memorie temporanee, che si aggiungono però al totale di memoria reale necessaria. Inoltre i programmi in esecuzione richiedono memoria per soddisfare le loro esigenze. Alcuni sistemi UNIX di grandi dimensioni, definiti per soddisfare un ambiente multiutente concorrente, dispongono di più di 40 megabyte di memoria reale. I più piccoli sistemi a uso personale possono naturalmente funzionare con una quantità di memoria molto inferiore, ma è consigliabile disporre di almeno 4, preferibilmente di 6 oppure 8 megabyte, per ottenere prestazioni di un certo rilievo.

PERSONALIZZAZIONE DEI SISTEMI UNIX

La maggior parte degli utenti non resta a lungo completamente soddisfatta dei pacchetti software in versione standard che esistono in commercio. Come per l'MS-DOS, sono disponibili molti prodotti software di completamento per la maggior parte dei sistemi UNIX. Alcuni di questi risolvono problemi particolari, come per esempio i word processor o pacchetti di grafica di elevata qualità. La scelta di un sistema operativo dipende di solito dalla varietà delle applicazioni disponibili. Comunque, l'ambiente UNIX incoraggia modifiche e miglioramenti al software del sistema standard e sono disponibili molti pacchetti software che sostituiscono i principali sottosistemi. Per esempio, è possibile migliorare il sistema di comunicazione e invio su stampante di file, così come sono disponibili diversi programmi di shell, che forniscono l'interfaccia utente al sistema operativo.

Inoltre, alcune case costruttrici hanno notevolmente arricchito o modificato il sistema per renderlo più commerciale o semplicemente per migliorare parte dei comandi o sottosistemi. In conclusione possiamo dire che esistono diverse varianti dei comandi, che sono nate nel corso delle evoluzioni

cui è stato soggetto inevitabilmente il sistema UNIX. Alcuni utenti preferiscono acquistare versioni UNIX con comandi che rispondono meglio a loro particolari preferenze, piuttosto che ricorrere a quelli standard. È possibile che alcuni comandi presenti su versioni commercializzate abbiano un comportamento che differisce, anche se di poco, dagli standard SVR4. Lo stesso file system viene spesso riorganizzato, talvolta in maniera totale, per meglio rispondere alle esigenze di una versione. Possono essere variati o aggiunti alberi dei directory, modificati i nomi di file per i dispositivi esterni, ed altro. Le case costruttrici che rilasciano sistemi UNIX possono fornire anche strumenti di proprio sviluppo per facilitare la gestione del sistema, in particolare alcune funzioni di user agent, che però comportano una inevitabile perdita di flessibilità ed efficienza dei sottosistemi. Queste funzioni di user agent spesso differiscono parecchio tra loro.

Se questo non fosse abbastanza complesso, tenete presente che tutti gli aspetti del sistema UNIX cambiano nel tempo ogni volta che il sistema viene potenziato, modificato o incrementato di caratteristiche; per esempio, le Release 4.1 e 4.2 differiscono in molti punti dalla Release 4.0.

Poiché è impossibile trattare tutte queste variabilità in un singolo libro, questo testo è orientato alla versione ufficiale di comandi e strumenti di System Administration in SVR4. Lo standard considerato è quello di System V, Release 4.0 *porting base*, come rilasciato dalla AT&T per macchine PC AT 80386/80486. Quando opportuno, il testo indicherà le variazioni o aree che rimangono non stabilizzate tra le diverse release. In molti casi, le variazioni consistono in dettagli nelle opzioni o nell'output dei comandi; vi sono poche differenze nei concetti tra le sottoversioni di SVR4 e questo libro potrà essere in ogni caso di aiuto anche se non dovesse essere del tutto esatto. In generale, queste variazioni sono presenti negli argomenti trattati nella seconda metà del libro; quanto trattato nella prima metà (fino al Capitolo 12) è coerente con quasi tutti i sistemi UNIX attualmente esistenti.

COMPATIBILITÀ CON BSD E XENIX

Storicamente le versioni BSD sono sempre state notevolmente diverse dalle versioni System V, e nel corso degli anni si sono evidenziate numerose incompatibilità tra le due. SVR4 sta portando avanti un serio tentativo di integrare le caratteristiche di queste due versioni; in larga parte questo sforzo ha avuto successo, specialmente nel supporto delle applicazioni BSD che sono state adattate per l'ambiente SVR4. Tuttavia, alcuni dei comandi e degli strumenti BSD previsti in SVR4 sono leggermente differenti da quello che un utilizzatore di BSD si aspetterebbe; infatti, anche se la piena compatibilità con BSD è una meta di SVR4, questo è prima di tutto un prodotto System V e la filosofia dell'implementazione è basata sul System V.

In conseguenza dell'orientamento verso l'ambiente System V, gli strumenti BSD e XENIX sono forniti in speciali *compatibility package* che la-

sciano al gestore del sistema la scelta se rimanere in System V oppure, se lo desidera, aggiungere il pacchetto di compatibilità. D'altra parte, molte funzioni e caratteristiche di BSD e XENIX sono state integrate nella release base SVR4: i pacchetti di compatibilità non sono così sempre necessari.

Tra l'altro, la compatibilità con XENIX è risultata molto più facile da realizzare che non quella con BSD, e il sistema SVR4 riflette questo fatto. Molti comandi e strumenti XENIX sono stati integrati nell'architettura principale del sistema; il pacchetto speciale di compatibilità XENIX risulta molto più ridotto dell'equivalente pacchetto BSD.

In questo testo sono trattate le caratteristiche standard, senza soffermarsi estesamente sui pacchetti di compatibilità. Se avete esperienza di uso del sistema BSD e trovate che lo SVR4 standard non si comporta come vi aspettavate, potete decidere di integrare nel vostro sistema il pacchetto di compatibilità BSD e usarne gli strumenti in luogo di quelli standard.

Capitolo 2

Introduzione all'uso del sistema UNIX

Per cominciare, esaminiamo come si svolge una breve sessione di lavoro introducendo alcuni concetti fondamentali e presentando alcuni esempi di interazione tra l'utente e il sistema operativo. Alla fine di questo capitolo sarete in grado di entrare e uscire senza difficoltà da UNIX.

Supponiamo che esista già il collegamento tra il terminale e la macchina sulla quale è caricato UNIX. La maggior parte dei neofiti ha bisogno che utenti più esperti inizializzino la macchina, ma questa operazione potete eseguirla facilmente dopo aver appreso alcuni nozioni preliminari su un sistema operativo qualunque. Se disponete di una macchina su cui non avete mai operato, è consigliabile leggere prima il Capitolo 25 e poi esaminare il Capitolo 21. Quando il sistema è caricato e inizializzato, potete ritornare a leggere questo capitolo.

Il vostro terminale può essere una console di sistema, un terminale funzionante in modalità carattere collegato direttamente alla macchina tramite una porta RS232, oppure un terminale remoto connesso via linea telefonica e modem, o anche un terminale connesso tramite una rete locale (Local Area Network, LAN). Non ha alcuna importanza il tipo di terminale che

utilizzate, poiché il sistema si comporta allo stesso modo per ognuno di questi terminali. I terminali e le loro configurazioni sono argomento del Capitolo 12.

L'accesso standard al sistema si verifica completamente in modalità carattere e non compare su video nessun grafico o icona. Poiché UNIX è stato originariamente sviluppato per terminali non intelligenti, gran parte del sistema richiede terminali che funzionino in modo carattere.

UNIX utilizza il set ASCII (American Standard Code for Information Interchange), per cui i terminali del tipo IBM 3270 non operano di solito con UNIX e i terminali grafici devono disporre del modo ASCII.

Per usare X Window System, l'usuale interfaccia a finestre, dovrete usare la console di sistema oppure un terminale intelligente collegato a una rete.

2.1 Accesso al sistema (logging in)

Una volta collegato il terminale alla macchina, può essere necessario premere il tasto di ritorno a capo (RETURN) una o due volte per risvegliare il sistema UNIX. A questo punto compare sul video un prompt (login) che segnala la disponibilità del sistema a ricevere i vostri comandi:

```
Welcome to the AT&T 386 UNIX System.
System name: my_sys

login:
```

Se non vedete il prompt **login:** dovete chiedere l'intervento del gestore del sistema perché corregga qualcosa di errato nei parametri del sistema.

Poiché UNIX è un sistema multiutente, richiede subito di potervi identificare con assoluta certezza. In questo modo il sistema vaglia il vostro intendimento di accedere ai file privati e assegna alla sessione di lavoro la priorità che avete stabilito. Ogni utente che è connesso al sistema viene referenziato tramite un identificatore (login id), che generalmente corrisponde al proprio nome, alle proprie iniziali o a qualche altra parola. L'unica restrizione è che questo identificatore deve avere una lunghezza inferiore a otto caratteri e contenere solo lettere e numeri. In particolare, esiste differenza tra lettere maiuscole e lettere minuscole, cosicché l'identificatore "GIORGIO" è diverso da **giorgio**. Tutti gli identificatori devono iniziare con una lettera minuscola, altrimenti il sistema pensa che stiate collegandovi da un terminale che funziona solo con lettere maiuscole e non esisterebbe corrispondenza tra ciò che siete e ciò che il sistema pensa di voi. L'identificatore personale deve essere unico, cioè non può esserci nel sistema un altro utente che si identifichi come voi.

Se avete scelto come identificatore **giorgio**, rispondete al prompt di sistema inserendo la stringa **giorgio**:

```
Welcome to the AT&T 386 UNIX System.
System name: my_sys

login: giorgio
```

e terminate premendo il tasto RETURN.

Diversamente da ciò che accade di solito quando scrivete testo in ambiente UNIX, non è possibile generalmente correggere con il tasto BACKSPACE eventuali errori che si compiono sulla linea del prompt di login, poiché il sistema non vi ha ancora identificato. L'impiego del tasto BACKSPACE per cancellare un carattere è considerato una opzione a livello utente. Se fate un errore nel digitare identificatore o password dovete completare comunque l'operazione; se sbagliate la procedura di login il sistema vi consente di ripeterla una seconda volta.

Dopo che avete inserito il vostro identificatore, il sistema vi richiede la password:

```
Welcome to the AT&T 386 UNIX System.
System name: my_sys

login: giorgio
Password:
```

Il sistema operativo UNIX chiede conferma che voi siate le persone autorizzate a utilizzare l'identificatore specificato in precedenza, essendo questo generalmente di dominio pubblico. Questa cautela è fondamentale per proteggere i file e altri dati privati dalla curiosità di altri utenti.

Inserite dunque la vostra password e terminate l'operazione premendo il tasto RETURN. Diversamente da ciò che accade quando il sistema vi chiede di specificare l'identificatore, il sistema ora non visualizza i caratteri della password man mano che li inserite da tastiera. (Questa operazione nella terminologia UNIX si chiama **echo**.) Si tratta di una ulteriore misura di sicurezza che impedisce a chi vi sta alle spalle di scoprire la vostra password.

Se inserite un identificatore di utente o una password errati, oppure se non siete autorizzati a utilizzare la macchina, il sistema risponde nel modo seguente:

```
Welcome to the AT&T 386 UNIX System.
System name: my_sys

login: giorgio
Password:
Login incorrect
login:
```

Il sistema non specifica chi, tra l'identificatore e la password, abbia generato l'errore, ma vi concede una seconda possibilità. Battete di nuovo il vostro identificatore, seguito successivamente dalla password.

Una volta che il sistema ha accettato la coppia identificatore-password, visualizza una intestazione iniziale e alcuni messaggi:

```
UNIX System V/386 Release 4.0
System my_sys
Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T
Copyright (c) 1987, 1988 Microsoft Corp.
All Rights Reserved

The my_sys system will be down for software installation
from 6:00 to 7:00 PM tonight.    = pat

news: new_user

You have mail.

$
```

Questi messaggi sono probabilmente diversi da quelli che appaiono sul vostro sistema, ma questo esempio è tipico di un sistema UNIX multiutente. Il primo messaggio:

```
UNIX System V/386 Release 4.0
```

è solamente una intestazione che identifica la versione di sistema in uso. Il messaggio successivo **my_sys** identifica la macchina. Ogni sistema UNIX possiede un nome che lo identifica univocamente.

Seguono alcuni informazioni relative al copyright del software che è installato sulla macchina. Queste informazioni possono riguardare chi ha introdotto sul mercato quella specifica versione di sistema UNIX e quale altro software è installato sulla macchina. Talvolta può seguire anche una più o meno lunga tabella che riassume l'utilizzazione dello spazio su disco; spesso viene omessa nei sistemi di grandi dimensioni. Riserviamo la trattazione di queste informazioni al Capitolo 18.

Il messaggio che segue costituisce il "messaggio del giorno"; è inserito dal gestore del sistema e viene di solito utilizzato per annunciare alcune informazioni di interesse comune.

La linea seguente:

```
news: new_user
```

rappresenta un'altra caratteristica del sistema UNIX denominata *news*, che di solito il gestore del sistema utilizza per inviare informazioni che hanno durata maggiore di quelle contenute nel messaggio del giorno e, una volta lette, il prompt scompare al momento dell'allocazione. Il messaggio del giorno invece viene visualizzato ogni volta che accedete al sistema, fino a

quando il gestore non decide di cambiarlo o di cancellarlo. Il messaggio **news** specifica anche l'oggetto delle informazioni, che in questo caso riguardano l'utente (**new_user**). L'ultima linea visualizzata:

```
You have mail.
```

vi annuncia la presenza nella vostra "casella" di un messaggio indirizzato a voi da un altro. Questo messaggio compare ogni volta che accedete al sistema fino a quando non evadete la posta in giacenza. Diversamente da quanto accade per il messaggio **news**, una volta eseguita la lettura della posta, il messaggio **You have mail.** non scompare automaticamente al momento della vostra allocazione nel sistema.

Finalmente il processo di allocazione è terminato; il sistema vi cede il controllo, visualizzando il prompt $, e attende un vostro comando, che viene analizzato dallo shell (l'interprete di comandi).

2.2 X Window System

Se il vostro login è stato configurato per lanciare X Window System e state usando un terminale intelligente o un PC, lo schermo risulterà del tutto differente da come mostrato nel paragrafo precedente; invece di visualizzare il prompt **$** il vostro schermo apparirà simile alla Figura 2.1. Solitamente in qualche parte dello schermo apparirà un menu Workspace (area di lavoro), che viene usato per controllare la vostra sessione e scegliere tra alcune opzioni. Può essere presente una finestra File Manager simile allo schermo di Macintosh o MS Windows. Il File Manager agevola gli spostamenti dentro il file system.

Probabilmente vedrete una o più finestre **xterm**, ciascuna contenente il prompt **$**. Ognuna di queste finestre emula un terminale e consente di usare la normale linea di comandi del sistema UNIX: per questo in ognuna appare il prompt **$**. Potrete vedere anche la finestra Mailbox (**xbiff**), che avverte della presenza di posta in arrivo con una bandierina alzata sulla cassetta delle lettere nell'icona.

X Window System viene controllato mediante mouse o tastiera, quindi il vostro schermo mostrerà da qualche parte anche un puntatore di mouse; spostando il mouse vedrete il puntatore cambiare posizione. Potrà anche cambiare aspetto spostandosi in differenti aree dello schermo. Spostando il mouse e premendo il bottone sinistro (in gergo, click), potrete rendere *attive* differenti aree dello schermo. Per esempio, se spostate il puntatore del mouse sulla finestra **xterm** e premete il bottone sinistro, *selezionate* quella finestra e i caratteri battuti dalla tastiera verranno introdotti nella finestra selezionata.

X Window System viene trattato nel Capitolo 23. Per il momento, assicuratevi che la finestra **xterm** sia attiva con lo spostamento e un "click" del

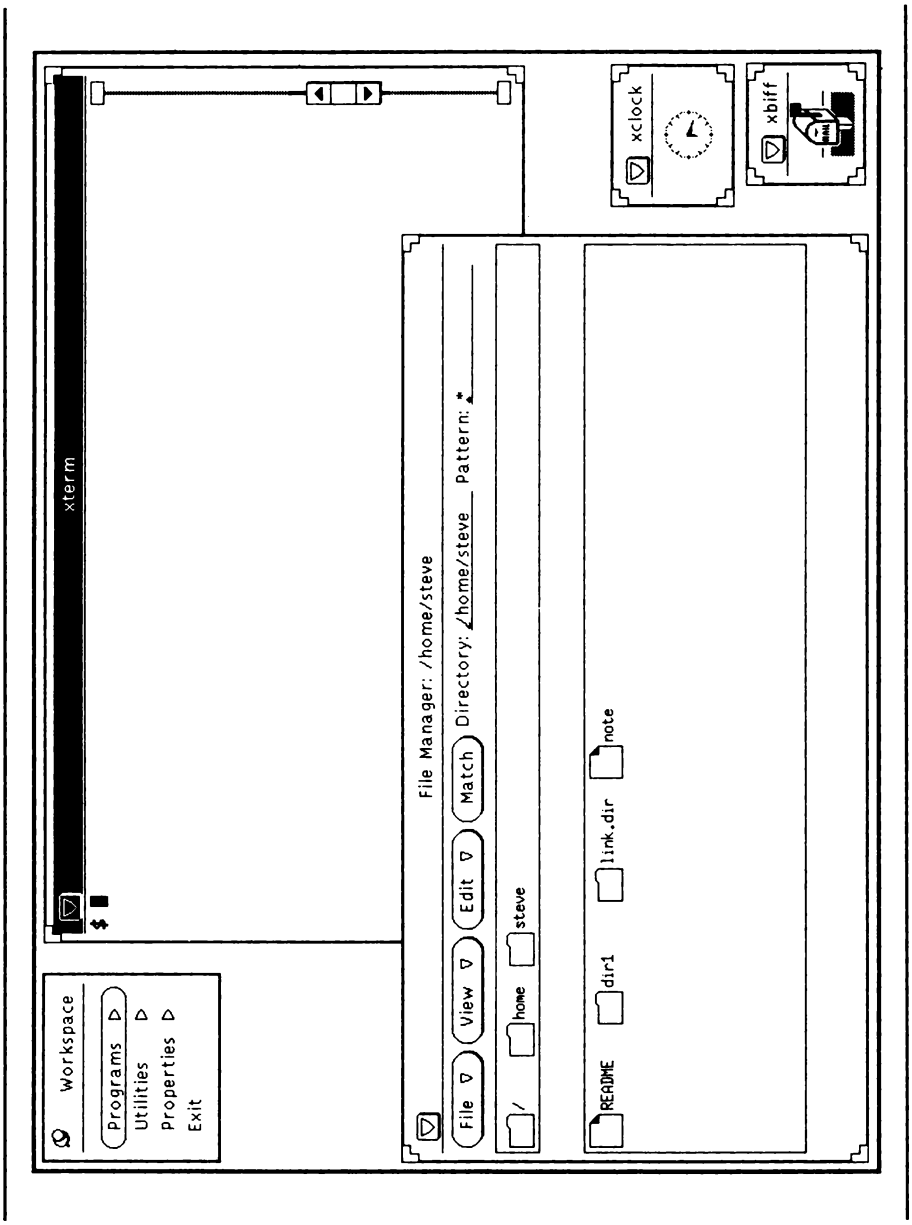


Figura 2.1 Tipica videata iniziale di X Window System.

mouse come descritto. Quando una finestra è attiva il suo bordo superiore risalta con un colore contrastante rispetto ai bordi delle altre finestre nello schermo. Potete eseguire i normali comandi del sistema UNIX all'interno della finestra attiva.

2.3 Lettura delle notizie

È consigliabile leggere innanzitutto le notizie di carattere generale in quanto possono condizionare il successivo utilizzo del sistema.

Per leggere queste notizie è sufficiente battere da tastiera:

```
$ news
```

seguito dal tasto RETURN. Il sistema visualizza il prompt $, che non deve quindi essere inserito esplicitamente da tastiera. Non dimenticate che il sistema interpreta diversamente i caratteri minuscoli e maiuscoli e che **news** e la maggior parte degli altri comandi vengono inseriti solo in caratteri minuscoli. A questo punto l'interprete di comandi (lo shell) esegue il comando **news**, per cui vengono visualizzate sullo schermo le informazioni non ancora esaminate. Quando inserite il comando **news**, la risposta del sistema è la seguente:

```
$ news

new_user (pat) Mon Jun 22 08:26:47 1991

The 'my_sys' system has a new user! Welcome 'giorgio'.... pat

$
```

Se sono presenti diverse notizie, queste verranno visualizzate successivamente. Ogni voce inizia con una linea di intestazione che identifica l'informazione, ne indica il mittente (tra parentesi) e la data di scrittura. Quando tutte le informazioni sono state visualizzate, il programma **news** termina e il controllo torna allo shell, che si mette in attesa di altri comandi.

2.4 Elenco dei file

È possibile avere l'elenco dei file che sono memorizzati nella vostra area dati privata. Il sistema UNIX dispone di un eccellente file system per memorizzare file e comandi. Dopo la fase di allocazione nel sistema, vi trovate in una zona specifica del file system, denominata directory preferenziale (home directory), che contiene generalmente i vostri file e i dati privati. Poiché

ogni utente possiede un home directory, non esiste condivisione di file tra gli utenti del sistema. Questa separazione di dati significa che utenti diversi possono scegliere gli stessi identificatori per referenziare i propri file, senza con questo generare problemi. Il file system viene trattato nel Capitolo 4.

Potete visualizzare l'elenco dei vostri file con il comando **ls** (list):

```
$ ls
note    PROVA
$
```

Inserite da tastiera il comando **ls** subito dopo il prompt $ e, una volta eseguito, il controllo ritorna allo shell che si mette in attesa del comando successivo.

In questo esempio, notiamo la presenza di due file: **note** e **PROVA**. Ricordatevi che esiste differenza tra lettere minuscole e lettere maiuscole, per cui il file **PROVA** non coincide con il file **prova**. Poiché siete principianti, il numero di file a vostra disposizione è esiguo, ma non è inusuale vedere listati che elencano un centinaio di file. Non esiste una limitazione al numero di file presenti in un directory.

In molti casi, un nuovo utente può non trovare alcun file nell'home directory e l'uscita prodotta dal comando **ls** risulta leggermente differente:

```
$ ls
$
```

Il comando **ls** non visualizza nulla e restituisce direttamente il controllo allo shell. Questo fatto dimostra una caratteristica generale del sistema UNIX: se non c'è niente da dire, il sistema non dice niente. Sebbene il comportamento del sistema possa sembrare troppo conciso in questa circostanza, il motivo del suo silenzio verrà spiegato nei capitoli successivi.

Come quasi tutti i comandi, anche **ls** dispone di diverse opzioni. Per esempio, è possibile usare la seguente linea:

```
$ ls -l
```

L'opzione **-l** (meno elle) induce il comando **ls** a generare una lista completa (long), una lista cioè che contiene un maggior numero di informazioni su un file di quante ne visualizza la versione standard del comando. Se volete che un comando esegua più azioni di quelle abituali, dovete indicarlo nella lista degli argomenti. Ogni argomento va preceduto dal segno - (meno) e prende il nome di flag od opzione. Lasciate uno spazio dopo il nome del comando, ma nessuno spazio deve separare il segno - dalla lettera che rappresenta l'argomento relativo. Il risultato del comando precedente può essere il seguente:

```
$ ls -l
total 2
-rw-rw-rw-   1 giorgio  utenti     138 Apr  5 19:34 PROVA
-rw-rw-rw-   1 giorgio  utenti     227 Apr  5 19:33 note
$
```

Questo esempio introduce alcune considerazioni sui file. Il primo termine a sinistra, **-rw-rw-rw-**, specifica i diritti di accesso al file, mentre **giorgio** indica il possessore del file, che fa parte del gruppo **utenti**; seguono la dimensione del file espressa in byte, di seguito una datazione del file, e da ultimo il nome del file. (Questi e altri attributi del file sono argomento del Capitolo 4.)

Gli argomenti nei comandi del sistema UNIX hanno la stessa funzione che viene ricoperta nell'MS-DOS e in altri sistemi operativi, cioè quella di modificare l'esecuzione del comando. A differenza di quanto accade in MS-DOS, comunque, gli argomenti dei comandi UNIX iniziano con il segno - (meno) invece che con il carattere / (barra), che in UNIX assume un altro significato.

I comandi generalmente dispongono di molti argomenti, che contribuiscono a rendere estremamente flessibile e potente il sistema. Potete consultare lo *UNIX User's Manual*, il più importante documento di riferimento sul sistema UNIX, per imparare l'esatta sintassi di tutti gli argomenti disponibili per ogni comando. (Il Capitolo 9 tratta in dettaglio lo *UNIX User's Manual.*)

2.5 Visualizzazione di file

Il vostro home directory contiene due file di cui potete visualizzare il contenuto. Il comando **cat** (concatenate file) assolve questo compito:

```
$ cat note
```

Questo comando mette in evidenza un altro aspetto generale dei comandi: i nomi dei file sono generalmente argomenti che non contengono nessun segno - o altro marcatore particolare.

Il seguente comando visualizza sullo schermo il contenuto del file:

```
$ cat note
Ciao, giorgio, benvenuto nel mio sistema. Ti divertirai sicuramente a
utilizzare il sistema UNIX; informami su come procede il tuo apprendistato
pat
$
```

Dobbiamo fare alcune precisazioni. In primo luogo, il comando **cat** visualizza il contenuto del file e poi restituisce direttamente il controllo allo shell, che si pone in attesa del comando successivo. In secondo luogo, il file viene

visualizzato senza particolari intestazioni o altro che non sia contenuto nel file. Quanto abbiamo detto rafforza le considerazioni fatte nel paragrafo precedente relative alla natura estremamente sintetica del sistema UNIX.

2.6 Cancellazione di file

Il comando **cat** non modifica il file, che rimane così ancora presente nel directory. Per cancellarlo, utilizzate il comando **rm** (remove).

```
$ rm note
$
```

Questo comando cancella il file e restituisce il controllo allo shell. Battendo ancora il comando **ls**, potete verificare l'assenza del file:

```
$ ls
PROVA
$
```

In ambiente UNIX, non è possibile recuperare un file cancellato con il comando **rm**. Essendo quindi questo comando così drastico, è utile che il sistema vi chieda conferma della cancellazione del file, prima di procedere. A questo scopo utilizzate l'opzione **–i** (interactive), per cui la linea di comando diventa:

```
$ rm –i note
rm: remove note: (y/n)?
```

Il comando **rm** vuole sapere se siete veramente decisi a cancellare il file e, se le vostre intenzioni sono queste, inserite da tastiera il carattere **y**, altrimenti qualunque altro carattere.

```
$ rm –i note
rm: remove note: (y/n)? y
$
```

Il file viene così cancellato. Se volete cancellare più di un file con un unico comando **rm –i**, per ogni file il sistema chiede la conferma dell'operazione. Potete interrompere l'esecuzione del comando, mentre è in attesa di dati, e ritornare direttamente allo shell, premendo il tasto DEL o BREAK.

```
$ rm –i note
rm: remove note: (y/n)? DEL
$
```

Il sistema elimina dal directory ogni file che avete deciso di cancellare, rispondendo affermativamente alla richiesta di conferma, prima di interrompere l'esecuzione del comando con DEL, mentre i file non interessati all'operazione di cancellazione rimangono nel directory.

Ricordate che può essere interrotta l'esecuzione di quasi ogni comando mentre è in attesa di dati in ingresso. Potete utilizzare questa tecnica per interrompere comandi che generano un'uscita troppo lunga. Pensate a cosa accade se premete il tasto DEL mentre lo shell è in attesa di vostri dati in ingresso.

2.7 Lettura della posta

Non dimenticate che quando accedete al sistema, sul video compare questo messaggio:

```
You have mail.
```

a indicare la presenza di posta che gli utenti del sistema vi hanno inviato. In caso di assenza di posta, il messaggio precedente non compare.

La posta elettronica funziona concettualmente come il servizio di posta tradizionale: permette la comunicazione tra utenti, ognuno dei quali è identificato da un indirizzo, e ogni messaggio ha una busta e un contenuto. Potete leggere la posta a voi indirizzata, utilizzando il comando **mail**:

```
$ mail
```

Questo avverte lo shell di eseguire il comando **mail** che guarda nella cassetta della posta e visualizza in sequenza ogni messaggio in essa contenuto.

```
$ mail
From Leo Mon Apr 6 18:24 MST 1991
Ciao, giorgio, benvenuto nel sistema UNIX. Perché non andiamo a pranzo
insieme per discutere del nostro progetto? – Leo

?
```

La prima linea successiva al comando rappresenta l'intestazione (postmark) e avverte che il messaggio proviene da **Leo** che lo ha inviato Lunedì 6 Aprile 1991 alle ore 18:24. Di solito non vi interessa sapere quando il messaggio è stato inviato, ma questa intestazione può essere utile in alcuni casi, come viene spiegato nel Capitolo 14.

Dopo aver visualizzato il messaggio, il programma **mail** si mette in attesa di dati in ingresso, visualizzando il carattere **?**, per sapere quali azioni intraprendere sul quel messaggio. Potete cancellare il messaggio, richiederne un

altro (se esiste), salvarlo in un file o richiedere altre azioni. Per cancellare il messaggio, è sufficiente premere **d**, seguito da un ritorno a capo:

```
$ mail
From Leo Mon Apr 6 18:24 MST 1991
Ciao, giorgio, benvenuto nel sistema UNIX. Perché non andiamo a pranzo
insieme per discutere del nostro progetto? – Leo

? d
$
```

Il programma **mail** cancella il messaggio senza dare spiegazioni.

Poiché non ci sono altri messaggi, il programma termina e il controllo ritorna allo shell. Se invece la vostra posta non si è esaurita, ciascun messaggio viene visualizzato dopo che avete finito di trattare il precedente. Potete dunque leggere ogni messaggio successivo ed eliminarlo in un modo o nell'altro.

Se state usando X Window System potrete notare che la bandierina della cassetta postale si abbassa non appena avete cancellato il messaggio dalla posta; poiché UNIX è un sistema multitasking numerosi fenomeni possono accadere simultaneamente sul vostro schermo. Ugualmente è normale che la bandierina venga rialzata non appena vi perviene un nuovo messaggio postale.

Il comando **mail** dispone di un aiuto interattivo per ricordarvi tutte le opzioni che avete a disposizione quando compare il prompt. Se non avete cancellato un messaggio, potete inserire il carattere **?** invece di **d**, come indicato in Figura 2.2. Nel Capitolo 14 tratteremo le diverse opzioni che sono disponibili con il comando **mail**, ma è importante notare per ora il nutrito insieme di operatori che potete impiegare per trattare i messaggi.

2.8 Invio della posta

Ovviamente, oltre a ricevere posta da un altro utente, potete anche inviare messaggi. Il comando **mail** viene utilizzato sia per mandare posta che per leggerla:

```
$ mail leo
```

Gli argomenti che specificate nel comando **mail** lo informano sull'azione da intraprendere. In questo caso abbiamo indicato il nome del destinatario del messaggio. Mentre il comando **cat** riceve come argomento il nome di un file, poiché la sua funzione è di concatenare e visualizzare i file, il comando **mail** rende possibile la comunicazione tra utenti, per cui come argomento è indispensabile specificare l'identificatore dell'utente (o degli utenti) a cui si intende spedire il messaggio. Molti comandi nel sistema UNIX fanno riferi-

```
$ mail
From leo Mon Apr 6 18:24 MST 1991
Ciao, giorgio, benvenuto nel sistema UNIX. Perché non usciamo a pranzo
insieme per discutere del nostro progetto? – Leo

? ?
Usage:
?          print this help message
#          display message number #
–          print previous
+          next (no delete)
! cmd      execute cmd
<CR>       next (no delete)
a          position at and read newly arrived mail
d [#]      delete message # (default current message)
dp         delete current message and print the next
dq         delete current message and exit
h a        display all headers
h d        display headers of letters scheduled for deletion
h [#]      display headers around # (default current message)
m user     mail (and delete) current message to user
n          next (no delete)
p          print (override any warnings of binary content)
P          override default 'brief' mode; display ALL header lines
q, ^D      quit
r [args]   replay to (and delete) current letter via mail [args]
s [files]  save (and delete) current message (default mbox)
u [#]      undelete message # (default current message)
w [files]  save (and delete) current message without header
x          exit without changing mail
y [files]  save (and delete) current message (default mbox)
?
```

Figura 2.2 Opzioni di aiuto del comando **mail**.

mento agli utenti mediante il relativo identificatore o login id, ovvero molti comandi si aspettano il login id in luogo del nome reale dell'utente. Come la maggior parte dei comandi, **mail** accetta più di un argomento. Potete scrivere da tastiera:

```
$ mail leo giorgio
```

per inviare lo stesso messaggio a **leo** e a voi stessi.

Una volta lanciato il precedente comando, il programma **mail** inizia l'esecuzione, che interrompe per permettervi di scrivere il messaggio. Per esempio:

```
$ mail leo giorgio
Leo, grazie per il benvenuto e l'invito a pranzo, che accetto volentieri.
Ci vediamo più tardi. giorgio
```

Potete scrivere un messaggio di qualsiasi lunghezza. Se fate un errore di battitura, potete correggerlo con BACKSPACE. Una volta che avete premuto il tasto RETURN, comunque, il sistema UNIX (e, in particolare, il programma **mail** in questo caso) ha acquisito i dati che da questo punto in poi non possono più essere facilmente cancellati. Se desiderate interrompere l'esecuzione del programma e ritornare allo shell senza inviare alcun messaggio, premete il tasto DEL.

Se non intendete interrompere l'esecuzione del programma **mail** ma semplicemente avvertirlo che avete terminato di inserire il testo del messaggio, premete CTRL-D, come nell'esempio seguente. Anche se spesso CTRL-D viene indicato con la D maiuscola, è sufficiente premere il tasto CTRL e direttamente il tasto D senza premere il tasto delle maiuscole.

```
$ mail leo giorgio
Leo, grazie per il benvenuto e l'invito a pranzo, che accetto volentieri.
Ci vediamo più tardi. giorgio
CTRL-D
You have mail.
$
```

Il programma **mail** invia il messaggio (se non ci sono errori) e restituisce il controllo allo shell, per l'esecuzione del comando successivo.

Se non esistono errori in fase di compilazione o esecuzione del comando, il sistema non visualizza alcun messaggio.

Nell'esempio precedente, si nota che lo shell visualizza un altro messaggio: **You have mail.** Infatti, poiché vi siete spediti una copia del messaggio, lo shell avverte che avete in giacenza della posta. Questo messaggio viene visualizzato una sola volta fino a quando non entrate di nuovo nel sistema e potete scegliere di non leggere subito la posta (soprattutto se ve la siete spedita). Poiché il programma **mail** invia il messaggio prima di restituire il controllo allo shell, e lo shell si accerta dell'eventuale giacenza di nuova posta, il messaggio **You have mail.** compare sul video. Questa situazione si verifica dopo che il programma **mail** ha terminato l'esecuzione, ma prima che lo shell visualizzi il prompt per ricevere un altro comando. Il sistema UNIX fornisce spesso servizi di questo tipo, per cui potete occasionalmente leggere su video alcuni messaggi inviati dallo shell o da altri programmi che sono contemporaneamente in esecuzione nel sistema. Poiché UNIX è un sistema multitasking, non c'è da stupirsi se esistono altri processi attivi oltre alla vostra sessione.

Il sistema UNIX dispone di molti strumenti efficienti (i Capitoli 14 e 15 ne descrivono le caratteristiche) per la comunicazione tra utenti e tra macchine differenti.

2.9 Elenco degli utenti in attività

In un sistema multiutente è possibile che oltre a voi altre persone stiano utilizzando il sistema da diversi terminali collegati alla macchina. Normalmente non notate la presenza di questi utenti perché il sistema vi mette in condizione di pensare di essere gli unici a sfruttare la macchina. Comunque esiste la possibilità di sapere chi oltre voi si trova nel sistema e avere un'idea di cosa stia facendo.

Un'altra caratteristica tipica del sistema UNIX è che gli utenti di una macchina costituiscono un gruppo di persone che hanno interessi in comune, come, per esempio, la comunicazione e la condivisione di file. Esiste dunque in UNIX una delicata contrapposizione tra la libertà di spaziare nel sistema concessa agli utenti e le restrizioni dettate da considerazioni di sicurezza. Il sistema fornisce strumenti efficienti per definire lo stato di protezione degli identificatori di utente, dei file e dei programmi di uso comune e dei file privati. (Il Capitolo 22 tratta questo argomento in maggior dettaglio.)

Potete scoprire chi sta utilizzando il sistema tramite il comando **who:**

```
$ who
```

Inserite il nome del comando, come al solito dopo il prompt dello shell, seguito da RETURN.

```
$ who
leo          console     Apr  7 14:05
giorgio      tty00s      Apr  7 16:41
$
```

L'esito del comando **who** compare su video e il controllo ritorna allo shell. Viene quindi visualizzata la lista degli utenti che sono presenti nel sistema: **leo** e **giorgio** (cioè voi) in questo caso. Sono disponibili altre informazioni riguardanti il tipo di terminale utilizzato: voi operate con il terminale remoto **tty00s**, mentre **leo** sta usando la console di sistema che è direttamente collegata all'elaboratore. La parte restante di ogni linea indica la data e l'ora in cui l'utente è entrato nel sistema: **giorgio** lo ha fatto alle 16 e 41, ora locale.

In un sistema UNIX di grandi dimensioni ci possono essere più di un centinaio di utenti allocati contemporaneamente, mentre in macchine più piccole può capitare che ce ne sia solo uno. Non importa; il sistema in ogni caso mette l'utente in condizione di pensare di essere l'unico utilizzatore della macchina.

2.10 Modifica della propria password

Prima di chiudere la sessione, uscendo dal sistema, potete modificare la vostra password. L'identificatore di utente e la password costituiscono la chiave di sicurezza nel sistema UNIX. Se qualcuno riesce a conoscerli, può accedere al sistema con grave rischio per i vostri file. Inoltre, potrebbe cambiare la vostra password, impedendovi di entrare nel sistema fino a quando il gestore non ripristina la situazione originale.

Quindi, tenete segreta la password e cambiatela di frequente. Naturalmente, il vostro identificatore deve essere noto agli altri utenti, se volete ricevere posta, ma la password è personale e non deve essere comunicata per nessun motivo.

Se siete nuovi utenti della macchina, è rischioso aspettare che il gestore del sistema vi designi un identificatore e una password di valori preassegnati. Questa infatti è una potenziale violazione di sicurezza perché la maggior parte dei gestori di sistemi utilizza per definire le password per i nuovi utenti la semplice regola di derivarla dall'identificatore o da qualche altra semplice parola facile da scoprire, per cui è consigliabile che scegliate rapidamente una vostra password privata.

Se il gestore del sistema vi ha assegnato un identificatore di utente ma non una password, probabilmente non compare su video il prompt **password:**, ma potete accedere direttamente al sistema tramite il prompt **login:**. In ogni caso definite rapidamente una password.

Nessuno può scoprire la vostra password perché viene subito codificata, per cui se ve la dimenticate, nessuno può aiutarvi a ricordarla. Il solo aiuto che può darvi il gestore del sistema è di cancellare la vecchia password e di assegnarvene una nuova. Quindi fate attenzione a non dimenticarvi la password, adottando metodi intelligenti per ricordarvela.

È conveniente scegliere per la password un termine che è facile da ricordare per voi, ma difficile da scoprire per gli altri. Il sistema generalmente richiede che la password non sia più corta di sei caratteri e che contenga almeno una cifra o un altro carattere non alfabetico. Queste regole, che derivano da considerazioni di sicurezza, permettono di comporre un elevato numero di password, rendendo estremamente impossibile una loro identificazione da parte di persone non abilitate.

Una volta scelta la nuova password, potete informare il sistema utilizzando il comando **passwd**. Notate la particolare compitazione del nome di questo comando.

```
$ passwd
passwd: Changing password for giorgio
Old password:
```

Il sistema non permette di cambiare password facilmente. Prima dovete inserire la password corrente (quella vecchia), che non viene in ogni caso vi-

sualizzata, per convincere il programma **passwd** che siete autorizzati a modificarla. Se, nell'inserimento della password, fate un errore di battitura, il programma **passwd** termina e restituisce il controllo allo shell, dandovi una seconda opportunità di modifica. Quando finalmente la password è stata scritta correttamente, il comando **passwd** vi chiede di specificare la nuova password:

```
$ passwd
passwd: Changing password for giorgio
Old password:
New password:
```

Inserite la nuova password, che non viene visualizzata.

```
$ passwd
passwd: Changing password for giorgio
Old password:
New password:
Re-enter new password:
```

Ora il programma **passwd** vi chiede di inserire una seconda volta la nuova password, per eseguire una doppia verifica. Se le due stringhe non coincidono, il comando **passwd** visualizza il messaggio di errore e vi dà l'opportunità di ritentare.

```
$ passwd
passwd: Changing password for giorgio
Old password:
New password:
Re-enter new password:
They don't match; try again.
New password:
```

Nessun cambiamento è stato apportato e potete inserire nuovamente la password. Se ora premete il tasto DEL, **passwd** termina l'esecuzione mantenendo la vecchia password. Se avete inserito la stessa password per due volte, il sistema assume come corrente la nuova password e restituisce il controllo allo shell che si mette in attesa del successivo comando.

2.11 Uscita dal sistema (logging out)

Al termine della sessione di lavoro, dovete eseguire la procedura di chiusura colloquio. Una volta che uscite, UNIX libera il terminale e lo rende disponibile ad altri utenti, o a un vostro riutilizzo, anche con un identificatore utente diverso. Scollegandovi dal sistema si evita che persone non abilitate

possano utilizzare il vostro terminale, soprattutto se il terminale si trova in un luogo pubblico. I due più importanti fattori che garantiscono la sicurezza del sistema UNIX sono di tenere le password al sicuro e di scollegarsi ogni volta che ci si allontana dal terminale. Se la macchina si trova però in una stanza alla quale solo voi avete accesso, queste misure di sicurezza sono superflue.

La procedura di uscita differisce a seconda che la vostra sessione sia stata o no lanciata in X Window System. Se siete sotto X, selezionate col click il menu Workspace e scegliete col click la voce Exit. Può esservi richiesto di confermare la richiesta con una finestra temporanea; in tal caso date il click sul bottone Yes. Dopo questo, X Window System scompare dal video e potrete venire scollegati dal sistema oppure lasciati in un normale shell. Introducendo **exit** in una finestra **xterm**, la finestra verrà cancellata dal video (e dal sistema), tuttavia non verrà annullato l'intero ambiente X e voi non verrete scollegati. Se state usando X dovete selezionare Exit dal menu Workspace per essere certi di venire scollegati dal sistema.

UNIX definisce due procedimenti differenti per scollegarsi. Il primo utilizza il comando **exit**:

```
$ exit

Welcome to the AT&T 386 UNIX System.
System name: my_sys

login:
```

Il sistema UNIX interrompe l'esecuzione dello shell, si resetta e visualizza il prompt iniziale **login**. Potete ora spegnere il terminale o entrare nuovamente.

Al posto di utilizzare il comando **exit**, potete uscire dal sistema premendo CTRL-D, che rappresenta il carattere di fine file e che lo shell interpreta come un segnale di disallocazione. Il Capitolo 3 spiega il significato del carattere di fine file.

Alcuni sistemi UNIX rendono immediato il processo di scollegamento se spegnete il terminale o la console. I terminali che sono collegati alla macchina via linee telefoniche e modem di solito seguono questa procedura. Comunque, se non siete sicuri del procedimento adottato dal vostro sistema, uscite esplicitamente prima di agganciare il telefono o spegnere il terminale.

2.12 Approfondimenti

Al terminale, per concludere la linea di comando o altre linee di testo, premete il tasto RETURN. Nella terminologia UNIX viene abitualmente chiamato carattere *newline* il simbolo posto alla fine di una linea di testo. In questo

libro il termine *newline* potrà essere usato per indicare il tasto di fine linea. Usualmente nei terminali il tasto RETURN agisce come il newline, in realtà il carattere ASCII per l'avanzamento linea è il carattere *linefeed* (LF).

In pratica, il tasto RETURN (CR+LF) ha lo stesso effetto di LF; verificate cosa accade sul vostro terminale e al vostro sistema se terminate una linea di testo con LF (CTRL-J) invece di RETURN.

CONTROLLO DELL'OUTPUT SU TERMINALE

Talvolta potrete riscontrare che l'output di un comando consiste in un numero di linee maggiore di quelle che il vostro terminale (o la vostra finestra **xterm**) può visualizzare; in questo caso, l'output scorrerà fuori della linea superiore del video prima che possiate averlo letto. Il sistema UNIX dispone di due strumenti per ovviare all'inconveniente controllando l'output. Un primo metodo usa il comando filtro **more** che consente di far procedere l'output di una schermata alla volta; il Capitolo 7 tratta il comando **more** e la sua utilizzazione. Un secondo metodo può essere impiegato quando non avete previsto la necessità di usare il comando **more**; in questo caso potete digitare CTRL-S per arrestare immediatamente l'output sul video e CTRL-Q per lasciarlo ripartire esattamente dal punto ove l'avete interrotto. Potete arrestare e rilanciare l'output tante volte quanto vi occorre con quest'ultimo metodo, ma dovete aver cura di battere CTRL-Q per ultimo, o il vostro terminale apparirà congelato.

Capitolo 3

Introduzione allo shell

Lo shell è il programma che sta in attesa dei comandi introdotti dal terminale, li riceve e li traduce in istruzioni nella sintassi interna del sistema. Il termine shell indica effettivamente la funzione che svolge, cioè una funzione di schermo o interfaccia che si colloca tra la parte più interna del sistema operativo e il mondo esterno. Lo shell esegue un gran numero di compiti, tra i più importanti e meno scontati di UNIX, ed è oggetto di continui miglioramenti. Molte funzioni offerte da UNIX derivano dallo shell, che può quindi essere considerato una parte integrante del sistema. In questo capitolo esaminiamo alcune funzioni di UNIX, focalizzando l'attenzione principalmente sulle utility offerte dallo shell.

3.1 I comandi nel sistema UNIX

I comandi sono generalmente programmi che lo shell ricerca ed esegue in risposta a istruzioni che vengono dettate da tastiera, come per esempio **mail** e **who**. Lo shell è in realtà un supporto più utile e organico di una semplice interfaccia tra l'utente e il sistema che esegue i comandi.

Lo shell vi aiuta a utilizzare la macchina, essendo, innanzitutto, un interprete di comandi che può espandere e modificare il comando (in accordo a regole predefinite) prima di eseguirlo. In quest'ottica, particolare importanza assumono i metacaratteri (wildcard) e gli operatori di connessione dei comandi, che possono rendere una linea di comandi più generalizzata e flessibile. D'altra parte i comandi possono sfruttare le utility che l'ambiente di lavoro gestito dallo shell mette a disposizione per migliorare la loro esecuzione.

3.2 La struttura di un comando

La struttura di un comando comprende generalmente, oltre al nome del comando, anche alcuni argomenti che rappresentano i flag di modifica (opzioni) del comando, preceduti dal segno – (meno), oppure nomi di file o altri identificatori di utente.

I comandi assumono questa forma:

```
$ pr -d note
```

Il comando **pr** permette di stampare i file. L'uscita prodotta da un comando viene inviata generalmente su terminale piuttosto che su stampante, per cui **pr** è simile al comando **cat**. Mentre **cat**, però, permette di concatenare i file, **pr** si occupa di impaginarli e incolonnarli.

Nella sintassi dei comandi del sistema UNIX, le opzioni seguono il nome del comando e precedono i principali argomenti. La linea di comando **pr -d** visualizza in doppia spaziatura il file **note** al terminale. L'argomento principale del comando è il nome del file, mentre l'opzione (**-d** in questo caso, a indicare la doppia spaziatura) ha il compito di modificare la funzione base del comando. Comunque, per la maggior parte dei comandi, non è necessario specificare né l'opzione né il nome del file come argomenti.

È possibile aggiungere al comando **pr** una seconda opzione:

```
$ pr -n -d note
```

In questo caso, **pr** realizza una stampa in doppia spaziatura (**-d**) con inclusa la numerazione di linea (**-n**). Ricordatevi che le opzioni precedono sem-

pre il nome dei file nell'elenco degli argomenti di un comando. Una forma equivalente del comando precedente può essere la seguente:

```
$ pr -nd note
```

Potete combinare le opzioni e farle precedere da un unico -.

L'esempio che abbiamo presentato genera un listato in doppia spaziatura sul terminale, con la numerazione di linea progressiva "1, 2, 3, 4, ...". È possibile utilizzare lo stesso comando per produrre un'uscita con la numerazione dispari di linea: "1, 3, 5, 7, ...".

```
$ pr -n2 -d note
```

L'opzione **-n** del comando **pr** può avere un proprio argomento, che indica l'intervallo tra un numero di linea e il suo successivo, per cui **pr -n2** conteggia le linee per due e non per uno. Potete utilizzare anche una delle seguenti forme alternative:

```
$ pr -n2 -d note
$ pr -d -n2 note
$ pr -d -n 2 note
$ pr -d              -n 2 note
$ pr -dn2 note
```

Sono tutte forme equivalenti dello stesso comando poiché il sistema non considera l'ordine con cui le opzioni sono scritte, né impone che ogni opzione sia preceduta dal segno - o che ci siano spazi aggiuntivi tra le opzioni.

Le seguenti linee di comando, comunque, non soddisfano i vostri intendimenti:

```
$ pr - n2 -d note
$ pr -nd2 note
$ pr note -nd2
```

È necessario anteporre all'opzione direttamente il segno -, senza inserire spazi, e far precedere le opzioni (se esistono) ai nomi dei file. Inoltre, dovete accertarvi che l'argomento che modifica una opzione, come per esempio il numero 2 nel nostro caso, segua l'opzione che modifica. Generalmente, quando fate un errore, il comando risponde con un messaggio che vi riconduce alla forma corretta, anche se qualche volta il messaggio è incomprensibile e il comando non esegue correttamente la sua funzione, pur non generando spiacevoli conseguenze.

È possibile aggiungere un secondo nome di file al comando **pr**

```
$ pr note PROVA
```

L'elenco dei nomi di file, separati da spazi o da caratteri di tabulazione, compare sulla stessa linea di comando. Per separare gli argomenti è necessario interporre almeno uno spazio.

Nel seguente esempio:

```
$ pr -dl note PROVA
$ cat note
$ pr -d -l 2 PROVA
```

la prima linea di comando ha tre argomenti, la seconda uno. Gli argomenti vengono numerati a partire dal nome del comando, **pr** in questo caso, che viene assunto come argomento zero; **-dl** rappresenta il primo argomento e i nomi dei due file costituiscono rispettivamente il secondo e terzo argomento. La terza linea dell'esempio presenta il comando **pr** con quattro argomenti, anche se il numero 2 è associato all'opzione **-l**. Gli argomenti sono separati da spazi, cosicché il comando:

```
$ pr -d -l 2 PROVA
```

corrisponde a una linea di comando **pr** con quattro argomenti, di cui due sono opzioni, uno è associato a una opzione e l'ultimo è il nome di un file. Molti comandi hanno come argomenti più nomi di file. Confrontate le differenze esistenti tra l'uscita prodotta dal comando

```
$ pr note PROVA
```

e l'uscita generata dal seguente comando:

```
$ cat note PROVA
```

Le differenze esistenti sono dovute alle diverse funzioni che i due comandi realizzano. Il comando **cat** viene utilizzato per concatenare più file, cosicché i due file si fondono in uno solo in uscita, mentre **pr** si occupa principalmente di impaginare i file e agisce in successione sui due file.

3.3 Espansione della linea di comando

Lo shell semplifica la specificazione dei nomi di file come argomenti di un comando. Tre caratteri speciali, denominati metacaratteri (wildcard), possono sostituire completamente o in parte i nomi dei file.

Nel directory che contiene i file **note** e **PROVA**, il comando

```
$ cat *
```

equivale a **cat note PROVA**. In altre parole, il carattere * (asterisco) induce lo shell a referenziare tutti i file presenti nel directory corrente. In realtà lo shell esamina i file, ricostruisce la linea di comando sostituendo i metacaratteri con i nomi completi dei file e poi esegue il comando.

Il metacarattere * può anche essere inserito all'interno di un nome di file. Per esempio,

```
$ cat PR*
```

visualizza tutti i file che iniziano con PR, mentre

```
$ cat *AD*
```

visualizza tutti i file che contengono AD al loro interno.

A differenza di quanto accade in MS-DOS dove il metacarattere * può solo sostituire il nome o l'estensione del file, in ambiente shell sostituisce qualunque sequenza di caratteri sia presente nel comando.

Il secondo metacarattere è ? (punto di domanda) che può sostituire qualunque carattere compaia nella posizione corrispondente del nome del file. Per esempio,

```
$ cat ?ROVA
```

visualizza il contenuto del file **PROVA**, analogamente al seguente comando:

```
$ cat ?R?VA
```

Al contrario, il comando

```
$ cat P?VA
```

non trova il file nel directory perché il metacarattere **?** è in grado di sostituire solo un carattere nel nome del file.

Il terzo metacarattere permette di specificare una lista di caratteri che possono sostituire un carattere nel nome di file, come l'operatore **?**, limitando la sostituzione a un insieme selezionato di caratteri. I caratteri parentesi quadra sinistra e destra ([ ]) possono racchiudere una lista di caratteri che possono essere presenti nella posizione corrispondente del nome file, per esempio:

```
$ cat PROV[AEO]
```

Questo visualizzerebbe i file PROVA, PROVE, PROVO, se esistono. Le parentesi quadre agiscono come il metacarattere **?** su di un solo carattere nel-

la posizione corrispondente; più caratteri possono venire individuati con più operatori parentesi quadre, per esempio:

```
$ cat [PpTt][Rr]OV[AEO]
```

Potete anche utilizzare questi metacaratteri nello stesso comando. Per esempio,

```
$ ls ??O*
```

visualizza i nomi di tutti i file che hanno come terzo carattere la lettera O. L'esempio precedente permette di trovare un file di nome **PRO**. In realtà, il metacarattere * induce lo shell a sostituire tutte le stringhe di zero o più caratteri. D'altra parte, il metacarattere ? sostituisce un carattere presente nel nome del file. Per esempio, il comando

```
$ ls PROVA?
```

non riesce a trovare il file **PROVA**, a differenza del comando

```
$ ls PROVA*
```

Alla stessa maniera, le parentesi quadre richiedono che un carattere situato nella posizione equivalente del nome del file corrisponda a uno dei caratteri contenuti nella lista; se non viene trovato alcun file con uno dei caratteri della lista in quella posizione del nome, allora nessun nome di file viene espanso nella linea di comando.

3.4 Variabili di ambiente

Lo shell permette di definire le cosiddette variabili di ambiente, cioè stringhe di caratteri che assumono il formato *nome=valore*, dove *nome* è una qualsiasi sequenza di caratteri che non comprende il segno $ (dollaro) e non contiene spazi, mentre *valore* è una qualsiasi sequenza di caratteri, spazi compresi. Molte variabili di ambiente sono generalmente associate agli identificatori di utente e variano in relazione al sistema, al software disponibile e alle vostre esigenze.

Potete inizializzare una variabile di ambiente specificando allo shell l'istruzione *nome=valore*:

```
$ ESEMPIO = "salve gente"
```

Lo shell riconosce questa istruzione, la interpreta come la definizione di una variabile di ambiente e ne memorizza il nome e il valore. Si adotta la

convenzione di nominare le variabili di ambiente in caratteri maiuscoli, anche se non è obbligatorio. Immediatamente dopo il nome, aggiungete il segno = (uguale) senza inserire spazi, e poi il valore che vi interessa. Dovete delimitare tra virgolette la parte *valore* dell'assegnamento indicato nell'esempio perché contiene un carattere spazio. Riprenderemo questo concetto più avanti.

Prima di utilizzarla, non è necessario definire o dichiarare il nome della variabile di ambiente. Lo shell verifica se quel nome è già stato impiegato e, in caso affermativo, modifica il valore della variabile. Se il nome non è mai stato utilizzato, lo shell lo associa alla variabile.

Quando volete ottenere il valore di una variabile di ambiente, anteponete al nome il segno $ (dollaro). Questa convenzione induce lo shell a riconoscere la stringa successiva come una variabile e a calcolarne il valore; in caso contrario lo shell interpreta la stringa come una semplice sequenza di caratteri.

Con il comando **echo** potete visualizzare il valore di una variabile di ambiente.

```
$ echo $ESEMPIO
```

Il seguente comando induce lo shell a scrivere il valore della variabile su terminale.

```
$ echo $ESEMPIO
salve gente
$
```

Immaginate cosa accade se inserite da tastiera il seguente comando:

```
$ echo ESEMPIO
```

Le variabili di ambiente possono sostituire i nomi dei comandi o i relativi argomenti. Per esempio:

```
$ XYZ="cat note"
$ $XYZ
Ciao, giorgio, benvenuto nel sistema. Aspettiamo che ci insegni a utilizzare
al meglio il sistema UNIX. pat
$
```

Prima di eseguire il comando, lo shell sostituisce in esso il valore associato al nome della variabile. Potete utilizzare questo meccanismo per assegnare ai comandi di notevole lunghezza e frequentemente usati una variabile di ambiente di lunghezza ridotta; così facendo, potete inserire da tastiera solo la variabile di ambiente (preceduta da $ perché sia interpretata dallo shell!) e lo shell esegue il valore associato.

Potete utilizzare il comando **echo** per visualizzare stringhe di caratteri su terminale.

```
$ echo salve $ESEMPIO gente
salve salve gente gente
$
```

Potete anche inserire la variabile di shell all'interno di un'altra stringa di caratteri.

```
$ echo salve${ESEMPIO}gente
salvesalve gentegente
$
```

Poiché lo shell non sa se intendete riferirvi alla variabile $ESEMPIO o alla variabile $ESEMPIOgente, dovete proteggere la stringa ESEMPIO quando non è delimitata da spazi. Le parentesi graffe assolvono questo compito, quando seguono immediatamente il segno $. Lo shell di solito ignora le variabili indefinite. Pensate a cosa può accadere se inserite da tastiera la seguente linea di comando:

```
$ echo salve$ESEMPIOgente
```

Esistono sempre da 10 a 30 variabili di ambiente che sono associate permanentemente agli identificatori di utente. Il sistema le assegna generalmente quando vi collegate e permangono fino a quando non uscite. Lo shell utilizza alcune di queste variabili permanenti, mentre altre possono essere utilizzate da alcuni pacchetti software applicativi per soddisfare le proprie esigenze. Per verificare quali siano le variabili di ambiente correntemente assegnate, utilizzate il seguente comando:

```
$ env
```

Il comando **env** visualizza una lista completa, i cui elementi presentano la forma *nome=valore*.

```
$ env
LOGDIR = /home/giorgio
HOME = /home/giorgio
SHELL = /usr/bin/ksh
MAIL = /var/mail/giorgio
EDITOR = /usr/bin/vi
LOGNAME = giorgio
TERM = xterm
PATH = :/home/giorgio/lib:/sbin:/usr/sbin:/usr/bin:/usr/X/bin:/usr/ucb
TZ = MST5MDT
$
```

Potete disporre di variabili di ambiente in quantità molto superiore rispetto all'esempio precedente.

È consentito aggiungere o cambiare variabili di ambiente a vostro piacimento, ma fate attenzione a non modificare il valore delle variabili di ambiente esistenti perché può esserci qualche comando o applicazione che le sta utilizzando. Prima di scegliere il nome di una variabile di ambiente, verificate se è già stata definita in precedenza. Se tentate di visualizzare il valore di una variabile di ambiente che non esiste, lo shell non visualizza niente.

```
$ echo $ASSENTE

$
```

Il comando **echo** in questo caso conferma che potete utilizzare la variabile di ambiente per le vostre esigenze.

3.5 Protezione degli argomenti della linea di comando

Lo shell interpreta una stringa di caratteri che contiene spazi come l'insieme di più argomenti in quanto il carattere spazio assume il compito di separatore. Consideriamo il seguente esempio:

```
$ ESEMPIO = salve gente
gente: not found
$
```

Il messaggio di errore che segue l'assegnamento della variabile di ambiente ESEMPIO compare su video perché lo shell interpreta la stringa **salve gente** come due argomenti separati. Poiché la coppia *nome* = *valore* della variabile di shell richiede che il valore sia un unico argomento, lo shell gli assegna il valore **salve**. Successivamente lo shell cerca di interpretare il successivo argomento, **gente**, come un comando da eseguire; comunque, poiché non esiste alcun comando denominato **gente**, lo shell deduce che il comando non è referenziabile.

Per risolvere questo problema dovete delimitare il valore tra virgolette. Per esempio:

```
$ ESEMPIO = "salve gente"
$ echo $ESEMPIO
salve gente
$
```

Una volta che avete racchiuso tra virgolette la stringa, lo shell la interpreta come unico argomento. Potete ripetere questo procedimento per ogni argomento presente sulla linea di comando.

Le virgolette di delimitazione possono contenere al loro interno anche variabili di ambiente che lo shell interpreta correttamente.

```
$ ESEMPIO = "Il mio identificatore di utente è $LOGNAME"
$ echo $ESEMPIO
Il mio identificatore di utente è giorgio
$
```

È possibile anche evitare la valutazione delle variabili di ambiente da parte dello shell, pur includendo i loro nomi nella linea di comando. Per esempio:

```
$ ESEMPIO = 'Utilizzate $LOGNAME per determinare il mio identificatore di
utente'
$ echo $ESEMPIO
Utilizzate $LOGNAME per determinare il mio identificatore di utente
$
```

Potete anche delimitare gli argomenti della linea di comando con un apice se avete delle virgolette presenti al loro interno:

```
$ echo '"salve gente"'
"salve gente"
$
```

In questo esempio le virgolette non sono più operatori shell perché sono protetti da apici, ma fanno parte della stringa di caratteri che intendete visualizzare.

3.6 PS1

Lo shell utilizza alcune variabili che non sono comprese nell'elenco prodotto dal comando **env**. Una di queste, **PS1** (prompt symbol, livello uno), rappresenta il valore del prompt dello shell, che finora è stato identificato dal simbolo $ (dollaro). Potete verificare il suo valore tramite il comando **echo**.

```
$ echo $PS1
$
$
```

In questo esempio, il comando **echo** ha visualizzato il valore del prompt; poi lo shell lo ha stampato per avvertire che è in attesa del successivo comando. Potete modificare il valore di PS1 proprio come accade per qualunque altra variabile di shell.

```
$ PS1 = "salve: "
salve:
```

Ora il prompt di shell non è più **$**, ma il valore che avete assegnato, cioè **salve:** in questo caso, ma un qualunque carattere in generale.

Le variabili di shell vengono valutate nel momento in cui sono inserite da tastiera e non quando vengono eseguite. Per esempio:

```
$ ESEMPIO = "Inserisci il comando:"
$ PS1 = $ESEMPIO
Inserisci il comando: ESEMPIO = salve
Inserisci il comando: echo $ESEMPIO
salve
Inserisci il comando:
```

PS1 è stato inizializzato al valore $ESEMPIO, ma quando cambiate di nuovo $ESEMPIO, $PS1 non cambia. Il comando per inizializzare PS1 a $ESEMPIO è stato valutato quando a $ESEMPIO era stato assegnato il valore **Inserisci il comando:**, per cui questo diventa il nuovo valore di PS1, non il valore di $ESEMPIO quando $PS1 viene visualizzato.

3.7 Standard input e standard output

Finora avete inserito i comandi battendoli sulla tastiera e i risultati prodotti sono stati visualizzati sullo schermo. Generalmente i comandi ricevono dati in ingresso che, una volta elaborati, producono una sequenza di caratteri in uscita. Queste sequenze di caratteri in ingresso e in uscita sono paragonabili a flussi di dati (stream) in quanto non presentano una struttura interna ben definita. Il comando dunque riceve in ingresso una lunga sequenza di caratteri, anche se alcuni di essi (come lo shell!) si mettono in azione su sequenze di caratteri che terminano con un LF (ASCII 10 ovvero line feed o newline come spesso è citato nei manuali UNIX). Il file di ingresso a un comando viene denominato standard input, mentre il file di uscita prende il nome di standard output. Per esempio, il comando:

```
$ echo $PS1
```

scrive sullo standard output.

Lo shell generalmente si organizza in modo che lo standard input di un comando provenga da tastiera, mentre lo standard output sia inviato su video. Comunque, è possibile ridirigere lo standard input e lo standard output in altri file. Per esempio:

```
$ cat note > n.copia
$
```

Il carattere > (maggiore di) induce lo shell a ridirigere lo standard output a un file che nel nostro esempio si chiama **n.copia**. Lo shell crea il file se non esiste oppure lo cancella, se è già esistente, prima di riportare i risultati prodotti dall'esecuzione del comando. In questo modo, potete facilmente distruggere un file esistente, per cui verificate bene prima di ridirigere l'uscita.

Potete anche ridirigere l'ingresso, utilizzando il simbolo < (minore di), in modo che i dati provengano da un file invece che dal terminale.

```
$ mail leo < note
$
```

In questo esempio, il contenuto del file **note** rappresenta l'ingresso al comando **mail leo**, per cui Leo trova il file nella sua cassetta della posta. Si ottiene lo stesso effetto sempre con il comando **mail leo**, ma inserendo da tastiera il contenuto del file **note**.

Potete combinare le operazioni di ridirezione dello standard input e dello standard output. Per esempio:

```
$ pr < note > n.copia
$
```

In questo esempio lo shell esegue il comando **pr** che riceve in ingresso il file **note** e memorizza l'uscita nel file **n.copia**. Lo shell richiede che il nome del comando preceda gli operatori di ridirezione. Potete scrivere indifferentemente qualunque dei seguenti comandi:

```
$ pr > n.copia < note
$ pr >n.copia <note
$ pr <note >n.copia
$ pr >n.copia<note
```

ma la linea di comando

```
$ n.copia > pr < note
```

non è corretta. È consigliabile separare con spazi gli operatori di ridirezione dal nome del comando e dagli argomenti, anche se non è necessario. La linea di comando può anche contenere le variabili di ambiente.

```
$ ESEMPIO=note
$ pr < $ESEMPIO > n.copia
$
```

Il concetto di ridirezione dell'input/output (I/O) genera implicazioni molto importanti e costituisce un aspetto tra i più generali e utili di UNIX. La maggior parte dei comandi riceve in input e produce in output semplici se-

quenze di byte che possono essere facilmente indirizzate da una periferica a un file, a un dispositivo di memoria.

Alcuni comandi vi permettono di indicare direttamente i nomi di file come argomenti invece di richiedervi di ridirigere lo standard input da un file al comando. Questa è una caratteristica del comando e non è stabilita dal sistema operativo o dallo shell, per cui non viene sempre seguita. Per esempio, i due comandi:

```
$ cat < note
$ cat note
```

sono equivalenti, poiché il comando **cat** opera su un file o sul suo standard input. Come molti altri comandi, **cat** permette di specificare sulla sua linea di comando sia i file che lo standard input. Anche le seguenti forme sono equivalenti:

```
$ cat note PROVA n.copia
$ cat note - n.copia < PROVA
```

Se specificate solo il segno - (meno) come argomento e nessuna opzione che lo segue, il comando lo interpreta come il suo standard input. Nel secondo esempio precedente, il comando **cat** analizza il file **note**, esamina lo standard input che è stato ridiretto dal file **PROVA** e poi processa il file **n.copia**.

Allo stesso modo, il comando

```
$ cat note - n.copia > output
```

esamina il file **note**, aspetta l'ingresso dalla tastiera e poi processa il file **n.copia**. Poiché non avete ridiretto lo standard input, lo shell lo considera ancora associato alla tastiera.

3.8 Carattere di fine file

Se utilizzate la tastiera come standard input, dovete informare i comandi come **cat** che state inserendo caratteri in ingresso e successivamente che avete terminato l'operazione di scrittura. UNIX definisce uno speciale carattere che viene utilizzato come carattere di fine file e occupa la prima posizione sulla linea. Questo carattere non viene memorizzato su disco alla fine del file perché il sistema è in grado di distinguerlo automaticamente. Comunque, quando usate la tastiera come un file, il sistema non sa quando avete terminato l'operazione di inserimento di dati e quindi dovete specificarlo espressamente attraverso la combinazione di tasti CTRL-D. Per generare questo carattere, mantenete premuto il tasto CTRL sulla tastiera e battete il tasto D.

Nella linea di comando

```
$ cat note - n.copia > output
```

cat processa il file **note**, aspetta l'input da terminale e legge i dati che inserite fino a quando battete CTRL-D e poi va a processare il file **n.copia**. Tutti i tre file sorgente vengono concatenati nel file di uscita.

3.9 Ridirezione dello standard output in un file

L'operatore > permette di annullare e ricreare un file già esistente, perdendo però il contenuto precedente. È possibile inoltre, con diverse modalità, concatenare il risultato prodotto da un comando alla fine di un file già esistente. La sequenza di comandi

```
$ pr < note > n.copia
$ cat note n.copia > dup.note
$ rm n.copia
$
```

genera il file **dup.note**, che contiene sia il file **note** che quello prodotto dal comando **pr**. Un altro modo per ottenere lo stesso risultato è di utilizzare l'operatore >>, che induce lo shell a ridirigere lo standard output su un file, accodandolo alla fine del file. Questa sequenza di comandi:

```
$ cat note > dup.note
$ pr note >> dup.note
$
```

produce lo stesso risultato dell'esempio precedente, utilizzando però un comando in meno. Se il file che compare a destra dell'operatore >> non è presente nel sistema, viene automaticamente creato. Tenete presente che l'operatore >> non può contenere spazi tra i caratteri > e che non esiste un equivalente operatore << per lo standard input.

3.10 Standard error

Abbiamo parlato di standard input e di standard output, ma il sistema definisce un altro file che viene comunemente chiamato standard error.

La maggior parte dei comandi utilizza il file standard error per visualizzare i messaggi di errore oppure altri risultati anomali che non debbono apparire nello standard output. Non è opportuno infatti che i messaggi di errore compaiano insieme ai risultati e quindi il sistema li riporta su un file

distinto. Se intenzionalmente scrivete un comando non corretto, analizzate poi il contenuto dello standard error. Per esempio, se cercate di utilizzare il comando **cat** su un file che non esiste, il sistema genera un messaggio di errore.

```
$ cat no.file
cat: cannot open no.file
$
```

Se ora ridirigete lo standard output del comando **cat**,

```
$ cat no.file > output
cat: cannot open no.file
$
```

il file **output** risulta vuoto e il messaggio di errore compare ancora sul terminale, perché viene scritto in standard error. Potete ridirigere lo standard error utilizzando l'operatore 2> oppure 2>>, se intendete rispettivamente creare un nuovo file oppure aggiungere dati a un file esistente:

```
$ cat no.file 2> output
$ cat output
cat: cannot open no.file
$
```

Abbiano utilizzato la notazione 2> per referenziare i canali di I/O a cui sono assegnati precisi identificatori numerici: 0 si riferisce infatti allo standard input, 1 allo standard output e 2 allo standard error. Questi identificatori numerici sono di competenza principalmente dei progettisti software anche se nel nostro esempio permettono di accedere allo shell a livello utente.

3.11 Combinazione di comandi tramite pipeline

È possibile che i dati prodotti in uscita da un comando vengano impiegati come dati in ingresso a un altro comando. Per esempio, è lecito scrivere:

```
$ cat note PROVA > temp
$ pr < temp > output
$ rm temp
$
```

Se non volete creare file intermedi, l'uscita del comando **cat** può essere direttamente incanalata in ingresso al comando **pr**. Anche se la sequenza di comandi indicata nell'esempio precedente svolge correttamente il suo compito, lo shell propone una soluzione più elegante ed efficiente tramite l'ope-

ratore *pipe* |. Una linea di comando che contiene questo operatore prende il nome di *pipeline* e lo shell interpreta automaticamente il passaggio di un flusso di dati da un comando a un altro. Per esempio, la linea di comando

```
$ cat note PROVA | pr > output
$
```

concatena i file **note** e **PROVA** e utilizza lo standard output di **cat** come standard input per il comando **pr**. Lo standard output di **pr** viene indirizzato al terminale, ma potete ridirigerlo ovunque scrivendo:

```
$ cat note PROVA | pr > output
$
```

Esistono altre linee di comando che permettono di ottenere la stessa funzione. Per ridirigere l'ingresso al comando **cat** potete inserire da tastiera:

```
$ cat - PROVA < note | pr > output
```

In questo esempio abbiamo due comandi che sono separati dall'operatore : **cat - PROVA < note** e **pr > output**. Lo standard output del primo comando è collegato allo standard input del secondo. L'operazione di ridirezione risulta locale alla parte di pipeline interessata, per cui il seguente comando non è equivalente al precedente:

```
$ cat - PROVA | pr > output < note
```

Generalmente, lo standard input si trova all'inizio del pipeline (leggendo da sinistra a destra sulla linea di comando) mentre lo standard output compare alla fine o nella seconda metà del pipeline. In ogni caso dovete iniziare sempre una linea di comando specificando il nome di un comando tra quelli disponibili. La seguente linea di comando non è corretta:

```
$ note > cat - PROVA | pr > output
```

Inoltre l'operatore | connette solo i comandi e quindi dovete utilizzare gli operatori > e < per indicare la ridirezione di file di dati. È scorretto scrivere anche:

```
$ note | cat - PROVA | pr > output
```

a meno che il sistema non disponga di un comando di nome **note**.

3.12 Filtri

Il meccanismo di pipeline non è limitato solo a due comandi. Potete definire pipeline di qualunque lunghezza collegando solo lo standard output di un comando allo standard input del successivo. UNIX possiede molti comandi che funzionano così. Sono denominati *filtri*, poiché acquisiscono dati in ingresso, svolgono operazioni di filtraggio stabilite dal particolare programma e generano un risultato.

Il programma **tr** (translate) è un classico esempio di filtro. Riceve due argomenti in ingresso, che interpreta come insiemi di caratteri (non stringhe di caratteri); copia il suo standard input nello standard output e sostituisce ogni elemento appartenente all'insieme di caratteri che costituisce il primo argomento con l'elemento che occupa la stessa posizione nell'insieme di caratteri che rappresenta il secondo argomento.

```
$ echo "salve gente" | tr a 3
s3lve gente
$
```

Il comando **tr** si differenzia dai soliti comandi in quanto non richiede in ingresso il nome di un file, ma legge solo lo standard input. Una ottimizzazione del comando è la seguente:

```
$ echo "salve gente" | tr ae 34
s3lv4 g4nt4
$
```

CAMPI E DELIMITATORI

Uno dei comandi che viene utilizzato più di frequente come filtro è **cut**. Questo comando permette di copiare nello standard output alcune parti delle linee del suo standard input. Un campo di una linea è una stringa di caratteri separata da un delimitatore prefissato. UNIX utilizza vari delimitatori di campo a seconda dei casi, anche se i tre più comuni sono i caratteri *tab*, *spazio* e *due punti*.

La notazione **t** serve per indicare il carattere di tabulazione all'interno di un testo, mentre per crearlo effettivamente dovete premere il tasto TAB sulla tastiera. L'idea di suddividere una linea di un file in campi ricorda le modalità di numerazione degli argomenti di una linea di comando per lo shell, anche se in quel caso veniva utilizzato come delimitatore il carattere spazio. In molte altre circostanze si utilizza invece come delimitatore il carattere : (due punti).

Il comando **cut** serve per copiare in uscita solo alcuni campi di una linea di ingresso. Nel creare una linea di comando per **cut**, dovete specificare

quali campi siano da ricopiare nello standard output e quale carattere utilizzare come delimitatore dei campi delle linee nello standard input. Consideriamo il seguente file come costituito da un insieme di record di dati, ognuno dei quali rappresenta una linea:

```
$ cut datafile
123:543:654:234
987:753:123:765
435:765:135:963
$
```

Ogni record in questo esempio si compone di quattro campi che sono separati dal delimitatore : e la fine della linea delimita ogni record. Se occorre utilizzare il carattere : all'interno di un campo, sceglietevi un altro carattere come delimitatore, facendo molta attenzione che non compaia già nei dati.

Potete utilizzare il comando **cut** per visualizzare una parte dei dati memorizzati in questo semplice database. Per esempio, per visualizzare solo il secondo campo di ogni linea, inserite la seguente linea di comando:

```
$ cut -f2 -d: < datafile
```

Per specificare al comando **cut** il carattere che avete scelto come delimitatore (cioè il carattere : in questo caso), utilizzate l'argomento **d** (delimiter), mentre l'argomento **-f** (field) permette di copiare nello standard output solo il secondo campo.

```
$ cut -f2 -d: < datafile
543
753
765
$
```

Si può notate che il carattere delimitatore non è presente in uscita e che ogni linea in ingresso genera una linea in uscita.

Il comando **cut** permette di salvare più di un campo delle linee in ingresso.

```
$ cut -f1,2,4, -d: < datafile
123:543:234
987:753:765
435:765:963
$
```

In questo esempio vengono selezionati, specificando l'argomento **-f1,2,4** (ogni campo che intendete copiare viene indicato con una notazione numeri-

ca, separata dalla virgola), i campi 1, 2 e 4 del file di ingresso, mentre il campo 3 viene tralasciato. In uscita, i campi vengono separati dal delimitatore originale, cioè dal carattere :.

Potete anche cambiare il carattere delimitatore. Se lo sostituite, per esempio con il carattere 3, il sistema non si oppone, poiché non esistono controindicazioni all'utilizzo di tale carattere come delimitatore. La scelta del delimitatore nasce da considerazioni contingenti. Molti comandi utilizzano come delimitatore il carattere :, ma nulla vieta di utilizzare qualunque altro carattere.

```
$ cat datafile
123:543:654:234
987:753:123:765
435:765:135:963
$ cut -f2 -d3 < datafile
:54
:12
5:765:1
$
```

Il risultato prodotto non sembra a prima vista molto chiaro, ma è solo una impressione. Esaminiamo, infatti, la seguente linea:

```
123:543:654:234
```

È costituita da quattro campi separati dal carattere 3: 12, :54, :654:2 e 4. In UNIX, i file sono sequenze di caratteri, senza una predefinita struttura interna.

Il comando **cut** può anche selezionare colonne di dati. Il termine "colonna" è entrato in uso dai tempi delle schede perforate, che si componevano di 80 colonne (questo è il motivo per cui la maggior parte dei terminali dispone di 80 caratteri per linea). Supponiamo di suddividere in colonne le linee dello standard input, anche se il sistema tratta i dati come una sequenza di caratteri senza alcuna suddivisione in colonne. Supponiamo inoltre che il carattere newline delimiti l'inizio di una linea nel file di ingresso e analogamente un altro newline ne delimiti la fine.

È possibile definire un comando **cut** che opera sui campi disposti in colonna.

```
$ cut -c5-8 datafile
```

Il comando **cut** può avere come argomenti il nome di un file o una lista di nomi di file e leggere direttamente lo standard input. L'esempio precedente contiene come argomento per selezionare le colonne comprese tra la quinta e l'ottava del file di ingresso **-c5-8** (c sta per colonna). Per indicare i campi compresi tra due valori estremi si utilizza il carattere lineetta (-).

```
$ cat datafile
123:543:654:234
987:753:123:765
435:765:135:963
$ cut -c5-8 datafile
543:
753:
765:
$
```

Inoltre, il comando **cut** opera una suddivisione in colonne di ciascuna linea del file e quindi il carattere : non ha un particolare significato in questo esempio.

Un altro comando utilizzato frequentemente come filtro è **sort**, che permette di ordinare numericamente o alfabeticamente le linee del file in ingresso.

```
$ sort < datafile
123:543:654:234
435:765:135:963
987:753:123:765
```

Il comando **sort** non modifica il contenuto delle linee, ma le riordina numericamente o alfabeticamente, e seleziona automaticamente il file utilizzando l'inizio della linea come campo chiave, cioè la stringa di caratteri che serve a ordinare alfabeticamente le linee. In ogni caso, **sort** permette di referenziare qualunque campo sia presente nel file e le regole di individuazione dei campi sono simili a quelle discusse per il comando **cut**. (Il Capitolo 7 tratta dettagliatamente le caratteristiche del comando **sort**.)

Potete combinare i due comandi **cut** e **sort** in un pipeline e ottenere così risultati interessanti. Per esempio, in questo modo è possibile ordinare alfabeticamente tutti i nomi delle variabili di ambiente.

```
$ env | cut -f1 -d= | sort
EDITOR
HOME
LOGDIR
LOGNAME
MAIL
PATH
SHELL
TERM
TZ
UMASK
$
```

Abbiamo utilizzato il segno = come delimitatore di campo e il comando ha ordinato alfabeticamente la variabile di ambiente in base alla parte nome della coppia *nome=valore*. Il risultato così ottenuto è stato memorizzato

nello standard output (cioè il terminale) in quanto non abbiamo eseguito alcuna operazione di ridirezione.

I filtri generalmente costituiscono l'ultima parte di un pipeline. Un buon esempio è costituito dal comando **more**, che evita uno scorrimento troppo rapido dell'immagine su video. La sua funzione infatti è di arrestare la visualizzazione dopo il riempimento dell'intero video, visualizzando il messaggio – –**More**– – sulla riga finale e restando in attesa di input da tastiera. Quando abbassate la barra dello spazio, **more** visualizza la successiva schermata; questo procedimento continua fino al raggiungimento del fine file, quando **more** restituisce il controllo allo shell. Il comando **more** è stato adattato anche in un comando simile di nome **pg** (pager); il vostro sistema potrebbe avere l'uno o l'altro di questi comandi consimili. Provate la seguente linea di comando:

```
$ echo /etc/profile ¦ more
```

Se volete interrompere l'esecuzione del programma **more**, prima della conclusione, premete il tasto DEL e il controllo ritorna immediatamente allo shell.

Potete definire un pipeline adatto per ogni vostra esigenza che si compone anche di cinque o sei comandi, alcuni dei quali utilizzabili come filtri.

È possibile sviluppare applicazioni abbastanza sofisticate impiegando solo sequenze di pipeline, ma il maggiore beneficio che proviene dai concetti di file non strutturati, standard I/O, ridirezione di I/O e pipeline di comandi si ha nella risoluzione semplice e rapida al terminale di problemi connessi ai file e ai dati. L'esempio precedente, di listato, filtraggio e ordinamento di variabili di ambiente, può aver richiesto lo sviluppo su un altro sistema operativo di un programma applicativo in BASIC, C o Pascal. In UNIX la struttura a pipeline può risolvere molti problemi legati ai file di dati che consistono in sequenze di caratteri e linee di testo ASCII.

3.13 Valori restituiti dai comandi

Oltre allo standard output e allo standard error, i comandi restituiscono allo shell un *codice di ritorno* numerico (return code). Questo codice non viene visualizzato; il valore è di solito zero se il comando termina correttamente e diverso da zero in presenza di errori. In questa seconda ipotesi, il valore restituito è compreso tra 0 e 255 a seconda dell'errore che il comando ha generato. Potete analizzare questo valore utilizzando la variabile $? (dollaro punto interrogativo). Al termine dell'esecuzione del comando il valore della variabile viene azzerato, per cui se intendete salvarlo dovete definire una nuova variabile di ambiente. Questo assegnamento in realtà comporta la

modifica del valore di $?, ma fortunatamente questo accade dopo che lo avete salvato nella nuova variabile di ambiente. Per esempio:

```
$ cat no.file
cat: cannot open no.file
$ SALVA=$?
$ echo $?
0
$ echo $SALVA
2
$
```

Nella variabile $? viene salvato solo il valore di ritorno dell'ultimo comando.

La variabile $? non è in realtà una variabile di ambiente come quelle di cui abbiamo parlato precedentemente, ma appartiene a un'altra classe di oggetti, denominati *variabili di shell*. Questo perché le variabili di shell non sono disponibili per i comandi in esecuzione come invece le vere variabili di ambiente, ma vengono gestite all'interno dello shell. Le variabili di shell si compongono di stringhe di caratteri di lunghezza unitaria, ma come al solito, se volete ricavare il loro valore, dovete anteporre al nome il carattere $. Ci sono molte altre variabili di shell oltre ?, tra cui # (memorizza il numero di argomenti della linea di comando) e $ (memorizza il numero di processi). Il valore della variabile di shell $ si ricava utilizzando la notazione $$. (Il Capitolo 8 tratta dettagliatamente le variabili di shell.)

3.14 L'operatore "accento grave"

Potete anche assegnare lo standard output di un comando a una variabile di ambiente, usando l'operatore ` (accento grave). Tenete presente che la lunghezza della parte valore della coppia *nome=valore* nella definizione di una variabile di ambiente è limitata in SVR4 a circa 5120 caratteri. Per esempio:

```
$ ESEMPIO=`echo $LOGNAME`
```

Questa notazione permette di assegnare lo standard output del comando **echo $LOGNAME** alla variabile di ambiente ESEMPIO.

```
$ ESEMPIO=`echo $LOGNAME`
$ echo $ESEMPIO
giorgio
$
```

Notate che l'accento grave ` (ASCII 96) non deve essere confuso col consueto apice singolo ' (ASCII 39).

Il comando **wc** (word count) fornisce un ottimo esempio di come utilizzare l'operatore ' in quanto agisce come filtro sullo standard input (o un nome di file specificato come argomento), di cui calcola il numero di linee, di parole e di caratteri. Le parole sono stringhe di caratteri limitate da spazi. Per esempio:

```
$ wc note
       4      44      227 note
$
```

Il file **note** contiene quattro linee, 44 parole e 227 caratteri. Il comando

```
$ wc < note
```

e il comando

```
$ cat note ¦ wc
```

daranno lo stesso risultato, eccetto che, poiché nessun nome di file è dato come argomento a **wc**, l'uscita non comprende nomi di file.

```
$ cat note ¦ wc
       4      44      227
$
```

Il comando **wc** può avere tre argomenti: **−l**, **−w** e/o **−c**, per limitare il risultato rispettivamente alle linee, alle parole o ai caratteri. Per esempio:

```
$ wc −w < note
       44
$
```

Potete utilizzare questa forma al termine di un pipeline e assegnare il risultato a una variabile di ambiente, perché possa essere impiegata successivamente:

```
$ PAROLE=`cat * ¦ wc −w`
$ echo $PAROLE
73
$
```

In questo esempio abbiamo definito il pipeline **cat * ¦ wc −w** e assegnato lo standard output alla variabile di ambiente PAROLE. Ricordatevi che lo shell impiega il carattere * per referenziare tutti i nomi di file che sono contenuti nel directory corrente prima di eseguire il comando.

3.15 Approfondimenti

Le funzioni di shell che presentiamo in questo testo includono la maggior parte delle utility che lo shell mette a disposizione dell'utente. Lo shell rappresenta effettivamente un linguaggio di programmazione di grande utilità ed efficienza che permette di definire strumenti di vario genere, come per esempio funzioni fondamentali del sistema operativo. Nei Capitoli 8 e 16 tratteremo l'uso dello shell come linguaggio di programmazione, dopo aver esaminato le caratteristiche del file system e qualche altro comando di utente.

COMBINAZIONE DI COMANDI

Lo shell definisce diversi operatori per combinare più comandi su una stessa linea. Potete utilizzare l'operatore ; (punto e virgola) che, a differenza dell'operatore pipe, non connette i comandi, ma li mantiene separati sulla linea. Per esempio:

```
$ ls ; echo salve
PROVA   note
salve
$
```

L'uscita prodotta dal comando **echo** segue direttamente su terminale l'uscita del comando **ls**.

Potete anche utilizzare questa sequenza di comandi:

```
$ ESEMPIO = 'ls ; echo salve'
$ echo $ESEMPIO
PROVA note salve
$
```

In questo esempio, la variabile di ambiente ESEMPIO contiene una lista di nomi di file e aggiunge un identificatore che non rappresenta un file esistente. Il risultato prodotto da una sequenza di comandi separati dall'operatore ; costituisce il valore dell'ultimo comando inserito, che in questo caso corrisponde al comando **echo**. Esistono moltissime possibilità di combinazione dei comandi, come vedremo nei capitoli successivi.

RIDIREZIONE DELLO SHELL

Lo shell viene referenziato tramite l'identificatore **sh** (shell) ed è eseguibile come un qualunque comando. L'istanza di shell che è in attesa dei comandi inseriti da tastiera è denominata *shell di login* poiché va in esecuzione quan-

do accedete al sistema e termina quando uscite. Il suo standard input è collegato alla tastiera e lo standard output è collegato allo schermo del terminale. Potete eseguire come comando un'altra istanza di shell.

```
$ sh
```

In questo modo avete creato ricorsivamente una seconda copia di shell che è in attesa di eseguire i comandi. Comunque, lo shell di login è ancora attivo e attende che termini l'esecuzione del comando corrente, per tornare in esecuzione. È possibile uscire dal secondo shell con il comando **exit**, e ritornare così allo shell originale, oppure potete interromperne l'esecuzione premendo CTRL-D che rappresenta il carattere di fine file per i dati in ingresso allo shell. Potete anche uscire dallo shell di login nel momento in cui vi scollegate. Quando definite un secondo shell, questo visualizza il prompt PS1 e può risultare molto difficoltoso determinare quale shell è in esecuzione, se non distinguete i prompt dei due shell. In questo modo siete in grado di utilizzare uno shell differente da quello di login, uno shell per esempio che vi permette di correggere le linee di comando prima di battere RETURN per lanciarlo. Tratteremo in seguito shell che offrono questa possibilità: Korn e C shell.

Esaminiamo un altro caso di impiego di shell come semplice comando eseguibile. Immaginate di scrivere un file che contenga alcuni comandi, quelli di uso più frequente, invece di inviare i comandi allo shell direttamente da tastiera. Potete creare questa lista di comandi con un editor di testo o semplicemente scrivendo:

```
$ cat > cmds
```

Potete inserire i comandi fino a quando non premete il tasto CTRL-D per segnalare a **cat** di aver terminato il file di ingresso. Questi comandi, invece di essere eseguiti nel momento in cui li battete da tastiera, vengono salvati nel file **cmds**, a cui avete ridiretto l'uscita di **cat**.

Ora potete prendere questo file di comandi e utilizzarlo come standard input ridiretto del comando **sh**.

```
$ sh < cmds
```

In questo modo abbiamo definito una seconda copia dello shell, che riceve in ingresso il file **cmds**. Lo shell (non quello di login, ricordatelo) legge ed esegue i comandi contenuti nel file, scrivendo i risultati nello standard output, cioè su terminale. Se intendete ridirigere l'uscita di questo comando, potete salvare il risultato in un file:

```
$ sh < cmds > output
```

Potete anche salvare lo standard error in un altro file, se si verifica un errore nel corso dell'esecuzione del comando.

```
$ sh < cmds > output 2> errore
```

Lo shell utilizza questi concetti, anche se in forma più semplice, per realizzare *programmi di shell.*

Capitolo 4

Il file system

Il *file system* rappresenta una caratteristica fondamentale di UNIX. La gestione dei file in ambiente UNIX è estremamente flessibile ed efficiente tanto da avere influenzato la struttura di molti altri sistemi operativi (come per esempio MS-DOS). UNIX definisce uno schema *gerarchico di directory*. Un directory è un raccoglitore di file e può fare parte a sua volta di altri directory, per cui si viene a creare una struttura complessa e ramificata a forma di albero. I comandi, i file di dati e anche le unità periferiche possono essere referenziati come i directory. UNIX adotta una semplice, ma nello stesso tempo efficiente, notazione di indirizzamento di file e directory all'interno del file system, in modo che i comandi possono facilmente reperire ciò che avete indicato sulla linea di comando.

4.1 File e directory

Un file corrisponde a una sequenza di byte che risiede in forma semipermanente su un supporto fisso, cioè un disco magnetico o un nastro. Un file può essere quindi un programma eseguibile (un comando), una parte di testo (un messaggio o un manoscritto), un database, un insieme di bit che memorizza

una immagine su video, ecc. Se tutte queste informazioni le memorizzate su un disco o su un nastro assegnandogli un nome, avete definito un file. Anche se il sistema considera come file una semplice, indifferenziata sequenza di byte, un utente o un programma applicativo può aggiungere al contenuto del file una struttura di riconoscimento e assegnargli così un preciso significato. Questa situazione si verifica infatti nei file di testo, in cui viene utilizzato uno speciale carattere ASCII denominato newline per strutturare logicamente il file in linee. Un altro esempio è costituito dai dati in forma binaria che sono contenuti in un programma eseguibile come **cat** oppure **wc**. Potete trovare file con questi nomi nel file system e conteggiare i caratteri in essi contenuti tramite il comando **wc**, proprio come accade nei file di testo. Il comando **cat** viene utilizzato raramente nei file che non contengono testo, ma può essere utile per visualizzare il contenuto di un file binario (non ASCII) o di un qualunque altro file. I comandi **cat** o **wc** possono occasionalmente fare parte di un pipeline quando state effettuando qualche operazione su file non ASCII. Per esempio, potete qualche volta concatenare file binari e inviarli direttamente in uscita a una stampante o a un terminale grafico.

Un tipo di file particolarmente interessante è il *directory*, che non contiene testo o un programma eseguibile, ma una lista di nomi di file e qualche informazione a essi associata. Un directory è un file come gli altri, ma viene utilizzato in maniera differente.

Un directory rappresenta la collocazione dei file nel file system. Poiché il sistema UNIX tratta tutti i file come sequenze di byte di dati, potete inserire qualunque tipo di file in qualsiasi directory o addirittura un directory in un altro directory, senza alcun limite allo sprofondamento di annidamento dei subdirectory. In questo modo l'immagine di un file system corrisponde a una gerarchia di directory e subdirectory; la Figura 4.1 illustra graficamente quanto abbiamo detto. Un directory di nome **giorgio** può contenere i file **note**, **PROVA** e **dir1**. I file **note** e **PROVA** sono semplici file di testo, ma **dir1** è un altro directory, che a sua volta contiene i file **salve**, **ciao** e **dir2**. Se anche **dir2** è un directory, può contenere altri file, come **su** e **giù**. Potete pensare (e disegnare) i directory come gli elementi di una gerarchia di subdirectory.

Quando eseguite il comando **ls** per avere il listato dei file, non viene visualizzato l'elenco di tutti i file disponibili nel sistema, ma solo quelli presenti in un particolare directory. Non potete capire da un semplice elenco se un nome è associato a un file o a un directory, ma in questo modo ricavate solo i riferimenti agli elementi contenuti nel *directory corrente*. Potete specificare un argomento al comando **ls** nel modo seguente:

```
$ ls note
note
$
```

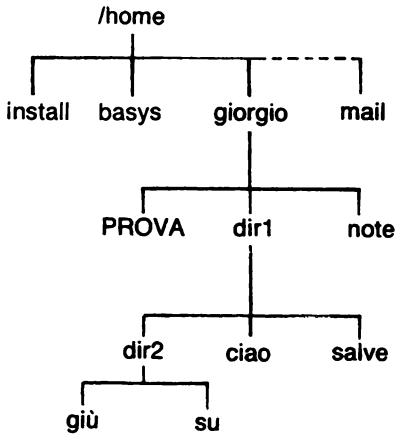


Figura 4.1 Una semplice gerarchia di directory.

Se l'argomento referenzia un file, **ls** scrive solo il nome del file sullo standard output, mentre se referenzia un directory, **ls** lo analizza e scrive i nomi di tutti i file presenti nel directory:

```
$ ls dir1
salve       ciao       dir2
$
```

Esistono altri modi per distinguere tra file e directory, come verificheremo in seguito.

REGOLE DI FORMAZIONE DEI NOMI DI FILE

Ai file e ai directory vengono assegnati nomi che devono obbedire alle medesime regole: sono permessi tutti i caratteri ASCII, e i caratteri maiuscoli e minuscoli non sono equivalenti, per cui un file chiamato **NOTE** risulta distinto dal file **note**. Potete usare qualunque carattere nel nome di file, senza distinzione fra nome ed eventuale estensione. Molti nomi di file in sistema UNIX usano estensioni, come **main.c** o **rc2.c**, ma questa è solo una convenzione stabilita e accettata fra gli utenti, non una proprietà del sistema.

Nell'assegnare il nome a un file o a un directory è imprudente includere i metacaratteri di shell o altri caratteri speciali. Evitate di utilizzare i caratteri $ (dollaro), ; (punto e virgola), \ (barra inversa), & (e commerciale), ! (punto esclamativo), * (asterisco) e ¦ (pipe), per non creare confusione, per-

ché altrimenti quando li referenziate dovete racchiuderli tra apici per impedire allo shell di interpretarli come operatori invece di normali caratteri. Solo il carattere / (barra) non può essere utilizzato all'interno di un nome in quanto ha una precisa funzione nella composizione dei nomi di file.

Nei sistemi SVR4 in alcuni directory la lunghezza dei nomi di file può essere limitata a 14 caratteri; in altri directory possono essere usati nomi lunghi fino a 256 caratteri. Sfortunatamente non risulta sempre facile conoscere se un particolare directory rientra nella limitazione di 14 caratteri. Generalmente, se un directory rientra in tale limitazione, tutti i subdirectory sottostanti rientrano nella stessa limitazione; tuttavia neanche questa regola è assoluta, salvo nel caso che voi stesso abbiate creato i subdirectory. Nel directory che volete usare potete creare un file di prova col nome più lungo di 14 caratteri; se il nome viene troncato al limite di 14 byte, dovrete curare di non superare tale limite in quel directory. Per esempio:

```
$ cat /etc/profile > 12345678901234567890
$ ls
12345678901234
$
```

Come potete vedere, il directory corrente rientra nella limitazione di 14 caratteri. Se in output di **ls** il nome del file appare completo, potete con sicurezza creare file con nomi più lunghi di 14 caratteri.

Questa complessità rispetto ai nomi di file deriva dalle due nuove importanti caratteristiche del sistema SVR4: i tipi di file system (trattati nel Capitolo 18) e i collegamenti simbolici (trattati più avanti in questo capitolo). Tuttavia al più presto la limitazione di 14 caratteri verrà completamente eliminata.

Le regole di lunghezza per i nomi di directory sono le stesse dei nomi di file.

NOMI DEI DIRECTORY

È possibile che il nome completo di un file contenga al suo interno il riferimento a qualche directory. Il carattere / (barra) permette di separare i singoli componenti del nome di un file. Per esempio, il nome **dir1/ciao** referenzia il file **ciao** contenuto nel directory **dir1**, **dir1/dir2** referenzia il file (un directory in questo caso) **dir2** contenuto nel directory **dir1** e **dir1/dir2/giù** referenzia il file **giù** contenuto nel directory **dir2**, che a sua volta si trova nel directory **dir1**. Questi nomi di file possono avere qualunque lunghezza, ma ognuno dei suoi componenti può rientrare nella limitazione di 14 caratteri. Solo l'ultimo componente di un file referenziato in questo modo può essere un file o un directory, a seconda delle vostre esigenze, mentre tutti i componenti intermedi devono essere directory.

Questo tipo di notazione prende il nome di *path* o *pathname*, in quanto la lista dei directory e subdirectory che compongono il nome descrive il cammino (path) che è necessario seguire lungo la struttura gerarchica di subdirectory del sistema per raggiungere il file.

Potete usare i normali metacaratteri di shell nella costruzione di nomi di percorso completi, per esempio:

```
$ cat /home/giorgio/*/*
```

Questo produce la visualizzazione di tutti i file contenuti in tutti i subdirectory sottostanti a **/home/giorgio**. Anche gli altri metacaratteri di shell mantengono le loro usuali proprietà.

4.2 Il directory di lavoro

Quando non specificate nessun argomento per il comando **ls**, viene visualizzato il contenuto del directory corrente. Vi trovate sempre logicamente allocati in un directory, che prende il nome di *directory corrente* o *di lavoro*. Per esempio, il pathname **dir1/ciao** referenzia il directory **dir1**, che è localizzato nel directory corrente, e il file **ciao** che è contenuto nel directory **dir1**. Il comando **pwd** (print working directory) ci informa sul directory corrente.

```
$ pwd
/home/giorgio
$
```

Il comando **pwd** memorizza sul suo standard output il pathname del directory corrente. Di solito questo comando viene associato a una variabile di ambiente, per facilitare il reperimento del directory mentre scorrete nel file system.

```
$ QUI = 'pwd'
$ echo $QUI
/home/giorgio
$
```

Utilizzate il carattere ' (accento grave) per assegnare lo standard output di un comando a una variabile.

4.3 Scorrimento nella gerarchia di directory

Potete percorrere tutto l'albero dei directory tramite il comando **cd** (change directory):

```
$ ls
PROVA     dir1        note
$ cd dir1
$ ls
dir2      ciao        salve
$ pwd
/home/giorgio/dir1
$
```

In questo modo avete cambiato il directory corrente.

Il carattere . (punto o dot) sostituisce il nome del directory corrente. Potete utilizzare il pathname **./note** per referenziare il file **note** nel directory corrente oppure **./dir1/ciao** per referenziare il directory **dir1** contenuto nel directory corrente.

```
$ pwd
/home/giorgio/dir1
$cd ./dir2
$pwd
/home/giorgio/dir1/dir2
$
```

La presenza del carattere . all'inizio di un pathname è generalmente ridondante, perché il sistema inizia a interpretare il pathname dal directory corrente, ma qualche volta è indispensabile specificarlo.

Nei precedenti esempi, siete discesi all'interno del file system fino a raggiungere i livelli più interni utilizzando il comando **cd**. Potete anche spostarvi verso i livelli più esterni del file system tramite l'operatore .. (punto punto) per individuare il directory padre del directory corrente.

```
$ pwd
/home/giorgio/dir1/dir2
$ cd ..
$ pwd
/home/giorgio/dir1
$
```

Potete muovervi verso il basso, scorrendo i subdirectory, fino a quando esistono directory; al termine della gerarchia di directory, non esistono più subdirectory, ma solamente file, quindi non potete spingervi oltre senza creare un nuovo directory al livello più basso. Verifichiamo cosa accade se vi spostate progressivamente verso la radice dell'albero di directory.

```
$ pwd
/home/giorgio/dir1
$ cd ..
$ pwd
```

```
/home/giorgio
$ cd ..
$ pwd
/home
$ cd ..
$ pwd
/
$
```

Avete finalmente raggiunto la radice dell'albero dei directory, oltre la quale non esistono più directory. Questo punto di riferimento è denominato *directory radice* o semplicemente radice (root) del file system. La sua collocazione è fissa e partendo da esso potete raggiungere tutti i file presenti nel sistema, scorrendo tutti i subdirectory.

Se definite un pathname che inizia con il carattere / (barra), il sistema lo riferisce al directory radice, mentre se il pathname non inizia con il carattere /, il sistema assume come punto di partenza il directory corrente.

```
$ pwd
/home/giorgio
$ ls dir1
dir2        ciao        salve
$ ls /home/giorgio/dir1
dir2        ciao        salve
$
```

Questi due comandi referenziano lo stesso directory, uno parte dal directory radice, mentre l'altro dal directory corrente. Il primo viene di solito denominato pathname relativo, mentre il secondo pathname completo o pathname assoluto. Si utilizzano entrambe le notazioni nella definizione degli argomenti per i comandi, ma a seconda dei casi conviene scegliere una o l'altra.

Potete utilizzare un pathname come argomento di un comando ogni volta che referenziate un semplice nome di file. La seguente linea di comando

```
$ cat /home/giorgio/note
```

è equivalente a

```
$ cat note
```

se il directory corrente è **/home/giorgio**. Questa possibilità di referenziare un file in due modi diversi rappresenta una caratteristica molto utile del sistema UNIX. In realtà il pathname completo è l'unico identificatore del file, mentre il pathname relativo è una conveniente abbreviazione, anche se il sistema riconosce sempre un pathname completo quando inserite il nome di un file.

4.4 Cambiamento della gerarchia di directory

Potete creare un numero qualunque di nuovi directory e cancellare quelli che non vi servono. Per creare un nuovo directory utilizzate il comando **mkdir** (make directory).

```
$ ls
PROVA     dir1          note
$ mkdir n.directory
$ ls
PROVA     dir1          n.directory          note
$
```

Il comando **mkdir** riceve in ingresso come argomenti una lista di pathname che rappresentano i nomi dei directory che intendete creare. Se specificate un pathname completo, il nuovo directory non ha relazione con il directory corrente, mentre se si tratta di un pathname relativo, il nuovo directory viene considerato un subdirectory del directory corrente. La notazione dei directory segue le stesse regole adottate per gli altri file; in particolare, non potete creare un directory con il nome di un altro file o directory che si trova nello stesso directory padre. Al momento della sua creazione, il directory è vuoto, per cui non contiene nessun file.

Utilizzate il comando **rmdir** (remove directory) per cancellare un directory.

```
$ rmdir n.directory
$ ls
PROVA     dir1          note
$
```

Il comando **rmdir** genera un messaggio di errore se cercate di cancellare un directory che non è vuoto.

```
$ pwd
/home/giorgio
$ rmdir dir1
rmdir: dir1 not empty
$
```

In questo modo non potete eliminare un directory che si trova nel file system, perché altrimenti lascereste orfani i file che sono contenuti nel directory che cercate di cancellare.

Potete invece cancellare un directory e ogni elemento in esso contenuto, inclusi subdirectory e file. Si tratta di un procedimento molto pericoloso, perché cancellate una appendice della gerarchia di directory. Utilizzate a tal proposito il comando **rm**, ma non **rmdir**, specificando l'argomento **–r** (recursive).

```
$ rm -r dir1
$
```

Se tutto procede correttamente, **rm** esegue quanto richiesto, cancellando **dir1** e ogni elemento in esso contenuto. Non dimenticatevi che si tratta di un'operazione molto rischiosa.

4.5 Il directory privato (home directory)

Quando entrate nel sistema UNIX, iniziate a operare da un punto particolare del file system che corrisponde al vostro directory privato o "home directory", in cui potete liberamente creare file e subdirectory, senza che altri utenti possano accedervi. La maggior parte degli utenti generalmente opera nell'home directory o in un subdirectory e non esiste conflitto tra i corrispondenti home directory.

Se il vostro directory privato rientra nella limitazione a 14 caratteri dei nomi di file, ogni subdirectory da voi creato sarà soggetto alla stessa limitazione; se il vostro directory non è soggetto a tale limitazione non vi saranno soggetti neanche i subdirectory.

Non preoccupatevi di dove vi trovate nel file system in quanto potete ritornare nell'home directory in qualunque istante tramite il comando **cd**. Negli esempi precedenti abbiamo utilizzato questo comando con un argomento, per referenziare una precisa locazione del file system (relativa o assoluta) in cui operare. Se non specificate nessun argomento al comando **cd**, ritornate immediatamente nel vostro home directory.

```
$ pwd
/home/giorgio/dir1/dir2
$ cd
$ pwd
/home/giorgio
$
```

Il sistema definisce una variabile di ambiente, denominata "HOME", che contiene il pathname associato al vostro home directory e utilizzato da quasi tutti i comandi che creano file.

```
$ echo $HOME
/home/giorgio
$
```

I seguenti due comandi

```
$ cd
$ cd $HOME
```

generano gli stessi effetti. Ogni sistema definisce la variabile di ambiente HOME, che non dovete modificare. Poiché alcuni comandi e applicazioni utilizzano questa variabile, alcuni sistemi possono impedirvi di modificarla, proteggendo così l'integrità del sistema.

4.6 Comandi per elaborare file

Il sistema definisce molti comandi per elaborare i file e trasferirli all'interno del file system. Utilizzate il comando **cp** (copy) per realizzare una esatta copia del file. Per esempio:

```
$ ls
PROVA     dir1        note
$ cp note note.copia
$ ls
PROVA     dir1        note        note.copia
$
```

In questo esempio, il primo argomento rappresenta il file esistente, mentre il secondo corrisponde al file che si vuole creare. Potete come al solito definire un pathname invece di un semplice nome di file:

```
$ cp ./note /home/giorgio/note.copia
```

Questa linea di comando produce lo stesso effetto dell'esempio precedente. Se il file o pathname referenziato esiste, **cp** cancella il vecchio file e ne fa una copia, per cui dovete assicurarvi che il nome del file referenziato sia corretto. Se il pathname referenziato rappresenta un directory esistente invece che un nome di file, la copia viene inserita nel directory specificato, utilizzando lo stesso identificatore come nome del file sorgente:

```
$ pwd
/home/giorgio
$ ls dir1
dir2        ciao        salve
$ cp note dir1
$ ls dir1
dir2        ciao        salve       note
$
```

Il comando **cp** accetta anche una lista di file che copia nel directory specificato. Quando ci sono più file da copiare è un errore specificare come destinazione un file e non un directory. In questo caso, il comando **cp** non ha effetto e viene evidenziato l'errore. Comunque, se il destinatario è un directory, **cp** vi copia tutti i file specificati.

```
$ pwd
/home/giorgio/dir1
$ ls dir2
alto        basso
cp salve ciao dir2
$ ls dir2
alto        basso     ciao      salve
$
```

Non potete usare come primo argomento un directory. Se intendete copiare il contenuto di un directory in uno nuovo, dovete indicare il nome di ogni file che volete copiare o esplicitamente o tramite un metacarattere.

```
$ pwd
/home/giorgio
$ cp dir1/* .
$
```

Se il comando **cp** termina l'esecuzione correttamente, lo shell riassume il controllo, in caso contrario **cp** invia un messaggio di errore sullo standard error.

Un comando simile al precedente **cp** e che segue le stesse regole di composizione della linea di comando è il comando **mv** che trasferisce il file da una locazione a un'altra o permette di cambiare il nome del file. Per esempio:

```
$ ls
dir1        ciao      salve
$ mv salve benvenuto
$ ls
dir1        ciao      benvenuto
$
```

Potete trasferire un file in un directory esistente, mantenendo lo stesso nome, o cambiare il nome del file quando lo trasferite, a seconda che l'argomento destinazione sia un directory esistente o il nome di un file.

```
$ pwd
/home/giorgio
$ mv note dir1/traccia
$ ls dir1
dir2        ciao      salve     note      traccia
$
```

Anche qui l'argomento può essere una lista di file o di pathname, di cui l'ultimo rappresenta l'argomento destinazione. In questo caso, l'argomento destinazione è generalmente un directory, che deve già esistere, se utilizzate **cp** o **mv**.

Il comando **mv** viene usato anche per cambiare il nome di un file o di un directory, per esempio:

```
$ mv vecchionome nuovonome
$
```

Questo rinomina il file *vecchionome* col nome *nuovonome*. Potete rinominare sia file che directory in questo modo; potete anche indicare se necessario una diversa locazione di destinazione.

Infine, il comando **ln** (link) agisce in modo analogo, tranne nel fatto che la sua funzione è di creare due nomi per lo stesso file.

```
$ ls
dir1        ciao        salve
$ ln salve sss
$ ls
dir1        ciao        salve       sss
$
```

In questo esempio non avete creato una copia del file **salve**, ma semplicemente un secondo nome che referenzia lo stesso file fisico. Se avete scritto o modificato il file **salve**, le modifiche compaiono anche nel file **sss**, a differenza di quanto accade facendo una copia del file con il comando **cp**. Il comando **ln** viene di solito utilizzato per avere lo stesso file in più di un directory.

Come succede per i comandi **cp** e **mv**, potete utilizzare il comando **ln** specificando una lista di nomi di file come argomenti sorgenti e un unico directory esistente come argomento destinazione (l'ultimo argomento della linea di comando). Non potete avere un directory come argomento sorgente, senza indicare il nome di tutti i file da collegare.

Quando, tramite il comando **ln**, create un altro file, il file acquisisce un "link" ulteriore. Se cancellate una delle due versioni del file con il comando **rm**, non cancellate in realtà il file, ma annullate uno dei diversi riferimenti del file. Il file scompare realmente dal sistema solo dopo aver eliminato l'ultimo riferimento o l'ultimo nome con cui il file veniva referenziato all'interno del file system. Una volta fatto questo, lo spazio su disco viene reso disponibile e il file cancellato non può più essere recuperato. Un file è sempre un componente di un directory e, se possiede più riferimenti, può essere allocato in uno o più directory contemporaneamente (con lo stesso nome o con nomi diversi). Comunque, questo discorso non vale per i directory, poiché non è possibile referenziare un directory con nomi differenti. Il file system è una struttura gerarchica, per cui un directory deve avere una collocazione fissa all'interno della gerarchia, con un directory padre e possibilmente alcuni subdirectory (directory figli). Un file può essere contemporaneamente presente in più locazioni del file system, ma non un directory.

4.7 Collegamenti simbolici (symlink)

Un'apparente eccezione alla stretta regola di gerarchia viene data dai collegamenti simbolici, *symbolic link* o *symlink*, che sono file che *puntano* ad altri file o a directory. Un collegamento simbolico ha un nome e una locazione nell'albero di directory, come un normale file o directory, ma, diversamente da un reale file o directory, un collegamento simbolico non ha alcun contenuto; esso agisce semplicemente da puntatore a un altro file o directory. Quando il sistema apre un normale file (come quando eseguite il **cat** su un file) esso legge il contenuto del file; tuttavia, quando il sistema apre un collegamento simbolico, legge unicamente un percorso dal collegamento simbolico, e in base a questo legge il file o directory al quale il collegamento punta. I collegamenti simbolici complicano notevolmente il file system UNIX; riserviamo la trattazione di alcune delle proprietà che essi presentano alla fine di questo capitolo.

Potete creare un collegamento simbolico con l'opzione **-s** (symlink) del comando **ln**, esempio:

```
$ ln -s note link.note
$
```

Il nome di file esistente precede il nome del collegamento simbolico nella linea di comando di **ln**. D'ora in avanti, quando specificate il file **link.note** il sistema realmente elaborerà il file **note**. I collegamenti simbolici possono interessare sia file che directory, per esempio:

```
$ ln -s dir1 link.dir
$
```

Questo comando crea nel file system un nuovo oggetto col nome **link.dir**, che agisce come un directory, come l'esempio dimostra:

```
$ ls
PROVA     dir1      note
$ ln -s dir1 link.dir
$ ls
PROVA     dir1      link.dir     note
$ ls dir1
dir2      ciao      salve
$ cd link.dir
$ ls
dir2      ciao      salve
$
```

Questo è il solo modo per avere un directory in due posizioni allo stesso tempo. Non si possono effettuare reali collegamenti fra nomi di directory;

un collegamento simbolico consiste solo in un puntatore a un altro file o directory, ma non esiste una reale connessione fra i due oggetti. Per questo motivo, è possibile cancellare l'oggetto reale senza influire sul collegamento simbolico, per esempio:

```
$ cd
$ mv dir1 DIR1
$ cd link.dir
link.dir: does not exist
$
```

Questo errore è comune. Ugualmente, molti comandi che percorrono la gerarchia di directory non seguono automaticamente i collegamenti simbolici come possono fare **cat** e **cd**. Se fate uso di collegamenti simbolici nei vostri directory dovete prendere delle precauzioni particolari per essere certi che i contenuti dei directory sotto i collegamenti simbolici vengano copiati o trattati come intendete.

I collegamenti simbolici non sono directory, pertanto non potete cancellarli con **rmdir**, anche se puntano a directory; dovete usare il comando **rm**, per esempio:

```
$ ln -s dir1 link.dir
$ rmdir link.dir
rmdir: link.dir: Path component not a directory
$ rm link.dir
$
```

Ricordate che quando cancellate un collegamento simbolico *non* viene cancellato il reale file o directory, ma solo il puntatore a tale oggetto.

4.8 Opzioni del comando ls

Quando abbiamo utilizzato in precedenza **ls** per visualizzare il contenuto di un directory, i file venivano elencati su video in ordine alfabetico, allineati in colonne dello schermo. Questo formato di visualizzazione si dimostra utile sul video, ma se l'output è destinato a un pipeline, una variabile di ambiente, o ad altri scopi, è necessario avere un file o directory per ogni linea. Il comando **ls** prevede parecchie opzioni che controllano l'immagine in uscita e altre che controllano la selezione dei file o directory che devono essere visualizzati. Se utilizzate l'opzione **-1** (1 colonna) per il comando **ls**, l'uscita su video visualizza un elemento per ogni linea; per esempio:

```
$ ls -1
dir1
PROVA
```

```
note
$ ls
dir1            PROVA           note
$
```

Generalmente il formato incolonnato si adatta meglio per il terminale. Attualmente **ls** utilizza automaticamente il formato di un elemento per linea se l'uscita viene ridiretta in pipeline, pertanto l'opzione –1 spesso può non essere necessaria. Se occorre, potete forzare l'output incolonnato con l'opzione –**C** (C maiuscolo per Columns). L'ordine dei file nella visualizzazione a colonne varia molto da sistema a sistema e il formato può dipendere da quanti file sono elencati. Verificate come si comporta sotto questo aspetto il vostro sistema.

Un'altra utile opzione per il comando **ls** è –**a**. Di fatto, **ls** non visualizza i nomi di tutti i file che sono contenuti nel directory, ma per esempio non visualizza quelli che iniziano con il carattere . (punto). Potete ottenere la lista di tutti i file presenti nel directory utilizzando il seguente comando:

```
$ pwd
/home/giorgio
$ ls
dir1        PROVA  note
$ ls –a
  .           . profile     dir1
  . .         PROVA         note
```

Ora si vedono diversi file che prima non erano presenti.

La maggior parte dei sistemi, infatti, inserisce dei file nascosti nel directory privato di ogni utente; il numero esatto di tali file dipende dal particolare sistema e dal software applicativo installato. I file che si vedono nell'ultimo esempio, però, sono di particolare interesse: il . (punto) corrisponde al directory corrente e .. (punto-punto) al directory padre. Tali file sono effettivamente elementi del directory in quanto ciascun directory contiene un puntatore a se stesso e al directory padre; l'implementazione dei directory nel file system avviene quindi tramite *liste concatenate*. La radice è un caso speciale in cui . e .. corrispondono allo stesso directory.

Il comando **ls** –**F** contrassegna i file a seconda del tipo: / corrisponde ai directory, * ai file eseguibili e @ ai collegamenti simbolici. Tali caratteri aggiuntivi non fanno parte dei nomi di file, ma vengono aggiunti da **ls**.

```
$ pwd
/home/giorgio
$ ln /usr/bin/cat .
$ ln –s dir1 link.dir
$ ls –F
PROVA     cat*     dir1/     link.dir@     note
$
```

In questo directory **cat*** è eseguibile, **dir1** è un subdirectory visualizzato con il carattere / e **link.dir@** è un collegamento simbolico.

Un'altra utile opzione di **ls** è **-R**, che sta per ricorsivo. È obbligatorio usare la *R* maiuscola in questo caso, perché la *r* minuscola corrisponde a un'altra opzione, che inverte l'ordine dell'uscita. Il comando **ls -R** percorre tutta la gerarchia di directory a partire da quello specificato come argomento (per default parte dal directory corrente) e visualizza tutti i file e i subdirectory che incontra.

```
$ pwd
/home/giorgio
$ ls -RF
PROVA     cat*     dir1/     link.dir@     note
./dir1:
dir2/     ciao     salve
/home/giorgio/dir1/dir2:
alto      basso
$
```

Questo comando per default fa riferimento al directory corrente in assenza di argomenti sulla linea di comando.

L'uscita del comando può essere ridiretta in un file o utilizzata come standard input per un comando di manipolazione di file. UNIX dispone di un altro comando, **find**, che viene spesso usato quando si desidera un elenco completo dei nomi dei file in una *sottostruttura* di directory, per esempio per creare una copia di backup su floppy disk di una parte del file system. **find** verrà descritto in dettaglio nel Capitolo 7.

4.9 Diritti di accesso dei file

Tutti i file e i directory presenti nel file system possiedono molti attributi, oltre al nome, che potete elencare tramite il comando **ls -l** (long listing). Per esempio:

```
$ pwd
/home/giorgio
$ ls -l
total 3
-rw-rw-rw-   1 giorgio   utenti   138 Apr  5   19:34   PROVA
drwxrwxrwx   3 giorgio   utenti    80 Apr  5   19:43   dir1
lrwxrwxrwx   1 giorgio   utenti     8 Apr  5   19:56   link.dir->dir1/
-rw-rw-rw-   1 giorgio   utenti   227 Apr  5   19:33   note
$
```

Il nome del file compare all'estrema destra e a ogni directory è associata una linea in uscita. La parte di linea che precede il nome del file, per esempio **Apr 5 19:34**, rappresenta la data e l'ora in cui avete modificato per l'ultima volta il file. L'indicazione dell'ora viene di solito utilizzata quando confrontate i file o volete risalire alla data di scrittura.

La colonna immediatamente a sinistra, contenente **138** o **227**, indica la dimensione del file in byte, che è compresa tra zero, per file appena creati, e uno o più megabyte. Il sistema definisce la variabile speciale **ulimit** per controllare la massima dimensione che può avere un file; spesso questa variabile ha un valore di due megabyte. Ricordatevi che anche un directory è un file, per cui possiede una dimensione in byte che però viene utilizzata raramente.

Se l'oggetto elencato è un collegamento simbolico, come **link.dir** nell'esempio precedente, il campo nome all'estrema destra mostra la particolare forma –> (puntatore a); il nome del collegamento simbolico appare per primo e *punta* a un secondo nome che è l'oggetto reale.

POSSESSO DEI FILE

Poiché UNIX è un sistema multiutente, ogni utente può creare i propri file, che referenzia fino a quando li cancella oppure li passa ad altri utenti, e viene inserito in un gruppo per cui può condividere i file con gli altri membri del suo gruppo, ma non con gli utenti che appartengono ad altri gruppi. Il Capitolo 22 tratta dettagliatamente i gruppi. Utilizzando il comando **ls –l** potete verificare quale utente e quale gruppo possiede un certo file: **giorgio** è il possessore del file e **utenti** è il nome del gruppo a cui **giorgio** appartiene. Quando create un file, il sistema lo assegna al gruppo di cui fate parte. Comunque, potete modificare l'assegnazione di un file a un gruppo o a un utente, per cui l'utente che compare in uscita dal comando **ls –l** può in realtà non appartenere al gruppo indicato.

La penultima colonna a sinistra nell'output di **ls –l**, contiene un numero di una sola cifra, che rappresenta il numero di riferimenti che il file possiede. Quando utilizzate **ln** per creare un altro nome del file, il numero si incrementa di una unità; quando cancellate un nome del file con **rm**, il numero si decrementa di una unità e quando raggiunge il valore zero il file viene cancellato.

INTERPRETAZIONE DEI PERMESSI DEI FILE

La colonna più a sinistra della linea di uscita, tipo **–rw–rw–rw–**, indica i permessi o diritti di accesso ovvero le modalità di utilizzo del file. La posizione più a sinistra dell'output è – (lineetta) se si tratta di un normale file, **d** se tratta di un directory, **l** (elle) se si tratta di un collegamento simbolico;

se in quella posizione compare una **b** o una **c** si tratta di un file *speciale*, utilizzato per controllare I/O con unità periferiche.

Le successive tre posizioni nella linea di uscita indicano i diritti di accesso per chi deve referenziare il file; le altre tre che seguono indicano i diritti di accesso per il gruppo che utilizza il file, mentre le ultime tre rappresentano i diritti di accesso per tutti gli altri utenti presenti nel sistema. Ognuno di questi tre gruppi è suddiviso in tre parti: accesso in *lettura* al file, accesso in *scrittura* e accesso in *esecuzione*. L'accesso in lettura stabilisce se il proprietario, il gruppo o tutti gli altri utenti possono leggere il file, per esempio, con **cat**. L'accesso in scrittura stabilisce chi è autorizzato a scrivere nel file, tramite editor oppure ridirigendovi l'uscita. L'accesso in esecuzione stabilisce chi può eseguire il file come un comando, referenziandolo cioè per nome come accade per esempio per **ls** o **rm**. Nel caso di directory, i significati di questi tre gruppi di caratteri differiscono leggermente. L'accesso in lettura indica chi può analizzare il contenuto di un directory, per esempio, con **ls**; l'accesso in scrittura indica che l'utente può creare un file in quel directory, mentre l'accesso in esecuzione indica che l'utente può attraversare il directory e addentrarsi nei subdirectory.

I collegamenti simbolici possiedono i permessi degli oggetti a cui puntano; i permessi dei collegamenti stessi sono privi di significato.

Il comando **ls −l** visualizza i diritti di accesso, specificando **r** per l'accesso in lettura, **w** per l'accesso in scrittura e **x** per l'accesso in esecuzione. Per esempio, la sequenza di caratteri **−rw−r−−−−−** indica che il proprietario del file può leggere e scrivere nel file, gli appartenenti allo stesso gruppo del proprietario del file possono leggere, ma non scrivere nel file, mentre gli altri utenti non hanno alcun accesso. La sequenza **−r−−−−x−−x** indica che il proprietario del file può leggerlo, ma non scriverci o eseguirlo, mentre gli appartenenti allo stesso gruppo del proprietario del file e tutti gli altri utenti possono eseguire il file, ma non leggerlo o scriverci.

LISTA DEI PERMESSI DI DIRECTORY

Esiste una importante opzione per il comando **ls**. Se scrivete **ls −l**, compaiono su video le informazioni relative al file, ma lo stesso comando riferito a un directory produce in uscita l'elenco dei file presenti in quel directory.

```
$ ls −l dir1
total 1
drwxrwxrwx      2 giorgio   utenti   64 Apr 18 12:43 dir2
−rw−rw−rw−      1 giorgio   utenti    0 Apr 17 17:42 ciao
−rw−rw−rw−      1 giorgio   utenti    0 Apr 17 17:42 salve
$
```

Questa è l'informazione più importante sul contenuto di un directory.

Per avere informazioni sui diritti di accesso al directory stesso, potete utilizzare il comando **cd** per referenziare il directory padre e lanciare il comando **ls -l**, come abbiamo fatto nell'esempio precedente per il directory **dir2**. Per avere informazioni sul directory **dir1** quando vi trovate nel directory preferenziale, operate nel modo seguente:

```
$ cd ..
$ ls -l giorgio
```

Il comando **ls** dispone di un modo più efficiente per ottenere gli stessi risultati, senza richiedere di cambiare directory.

```
$ ls -ld dir1
drwxrwxrwx      3 giorgio   utenti     80 Apr 18 12:43 dir1
$
```

Utilizzate **ls -d** (directory listing) per avere il nome di un directory invece che il suo contenuto. Aggiungendo a questo comando anche l'opzione **-l**, potete verificare lo stato del directory.

MODIFICA DEL POSSESSO DI FILE

Il sistema UNIX mette a disposizione diversi comandi che permettono di trattare il possesso e i diritti di accesso di file e directory. Se siete proprietari di un file, potete cederlo a un altro utente con il comando **chown** (change owner).

```
$ ls -l note
-rw-rw-rw-   1 giorgio   utenti    227 Apr 5 19:33 note
$ chown note  leo
$ ls -l note
-rw-rw-rw-   1 leo       utenti    227 Apr 5 19:33 note
```

Il comando **chown** riceve come argomenti un file o una lista di file e, come ultimo argomento, l'identificatore di utente a cui volete cedere il file, che deve essere già stato definito nel sistema. Una volta che il file è stato assegnato a un altro utente, non potete utilizzare il comando **chown** per riaverlo indietro.

Utilizzate il comando **chgrp** per modificare il gruppo che possiede un file, se siete proprietari di quel file e appartenete a quel gruppo:

```
$ ls -l note
-rw-rw-rw-   1 giorgio   utenti    227 Apr 5 19:33 note
$ chgrp note bin
$ ls -l note
-rw-rw-rw-   1 giorgio   bin       227 Apr 5 19:33 note
```

Inoltre, quando avete ceduto il file a qualcun altro, non lo potete più referenziare e quindi avere indietro. Naturalmente, se siete autorizzati a leggere il file, potete farne una copia con il comando:

```
$ cat note > mio.note
```

o con un comando analogo. Quando create un file, ne diventate automaticamente proprietari.

MODIFICA DEI PERMESSI DI FILE

Da ultimo, potete cambiare i diritti di accesso di un file di cui siete proprietari tramite il comando **chmod** (change mode). Un file può avere tre categorie di diritti di accesso: utente, gruppo e altri, che vengono rispettivamente indicate con **u** (user), **g** (group) e **o** (other) e che corrispondono alle tre categorie che sono elencate dal comando **ls -l**. Ogni categoria di utenti può avere accesso al file in lettura, scrittura o esecuzione, e questa situazione viene indicata rispettivamente con i caratteri **r**, **w** e **x**. Potete utilizzare questa notazione per definire un comando **chmod**:

```
$ ls -l note
-rw-rw-rw-   1 giorgio    utenti     227 Apr 5 19:33 note
$ chmod -w note
$ ls -l note
-r--r--r--   1 giorgio    utenti     227 Apr 5 19:33 note
$ chmod +w note
$ ls -l note
-rw-rw-rw-   1 giorgio    utenti     227 Apr 5 19:33 note
$
```

Nel primo caso abbiamo specificato per **chmod** l'argomento **-w**, per cui viene rimosso il diritto di accesso in scrittura al file. Il secondo esempio aggiunge di nuovo il diritto di accesso in scrittura al file. L'uso del carattere + (segno più) come flag è una eccezione all'uso normale di -, ma potete capirne il significato: il segno - permette di rimuovere il diritto di accesso, mentre il segno + provvede a ripristinarlo.

Potete cambiare con lo stesso comando più di un diritto di accesso, premettendo una o più lettere al carattere - o +, per esempio:

```
$ chmod -w+x note
$ ls -l note
-r-xr-xr-x   1 giorgio    utenti     227 Apr 5 19:33 note
$ chmod -wx note
$ ls -l note
-r--r--r--   1 giorgio    utenti     227 Apr 5 19:33 note
$
```

In tutti questi esempi, **chmod** ha cambiato il diritto di accesso per tutte le tre classi di utenti. Potete anche effettuare modifiche per ogni singola classe aggiungendo una lettera prima del segno – o del segno +.

```
$ ls -l note
-rw-rw-rw-   1 giorgio    utenti    227 Apr 5 19:33 note
$ chmod u-w note
$ ls -l note
-r--rw-rw-   1 giorgio    utenti    227 Apr 5 19:33 note
$ chmod go+wx note
$ ls -l note
-r--rwxrwx   1 giorgio    utenti    227 Apr 5 19:33 note
$
```

La sintassi del comando **chmod** consiste nella classe di utente (**u**, **g** od **o**), seguita dall'azione da intraprendere (– o +), seguita dal diritto di accesso da cambiare (**r**, **w** o **x**). La lista dei nomi dei file o dei directory da cambiare completa la linea di comando.

Il comando **chmod** può anche ricevere un argomento numerico che indica la classe di utente e il diritto di accesso da cambiare come sequenza di bit.

```
$ chmod 0466 note
```

Questa notazione comunque è più propensa a errori rispetto alle precedenti. Il numero 0466 è nella notazione ottale. Inoltre il comando **chmod** può assegnare ai diritti di accesso i valori **s** e **t**, oltre a **r**, **w** e **x**. I diritti di accesso **s** e **t** vengono utilizzati da alcuni programmi eseguibili, ma non dai singoli utenti, per modificare l'ambiente di esecuzione e in alcuni sistemi possono essere visualizzati in uscita dal comando **ls -l**.

4.10 Approfondimenti

Molte delle caratteristiche interessanti del file system sono associate con l'esecuzione dei comandi, come accade per il diritto di accesso **s**. Alcune di queste caratteristiche migliorano il collegamento esistente tra il nome di un file in un directory e la memorizzazione fisica effettiva dei dati su disco (o su altri supporti). Il sistema permette di definire altri file system in qualunque locazione interna alla gerarchia di directory. Potete perfino configurare il sistema in modo tale da utilizzare una rete locale per permettere a un file di risiedere su un disco di un'altra macchina. Molte di queste caratteristiche sono associate alla gestione e amministrazione del sistema, concetti questi che tratteremo nei Capitoli 12, 23 e 24.

FILE MANAGER DI X WINDOW SYSTEM

Se sul vostro sistema è disponibile X Window System potete usare il File Manager in luogo delle funzioni di linee di comando descritte in questo capitolo. Per un esempio di visualizzazione del File Manager, tornate alla Figura 2.1; il File Manager di OPEN LOOK appare a sinistra in basso nel video. Molti pacchetti X includono un'applicazione simile a questa. Per maggiori informazioni sulla gestione delle finestre in X, consultate la prima parte del Capitolo 23.

Il File Manager consente di spostarsi all'interno del file system, di selezionare un sottoinsieme di file in un directory per eseguire una operazione come copia o cancellazione, e aprire subdirectory. Il contenuto di directory viene visualizzato nell'area in basso nella finestra File Manager, il percorso assoluto necessario per l'accesso a ciascun directory compare in una linea di componenti al centro della finestra; vedi Figura 2.1.

Accanto al nome di ogni componente del contenuto di un directory appare una icona che si richiama alla funzione o alla natura dell'oggetto: i directory appaiono come cartellette di raccoglitori, i file presentano un angolo ripiegato e i programmi eseguibili mostrano una finestra con bordi. Possono comparire anche molti altri tipi di icone.

Con un click col tasto sinistro del mouse su di un oggetto, l'oggetto viene selezionato per eseguire una qualche azione che potrà di seguito essere selezionata tra i menu nella finestra File Manager, a sinistra in alto. Con un doppio click col tasto sinistro del mouse su di un oggetto, quell'oggetto viene aperto. Se l'oggetto è un directory (cartelletta), il contenuto del directory rimpiazza il contenuto corrente della finestra, e il directory compare nella lista al centro della finestra File Manager; altri oggetti causano azioni differenti. Nuove finestre appaiono se selezionate una elaborazione di testo, o se lanciate l'esecuzione di un programma. Quando un'applicazione termina, o viene chiusa con l'opzione Quit nel menu Window, la finestra dell'applicazione scompare.

In alternativa, per spostarsi nel file system, in luogo del doppio click sulle icone di cartellette, potete digitare un nome di percorso nella linea di directory in alto a destra, e poi col click sulla voce Match nel menu saltare a quel directory.

Potete selezionare un sottoinsieme di file nel directory selezionato digitando uno schema di nome di file con metacaratteri (operatori * ? [ ]) nella zona Pattern e poi dare il click su Match.

Potete accedere a più complesse operazioni in file system tramite i menu File..., View..., Edit. Questi menu consentono di prestabilire l'ordine di visualizzazione di file, di esaminare i permessi e altri attributi di file, di creare file e directory.

I COMANDI basename E dirname

Il sistema UNIX definisce alcuni comandi per derivare nomi di file e directory dai pathname completi e viceversa. Per esempio, potete avere una variabile di ambiente che contiene un pathname completo e chiedere di conoscere solo il nome di un particolare file.

```
$ echo $EDITOR
/usr/bin/vi
$ MYEDIT = 'basename $EDITOR'
$ echo "Il mio editor è $MYEDIT"
Il mio editor è vi
$
```

Utilizzate il comando **basename** per avere sullo standard output l'ultimo componente del pathname completo che indica generalmente il nome di un file spogliato del suo directory. Il comando **dirname**, al contrario, fornisce solo la parte directory, spogliata del nome del file.

```
$ EDDIR = 'dirname $EDITOR'
$ echo "Il mio editor $MYEDIT è nel directory $EDDIR"
Il mio editor vi è nel directory /usr/bin
$
```

Potete utilizzare in molti modi questi strumenti di gestione dei pathname quando parleremo della programmazione di shell nel Capitolo 8.

FILE SPECIALI (DEVICE FILE)

Abbiamo trattato i file, symlink e directory di uso abituale, ma il comando **ls** evidenzia anche un terzo tipo di file, che prende il nome di file speciale o semplicemente dispositivo (device). Il sistema UNIX fornisce una interfaccia standard tra le periferiche e il sistema operativo che è strutturata come un file normale, per cui le operazioni di I/O all'hardware seguono regole analoghe a quelle trattate nella scrittura di un file. Non si tratta di un file su disco come quelli di cui abbiamo parlato, ma di uno speciale pathname che referenzia il canale di I/O all'hardware. Come per la ridirezione dello standard I/O a un file normale, potete ridirigere facilmente l'uscita ai dispositivi a livello di shell. Questo è argomento di particolare interesse per chi sviluppa e crea sistemi operativi e in questo contesto ne viene data solo una breve introduzione.

Il comando **tty** (da Teletype Corporation, telescrivente un tempo usata come terminale) restituisce il pathname associato con il vostro terminale.

```
$ tty
/dev/tty00s
$
```

È possibile accedere al vostro terminale tramite il file **/dev/tty00s**; lo standard input allo shell, che è in attesa dei vostri comandi (shell di login) è collegato a questo file quando accedete al sistema. I pathname dei dispositivi variano a seconda del sistema e cambiano quando vi collegate con il sistema da differenti terminali, per cui utilizzate il comando **tty** se avete necessità di determinare la periferica da cui vi collegate. Il directory **/dev** contiene molti altri dispositivi (device file) oltre ai terminali. Potete utilizzare questo pathname secondo le modalità abituali all'interno del file system.

```
$ ls -l /dev/tty00s
crw-rw-rw-   1 giorgio     utenti        7,  2 Apr 19 13:21 /dev/tty00s
$
```

Il primo carattere della linea di uscita, **c**, indica che non si tratta di un file normale o di un directory, ma di uno *speciale file di caratteri*, designato per trasferire dati, carattere per carattere, come un terminale, un modem o una stampante. Un altro tipo di file speciale è l'unità periferica a blocchi, contrassegnata dalla lettera **b** (block) in prima posizione, che permette di trasferire grandi quantità di dati contemporaneamente, come i disk drive e alcune unità periferiche a nastro. Potete utilizzare questa unità periferica come un file system.

I device file possiedono diritti di accesso come tutti gli altri file e alcuni sono referenziabili dall'utente corrente. Nell'esempio che abbiamo discusso, potete leggere e scrivere nel file, che permette di ricevere e trasferire dati sul vostro terminale, e altri utenti possono leggere il file, cioè seguire, se lo vogliono, la vostra sessione di lavoro. Per fare questo, inserite la seguente linea di comando:

```
$ cat - < /dev/tty00s
```

Provate ancora questo sul vostro terminale, battendo:

```
$ cat - < 'tty'
```

Potete digitare alcuni caratteri, poi premere il tasto DEL per interrompere l'esecuzione del comando **cat** e riportare il terminale nella situazione normale.

Verificate cosa accade con quest'altra versione del comando:

```
$ cat - > 'tty'
```

IL COMANDO mesg

Nell'esempio precedente potevate direttamente leggere e scrivere in un device file che rappresenta il vostro terminale. Se i diritti di accesso elencati dal comando **ls -l 'tty'** mostrano che altri utenti possono leggere e scrivere nel vostro device file, allora sono in grado di seguire la vostra sessione di lavoro o scrivere direttamente sul vostro terminale ridirigendo l'I/O. Questa situazione rappresenta un potenziale rischio per la sicurezza, in quanto altri utenti possono intercettare il vostro I/O. Comunque, il gestore del sistema, o il sistema stesso attraverso il software, qualche volta decide di inviare un messaggio direttamente al vostro terminale. Questo può accadere quando il sistema si spegne e il gestore vi vuole avvisare in modo che possiate salvare i file e disallocarvi senza problemi. I comandi **wall** (write to all users) e **write** (writing message between users) utilizzano direttamente l'I/O al vostro terminale per le comunicazioni. Il Capitolo 14 tratta diffusamente di questi comandi.

Potete controllare l'accesso di altri utenti al vostro terminale semplicemente cambiando i diritti di accesso sul vostro file di periferica. Potete fare questo direttamente con il comando **chmod**, ma il sistema definisce espressamente un comando per permettere o proibire messaggi come questo. Utilizzate il comando **mesg** (message) per autorizzare o impedire ad altri utenti di scrivere o leggere dal vostro terminale. Questo comando accetta un solo argomento, **y** o **n** rispettivamente per accettare o rifiutare i messaggi. Per esempio:

```
$ ls -l /dev/tty00s
crw--rw--rw-   1 giorgio     utenti       7,  2 Apr 20 18:37   /dev/tty00s
$ mesg n
$ ls -l /dev/tty00s
crw--r----r--  1 giorgio     utenti       7,  2 Apr 20 18:38   /dev/tty00s
$ mesg y
$ls -l /dev/tty00s
crw--rw--rw-   1 giorgio     utenti       7,  2 Apr 20 18:37   /dev/tty00s
$
```

Il comando **mesg** cambia proprio i diritti di accesso sul vostro file di periferica.

ALTRI FILE SPECIALI

Oltre ai dispositivi esterni **tty**, esistono molti altri device file nel directory **/dev**. Uno tra i più interessanti è **/dev/null**, che rappresenta un contenitore di dimensione infinita da utilizzare per ridirigere i risultati da eliminare. Per esempio:

```
$ cat /etc/passwd > /dev/null
```

L'uscita del comando **cat** viene in questo caso annullata.

Un altro device file di particolare interesse è **/dev/kmem** (kernel memory), che costituisce una rappresentazione della memoria reale presente nella macchina. Programmi speciali come i rivelatori di errore (debugger) possono leggere **/dev/kmem** per controllare in qualunque momento come il sistema utilizza la memoria. Comunque, agli utenti abituali viene impedito di analizzare (o scrivere!) **/dev/kmem**, perché l'accesso al sistema rappresenta una violazione di sicurezza.

Inoltre, tutte le periferiche collegate al sistema, come per esempio le porte di terminale, le porte di stampante e i disk drive vengono rappresentate simbolicamente nel directory **/dev**.

COLLEGAMENTI SIMBOLICI

Come detto in precedenza, l'introduzione dei collegamenti simbolici complica notevolmente il file system UNIX; nel sistema i collegamenti simbolici sono largamente presenti. Diversi collegamenti simbolici possono puntare allo stesso file, e un collegamento può puntare fuori da un file system (uno che non prevede limite di 14 caratteri) dentro un altro (che potrebbe avere il limite di 14 caratteri); di conseguenza non possono essere applicate le vecchie regole della stretta gerarchia dell'albero dei directory. In SVR4 l'organizzazione dell'intero sistema (cioè partendo dal directory root /) è molto complessa, e spostandosi nel file system occorre tener conto dei collegamenti simbolici e trattarli accuratamente. Molti programmi che percorrono il file system, come **find**, per default non seguono i collegamenti simbolici, che richiedono un trattamento molto particolare. Per esempio, avendo un collegamento che punta a un directory, come **link.dir** negli esempi precedenti, potete usare **cd** per spostarvi su quel directory e il comando **pwd** indicherebbe correttamente la collocazione del directory, come mostrato qui di seguito:

```
$ cd link.dir
$ pwd
/home/giorgio/link.dir
$
```

Questo dà l'impressione che il collegamento simbolico sia realmente un directory, e che **link.dir** sia il vostro directory corrente; effettivamente, uno degli scopi del symlink è appunto di creare questa impressione. In realtà, **link.dir** *non* è un directory e il vostro vero directory corrente è **dir1**, supponendo che sia puntato da **link.dir**; comunque questi due comandi sono equivalenti:

```
$ cd /home/giorgio/dir1
$ cd /home/giorgio/link.dir
$
```

Nonostante ciò l'output di **pwd** risulta differente nei due casi che seguono:

```
$ cd /home/giorgio/dir1
$ pwd
/home/giorgio/dir1
$ cd /home/giorgio/link.dir
$ pwd
/home/giorgio/link.dir
$
```

L'esempio dimostra come sia possibile raggiungere una singola locazione nel file system attraverso percorsi differenti. Se non tenete presente ciò, potete ritenere che questi differenti percorsi conducano a locazioni effettivamente differenti e apportare delle modifiche a una destinazione apparente, producendo danni alle altre.

Usualmente, i comandi **pwd** e **cd** sono integrati nello shell, e anche questo fatto contribuisce a fornire l'impressione che il file system abbia realmente la gerarchia che *voi* usate per spostarvi nel suo interno, mentre con i symlink possono esistere molte altre versioni (può esistere più di un symlink a un file o directory). In molti casi questo semplifica le cose, perché voi vedete sempre la versione di file system che state usando; tuttavia esiste una rappresentazione di file system *reale* o *obiettivo* che rispetta la stretta gerarchia. Dato che i symlink sono dei puntatori, devono puntare a qualcosa di reale, e quello è il file system *obiettivo* costituito da file e directory reali. Tutti i comuni programmi shell (**sh**, **csh**, **ksh**) implementano **pwd** restituendo il percorso che avete usato per raggiungere un particolare directory, non il percorso reale. Al contrario, il programma **/usr/bin/pwd** non segue questa convenzione di shell e restituisce sempre il percorso reale del directory corrente di lavoro, come mostrato qui:

```
$ cd link.dir
$ pwd
/home/giorgio/link.dir
$ /usr/bin/pwd
/home/giorgio/dir1
$
```

Se state scorrendo il file system per esaminarlo, usate spesso **/usr/bin/pwd** per verificare in quale directory vi trovate *realmente*.

ESAME DEL FILE SYSTEM

In questo capitolo abbiamo focalizzato l'attenzione principalmente sull'home directory e sui file e directory che create in esso per soddisfare le vostre esigenze. Naturalmente si tratta solo di una parte del file system, che risulta nel suo complesso una struttura enorme e diversificata. Nei sistemi com-

pleti SVR4 esistono più di 6500 file e 850 directory, anche se circa un quarto del totale è costituito da strumenti di System Administration e circa un decimo dal Software Development Set, che possono non essere installati sul vostro sistema. Circa 800 file sono comandi, i collegamenti simbolici sono circa 350. Inoltre, i singoli utenti e il software applicativo possono incrementare notevolmente i valori indicati. Molti di questi file svolgono funzioni particolari all'interno del sistema e la loro mancanza, o l'errata conformità dei diritti di accesso alle reali esigenze, genera seri problemi e risultati estremamente variabili. Quando si verificano situazioni anomali, è difficile trovare e risolvere il problema senza eseguire una operazione di ricaricamento del sistema. Di solito dovete essere molto sicuri quando modificate o cancellate un file che non si trova nel vostro home directory e non dovete avere nessuna incertezza quando cambiate i diritti di accesso di un file presente nel sistema. D'altra parte, inoltrarsi nel file system può essere motivo di grande divertimento e di conoscenza del funzionamento più intimo del sistema. Solo pochi file non sono leggibili, ma possono diventarlo se le considerazioni sulla sicurezza non sono di fondamentale importanza nella vostra macchina.

Partendo dal directory "root", esistono diversi file e subdirectory di notevole importanza, come viene indicato qui di seguito:

```
$ ls -F /
bin@      export/    lib@          proc/     tmp/      var/
boot/     home/      lost+found/   sbin/     u/
dev/      home2/     mnt/          shlib/    unix*
etc/      install/   opt/          stand/    usr/
$
```

Questi pochi directory provvedono ai punti di entrata nella intera gerarchia sottostante; da qui potete esaminare l'intero file system. La Tabella 4.1 contiene un sommario dei principali file e directory in sistemi SVR4; se andate a esplorare il file system consultate questa tabella per avere informazioni su ciascun directory incontrato.

RIORGANIZZAZIONE DEL FILE SYSTEM IN SVR4

L'intera organizzazione del file system di SVR4 è profondamente cambiata rispetto a SVR3 e release precedenti, principalmente a causa delle esigenze dei sistemi in rete o sistemi distribuiti. Alcune parti del sistema sono specifiche di una particolare macchina, come certi file di controllo o file del giornale storico di macchina, mentre altre possono essere condivise fra differenti macchine dello stesso tipo, e altri ancora possono essere condivise fra quasi tutte le macchine indipendentemente dalla versione di sistema UNIX utilizzata. Questo ha portato a situazioni per cui molte parti note del siste-

Tabella 4.1 La gerarchia del file system SVR4.

Path	Descrizione
/bin	Symlink a **/usr/bin**
/boot	File di inizializzazione
/dev	File speciali per i device
/dev/dsk	Device dischi
/dev/fd	Device per descrittori di file aperti
/dev/kd	Device tastiera e video
/dev/kmem	Memoria
/dev/null	Device nullo
/dev/osm	Messaggi di errore del kernel
/dev/pts	Pseudo **ttys**; uguale a **/dev/pts***
/dev/rdsk	Device dischi raw
/dev/term	Terminali; uguale a **/dev/tty***
/dev/xt	Pseudo **ttys**, per DMD
/etc	Gestione specifica della macchina
/etc/Backup	Lista di directory per salvataggi
/etc/Ignore	Lista di directory da ignorare nei salvataggi
/etc/X0.hosts	Lista di abilitazione host X
/etc/bkup	File di controllo dei salvataggi
/etc/bupsched	Orario per auto-salvataggi (**ckbupscd**)
/etc/conf	Materiale per riconfigurazione kernel
/etc/cron.d	File di controllo per **cron**
/etc/default	File di controllo per i default a inizializzazione
/etc/fs	Strumenti eseguibili per tutti i tipi di file system
/etc/group	Attributi dei membri di gruppi
/etc/inittab	Caratterizzazione dei processi a inizializzazione
/etc/issue	"Welcome ..." stampato al login
/etc/lp	File di informazione e interfaccia per **lp**
/etc/mail	File di controllo della posta; configurato per **/bin/mail**
/etc/mnttab	Tabella dei file system correntemente montati
/etc/motd	Messaggio del giorno
/etc/net	File di controllo per servizi di rete e host
/etc/profile	Profilo di login a livello sistema
/etc/rc?.d	Directory per procedure di inizializzazione
/etc/rfs	File e comandi per RFS
/etc/rstab	Mount remoti
/etc/saf	File di controllo per SAF
/etc/shadow	File delle password
/etc/ttydefs	File di controllo per SAF
/etc/ttytype	Default per login su canali **tty**
/etc/uucp	File di controllo per **uucp**
/export	Esportati in rete
/home	Home directory degli utenti
/home/oasys	File di supporto gestione OA&M
/home/vmsys	File di supporto gestione OA&M

Tabella 4.1 La gerarchia del file system SVR4 (*continua*).

Path	Descrizione
/home2	Home directory degli utenti
/install	Pacchetti aggiuntivi
/lib	Symlink a **/usr/lib**
/mnt	Punto di mount
/mnt1	Punto di mount
/opt	Materiale aggiuntivo
/proc	Processi correnti
/sbin	**bin** di gestione
/shlib	Librerie condivise
/stand	File di inizializzazione e kernel
/tmp	File temporanei
/u	Home directory degli utenti
/usr	File che non cambiano (read only)
/usr/X	Strumenti di supporto X Window System
/usr/X/bin	Eseguibili di X
/usr/X/clients	File dati di supporto OL
/usr/X/include	File di inclusione per X
/usr/X/lib	File dati di supporto X
/usr/add – on	File di supporto per pacchetti addizionali
/usr/admin	Menu di gestione per pacchetti addizionali
/usr/bin	Comandi eseguibili dagli utenti
/usr/ccs	Sistema di compilazione C
/usr/include	Definizioni del sistema in file di testata
/usr/lbin	**bin** locale, comandi **installpkg**
/usr/lib	Librerie e librerie condivise
/usr/lib/acct	Strumenti di contabilizzazione
/usr/lib/class	File di controllo per le classi di priorità
/usr/lib/installed	Informazioni sui pacchetti installati
/usr/lib/layersys	File di controllo per layer
/usr/lib/locale	File e directory specifici per nazionalità e ora
/usr/lib/lp	Strumenti statici per **lp**
/usr/lib/mail	File di controllo per **/bin/mail**
/usr/lib/netsvc	File di controllo per comandi NFS
/usr/lib/nfs	Demon e file di controllo per NFS
/usr/lib/rsh	Shell con limitazioni
/usr/lib/sa	File di controllo per sistema di contabilizzazione **sa**
/usr/lib/saf	Servizi per SAF
/usr/lib/spell	File di controllo per **spell**
/usr/lib/terminfo	Symlink a **/usr/share/lib/terminfo**
/usr/lib/uucp	File di controllo per **uucp**
/usr/net	File di controllo per RFS e **nls**
/usr/news	File notizie; symlink a **/var/adm/news**
/usr/nserve	Gestione RFS; symlink a **/etc/rfs**
/usr/options	Nomi dei pacchetti installati

Tabella 4.1 La gerarchia del file system SVR4 (*continua*).

Path	Descrizione
/usr/sadm	Strumenti di gestione del sistema per OA&M
/usr/sbin	Eseguibili dei comandi di gestione
/usr/share	Condivisa fra architetture di macchina
/usr/share/lib	Materiale condiviso specifico per applicazioni
/usr/share/lib/spell	File di controllo per **spell**
/usr/share/lib/terminfo	File di controllo terminali per **vi**, ecc.
/usr/spool	Symlink a **/var/spool**
/usr/src	Codice sorgente, quando presente
/usr/tmp	File temporanei durante compilazioni
/usr/ucb	**bin** per strumenti compatibilià BSD
/usr/ucbinclude	File di include per strumenti compatibilità BSD
/usr/ucblib	Librerie per strumenti compatibilità BSD
/usr/vmsys	Strumenti per gestione di programmi aggiuntivi SVR3
/var	File che cambiano durante la vita del sistema
/var/adm	Controlli di contabilizzazione; symlink a **/usr/adm**
/var/lp	Giornale di macchina per **lp**
/var/sadm	File di supporto OA&M
/var/spool	File temporanei per **uucp**, stampe, ecc.
/var/uucp	Giornale di macchina per **uucp**

ma, raggruppate logicamente in un sottoalbero di un directory secondo la *funzione*, risiedono adesso in directory differenti. Il file system è cambiato da una organizzazione funzionale a una organizzazione che è determinata dalla *macchina* che in un ambiente eterogeneo di rete ha la responsabilità della manutenzione e dell'uso del file.

Questa riorganizzazione si è resa necessaria per supportare pienamente una versione in rete del sistema UNIX; se avete esperienza dei precedenti sistemi UNIX dovrete imparare di nuovo la collocazione di ogni file e directory. D'altra parte, i collegamenti simbolici nel sistema puntano i vecchi nomi alle nuove collocazioni; se, scorrendo per il file system ne esaminate il contenuto, fate attenzione di non confondere i file reali con i symlink che si presentano come se fossero il reale file system. Usate frequentemente **/usr/bin/pwd** per verificare dove realmente siete posizionati nella nuova organizzazione.

CONVENZIONI PER NOMI DI FILE E DIRECTORY

Il sistema UNIX segue alcune convenzioni relativamente standard che riguardano la denominazione dei directory; alcune nuove convenzioni sono introdotte in SVR4. Usualmente, anche differenti applicazioni e gli stessi

utenti seguono queste convenzioni nei propri directory, per quanto le regole non siano obbligatorie fuori dei directory di sistema.

I nomi di directory **bin** (binary), **sbin** (standalone bin), **lib** (library), **src** (source), **man** (manual), **usr** (users) ed **etc** (etcetera) sono di uso ricorrente. La maggior parte dei programmi eseguibili risiede nel directory **bin**, gran parte delle librerie di sviluppo e altro materiale di supporto risiede nel directory **lib**, la maggior parte del codice sorgente per applicazioni e comandi si trova nel directory **src**, gran parte della documentazione risiede nel directory **man**, la maggior parte del materiale degli utenti risiede in un directory **home** o **usr** e gran parte del materiale di supporto risiede nel directory **etc**.

Potete osservare molti di questi nomi se chiedete di visualizzare il contenuto del directory "root". Il directory **/usr/bin** contiene la maggior parte dei comandi, **/bin** è spesso un symlink a **/usr/bin**, molti dei restanti comandi sono nel directory **/sbin** e **/usr/sbin**. Usualmente i directory **sbin** contengono i principali comandi di gestione, i directory **bin** contengono i comandi di utente. Molti sistemi possono includere un altro directory, **/usr/local** o **/local/bin** per contenere comandi che non fanno parte del sistema UNIX ufficiale, ma sono stati aggiunti a quella specifica macchina.

Il directory **/etc** contiene una grande quantità di file e strumenti software che vengono utilizzati nella gestione del sistema UNIX. Per esempio, **/etc/rc2.d** è un directory che contiene i file che vengono utilizzati quando il sistema viene caricato, mentre **/etc/passwd** è un file che contiene la lista di utenti che sono autorizzati ad accedere alla macchina.

Le parti del sistema operativo UNIX che costituiscono il kernel sono contenute in **/etc/conf**; questo materiale viene utilizzato per aggiornare il nucleo quando aggiungete alla macchina nuovo hardware che richiede device driver (programmi pilota) di periferiche. La versione corrente di kernel caricata quando la macchina viene inizializzata si trova sotto il directory **/stand**; una copia viene mantenuta in **/unix** per compatibilità con il vecchio software, ma non viene attualmente usata come kernel di inizializzazione. Non dovete mai modificare o cancellare manualmente il contenuto di questi directory, anche se occupa una grande quantità di spazio su disco. Il directory **/shlib** (shared libraries) contiene le librerie di subroutine caricate dinamicamente per supportare molte applicazioni e parti addizionali del sistema operativo UNIX.

Il Software Development Set risiede principalmente in **/usr/ccs** e suoi subdirectory, in parte sotto **/usr/include** e **/usr/lib**. Spesso **/lib** è un collegamento simbolico a **/usr/lib**.

Il directory **/dev** contiene i file speciali che gestiscono le connessioni coi dispositivi esterni, **/tmp** viene utilizzato nelle applicazioni come memoria temporanea di file, specialmente in fase di sviluppo di software. Potete utilizzare **/tmp** anche per i vostri file temporanei. Questi file vengono cancellati quando il sistema è ricaricato. Anche **/var/tmp** è un directory per file temporanei e in alcuni sistemi non viene svuotato all'inizializzazione.

I directory privati dei diversi utenti risiedono generalmente sotto il directory **/home**, ma alcuni sistemi possono usare in alternativa **/usr** o **/u**. Il codice sorgente completo del sistema UNIX si trova generalmente nel directory **/usr/src** e nei subdirectory relativi, anche se questo è raramente presente sui microelaboratori. Il directory **/usr/lib/uucp** comprende il supporto per il sottosistema di comunicazione **uucp**, **/usr/share/lib/terminfo** contiene le descrizioni delle caratteristiche specifiche dei terminali che possono essere collegati al sistema. Se presente, **/usr/man** contiene la documentazione del sistema tra cui il testo dello *UNIX User's Manual*.

Il directory **/usr** contiene numerosi altri oggetti, molti dei quali sono collegamenti simbolici a directory in **/var** (variable), i quali contengono oggetti che sono soggetti a modifiche nel corso di operazioni di gestione della macchina. Molti giornali di macchina, directory di spool e altre variabili sono contenuti in **/var**.

Il sottoalbero **/export** è destinato a contenere file e directory che sono condivisi, o *esportati*, su altre macchine in rete.

Il pacchetto di compatibilità BSD, se installato, risiede in **/usr/ucb** e contiene i comandi eseguibili; potete aggiungere questo directory nel vostro PATH per utilizzare i comandi BSD. Il file **/usr/ucblib** contiene materiale di supporto per il pacchetto di compatibilità BSD.

Gli strumenti per X Window System e il materiale di supporto possono risiedere in **/usr/X**, o in **/usr/bin/X11**, oppure in **/usr/lib/X11**.

I menu e le maschere di System Administration sono usualmente memorizzati sotto **/usr/admin** e **/usr/vmsys**; questi sottoalberi possono essere molto grandi se la macchina include tutti gli strumenti di gestione previsti in SVR4.

Infine, gli sviluppatori di applicazioni si costruiscono una parte della gerarchia di directory a uso esclusivo delle loro applicazioni; spesso vengono usati **/install, /opt** oppure **/usr/add-on**, altrimenti ogni applicazione può avere un suo proprio schema. In conseguenza, il vostro sistema può avere molti più directory di quelli finora menzionati. Ispezionate attraverso il vostro proprio sistema e studiate la sua struttura; incontrerete molti di questi file e directory nei prossimi capitoli.

Capitolo 5

Elaborazione di testi con vi ed emacs

Il sistema UNIX presenta tra i suoi componenti standard molti editor che differiscono parecchio tra loro e sono di solito ottimizzati per svolgere un particolare sottoinsieme di operazioni di elaborazione di testi. Nessuno di essi è un vero elaboratore di testi come se ne trovano in altri piccoli sistemi o personal in quanto in ambiente UNIX esiste un ben definito insieme di strumenti software che esegue l'elaborazione di testi o la formattazione di documenti. Oltre agli editor standard, sono disponibili sul mercato molti elaboratori di testi che sicuramente soddisfano le vostre esigenze, ma in questo libro tratteremo solamente degli strumenti di uso più diffuso.

Il **vi** (visual) è un editor originato in release BSD e tuttavia incluso in release System V da molti anni. L'altro principale editor, **emacs**, viene largamente usato e supportato in molti sistemi, tuttavia non viene incluso in release standard. Dovrebbe venire acquistato a parte, ma numerose versioni sono disponibili gratuitamente. Una delle migliori è la versione *gnu emacs* distribuita da Free Software Foundation. Altri editor, come il venerabile **ed** (editor) e **sed** (stream editor) vengono descritti nel prossimo capitolo.

Le tecniche di elaborazione di testi, come molte altre caratteristiche del sistema UNIX, introducono molti concetti innovativi che trovano applicazione pratica nelle operazioni di aggiornamento e filtraggio di stringhe di caratteri, essendo i file di testo semplici sequenze di caratteri. Tra questi concetti, sicuramente il più significativo risulta la definizione di *espressione regolare* che trascende dall'elaborazione di testi e influenza l'intero ambiente di programmazione. In questo capitolo vengono introdotte le tecni-

che fondamentali di uso dei comuni editor per creare, modificare ed esaminare file. Nel prossimo capitolo vengono illustrati concetti e strumenti più progrediti e più veloci per elaborare testi.

5.1 L'apprendimento degli editor

Gli strumenti di elaborazione di testi in ambiente UNIX presentano una interfaccia utente molto concisa e di rapido impiego che favorisce più l'esperto che il principiante. La tecnica migliore per imparare a manipolare i testi è esercitarsi intensamente su un piccolo file, perché occorre acquisire abilità e automaticità che non si apprendono con la sola lettura dei manuali. Osservate l'operato di un utente esperto e chiedete spiegazioni quando vedete eseguire operazioni che non conoscete. Se non potete disporre dell'aiuto di un esperto la maniera migliore di procedere è quella di sperimentare passo passo mentre leggete il manuale e assimilare progressivamente nuove caratteristiche quando siete in grado di padroneggiare quanto appreso precedentemente. Se vi sembra che un editor non sia in grado di soddisfare le vostre esigenze, il motivo generalmente risiede nella vostra incapacità di utilizzarne appieno le potenzialità.

Se siete nuovi del sistema UNIX è forse preferibile iniziare con **emacs** invece che non **vi**. L'esperienza dimostra che pur tenendo conto delle preferenze personali per l'uno o l'altro editor, **emacs** risulta di più facile apprendimento, di più agevole uso e di più ampia configurabilità rispetto a **vi**. Tuttavia, le reali differenze nell'uso sono piccole, cosicché se sul vostro sistema gli utenti sono esperti di **vi** oppure non è installato **emacs**, il **vi** risulta pienamente accettabile.

5.2 Il text editor a tutto schermo vi

L'editor **vi** è un text editor a tutto schermo che visualizza una finestra nell'intero file mantenuto in un buffer interno. Il cursore del video si trova sempre in una posizione sulla linea in uso, o linea corrente, e molte operazioni vengono eseguite in quella posizione. Quando il cursore viene spostato in una nuova posizione nel file, la pagina video viene rivisualizzata in modo che in essa compaia sempre la linea corrente. Comunque **vi** non è un word processor che formatta testo o un sistema grafico, ma, poiché è indipendente dal tipo di terminale, potete utilizzarlo sia da una console di sistema che da un terminale remoto.

L'editor **vi** deriva dall'editor **ex** e una parte della sua documentazione fa riferimento ad alcuni comandi di **ex**, tuttavia attualmente **ex** non viene utilizzato di frequente, per cui tratteremo principalmente **vi**.

PREDISPOSIZIONE DEL TIPO DI TERMINALE

Per lanciare **vi** da terminali diversi, dovete informare il programma del tipo di terminale col quale state lavorando memorizzando questa informazione nella variabile di ambiente **TERM** (terminale). L'editor **vi**, come molte altre applicazioni, legge questa variabile di ambiente al momento in cui viene lanciato e adatta la sua uscita al tipo di terminale impiegato. La maggior parte dei sistemi non definisce automaticamente **TERM** quando vi collegate a meno che non operiate da una console di sistema. Comunque, l'uso di **TERM** è abbastanza comune in molte parti del software, per cui assicuratevi che questa variabile sia definita oppure inizializzatela direttamente da shell, come qui di seguito.

```
$ TERM = ansi
$ export TERM
```

La dichiarazione **ansi** in minuscolo è corretta per terminali standard ANSI o similari; sono previsti altri valori per i terminali più diffusi. Se commettete un errore nel dichiarare la variabile **TERM**, le applicazioni che formattano lo schermo potranno visualizzare strane sequenze di caratteri in maniera del tutto imprevedibile. Il secondo comando dell'esempio precedente, **export TERM**, permette allo shell di login e a tutti i vostri subshell di utilizzare la variabile **TERM**. Potete inserire questi due comandi nel vostro file **.profile**, in modo che **TERM** sia sempre inizializzata quando vi collegate nel sistema. Il file **.profile** e la procedura utilizzata per inizializzare la variabile **TERM** sono argomento di discussione del Capitolo 8.

L'editor **vi** e altri programmi che formattano lo schermo leggono la variabile **TERM** e poi cercano in un database **terminfo** una descrizione simbolica del terminale specificato. Nei sistemi SVR4, questo database si trova nel directory **/usr/share/lib/terminfo** (noto anche come **/usr/lib/terminfo**), che contiene un subdirectory per ogni carattere iniziale (lettera o numero) delle sigle dei tipi di terminali che vengono utilizzati da programmi di gestione di video. Per esempio, il directory **/usr/share/lib/terminfo/a** contiene una lista di terminali il cui nome inizia per *a*, come **ansi**, mentre il directory **/usr/share/lib/terminfo/2** contiene una lista di terminali il cui nome inizia con 2, come per il terminale HP 2626. Potete verificare se la descrizione del vostro terminale è inclusa nel database, ispezionando i file nel directory *terminfo*. Le versioni di sistema operativo precedenti utilizzano uno schema diverso di database per i terminali e la descrizione di tutti i terminali si trova nel file **/etc/termcap**. Se il directory **/usr/share/lib/terminfo** è presente sulla macchina, sostituisce il database **/etc/termcap** per l'editor **vi** e per altre applicazioni che devono gestire il video.

I sistemi con microprocessore 80386 oppure 80486 basati sul bus AT utilizzano la dichiarazione **TERM=AT386** per utenti che lavorano su console. Sotto X Window System si deve dichiarare **TERM=xterm**, mentre

per un terminale remoto generalmente funziona la dichiarazione **TERM=ansi** (ANSI standard).

LANCIO DELL'EDITOR vi

Lo shell può lanciare l'editor **vi** con una linea di comando che contiene come argomenti una lista opzionale di nomi di file.

```
$ vi old.file
$ vi new.file
$ vi new.file old.file
$ vi
```

Questi sono tutti esempi di comandi validi. Se il file in argomento non esiste **vi** lo crea: questo è il metodo normale di creazione di nuovi file in un sistema UNIX.

Se indicate come argomento una lista di nomi di file, **vi** li tratta sequenzialmente uno per uno, caricando il primo nel buffer di testo al momento del lancio del programma. Sono disponibili comandi per trattare alternativamente un file o un altro, ma **vi** non può trattare simultaneamente file multipli. I comandi di **vi** accedono al file corrente.

Se al lancio di **vi** non avete specificato nessun nome di file come argomento, **vi** lavora regolarmente, ma alla fine del lavoro dovrete dichiarare un nome di file per poter scrivere il testo dal buffer nel file system.

Con tutti i comandi degli esempi precedenti **vi** inizierebbe con la prima linea del file come linea corrente.

L'editor **vi** accetta molti altri argomenti sulla linea di comando, oltre a nomi di file. Potete utilizzare per esempio l'opzione + seguita da un numero di linea per iniziare da quel numero come linea corrente. Questo comando

```
$ vi +45 old.file
```

definisce la linea 45 del file **old.file** come linea corrente iniziale, mentre il seguente comando

```
$ vi +$ old.file
```

inizia con la linea corrente sull'ultima linea del file. Se invece scrivete

```
$ vi +/stringa old.file
```

vi ricerca nel file **old.file** la prima occorrenza di *stringa* e, se esiste, assume la corrispondente linea come linea corrente. Dopo il carattere + potete specificare qualunque comando di **vi**: il comando verrà interpretato prima di iniziare l'esecuzione.

La Figura 5.1 mostra un tipico schermo del video durante una sessione di editor con un file corto. Normalmente **vi** riempie l'intero schermo con il contenuto del file di testo che ha in esame, lasciando libera la linea finale per eventuali messaggi; tuttavia, se il file è breve oppure se siete giunti alla fine del file, **vi** riempie la parte di schermo inutilizzata con linee vuote che iniziano con il carattere ~ (tilde). Quando lanciate **vi** senza specificare il file oppure specificando un file non esistente, sull'intero video compariranno righe col carattere tilde a sinistra, a indicare che il buffer non contiene alcun testo.

Le linee del testo nel buffer interno sono numerate da uno fino alla fine del file, anche se di norma i numeri delle linee non appaiono sullo schermo. Il cursore dello schermo appare su un carattere del testo visualizzato sullo schermo; la linea che contiene il cursore è la *linea corrente*, il carattere che contiene il cursore si trova nella *posizione corrente* nella linea corrente. Molte operazioni di edit agiscono sulla linea corrente o sulla posizione del cursore; **vi** mette a disposizione comandi per spostare il cursore e cambiare la linea corrente.

MODALITÀ DI vi

L'editor **vi** dispone di tre diverse modalità (o modi) di esecuzione, ovvero stati differenti in cui può trovarsi in momenti diversi. Potete passare da una modalità a un'altra per eseguire differenti operazioni di edit. In *modalità comando* potete introdurre comandi diretti all'editor, cambiare la linea cor-

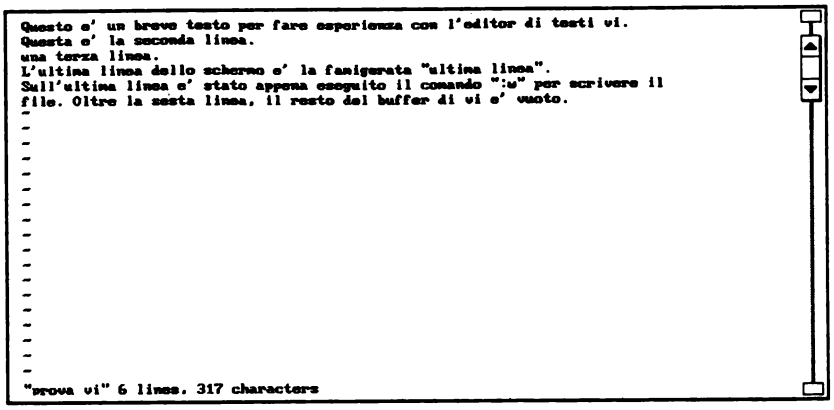


Figura 5.1 Videata di **vi**.

rente, abbandonare la sessione. In *modalità inserimento*, o *modalità testo*, potete introdurre del testo direttamente nel file. Normalmente iniziate **vi** in modo comando, ma potete passare in modalità inserimento per aggiungere del testo, quindi passare da una modalità all'altra nel corso del trattamento del testo. Questa può essere una fonte di errori perché è facile dimenticare la modalità in uso in un dato momento. Se introducete dei comandi e siete in modalità testo questi caratteri diventano parte del contenuto del file, mentre se introducete del testo e siete in modalità comando potreste causare impreviste modifiche nel vostro file. Il comando interno **set** consente di visualizzare la modalità corrente; questo comando viene descritto in seguito in questo capitolo. Per il momento potete attivare la visualizzazione del modo corrente usando il comando

```
:set showmode
```

Una terza modalità di esecuzione, denominata *modalità ultima linea*, consente di introdurre linee di comando nella linea finale dello schermo. Il carattere : (due punti) introdotto in modalità comando causa il passaggio alla modalità ultima linea; come si vede, il comando **set** del precedente esempio è anche un esempio di uso della modalità ultima linea. Potete entrare in ultima linea solo da modalità comando; se vi trovate in modalità testo il carattere : viene introdotto come testo nel buffer.

Quando passate in modalità ultima linea con il carattere : (e con altri caratteri di comando) il cursore salta all'ultima linea dello schermo dove **vi** si mette in attesa di una vostra linea di comando da eseguire. Nel caso del nostro precedente esempio introducete la linea di comando **set showmode** per attivare la visualizzazione della modalità corrente.

SCAMBIO DI MODALITÀ

La modalità ultima linea viene usata per introdurre una singola linea di comando nella linea finale del video; eseguito il comando **vi** esce dalla modalità ultima linea e ritorna in modo comando, pertanto la modalità ultima linea ha vita breve; al contrario, le altre due modalità restano attive finché non chiedete esplicitamente uno scambio di modalità.

Quando siete in modalità inserimento potete passare alla modalità comando premendo il tasto ESCAPE, generalmente marcato ESC sulle tastiere più diffuse. Potete passare dalla modalità comando alla modalità inserimento usando uno dei comandi di trattamento testo illustrati più avanti in questo capitolo.

Ciascuna modalità agisce in modi differenti e dispone di un proprio insieme di comandi. I comandi in modalità ultima linea vengono visualizzati sull'ultima linea mentre li introducete, non vengono invece visualizzati in alcun modo i comandi in modalità inserimento.

Consultate la Tabella 5.1 come sommario dei principali comandi **vi**, trattati nei paragrafi seguenti.

Tabella 5.1 I principali comandi dell'editor **vi**.

Per entrare in modalità inserimento	
i	Prima del cursore
a	Dopo il cursore
I	All'inizio della linea
A	Alla fine della linea
O	Apertura linea precedente
o	Apertura linea successiva
Delete	
dw	Cancella la parola
dd	Cancella la linea
D	Fino alla fine della linea
x	Carattere sul cursore
Altre funzioni	
u	Annulla ultima modifica
/	Ricerca in avanti
?	Ricerca a ritroso
n	Istanza successiva
.	Ripete l'ultima azione
Y	Accantona la linea
y	Accantona la linea
p	Inserisce sotto
P	Inserisce sopra
``	Marcatore
ZZ	Scrive ed esce
ESC	Cancella il comando
Modalità di ultima linea	
:w	Scrive il file
:q	Esce
:wq	Scrive ed esce
:n	File successivo
:r	Legge il file
:e	Elabora il file
:f	Nome del file
CTRL-G	Nome del file
:set	Modifica le opzioni
.!	Uscita allo shell
:*n*	Linea *n*
:addr	Salta alla linea addr
Spostamento del cursore	
l	Un posto a destra
SPAZIO	Un posto a destra
h	Un posto a sinistra
BACKSPACE	Un posto a sinistra
j	Una linea sotto
+	Una linea sotto
k	Una linea sopra
–	Una linea sopra
$	Fine di linea
^	Inizio di linea
w	Parola successiva
e	Fine di parola
b	Parola precedente
*n*G	Alla linea *n*
Sostituzione	
cw	Sostituzione di parola
cc	Sostituzione di linea
C	Fino a fine di linea
r	Carattere sul cursore
R	Fino a ESCAPE
Controllo del video	
CTRL-D	Scorrimento video in avanti
CTRL-U	Scorrimento video a ritroso
CTRL-F	Schermata successiva
CTRL-B	Schermata precedente
CTRL-L	Riformazione del video
In modalità inserimento	
BACKSPACE	Cancella carattere
CTRL-W	Cancella parola
ESC	Modalità comando

USCITA DALLA SESSIONE vi

Per uscire da **vi** e tornare allo shell potete usare il comando **:q** (quit)

```
:q
```

Questo comando viene eseguito solo se non avete modificato il file dall'ultima precedente registrazione.

Se usate il comando dopo che avete aggiornato il file, ma prima di averlo memorizzato su disco, **vi** non lo esegue, visualizzando il seguente messaggio:

```
:q
    No write since last change (:quit! overrides)
```

Potete forzare l'uscita da **vi** e abbandonare tutte le modifiche apportate al file nella corrente sessione accodando un **!** (punto esclamativo) al **:q**, per esempio:

```
:q!
```

Tenete presente che se forzate l'uscita da **vi** in questo modo tutte le modifiche apportate al file dopo la precedente registrazione sono perdute.

REGISTRAZIONE DI FILE

La modalità ultima linea mette a disposizione comandi per leggere e scrivere file tra i file system e il buffer di testo di **vi**. Il comando **:w** riscrive il file trattato su disco; userete questo comando abitualmente prima di terminare la sessione **vi**, tenendo presente che il file su disco viene sovrascritto. Il buffer interno viene conservato intatto, quindi potrete riscrivere le modifiche effettuate ogni volta che occorre. È consigliabile riscrivere il file frequentemente durante una sessione di edit per evitare errori di involontarie modifiche del testo, in quanto **vi** non provvede a registrazioni automatiche o copie di salvataggio.

Potete cambiare il nome del file per una operazione di scrittura, indicando un nome completo di percorso dopo **.w**, per esempio:

```
:w altro.file
```

Questo comando crea un nuovo file col nome **altro.file** nel directory corrente e registra il buffer nel file, quindi **vi** informa delle dimensioni del file scritto, per esempio:

```
:w altro.file
"altro.file" 4 lines, 96 characters
```

Questo messaggio compare nell'ultima linea in sostituzione del comando **:w**. Se il file **altro.file** esiste già, **vi** visualizza il seguente messaggio invece di eseguire l'operazione di scrittura:

```
:w altro.file
"altro file" File exists – use "w! altro.file" to overwrite
```

In questo modo evitate di riscrivere un file per errore.

Potete forzare **vi** a riscrivere il file in ogni caso accodando un **!** al **w**, come mostrato nello stesso messaggio d'errore. Se non disponete del permesso di scrittura per quel file, **vi** visualizza il seguente messaggio invece di eseguire la scrittura:

```
:w! altro.file
"altro.file" Permission denied
```

In questo caso dovrete cambiare i permessi di accesso al file, oppure scrivere il buffer con un diverso nome di file.

Tutti i messaggi d'errore appaiono sull'ultima linea dello schermo e si sovrappongono al comando **:**.

Potete scrivere solo una parte del file indicando una serie di linee con l'operazione **w**; dopo il **:** introducete il numero della prima linea che volete scrivere, seguito da virgola, quindi dal numero dell'ultima linea che volete scrivere e infine dal comando **w**, per esempio:

```
:5,30w altro.file
```

Solo le linee dalla cinque alla trenta vengono scritte nel file **altro.file**, il buffer interno resta invariato.

Potete combinare le operazioni di scrittura **w** e uscita **q**, per esempio:

```
:wq
```

Questo scrive il file poi ritorna allo shell. In modalità comando potete anche usare il comando **ZZ** (Z maiuscola) come sinonimo di **:wq** per scrivere il file e uscire ritornando allo shell.

LETTURA DI FILE

Il comando **r** è simile a **w**, con la differenza che *legge* un file nel buffer interno dopo la linea corrente. Anche in questa operazione **vi** fornisce le informazioni sul file, per esempio:

```
:r altro.file
"altro.file" 6 lines, 158 characters
```

Il comando può essere usato per caricare il file indicato in un buffer vuoto, oppure per inserire il contenuto del file nel testo del buffer corrente. Se prima della **r** indicate un numero di linea, come nell'esempio:

```
:6r altro.file
```

vi leggerà il file dopo la linea indicata invece che dopo la linea corrente. L'esecuzione del comando **r** sposta la linea corrente dopo l'ultima linea letta.

SCAMBIO DI FILE

Se siete entrati in **vi** con una lista di nomi di file, il comando **:n** (next) chiude il file corrente e lo scambia col successivo file nella lista di nomi. Se il file corrente non è stato scritto, dopo essere stato modificato, potete ancora forzare lo scambio di file col comando **:n!**.

Se volete conoscere il nome del file corrente potete usare il comando in modalità ultima linea **:f** (file), oppure l'operatore CTRL-G per visualizzare il nome e la dimensione del file corrente.

AGGIORNAMENTO DELLO SCHERMO

Se lo schermo risulta confuso o alterato per qualsiasi motivo, l'operatore CTRL-L in modalità comando provoca l'aggiornamento dello schermo con il contenuto corrente del buffer. Questo risulta utile quando messaggi di sistema su console disturbano la visualizzazione di una sessione di editing.

ESECUZIONE DI SHELL

Il carattere **!** assume diversi altri significati nella modalità ultima linea. Potete utilizzare

```
:! comando o pipeline
```

per sospendere temporaneamente **vi** ed eseguire un comando o pipeline in un subshell; al termine del comando o pipeline il controllo ritorna a **vi**. Per esempio, questo comando:

```
:!ls
```

esegue **ls** sul directory corrente. Per accedere allo shell dovendo eseguire più operazioni, potete utilizzare il seguente comando:

```
:sh
```

L'esecuzione di **vi** viene sospesa e viene visualizzato il prompt **PS1**. Potete introdurre comandi per lo shell; quando volete terminare le operazioni in shell, potete introdurre **exit** oppure premere CTRL-D e il controllo ritorna a **vi**.

5.3 Elaborazione di testi con l'editor vi

A questo punto conoscete le operazioni fondamentali di **vi** e potete cominciare a usare i numerosi comandi disponibili esercitandovi a eseguire reali elaborazioni di testi su vostri file di prova.

RECUPERO DI ERRORI

Uno dei più importanti comandi è l'operatore **u** (undo). L'editor **vi** memorizza il contenuto precedente all'ultima modifica fatta al file; potete pertanto annullare in ogni momento quest'ultima modifica e recuperare il contenuto precedente con **u**. Se avete eseguito una serie di modifiche potrete recuperare soltanto la più recente. Provate il comportamento del comando **u** e verificate cosa accade quando eseguite due comandi **u** successivi.

PASSAGGIO DALLA MODALITÀ COMANDO ALLA MODALITÀ INSERIMENTO

Molti comandi passano dalla modalità comando alla modalità inserimento. Potete spostare il cursore a una posizione particolare nel file, entrare in modalità inserimento e quindi introdurre quanto testo volete, inclusi anche *newline*. Dovete solo ricordare di battere il tasto ESC per ritornare in modalità comando quando avete finito di introdurre il testo aggiunto.

Il comando **o** (open) apre una nuova linea di testo dopo la linea corrente e posiziona il cursore all'inizio della nuova linea, mentre l'operatore **O** maiuscolo (lettera O) apre una linea sopra la linea corrente. Il carattere **a** (append) permette di attivare la modalità inserimento a partire dal carattere che si trova dopo il cursore e l'operatore **A** maiuscolo permette l'inserimento di testo alla fine della linea corrente. L'operatore **i** (input) rende possibile l'inserimento di testo a partire dalla posizione precedente a quella occupata dal cursore e **I** attiva la modalità inserimento a partire dall'inizio della linea corrente. Quando siete in modalità inserimento il testo alla destra del cursore si sposterà verso destra mentre introducete il testo aggiuntivo. Per ritornare in modalità comando, utilizzate il tasto ESC.

MODALITÀ INSERIMENTO

In **vi**, molte funzioni che risultano disponibili in modalità inserimento sono intuitive. Quando siete in modalità inserimento, tutto il testo che scrivete

entra nel file, compresi i caratteri ritorno a capo che vi danno appunto il modo di inserire nuove linee di testo. Se fate un errore di battitura, il tasto ← fa retrocedere il cursore di una posizione, cancellando a ritroso uno o più caratteri sulla linea corrente, fino all'inizio della linea; comunque **vi** non consente di retrocedere col tasto ← dall'inizio della linea. Per modificare una linea che precede quella corrente, dovete entrare in modalità comando, riposizionare il cursore sul testo con l'errore e poi effettuare la modifica. Potete premere il tasto CTRL-W per cancellare la parola corrente senza abbandonare la modalità inserimento.

Un errore comune è cercare di introdurre i comandi mentre siete in modalità inserimento senza prima premere il tasto ESC; anche utilizzatori esperti di **vi** commettono questo errore o quello inverso, cioè tentano di inserire testo in modalità comando. Il secondo errore è maggiormente distruttivo perché, mentre battete, potete effettuare molti cambiamenti involontari al file. Questi cambiamenti di solito non possono essere corretti con l'operatore **u**, dal momento che il comando **undo** può recuperare solo l'ultima modifica eseguita. Per evitare i problemi associati con questi errori, memorizzate frequentemente su disco il file con il comando **:w**, quando siete sicuri che sia corretto.

SPOSTAMENTI NEL BUFFER

Il modo comando viene principalmente utilizzato per spostarsi nel buffer del testo, e per passare al modo inserimento in vario modo per modificare il testo del file. Potete riposizionare la linea corrente, la qual cosa comporta spesso la rivisualizzazione dello schermo affinché la linea corrente risulti centrata sul video. Potete anche spostare il cursore all'interno della linea corrente. Modifiche o inserimenti di testo da voi effettuati nel buffer avranno luogo nella linea corrente, alla posizione corrente del cursore.

L'operatore **w** (word) sposta il cursore in avanti all'inizio della parola che segue quella corrente, mentre l'operatore **b** (back) sposta il cursore indietro all'inizio della parola che precede quella corrente. La barra di spazio sposta il cursore in avanti di un carattere, mentre il tasto ← lo sposta indietro di una posizione. RETURN trasferisce il cursore all'inizio della linea successiva e il carattere – (meno) lo trasferisce all'inizio della linea che precede quella corrente. Il carattere ^ sposta il cursore all'inizio della linea corrente, mentre $ lo sposta alla fine della linea corrente.

Tutti questi operatori di spostamento possono essere preceduti da un argomento numerico, o *contatore di ripetizione*. Per esempio, il comando **5w** trasferisce il cursore all'inizio della quinta parola che segue quella corrente, e **6** – lo trasferisce indietro all'inizio della sesta linea che precede quella corrente.

L'operatore **H** sposta il cursore sulla prima linea dello schermo, mentre **L** lo sposta sull'ultima linea dello schermo. Potete spostarvi a una determinata linea nel buffer con **G** (go) preceduto dal numero della linea voluta.

L'ultima linea del file è identificata col simbolo logico **$**, potete pertanto usare **$G** o semplicemente **G** per spostarvi alla fine del buffer.

Anche alcuni comandi in modalità ultima linea possono essere usati per spostarsi nel file. Per spostarsi sulla prima linea del file, potete scrivere **:1**, mentre per muoversi sull'ultima linea del file è ovviamente usato **:$**. La linea corrente nel file è identificata dal simbolo logico **.** (punto). Potete determinare il numero della linea corrente col comando in modalità ultima linea punto-uguale, per esempio:

```
:.=
```

In risposta **vi** visualizza alla fine dello schermo il numero della linea corrente. In modo simile

```
:$=
```

visualizza il numero dell'ultima linea del file.

Premendo CTRL-F vi spostate in avanti nel file di una intera schermata, mentre se battete CTRL-B vi posizionate sulla schermata che precede quella della linea corrente. Lo schermo viene rivisualizzato nel corso di questi movimenti.

Le tastiere di molti terminali includono tasti per il controllo del cursore e generalmente **vi** risponde anche a questi tasti in aggiunta agli operatori di movimento cursore descritti. Dovrete comunque verificare la coerenza di comportamento della vostra versione di **vi** col vostro terminale.

Se il vostro terminale non consente l'uso di tasti di controllo del cursore dovrete usare i tasti delle lettere **h** per spostarvi di un carattere a sinistra, **l** per spostarvi di un carattere a destra, **j** per spostarvi di una linea in basso alla stessa posizione del cursore, **k** per spostarvi di una linea in alto alla stessa posizione del cursore.

RICERCA DI TESTO

Una variante della modalità ultima linea consente di ricercare stringhe di testo nel file; la ricerca inizia dalla posizione corrente del cursore e continua fino alla fine del file, eventualmente ricomincia dall'inizio. Se la stringa non viene incontrata, la ricerca termina ritornando alla posizione corrente del cursore.

Usate / (barra) seguito da una stringa per eseguire una ricerca. Il comando:

```
/stringa
```

ricerca *stringa* nel file. Quando introducete / il cursore passa nell'ultima linea del video (la linea corrente non viene modificata) e viene visualizzata di

seguito la stringa che introducete. Quando premete RETURN la ricerca inizia; se viene trovata una stringa corrispondente, **vi** visualizza sullo schermo la parte del file con il cursore sulla stringa trovata. Potete ricercare a ritroso nel file col carattere **?**; la ricerca procede a ritroso a partire dalla posizione corrente del cursore, eventualmente ricomincia dalla file del file, e continua a ritroso fino a ritornare sulla linea corrente.

Una volta trovata una stringa con / o **?** potete ripetere la ricerca di un'altra presenza della stessa stringa battendo / o **?** e RETURN; potete invertire la direzione di ricerca senza limitazioni.

La stringa introdotta viene memorizzata da **vi** per le ricerche successive, ma se introducete / o **?** senza avere mai prima specificato una stringa, **vi** segnala il fatto:

```
/
No previous regular expression
```

La stringa di ricerca può avere una struttura molto più complessa di una semplice stringa di testo; vedremo questo aspetto trattando le espressioni regolari nel prossimo capitolo.

MODIFICA DI TESTO

Diversi operatori attivano la modalità testo e permettono di cambiare il testo esistente. Il comando **r** (replace), seguito da un solo carattere, sostituisce il carattere che si trova in corrispondenza della posizione corrente del cursore con il carattere specificato, lasciandovi in modalità comando, perché la sostituzione riguarda un solo carattere. L'operatore **R** permette invece di sostituire un numero qualsiasi di caratteri, a partire dalla posizione corrente del cursore, lasciandovi in modalità inserimento. Potete sostituire testo solo fino alla fine della linea corrente, mentre nuove linee introdotte con RETURN vengono aggiunte al file dopo la linea corrente. Quando ritornate in modalità comando premendo il tasto ESC dopo un'operazione **R**, i rimanenti caratteri a destra sulla linea corrente che non sono stati sovrascritti rimangono invariati. Il comando **C** (change) opera in maniera analoga a **R**, ma l'intera linea viene modificata anche se avete cambiato un solo carattere prima di ritornare in modalità comando.

Il comando **cw** cambia la parola corrente, mentre **6cw** modifica sei parole consecutive, l'operatore **cc** permette di cambiare il contenuto della linea corrente, **6cc** consente di cambiare sei linee consecutive.

L'operatore **J** (join) unisce due linee di testo eliminando il carattere ritorno a capo che separa la linea corrente dalla successiva. Se al contrario volete dividere una linea in due, dovete spostare il cursore nel punto in cui intendete effettuare la divisione, entrare in modalità inserimento e premere RETURN.

CANCELLAZIONE DI TESTO

Esistono anche operatori che permettono di cancellare parti di testo. L'operatore **x** cancella il carattere corrente, mentre **6x** cancella sei caratteri consecutivi a partire dalla posizione corrente. L'operatore **dd** cancella la linea corrente, mentre **6dd** cancella oltre alla linea corrente le successive cinque. Potete cancellare una sola parola con **dw** oppure cinque parole consecutive con **5dw**.

Il comando **D** cancella dalla posizione del cursore fino alla fine della linea corrente.

Potete usare anche la modalità ultima linea per cancellare una serie di linee indicando i numeri della prima e ultima linea. Per esempio, con il comando:

```
:3,5d
```

vengono cancellate le linee dalla numero tre alla numero cinque.

RIPETIZIONE DI OPERAZIONI

L'operatore . (punto) ripete, in corrispondenza della posizione corrente del cursore, l'ultima modifica apportata sia in aggiunta che in eliminazione e sostituzione di testo.

SPOSTAMENTO E COPIA DI TESTO

L'editor **vi** supporta alcuni buffer interni di transito che potete utilizzare per trasferimenti di blocchi di testo nel file corrente. Oltre al buffer principale di testo, esiste un buffer di transito di default; potete definire altri buffer di transito utilizzando un nome monocarattere. Quando eseguite una operazione con uno di questi buffer, lo referenziate per nome. Queste operazioni sono conosciute come *yank* e *put* (salva e metti) in terminologia **vi**; in altri ambienti sono spesso chiamate operazioni *cut* e *paste* (taglia e incolla) oppure *copy* e *paste* (copia e incolla). Tutte queste operazioni lavorano su blocchi di linee complete; per salvare (yank) una parte di linea dovete suddividere la linea in due e salvare la parte che vi interessa.

Potete trasferire una parte di testo nel buffer di transito con il comando **Y** (yank), che può essere preceduto da un numero di linee da salvare. Per esempio, il comando

```
7Y
```

trasferisce la linea corrente e le sei successive linee nel buffer di transito di default. Potete riposizionare il cursore e poi utilizzare il comando **p** (put)

per mettere nel file, dopo la linea corrente, le linee di testo copiate in precedenza. La lettera **P** maiuscola permette di inserire le linee nel file, prima della linea corrente. Queste sono operazioni copy e paste, perché yank non altera il testo, ma semplicemente lo copia nel buffer di transito di default.

Per eseguire un'operazione cut e paste potete cancellare un blocco di linee con un comando come **7dd**; questo cancella la linea corrente e sei successive linee dal buffer principale. Le linee cancellate vengono memorizzate nel buffer di transito di default, cosicché dopo una cancellazione come questa potete riposizionare il cursore e usare **p** o **P** per ricopiare (paste) il materiale nel file.

Quando eseguite un'operazione di *paste*, il materiale non viene cancellato dal buffer di transito, cosicché potete ricopiare lo stesso materiale diverse volte nel file. Tuttavia, non appena eseguite un'altra operazione di modifica, il materiale nel buffer di transito viene perduto e sostituito dalla più recente modifica. Il buffer transitorio di default viene utilizzato anche per il comando **u**; il contenuto precedente del testo oggetto di modifica viene ogni volta memorizzato nel buffer transitorio di default, **u** lo ricopia semplicemente nella sua locazione originale. Ogni modifica rimane nel buffer di transito di default fino a quando effettuate una qualsiasi altra operazione che modifica il testo in fase di edit.

Potete memorizzare più permanentemente del testo in un buffer definendo un vostro buffer di transito con un nome di un carattere, preceduto da ". Questo comando

```
7"aY
```

trasferisce sette linee nel buffer denominato **a**; è possibile definire contemporaneamente fino a 26 buffer, ognuno con un nome diverso. Il testo memorizzato in un vostro buffer di transito può essere trasferito nel file, dopo la linea corrente, con un comando come il seguente:

```
"ap
```

SPOSTAMENTO E COPIA DI TESTO USANDO NUMERI DI LINEA

La modalità ultima linea consente anche operazioni di spostamento e copia di blocchi di linee definiti mediante numeri di linea. Potete usare, per esempio:

```
:3,5m9
```

per spostare le linee dalla 3 alla 5 dopo la linea 9; le linee vengono cancellate dalla posizione originale. In maniera simile, potete usare

```
:3,5t9
```

per copiare le linee dalla 3 alla 5 dopo la linea 9, ma senza cancellare le linee originali dalla loro collocazione di partenza. Potete copiare o spostare una sola linea, usando un singolo numero, per esempio:

```
:5m9
```

I numeri di linea possono essere espressi anche con indirizzi mediante complete espressioni regolari, come tratteremo nel prossimo capitolo.

5.4 L'editor emacs

L'editor **emacs** è popolare quasi quanto **vi**, ma non fa parte del sistema standard UNIX. Tuttavia è disponibile per tutte le versioni di UNIX per qualsiasi macchina, e ne viene prontamente disponibile una nuova versione a ogni nuova release del sistema.

L'editor **emacs** differisce da **vi** come filosofia ma, come **vi**, non è un word processor nel senso moderno del termine. Come **vi**, **emacs** permette di utilizzare qualsiasi terminale, anche se alcune versioni di **emacs** non utilizzano il database di terminali **/usr/share/lib/terminfo**, ma possiedono un file di descrizione di terminali in **/usr/share/lib/emacs/terminals** oppure **/usr/lib/emacs/terminals**. Esistono altri directory in **emacs**, che possono differire leggermente tra le diverse versioni, come per esempio il directory **/usr/share/lib/emacs** che contiene l'elenco delle macro disponibili e altro materiale.

CONCETTI FONDAMENTALI DI emacs

In esecuzione, l'editor **emacs** differisce fondamentalmente da **vi**, in quanto non esiste una distinzione tra la modalità comando e la modalità inserimento, ma tutti i caratteri che inserite da tastiera finiscono nel testo in corrispondenza della posizione corrente del cursore. In altre parole, **emacs** si trova sempre in modalità inserimento e, inoltre, il termine *modalità* assume un significato diverso perché si riferisce non allo stato di comando dell'editor, ma allo stato corrente di attivazione delle opzioni. Per esempio, quando l'editor visualizza la numerazione di linea per il file in scrittura, si usa dire che è attiva la *modalità di numerazione di linee*.

I comandi per il movimento del cursore, per la lettura o scrittura di file, per l'apertura di nuove linee e per altre utility, sono rappresentati da speciali sequenze di tasti, il primo dei quali risulta ESC o CTRL. Per esempio, per uscire da **emacs** e passare il controllo allo shell, utilizzate la sequenza di tasti CTRL-X seguita da CTRL-C. Nella terminologia **emacs** questi tasti CTRL vengono indicati con la notazione **^x^c** (talvolta anche C-xC-c), dove il carattere ^ (oppure C-) che precede gli altri caratteri significa mantenere premuto il

tasto CTRL mentre battete i caratteri successivi. Se invece volete lanciare un comando, premete il tasto ESC seguito da un carattere, come nel caso di ESC-M (tasto ESC seguito dal tasto M) che visualizza le opzioni correnti di **emacs**. Nella terminologia **emacs**, queste sequenze di escape vengono indicate con la lettera **M** (meta) seguita dal relativo carattere, come nel caso di **M-d** che equivale al comando ESC-D, mentre qualsiasi sequenza di caratteri che non inizia con ESC o con CTRL indica una parte di testo da memorizzare nel file. In linea generale, l'editor **emacs** utilizza i comandi CTRL per richiamare funzioni varie su singole linee o singoli caratteri; i meta comandi per operare su parole o frasi; i comandi CTRL-meta invece identificano più complesse combinazioni di ambedue le operazioni.

La Tabella 5.2 riassume i comandi **emacs**.

Molti utenti preferiscono lo stile semplice di **emacs** a quello di **vi**, anche se l'insieme di comandi definiti in ambiente **emacs** è meno mnemonico rispetto ai comandi **vi** e l'apprendimento delle caratteristiche di **emacs** è più complicato rispetto a quello di **vi**.

LANCIO di emacs

Per lanciare l'editor **emacs** utilizzate il seguente comando, dove la specifica del file è facoltativa:

```
$ emacs old.file
```

Questo comando accetta solo un nome di file come argomento e, se è specificato, il file viene trascritto nel buffer principale di **emacs**. L'argomento opzionale +*n*, dove *n* è un numero, induce **emacs** a posizionare il cursore sulla linea *n*. Dopo l'operazione di caricamento, è possibile leggere altri file nei buffer addizionali di **emacs** e questo permette all'utente di aggiornare simultaneamente un massimo di 12 file. Il comando **^x^f** (f sta per file) rende possibile la lettura di un nuovo file in un nuovo buffer, a differenza del comando **^x^r** (r sta per read), che esegue la lettura di un file nel buffer corrente per sola lettura. Il comando **^xi** (insert) legge un file nel buffer alla posizione corrente del cursore, senza alterare l'eventuale contenuto corrente del buffer. Questi comandi, prima di eseguire la loro azione, richiedono all'utente il nome del file da leggere.

La Figura 5.2 mostra una tipica visualizzazione prodotta da **emacs**, dopo che il programma è stato caricato con il file **/etc/profile**; questa immagine può differire leggermente su altre versioni di **emacs**. La parte superiore della finestra grafica mostra la linea corrente e le linee adiacenti del file che state trattando. Potete modificare la dimensione della finestra e visualizzare o meno i numeri di linea con i comandi di **emacs**, usando la sequenza **^x^m** (change mode), trattata più avanti. La parte inferiore dello schermo presenta il nome **emacs**, seguito dal nome del file o buffer corrente, e dalle opzioni

Tabella 5.2 I principali comandi dell'editor **emacs**.

Movimenti del cursore

^f	Avanti di un carattere
^b	Indietro di un carattere
^n	Linea seguente
^p	Linea precedente
^a	Inizio linea
^e	Fine linea
M – f	Avanti di una parola
M – b	Indietro di una parola
M – a	Inizio di frase
M – e	Fine di frase
M – [	Inizio di paragrafo
M –]	Fine di paragrafo
M – <	Inizio di buffer
M – >	Fine di buffer
^v	Schermo successivo
M – v	Schermo precedente
M – *n*M – r	Muove avanti *n* linee
M – x	Salta a linea *n*

Comandi di file

^x^f	Legge file nel nuovo buffer
^x^s	Scrive buffer nel suo file
^x^w	Scrive buffer nel file
^xi	Aggiunge file nel buffer corrente

Comandi di regione

^SPAZIATRICE	Pone mark al cursore
^w	Cancella da mark a cursore
M – w	Copia nel kill stack
^x^x	Scambia cursore e mark

Comandi di ricerca

^s	Cerca stringa in avanti
^r	Cerca stringa indietro
M – s	Cerca avanti RE
M – r	Cerca indietro RE

Comandi di cancellazione

^d	Cancella un carattere avanti
BACKSPACE	Cancella un carattere indietro
^K	Cancella fino a fine linea
M – d	Cancella una parola avanti
M – k	Cancella una frase avanti
^y	Ripristina l'ultimo testo cancellato
M – y	Scambia testo yank con il buffer del testo cancellato

Comandi di buffer

^x2	Divide lo schermo tra 2 buffer
^x1	Unico buffer su schermo
^xb	Scambia buffer
^x^b	Elenca buffer
^xo	Scambia finestra
^xk	Annulla buffer
^y	Yank

Comandi vari

^x^c	Uscita
M – !	Richiama lo shell
^l	Ripristina lo schermo
^h	Help
^hw	Sequenza tasti del comando
^xu	Annulla ultima operazione (unstack)
^o	Inserisce linea vuota
^g	Annulla comando in esecuzione

Comandi di macro

^x(	Inizio macro di tastiera
^x)	Fine macro di tastiera

o modi in effetto (in questo caso **Fundamental**). Viene di seguito indicata la percentuale di file sullo schermo; in caso di piccoli file viene indicato **All** per significare che l'intero file appare sullo schermo. I trattini sulla sinistra significano che il file non è stato modificato da quando avete iniziato l'operazione di edit; compaiono anche due caratteri * non appena il contenuto del buffer viene modificato. L'ultima linea della finestra contiene informa-

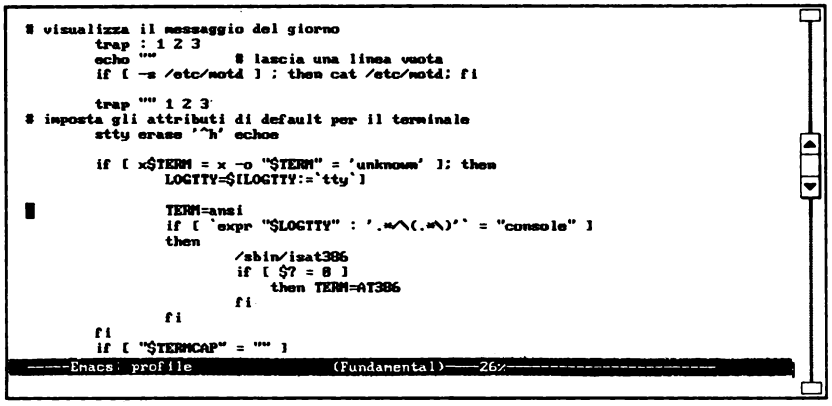


Figura 5.2 Una videata dell'editor **emacs**.

zioni di **emacs**, riservate per visualizzare speciali messaggi e comandi. Fra i vari messaggi, diverse versioni di **emacs** avvertono su questa linea se avete posta in attesa. Molte versioni di **emacs** generalmente supportano alcuni comandi particolari che permettono la lettura della posta in un buffer di testo o l'invio come posta del contenuto di un buffer.

SUDDIVISIONE DELLO SCHERMO

L'editor **emacs** può dividere lo schermo in due finestre in modo da permettervi di trattare simultaneamente il contenuto di due buffer, mentre è possibile operare contemporaneamente in 12 buffer diversi. Se richiedete di leggere un secondo file in un secondo buffer con il comando **^x^f**, potete allora dividere lo schermo in due finestre separate utilizzando il comando **^x2**; in questo caso, **emacs** vi chiede di specificare il nome o il numero del secondo buffer da visualizzare su video. Potete ritornare a un'unica finestra con **^x1** e in questo caso **emacs** richiede di specificare quale buffer mantenere sul video.

Tenete presente che quando visualizzate o nascondete un buffer, questo rimane nella memoria interna di **emacs**, per cui non viene chiuso il file né vengono annullate le modifiche. Prima di restituire il controllo allo shell, **emacs** aspetta che salviate ogni buffer o annulliate le modifiche.

L'editor segnala quale dei due buffer è attivo correntemente; la linea di informazione in fondo allo schermo visualizza il nome e il numero del buffer della finestra attiva.

Potete rendere attiva l'altra finestra con il comando **^xo** (other) oppure cambiare completamente i buffer visualizzati con **^x^b** (buffer).

SCRITTURA DEL FILE

Dopo che avete apportato le modifiche al buffer, potete scriverlo con il comando **^x^w** (write). **emacs** vi chiede conferma visualizzando alla base dello schermo:

```
Write file:
```

e restando in attesa di introduzione del percorso per il file da scrivere. Potete premere RETURN per riscrivere il buffer nel file corrente oppure introdurre il nome di un altro file.

Per riscrivere il buffer correntemente attivo nel file corrente, utilizzate il comando **^x^s** (save); potete anche annullare l'operazione di scrittura col comando **^g**.

USCITA DA emacs

Se volete uscire da **emacs** e ritornare allo shell, utilizzate il comando **^x^c** (close). Se tentate di uscire prima di avere scritto i file, **emacs** richiede conferma prima di ritornare allo shell e perdere le modifiche eseguite.

RICHIESTA DI TESTO DI AIUTO

Molte versioni di **emacs** dispongono di ampi testi di aiuto; il comando **^h** (help) richiama il sottosistema di aiuto. Dopo **^h** potete introdurre un **?**, oppure un altro comando **^h** per ottenere informazioni di aiuto all'uso del sottosistema di aiuto. In alternativa, potete usare

^hc *tasto*

per ottenere una breve descrizione di una qualunque sequenza di caratteri di comando; una descrizione più completa si può ottenere con

^hk *tasto*

L'inverso, cioè la sequenza di caratteri necessaria per eseguire una funzione, può essere ottenuto, se conoscete il nome esatto del comando, con

^hw *comando*

Il tempo impiegato per imparare a usare le funzioni di aiuto di **emacs** è ben speso.

SPOSTAMENTO DELLA POSIZIONE DEL CURSORE

In **emacs**, quando il cursore appare *sopra* un carattere, la *posizione corrente* trattata è quella *precedente* la posizione del cursore, pertanto il testo che introducete viene inserito *davanti* al cursore. Tenete ben presente questo comportamento mentre usate **emacs**.

In ambiente **emacs** è disponibile un vasto insieme di operatori che permette di modificare la posizione corrente del cursore. Il comando **^f** (forward) sposta il cursore in avanti di un carattere, mentre **^b** (backward) lo sposta indietro di una posizione. L'operatore **^n** (next) trasferisce il cursore sulla linea successiva, **^p** (previous) lo sposta sulla linea precedente, il comando **^a** muove il cursore all'inizio della linea corrente, mentre **^e** (end) lo muove alla fine della linea corrente. Il comando **M-<** sposta il cursore all'inizio del buffer, mentre **M->** lo sposta alla fine. L'opzione **M-f** muove il cursore in avanti di una parola, **M-b** lo muove indietro di una parola, il comando **^v** sposta il cursore in avanti di una pagina, mentre **M-v** sposta il cursore indietro di una pagina. La maggior parte di questi comandi accetta un argomento numerico per ripetere più volte l'azione e questi argomenti vengono specificati iniziando con il tasto ESC, poi il numero e infine l'opzione desiderata. Per esempio, per spostarsi in avanti di otto parole, premete ESC, poi 8, poi ESC e infine **f**.

CANCELLAZIONE DI TESTO

Poiché ogni carattere che battete finisce nel testo in corrispondenza della posizione corrente, non esiste difficoltà a inserire un testo con **emacs**, ma occorrono comandi specifici per cancellare un testo. Il tasto ← cancella il carattere che si trova a sinistra del cursore, mentre **^d** (delete) cancella il carattere alla posizione corrente del cursore. Il comando **^k** (kill) cancella la linea corrente a partire dalla posizione alla destra del cursore oppure concatena la linea corrente con la linea successiva se il cursore si trova alla fine della linea. Se l'argomento specificato è **0**, tutto il testo compreso tra l'inizio della linea e la posizione occupata dal cursore viene cancellato, mentre un argomento maggiore di 0 indica quante linee vengono cancellate a partire dalla posizione occupata dal cursore. Un argomento minore di 0, invece, indica il numero di linee che vengono cancellate prima del cursore.

MARK IN emacs

I comandi più complessi di trasferimento di testo e di cancellazione coinvolgono simboli denominati *mark*; **emacs** permette di specificare fino a 12

mark, uno per ogni buffer di testo. Potete collocare un mark alla posizione corrente del cursore col comando **M-BARRA SPAZIATRICE**; il mark non viene visualizzato sullo schermo in molte versioni di **emacs**. Quando spostate il cursore il mark rimane nella posizione di collocazione; potete allora trasferire o cancellare il testo compreso fra il mark e la posizione corrente. Quando scambiate **emacs** su un altro buffer, il mark nel buffer precedente rimane nella sua collocazione, e potete collocare un altro mark nel buffer corrente attivo.

Siccome i mark non sono visualizzati sullo schermo è facile dimenticarne la collocazione; potete scambiare la posizione del mark e del cursore col comando **^x^x**; per tornare alla posizione originale del cursore e ripristinare il mark, eseguite **^x^x** una seconda volta.

SPOSTAMENTO E COPIA DI TESTO IN emacs

Quando viene cancellata parte del contenuto di un buffer di testo, **emacs** lo salva in un *kill stack*; questo stack conserva il testo delle ultime otto operazioni di cancellazione effettuate. Per trasferire del testo, usualmente marcate l'inizio del blocco, spostate il cursore alla fine del blocco e poi cancellate il blocco con **^w**. Il testo in realtà non viene perso, per cui potete spostare il cursore alla locazione in cui volete riportare il testo e battete il comando **^y** per inserire l'ultimo blocco cancellato in corrispondenza della posizione corrente del cursore. Per cancellare una parte di testo, spostatelo semplicemente nel kill stack con **^w** e dimenticatelo.

Il comando **^y** è utile anche per riparare a eventuali errori nella cancellazione di testo, poiché il testo viene salvato sullo stack dopo l'operazione di cancellazione. In **emacs** esiste anche uno specifico comando a questo scopo.

Il comando **M-w** raccoglie il testo compreso tra il mark e la posizione corrente del cursore e lo trasferisce sullo stack, ma non modifica il buffer corrente; viene utilizzato insieme al comando **^y** per copiare blocchi di testo da un punto del file all'altro o da un buffer a un altro.

RICERCA E SOSTITUZIONE DI STRINGHE DI TESTO

L'editor **emacs** supporta operatori di ricerca particolarmente efficienti per localizzare nel buffer corrente stringhe di testo o espressioni regolari. Il comando **^s** (search) predispone l'azione di ricerca; **emacs** richiede l'introduzione della stringa di ricerca, utilizzando la sintassi delle espressioni regolari. La ricerca procede in avanti dalla posizione corrente del cursore fino alla fine del file. Il comando **^b** (backward) consente la ricerca a ritroso dalla posizione corrente del cursore fino all'inizio del file; anche in questo caso **emacs** richiede la stringa di ricerca. Quando viene trovata una stringa corrispondente, potete procedere alla ricerca della prossima presenza della stes-

sa stringa con un altro ^**s** (search) oppure con ^**r** (reverse); se non introducete una nuova stringa, viene riusata la stringa precedente. Raggiunta la fine del file potete far ripartire la ricerca dall'inizio del file con un altro ^**s**; potete interrompere una ricerca in corso con ^**g**.

Una procedura simile si applica per effettuare la sostituzione di stringhe corrispondenti alle espressioni regolari di ricerca. L'operazione inizia col comando:

M – x *stringa-sostituzione*

Dopo **M-x** dovete introdurre la *stringa-sostituzione*, quindi premere RETURN; **emacs** richiede di introdurre la stringa da cercare e quella che la sostituisce. La sintassi è ancora la stessa delle espressioni regolari; introducete la stringa da sostituire e RETURN, la stringa sostitutiva e ancora RETURN. Questa operazione deve essere effettuata con cura, perché comporta la sostituzione di *tutte* le occorrenze della prima stringa con la seconda.

Con il comando:

M – x *conferma-sostituzione*

seguito dalla stessa sequenza di operazioni, **emacs** attua un meccanismo di sostituzione interattiva; trovata la stringa richiesta richiede una vostra conferma per eseguire la sostituzione. Premete **y** se volete eseguire la sostituzione della corrente stringa trovata, altrimenti battete **n** per evitare quella sostituzione. In ambedue i casi **emacs** passa a ricercare la successiva occorrenza della stringa, se esiste. Potete anche premere **R** per indurre **emacs** a sostituire tutte le restanti occorrenze della stringa contenute nel file senza ulteriori richieste di conferma, oppure potete battere ^**g** per annullare l'operazione e ritornare a normali operazioni **emacs.**

ESECUZIONE DI SHELL

Come molti altri comandi, **emacs** permette di sospendere temporaneamente la propria esecuzione mentre state effettuando altre operazioni in ambiente shell. Introducendo **M-!**, **emacs** richiede una linea di comando da eseguire e resta in attesa fino a quando il comando viene completato; successivamente ripristina lo schermo e riprende la sua esecuzione. Potete richiedere l'esecuzione di uno shell interattivo con:

M – x shell

Quando uscite dallo shell, **emacs** riprende l'attività.

Il comando **M-|** (meta – pipe), seguito da una linea di comando, usa il testo fra il mark e il cursore come standard input del comando; il buffer resta

invariato. Potete anche introdurre l'output di un comando shell nel buffer corrente e sostituire la regione corrente, con **^uM-!**, seguito da una linea di comando.

5.5 Approfondimenti

In ambiente UNIX l'operazione di editing è molto complessa. In questo paragrafo tratteremo alcune caratteristiche particolari, prima di passare alla discussione delle espressioni regolari nel prossimo capitolo.

NOTE SULL'AMBIENTE X WINDOW SYSTEM

Le Figure 5.1 e 5.2 sono prese da un videoterminale operante sotto X Window System. Nella configurazione base di X, sullo schermo saranno presenti *scroll bar*, o barre di scorrimento, come appare alla destra delle Figure 5.1 e 5.2. Tuttavia gli editor **vi** ed **emacs** non sono implementati in maniera da sfruttare appieno i vantaggi delle scroll bar, che vengono gestite da X Window System anziché dagli editor. In conseguenza di ciò voi potete muovere il contenuto dello schermo manipolando le barre (come spiegato nel Capitolo 23), ma questi movimenti non avranno influenza sulla linea corrente usata dagli editor.

In pratica, quando cambiate la locazione di scorrimento nella barra e successivamente introducete un carattere per l'editor, X provvederà a rivisualizzare lo schermo alla linea corrente usata dall'editor. Le barre possono essere usate al meglio per copiare una regione di testo nella posizione corrente di editor. Per realizzare questo in **vi**, entrate in modo testo al punto dove volete copiare il materiale, muovete la barra per visualizzare il testo desiderato, quindi eseguite **copy** tramite X Tools. Se siete correttamente in modo testo nell'editor, X inserirà il materiale nella posizione corrente e lo schermo verrà aggiornato.

Per gli stessi suddetti motivi, non potete ridimensionare una finestra di X mentre un qualsiasi editor è in azione; se lo fate gli editor non si adatteranno alla nuova dimensione e le informazioni sullo schermo risulteranno illeggibili. La corretta procedura di ridimensionamento della finestra deve essere eseguita prima del lancio di un editor, in modo che l'editor possa correttamente adattarsi alla nuova dimensione della finestra.

LE OPZIONI DI vi

L'editor **vi** dispone di molte opzioni che alterano il suo abituale comportamento e che potete specificare in due modi. Il comando **:set**, già trattato in precedenza quando abbiamo parlato di **:set showmode**, è un esempio di co-

me utilizzare quasi 50 opzioni, oltre a **showmode**. Tutte le opzioni disponibili possono essere visualizzate dal comando

```
:set all
```

Quando volete modificare una opzione, specificate il nome dell'opzione dopo il comando **:set** per attivarla e fate precedere un **no** al suo nome per disattivarla. Questi comandi

```
:set showmode
:set noshowmode
```

rispettivamente attivano e disattivano l'opzione **showmode**.

Alcune delle opzioni accettano un argomento, come nel caso di

```
:set window = 10
```

che imposta la dimensione della finestra logica a dieci righe in altezza. Potete sperimentare sul vostro terminale le opzioni di **vi** per acquisire esperienza su come personalizzare l'editor secondo le preferenze personali o le esigenze del vostro terminale. Alcune delle opzioni più utilizzate sono l'opzione **terse**, per visualizzare messaggi di errore più concisi del solito, **autoindent**, per indentare una linea sulla stessa colonna di inizio di quella precedente, e **number**, per visualizzare la numerazione delle linee del file. L'opzione **tabstop** cambia il numero di posizioni di cui si sposta il cursore dopo aver premuto il tasto TAB, mentre l'opzione **ignorecase** annulla la distinzione tra caratteri minuscoli e caratteri maiuscoli durante le operazioni di ricerca di stringhe con le espressioni regolari. Sono disponibili molte altre opzioni.

Le opzioni attivate con la linea di comando **:set** valgono solo per la sessione di editing in corso; quando uscite dall'editor **vi** con **:q**, le opzioni vengono perse e dovete reinizializzarle alla successiva esecuzione di **vi**. Esistono tuttavia in ambiente **vi** due meccanismi che permettono di mantenere le opzioni permanentemente attive, per cui quando lanciate **vi** l'editor stesso si configura in base alle vostre istruzioni. Primo, la variabile di ambiente **EXINIT** può memorizzare il comando **set**, che viene eseguito quando lanciate **vi**. Per esempio, questo comando

```
$ EXINIT = 'set number tabstop = 4 ignorecase' ; export EXINIT
```

configura **vi** con le opzioni **number**, **ignorecase** e **tab** inizializzata a quattro spazi. Normalmente, assegnerete il valore voluto alla variabile di ambiente **EXINIT** nel vostro file **.profile**, che attiva le opzioni specificate ogni volta che vi collegate nel sistema. Secondo, potete inserire questi comandi in un file denominato **.exrc** (**ex r**un **c**ommands), presente nel directory HOME, che **vi** legge e interpreta quando entra in esecuzione.

Il comando **map** consente un altro modo di utilizzo della variabile **EXINIT** e del file **.exrc**. Potete usare questo comando per associare a ogni tasto una funzione **vi** o una serie di funzioni, in modo da configurare un ambiente **vi** in cui si possono eseguire azioni complesse con solo pochi tasti. Sfortunatamente, questa funzione utilizza un linguaggio di programmazione oscuro e difficile, per cui risulta utile soprattutto agli esperti di **vi**.

I FILTRI DI TESTO IN AMBIENTE vi

Potete aggiungere lo standard output di qualunque comando al vostro buffer corrente, scrivendo:

```
:r !comando
```

In questo modo viene letto e riportato lo standard output di *comando* nel vostro file in corrispondenza della linea corrente. Analogamente, questo comando

```
:w !comando
```

riporta il file nello standard input di *comando*.

Inoltre potete filtrare una parte del file corrente tramite un pipeline, scrivendo:

```
:3,56 !comando
```

Questo comando ridirige le linee dalla linea 3 alla linea 56 nello standard input di *comando*, cancella le linee dal buffer e introduce in sostituzione lo standard output di *comando*. Un semplice strumento di formattazione **fmt** consente di raggruppare assieme brevi linee di testo in linee di lunghezza ragionevole, per esempio:

```
:1,$ !fmt
```

Potete verificarne il comportamento con un file di prova.

MODI MAGGIORI IN emacs

L'editore **emacs** dispone di vari *modi maggiori* indipendenti che consentono l'efficiente trattamento di vari tipi di testo; esistono modi specifici per programmi C, per documenti sorgenti **troff**, programmi LISP, e molti altri. Usualmente, **emacs** deduce quale tipo di file sta trattando dal nome di file, e predispone appropriati modi; indentazioni, annidamenti, suddivisione linee ed altre funzioni, cambiano il loro comportamento in base al modo cor-

rente. L'elenco esatto dei modi disponibili dipende sia dalla versione di **emacs** in uso, che dalla personalizzazione dell'editor per la vostra installazione; consultate il vostro esperto locale di **emacs** per conoscere quali modi sono disponibili. Gli esperti trovano molto conveniente la predisposizione automatica dei modi, ma i principianti possono cadere in confusione.

Potete forzare **emacs** a usare un particolare modo per il trattamento di un file inserendo una stringa di comando di modo nella prima linea del file. Il comando di modo deve essere preceduto e seguito dalla stringa –*– (trattino, asterisco, trattino), come qui di seguito:

```
–*– nroff –*–
```

Questa linea predispone un modo appropriato all'edit di documenti sorgenti **nroff**. La stringa può comparire in qualunque posizione della prima linea del file; potete includerla quindi come un commento nel linguaggio sorgente del file:

```
\" –*– nroff –*–
```

MODI MINORI IN emacs

L'editor **emacs** dispone anche di *modi minori*, che sono usati per predisporre varie opzioni e preferenze. Il comando **M-x** seleziona un modo; potete introdurre un nome di modo dopo il comando, e quel modo verrà posizionato a *on* oppure *off*. Trattiamo qui solo pochi dei modi più importanti; tenete presente che i nomi dei modi variano ampiamente da sistema a sistema (consultate il vostro esperto **emacs** se vi occorrono maggiori informazioni).

Il modo *save* predispone **emacs** a salvare automaticamente il contenuto del buffer su disco dopo aver raggiunto un numero minimo di caratteri specificato dal modo *savetype*. Il file di salvataggio ha un nome diverso dal file originale, cosicché non dovete preoccuparvi che **emacs** sovrascriva automaticamente il vostro file dopo che avete commesso un errore; se il vostro file ha nome **file** il file di salvataggio avrà nome **#file#**.

Posizionate a *on* il modo *auto fill mode* se volete utilizzare le caratteristiche *autowrap* (a capo automatico): l'editor **emacs** aggiunge un newline, o ritorno a capo, quando avete introdotto in una linea un numero di caratteri maggiore di **fillcol**; con questo modo attivo **emacs** divide la linea intelligentemente alla fine delle parole. Il modo *overwrite* causa la sovrascrittura e la sostituzione dei caratteri esistenti nel file, dalla posizione del cursore, con gli ordinari caratteri che normalmente verrebbero inseriti nel file; questo modo viene usato per cambiare più rapidamente un file esistente. Il modo **set-number** attiva o disattiva la numerazione delle linee. I modi *height* e *width* permettono di specificare le dimensioni della finestra di video; risulta utile per X Window System. Esistono molte altri modi di esecuzione che differiscono nelle varie versioni di **emacs**.

FILE DI PERSONALIZZAZIONE emacs

Quando **emacs** parte, legge il file **.emacs** nel vostro HOME directory; questo file contiene un insieme di sequenze di caratteri tali e quali come potrebbero anche essere introdotti da tastiera. Il programma esegue questi comandi prima di accedere al video e di passare il controllo all'utilizzatore; potete usare il file **.emacs** per una personalizzazione dei modi e dell'ambiente di **emacs**.

MACRO IN emacs

L'editor **emacs** dispone di eccellenti strumenti di estensione e personalizzazione: *macro* composte da una serie di normali comandi **emacs**, che a loro volta possono contenere chiamate di altre macro predefinite. Molte versioni in commercio di **emacs** includono molte macro, che vanno dalla verifica ortografica automatica a strumenti che rimappano la tastiera per specifici terminali. Sono possibili quasi illimitate estensioni del sistema **emacs**; certi "maghi" di **emacs** lanciano intere sessioni dall'editor, usando strumenti talmente personali che il normale shell e i comandi risultano scarsamente visibili. Consultate la documentazione della vostra particolare versione di **emacs** se volete creare e usare le macro.

MACRO DI TASTIERA

Una forma di macro, la macro di tastiera, è molto facile da creare e usare; consente di predisporre una sequenza di tasti, cui assegnare un nome come una macro. Quando usate quel nome **emacs** esegue l'intera sequenza memorizzata come macro; questo risulta molto comodo quando in una sessione di edit dovete eseguire azioni ripetitive.

Per iniziare la memorizzazione di una sequenza di tasti come macro, usate il comando:

```
^x(
```

(CTRL-**x** parentesi aperta). Tutti i caratteri introdotti fino alla fine memorizzazione:

```
^x)
```

(CTRL-**x** parentesi chiusa) vengono memorizzati come parte della macro. Tenete conto che i caratteri battuti vengono eseguiti così come memorizzati nella macro durante l'introduzione.

Dopo completato una macro definizione, potete eseguirla con **^xe** (execute); questo comando può eseguire solo l'ultima macro definita. Se volete

avere più di una macro di tastiera definite simultaneamente, dovete dare un nome alle macro, per esempio:

^x *nome-macro-tastiera*

Le macro di tastiera vengono perdute uscendo da **emacs**, a meno che le loro definizioni siano incluse nel file **.emacs**.

Un'altra utile caratteristica di **emacs** sono le mappe di tasti, *keymap*; potete "ricollegare" la vostra tastiera per riadattare quasi tutti i tasti usati per i comandi. Molti utenti fanno estese modifiche con mappe di tasti nei loro file **.emacs** secondo loro preferenze. Consultate la documentazione relativa alla vostra versione di **emacs** per apprendere come predisporre le keymap.

Capitolo 6

Espressioni regolari ed elaborazione di testi

Nel sistema UNIX le elaborazioni di testi sono basate sul concetto di *espressione regolare*, una descrizione formale di stringhe di testo che consente potenti operazioni di comparazione di stringhe.

Le operazioni / e ? di ricerca in **vi** sono molto più potenti e flessibili di quanto è stato mostrato nel capitolo precedente. Usando le espressioni regolari in **vi** potete effettuare efficienti ricerche di molte diverse stringhe con lo stesso comando, e riconoscere ognuna di queste. Con le espressioni regolari potete effettuare anche complesse operazioni di ricerca e sostituzione di stringhe.

Le espressioni regolari sono ammesse non solamente in **vi**, ma anche in tutti gli altri editor standard, e anche in shell. Potete considerare i metacaratteri di shell come una forma semplificata delle espressioni regolari qui trattate. Sfortunatamente la sintassi delle espressioni regolari in shell non è identica alla forma usata in **vi**; anche utenti esperti possono cadere in confusione.

In questo capitolo introduciamo il concetto di espressioni regolari, discutiamo i comandi che le trattano direttamente, quindi torniamo a **vi** e altri editor per illustrare come le espressioni regolari possono accrescerne le caratteristiche.

6.1 Concetti base delle espressioni regolari

In **vi**, ogni volta che utilizzate la notazione */stringa* per cercare una stringa corrispondente, potete sostituire il termine *stringa* con una espressione regolare. In realtà la stringa di caratteri rappresenta la forma meno complessa di espressione regolare.

RAPPRESENTAZIONE DI UN CARATTERE

Una espressione regolare consiste di una sequenza di operatori che identificano i singoli caratteri da ricercare. Per ogni carattere all'interno di una stringa, è possibile utilizzare una espressione che lo rappresenti. Il carattere . (punto), per esempio, è uno degli operatori più utilizzati nelle espressioni regolari e sostituisce un singolo carattere.

L'espressione regolare **a.c** indica tutte le stringhe che iniziano con la lettera *a*, terminano con la lettera *c*, e contengono un carattere qualunque all'interno. Questa espressione regolare identifica quindi la stringa **adc** come pure **a#c** o **aSc**, ma non identifica la stringa **abdc** che include due caratteri tra *a* e *c*. In questo caso, è necessario utilizzare le espressioni regolari **a..c**, **a...** oppure **.....** tuttavia le ultime due corrispondono anche a molte altre stringhe che probabilmente non interessano. Usando le espressioni regolari è importante specificare esattamente l'insieme di stringhe che debbono essere identificate e non altre.

RAPPRESENTAZIONE DI UN INSIEME DI CARATTERI

Gli operatori [e] (parentesi quadra aperta e chiusa) rappresentano un insieme di caratteri, ognuno dei quali può essere considerato la stringa da cercare. L'espressione

```
[abc]
```

rappresenta uno qualunque tra i caratteri **a**, **b** o **c**, mentre la seguente espressione regolare

```
[aA]
```

equivale alla lettera *a* minuscola o maiuscola.

Questo tipo di espressione, cioè un insieme di caratteri racchiusi tra parentesi quadre, identifica un solo carattere nel file di ricerca e, per comporre una stringa di più caratteri, è sufficiente associare un numero di espressioni regolari uguale al numero di caratteri che costituiscono la stringa.

Per esempio, l'espressione

```
[aA][bB]
```

identifica una qualunque tra le stringhe **ab**, **aB**, **Ab** o **AB**, ma non **BA**. Poiché ogni sequenza di caratteri racchiusi tra parentesi quadre corrisponde a un solo carattere, il costrutto precedente rappresenta una stringa di due caratteri. In un comando di ricerca di stringa in **ed**, la seguente linea di comando

```
/[aA][bB]
```

ricerca la prima occorrenza di una delle quattro possibili stringhe corrispondenti.

Potete definire espressioni regolari di qualsiasi livello di complessità, ma non dimenticate che ogni sequenza di caratteri racchiusa tra parentesi quadre corrisponde a un singolo carattere.

RAPPRESENTAZIONE DI UNA SERIE DI CARATTERI

L'operatore parentesi quadra può associare l'espressione regolare a un carattere che appartiene a un insieme di caratteri ordinati alfabeticamente o numericamente. Per esempio, per referenziare tutte le occorrenze di numeri presenti in un file, potete scrivere:

```
[0123456789]
```

Una forma più contratta è la seguente:

```
[0-9]
```

Il carattere - (meno) identifica, se utilizzato all'interno della coppia di parentesi quadre, un insieme di caratteri. Per referenziare un carattere alfabetico qualsiasi, maiuscolo o minuscolo, potete usare la seguente espressione:

```
[A-Za-z]
```

PROTEZIONE DEGLI OPERATORI DI ESPRESSIONI REGOLARI

Per referenziare direttamente l'operatore speciale **[**, evitando che **vi** lo interpreti come l'inizio di una espressione regolare, dovete anteporgli il carattere \ (barra inversa). Per esempio,

```
/\[
```

In questo modo **vi** trova la prima presenza del carattere [nel buffer. Dovete proteggere in questo modo ogni operatore di espressione regolare per evitarne la speciale interpretazione.

SIMBOLI SPECIALI PER INIZIO E FINE LINEA

Potete fissare la vostra espressione regolare all'inizio o alla fine di una linea in modo da specificare il carattere che deve apparire all'inizio o alla fine della linea. Usate un carattere ^ per denotare inizio linea, e un carattere **$** per fine linea.

Per esempio:

```
/^Parola
```

identifica la stringa **Parola** solo se appare all'inizio di una linea. Così,

```
/Parola$
```

identifica **Parola** solo se direttamente seguita da newline. Per evitare i significati speciali di questi caratteri e identificare alla lettera i caratteri ^ e **$** dovete come in altri casi proteggerli col carattere \. Per esempio:

```
/\$25
```

identifica la stringa **$25**. Notate che poiché le espressioni regolari non estendono il loro effetto su linee multiple, espressioni come le seguenti:

```
/hello^goodbye
```

```
/hello$goodbye
```

vengono riconosciute da **vi** come prive di senso e i caratteri ^ e **$** sono trattati come caratteri normali in luogo di caratteri speciali di inizio o fine linea.

COSTRUZIONE DI ESPRESSIONI REGOLARI COMPLESSE

È possibile combinare espressioni regolari che referenziano un solo carattere per costruire stringhe più lunghe, come viene indicato qui di seguito:

```
/[0-9][0-9][0-9]
```

Questo comando trova la prima sequenza di tre cifre numeriche. Potete aggiungere l'operatore * (asterisco) per indicare zero o più occorrenze della

precedente espressione regolare. Per esempio, per trovare una sequenza di cifre di qualunque lunghezza, potete scrivere:

```
/[0-9][0-9]*
```

Poiché il carattere * indica zero o più occorrenze della stringa, il seguente comando

```
/[0-9]*
```

identifica l'inizio di ciascuna linea e non ottiene il risultato richiesto. Invece, un primo **[0-9]** richiede che almeno una cifra appaia nella stringa prima delle cifre opzionali.

Il successivo comando invece identifica un'intera linea del file:

```
/^.*$
```

Questo può essere letto nel seguente modo: a partire dall'inizio della linea (^), considera ogni carattere (.), ripetuto un numero qualunque di volte (*), fino a incontrare la fine della linea ($). Potete quindi definire espressioni regolari di qualsiasi complessità, utilizzando questa tecnica di composizione a blocchi.

LA MASSIMA STRINGA IDENTIFICATA

Una importante regola da tenere presente è quella che le espressioni regolari identificano la più lunga stringa possibile. Supponete di avere la stringa seguente

```
abc1234def
```

e di utilizzare il comando

```
/[0-9][0-9]*
```

L'espressione identifica tutte le cifre 1234, in quanto la notazione **[0-9]*** identifica tutte le cifre contenute nella stringa in oggetto.

6.2 Il comando grep

Il sistema UNIX dispone di un programma filtro che analizza un file alla ricerca delle stringhe specificate come espressioni regolari. Si tratta del comando **grep** (global regular expression and print) che legge lo standard in-

put o una lista di file indicati come argomento e restituisce allo standard output tutte le linee che contengono la stringa specificata come primo argomento. Per esempio, questa linea di comando

```
$ grep una_stringa file
```

ricerca *una_stringa* all'interno di *file*. Tenete presente che *una_stringa* può essere qualunque espressione regolare e che, nella forma più semplice, l'espressione regolare può essere una semplice sequenza di caratteri.

```
$ grep gente vecchio.file
salve gente
$
```

Ogni linea del file che contiene la stringa specificata nel comando **grep** viene riportata sullo standard output.

Potete usare qualsiasi espressione regolare per identificare una stringa per esempio:

```
$ grep "^[a-z]12" altro.file
```

La precedente linea permette di trovare ogni linea che inizia con una lettera minuscola e come secondo e terzo carattere presenta le cifre 1 e 2. Delimitate tra virgolette l'espressione regolare di ricerca per evitare che lo shell interpreti i caratteri di parentesi quadra prima di **grep**.

Il programma **grep** può selezionare anche ogni linea del file che *non* contiene la stringa specificata come argomento, purché nella linea di comando compaia l'opzione **-v**. Il seguente esempio:

```
$ grep -v gente vecchio.file
altra linea
una terza linea
$
```

visualizza infatti ogni linea di *vecchio.file* che non contiene la stringa *gente*. Il comando **grep** dispone di molte altre opzioni: per esempio, l'opzione **-c** (count) produce solo il numero di linee che contengono la stringa indicata, come si potrebbe ottenere con un pipeline tra l'uscita di **grep** e il comando **wc -l**; l'opzione **-n** (number) aggiunge all'uscita prodotta il numero che la linea occupa nel file sorgente, mentre l'opzione **-i** (ignore) ignora la distinzione tra lettere maiuscole e minuscole in fase di ricerca delle stringhe.

Una delle più importanti utilizzazioni di **grep** è data dalla ricerca di una particolare stringa all'interno di parecchi file. Se avete molti file in un directory e non ricordate quale file contiene un certo materiale, **grep** o una delle sue varianti risolve il vostro problema. Se includete più di un nome

di file in una lista di argomenti per **grep**, in testa a ogni linea di output viene indicato il nome del file a cui la linea appartiene:

```
$ grep una *
file.prova1:una altra linea
file.prova2:una terza linea
$
```

In questo modo potete immediatamente conoscere il file che vi interessa.

6.3 I comandi fgrep ed egrep

Esistono altre due versioni del comando **grep** che ottimizzano le operazioni di ricerca in alcuni casi particolari. Il comando **fgrep** (fixed grep) accetta, invece di una espressione regolare, solo una stringa fissa come elemento di ricerca; il comando **egrep** (extended grep) definisce un insieme più ricco di operatori delle espressioni regolari rispetto al comando **grep** e rappresenta una versione ottimizzata per ricerche più complesse, ma che risulta più lenta di **grep**. Nella maggior parte dei casi, la versione originale **grep** costituisce un buon compromesso tra diverse esigenze di ricerca.

Il comando **egrep** accetta operatori addizionali di espressioni regolari che differiscono leggermente dagli operatori di **grep**. Oltre all'operatore *, utilizzato per indicare zero o più occorrenze di un costrutto monocarattere, **egrep** prevede infatti anche l'operatore + (più) per specificare *una o più* occorrenze di un carattere. Per esempio, se utilizzate il comando **grep**, potete scrivere

```
[0-9][0-9]*
```

per indicare una sequenza di cifre di qualsiasi lunghezza. Lo stesso risultato si ottiene anche con **egrep**, ma in questo caso potete anche scrivere

```
[0-9]+
```

Analogamente, l'operatore ? (punto interrogativo) indica *zero o una occorrenza* del costrutto.

In **egrep**, l'operatore | (pipe) indica una tra due espressioni regolari. Per esempio, questo comando

```
^[0-9]?a|b
```

referenzia una linea che inizia con una cifra opzionale e presenta come secondo carattere la lettera *a* oppure *b*. Come altro esempio,

```
^a | b
```

referenzia una linea che inizia con *a* oppure *b*. Il comando **egrep** utilizza anche l'operatore parentesi per raggruppare espressioni regolari multicarattere. Per esempio

```
$ egrep "([0-9]+ab)*1234" file
```

ricerca ogni stringa che contiene ricorsivamente un costrutto che inizia con una sequenza di una o più cifre, seguita da *ab*, e continua ancora con una sequenza di una o più cifre, seguita da *ab*, ecc., fino al riconoscimento della stringa **1234**.

Nella linea di comando è disponibile inoltre l'opzione **-f nome.file** (f sta per file) che in **egrep** permette di specificare l'espressione regolare tramite il file specificato nell'opzione, semplificando così la stessa linea di comando, mentre in **fgrep** il file contiene l'elenco delle stringhe da cercare.

Il comando **grep**, o i suoi consimili, viene largamente usato in quanto costituisce un veloce e potente strumento per la ricerca di stringhe di caratteri all'interno di un file di testo. Quando lo usate, cercate di specificare la più precisa espressione regolare che soddisfa alle vostre esigenze, perché il comando **grep** può originare un output enorme se i termini della ricerca sono specificati troppo genericamente.

6.4 Ricerche con espressioni regolari in vi

Gli operatori di ricerca in modo ultima linea / e ? consentono di specificare con una qualsiasi espressione regolare la stringa da ricercare; una particolare stringa di caratteri non è altro che una forma semplificata di espressione regolare. Quando usate un editor, provate a usare le espressioni regolari; quanto più imparate come funzionano tanto più rapide ed efficienti risulteranno le vostre operazioni.

6.5 Sostituzioni in vi

Potete effettuare sofisticate operazioni di ricerca e sostituzione su stringhe, usando espressioni regolari per definire le corrispondenze che volete identificare.

L'operazione in modo ultima linea **:s** (substitute) consente di specificare un'espressione regolare per la ricerca, e una nuova stringa che sostituisce la stringa identificata. A seguito del **:s**, introducete il delimitatore /, l'espressione regolare da cancellare, un altro /, infine la nuova stringa sostitutiva, come mostrato qui sotto:

```
:s/powerful/flexible
```

Questo sostituisce la nuova stringa, **flexible**, alla prima occorrenza della stringa (o espressione regolare), **powerful**, sulla linea corrente. La prima parte del comando di sostituzione può essere specificata come una qualunque espressione regolare, la seconda parte deve invece essere una precisa stringa da inserire in luogo dell'espressione regolare.

Il precedente comando sostituisce solamente la prima presenza dell'espressione regolare sulla linea corrente, anche se la posizione del cursore si trova dopo la stringa identificata; i comandi di sostituzione agiscono sulla intera linea corrente, indipendentemente dalla posizione corrente del cursore nella linea.

Per sostituire *tutte* le occorrenze dell'espressione regolare sulla linea corrente, dovete aggiungere un / finale al comando, seguito da **g** (global), come nell'esempio seguente:

```
:s/old.re/newstring/g
```

Questo comando ricerca e sostituisce tutte le occorrenze sulla linea corrente.

Potete cancellare una stringa con questo comando, semplicemente specificando come sostituzione una stringa vuota, come mostrato qui:

```
:s/the//
```

Il / finale non è indispensabile, ma potete usarlo per chiarezza.

Potete usare gli operatori speciali ^ e **$** nelle espressioni regolari per designare l'inizio o la fine linea. Per aggiungere una stringa alla fine di una linea potete usare il comando:

```
:s/$/nuovo testo alla fine
```

Per aggiungere una stringa all'inizio di una linea usate il comando:

```
:s/^/nuovo testo all'inizio
```

Un / finale non è obbligatorio a meno che non dobbiate aggiungere un **g**.

SOSTITUZIONI USANDO LA STRINGA IDENTIFICATA

A volte può essere necessario indicare una stringa nella sezione ricerca del comando **:s**, senza che debba essere sostituita; per esempio, se occorre accodare una parola a una stringa di cifre, non potete usare:

```
:s/[0-9][0-9]*/new_string
```

perché la stringa **new_string** cancella e sostituisce la sequenza di cifre, mentre volevate accodarla. L'operatore speciale **&**, usato nella sezione sostituzione, consente di fare riferimento alla stringa identificata con la sezione ricerca. Per esempio, data la stringa

```
abc1234def
```

e il comando

```
:s/[0-9][0-9]*/&new_string
```

Il risultato sarà:

```
abc1234new_stringdef
```

L'operatore **&** ha sostituito la stringa identificata nella linea in output.

Come al solito, potete includere un **&** effettivo nel vostro output, proteggendolo con \.

SOSTITUZIONI ENTRO UN GRUPPO DI LINEE

Normalmente il comando **:s** cambia solo la prima occorrenza di una stringa nella linea corrente; potete specificare un ambito di linee entro il quale volete che **vi** operi la stessa azione, con i numeri iniziale e finale delle linee dopo :. Per esempio:

```
:3,7s/old/new
```

Questo sostituisce la prima presenza della stringa **old** con la nuova stringa **new** nelle linee dalla 3 alla 7; per sostituire tutte le presenze dell'espressione regolare nelle stesse linee, dovete aggiungere il **g** finale, per esempio:

```
:3,7s/old/new/g
```

Per sostituire tutte le presenze di una stringa nel file, usate

```
:1,$s/old/new/g
```

Qui, il $ viene interpretato come ultima linea del file. Notate i due diversi significati del **$** in **vi**: uno, all'interno di una espressione regolare indica la fine della linea; l'altro, come indirizzo di linea indica l'ultima linea del file. Anche il . presenta due significati: indica qualsiasi carattere quando viene usato in un'espressione regolare, mentre indica la linea corrente se usato in un indirizzo di linea.

INDIRIZZI DI LINEA

Potete anche operare su una sola linea, in luogo di un ambito di linee, specificando un solo numero di linea, per esempio:

```
:34s/old/new
```

Questo sostituisce la stringa specificata solo sulla linea 34 del file. Potete usare anche numeri logici di linea, per esempio:

```
:.s/old/new
```

che sostituisce nella linea corrente e

```
:$s/old/new
```

che sostituisce nell'ultima linea del file.

Sono ammesse anche semplici espressioni aritmetiche; potete specificare un ambito di linee prima e dopo la linea corrente fornendo un *indirizzo relativo*, per esempio:

```
:.-4,.+6s/old/new
```

Questo opera la sostituzione in un ambito di 11 linee, partendo da 4 linee prima della linea corrente e terminando 6 linee dopo quella corrente. Potete anche usare il **$** in questi calcoli, per esempio:

```
:.-4,$-2s/old/new
```

In caso di errore in queste espressioni aritmetiche, **vi** segnala se il risultato non è un indirizzo valido, per esempio:

```
:.,$+2s/old/new
Not that many lines in buffer
```

Usate con cautela queste espressioni aritmetiche, perché un risultato *legale* potrebbe comunque non essere *logicamente* corretto ai vostri fini, e questi operatori di sostituzione non chiedono conferma prima di agire.

INDIRIZZI PER CONTESTO CON ESPRESSIONI REGOLARI

Il **$** come indirizzo è un esempio di *indirizzo per contesto*. In aggiunta all'indirizzamento per numeri di linea (non troppo agevole, visto che **vi** per default non visualizza i numeri di linea), potete specificare le linee con un'espressione regolare; **vi** ricerca la prima linea che corrisponde all'espressio-

ne regolare, e la tratta come la linea indirizzata. Per esempio, potete sostituire la prima presenza di una stringa dopo la linea corrente, con:

```
:/old/s//new
```

In questo caso, **vi** tratta la prima espressione regolare come un operatore di ricerca, posiziona la linea corrente alla linea dove trova la prima corrispondenza, quindi esegue la sostituzione. Potete eseguire pure una ricerca a ritroso, per esempio:

```
:?old?s//new
```

Notate che quando **vi** ha in esistenza un'espressione regolare corrente, come quella definita in questo caso con **/old/**, la riusa fino a che ne viene definita una nuova. Per questo motivo il comando negli esempi precedenti non necessita di una espressione regolare nella parte di ricerca; viene assunta per default l'espressione corrente **old**. Naturalmente, se la stringa da sostituire non coincide con l'espressione regolare corrente, dovete specificarla. Per esempio, per trovare la prima linea che contiene **old** e in quella linea sostituire la stringa **hello**, dovete usare

```
:/old/s/hello/goodbye
```

Dopo questa operazione l'espressione regolare corrente è **hello**, in luogo di **old** come negli altri esempi.

Potete definire anche un ambito di indirizzi usando il formato con espressioni regolari, per esempio:

```
:/hello/,/goodbye/s/old/new/
```

In questo caso **vi** cerca dalla linea corrente fino a trovare una corrispondenza con la stringa **hello**, quindi esegue le sostituzioni fino a che trova una linea che contiene **goodbye**.

Questa forma di indirizzamento è più difficile da controllare dei numeri di linea, tuttavia riflette il concetto che in sistemi UNIX tutti i trattamenti di edit possono essere fatti senza numerazione delle linee; per questo motivo **vi** viene spesso definito *editor di contesto*.

6.6 L'editor sed

Il comando **sed** (stream editor) presenta alcune caratteristiche di **vi** in modalità ultima linea, e alcune di **grep**; pur essendo un programma filtro come **grep**, permette all'utente di apportare modifiche ai file. Il comando **sed** non è interattivo come **vi**, al contrario legge il file di input linea per linea e scri-

ve le linee una per una sullo standard output. Su ogni linea letta, **sed** applica un'operazione di sostituzione della forma di quella utilizzata da **vi** in modalità ultima linea; se l'esito della ricerca è positivo, la sostituzione ha luogo e la linea viene scritta modificata, altrimenti la linea viene trascritta invariata.

Un grande vantaggio di **sed** rispetto a **vi** o ad altri editor è che le linee vengono lette, modificate e scritte una per una; non esistendo un buffer di memoria interna per l'intero file, **sed** può modificare file di qualsiasi lunghezza, anche quelli che risultano troppo grandi per **vi** o per altri editor. Il comando **sed** risulta molto utile per trattare file di lunghezza maggiore di un megabyte. La maggior parte degli editor di testo non può trattare file di queste dimensioni, **vi** è limitato generalmente a 256 000 byte.

È possibile lanciare il comando **sed** con le stesse modalità impiegate per eseguire il comando **grep**, anche se non potete utilizzare l'operatore di sostituzione globale. Questo comando

```
$ sed "s/salve/ciao/" in.file
```

sostituisce la prima istanza di **salve** presente su ogni linea del file **in.file** con la stringa **ciao** e riscrive la linea così aggiornata sullo standard output.

```
$ echo "1234salve5678" ¦ sed "s/salve/ciao/"
1234ciao5678
$
```

Delimitate tra virgolette il comando di sostituzione per proteggerlo dalle interpretazioni di shell. Come al solito, la stringa di ricerca può essere una espressione regolare.

Il comando **sed** dispone di molte altre funzioni. Per esempio, il seguente comando può cancellare tutte le linee che contengono la stringa **salve**:

```
$ sed "/salve/d" in.file
```

In questo esempio, il comando cerca nel file **in.file** la stringa **salve** e, se la trova, cancella la linea che la contiene. Lo stesso risultato si ottiene con

```
$ grep -v salve in.file
```

Per cancellare dalla linea solo la stringa **salve**, senza eliminare completamente la linea, potete utilizzare questa forma alternativa:

```
$ sed "s/salve//" in.file
```

Come **vi**, il programma **sed** accetta anche un indirizzo di linea o un insieme

di indirizzi, se intendete limitare la ricerca solo a una parte di file. Questo comando

```
sed "3,7s/salve//" in.file
```

cancella la prima istanza di **salve** presente tra le linee 3 e 7 del file e lascia invariato il resto della linea. Potete anche utilizzare un indirizzo di contesto invece di un numero di linea.

```
sed "/salve/,/ciao/s/cattivo/buono/g" in.file
```

Questo comando ricerca la prima istanza della stringa **salve**, modifica tutte le istanze della stringa **cattivo** con **buono**, fino a quando incontra la stringa **ciao** o fino a quando termina il file. In questo esempio, se esiste un'altra istanza di **salve** dopo l'incontro di **ciao**, la sostituzione ricomincia fino alla successiva istanza di **ciao**.

COMPLESSI PROGRAMMI DI sed

In realtà, **sed** risulta ancora più efficiente di quanto abbiamo descritto finora. Se inserite il comando in un file invece che su una linea di comando, potete specificare l'opzione **–f** sulla linea di comando **sed**.

```
$ sed –f comando.file in.file
```

In questo comando gli operatori di espressioni regolari sono memorizzati nel file **comando.file**, mentre, se non utilizzate questa notazione, **sed** opera nelle modalità consuete. Con un unico comando, come indicato nei precedenti esempi, l'utilizzo del file di comandi non comporta un notevole miglioramento di efficienza, anche se le espressioni regolari complesse possono qualche volta essere più facilmente corrette se sono memorizzate permanentemente in un file. Comunque, il file di comandi ha una funzione più importante: potete definire per **sed** un inserimento multilinea, per cui possono essere eseguite una serie di operazioni su ogni linea di inserimento prima che **sed** lo riporti in uscita. Per esempio, potete creare un file denominato **comando.file** contenente la seguente lista di comandi:

```
s/salve/ciao/
s/ciaoatutti/notte/
```

Ora, se eseguite

```
$ echo "1234salve5678" ¦ sed –f comando.file
```

l'uscita diventa

```
1234notteatutti5678
$
```

Le operazioni memorizzate nel file vengono eseguite sequenzialmente su ogni linea di inserimento fino a quando la linea viene cancellata o si raggiunge la fine del file. Quando l'insieme dei comandi è completato, la linea viene scritta sullo standard output, viene letta la successiva linea di inserimento e il processo continua.

Esistono molte altre opzioni per il comando **sed**, il cui impiego, insieme a quello delle espressioni regolari in genere, si migliora con l'esperienza e con la lettura accurata dello *UNIX User's Manual.*

6.7 L'editor ed

Il programma **ed** (editor di testo) ha fatto parte del sistema UNIX sin dalle prime versioni, come originario editor di uso generale. Poiché **ed** è un editor di linea e non un editor a tutto schermo come la maggior parte degli editor moderni, non lo esamineremo approfonditamente. Il programma **ed** introduce alcuni concetti chiave che riguardano la scrittura e la manipolazione di testi e rappresenta lo strumento base per la conoscenza di altri editor, anche se è riduttivo dipendere da **ed**.

Potete lanciare **ed** da shell specificando come argomento un nome di file (opzionale):

```
$ ed old.file
260
```

Se il file esiste già, **ed** lo copia nel suo *buffer*, per cui gli aggiornamenti che apportate non modificano immediatamente la versione originale, e restituisce la dimensione del file in caratteri (**260** in questo esempio) a indicare l'avvenuta lettura del file. Se il file non esiste, **ed** lo crea. Il nome di file fornito come argomento viene denominato *file corrente*. Quando riportate su disco il file che è memorizzato sul buffer di **ed**, il nome che assume coincide con quello da voi specificato come argomento. Se non specificate alcun argomento, **ed** lavora correttamente, ma non essendo dichiarato alcun file corrente, siete costretti a indicare esplicitamente un nome da assegnare al file su disco.

Se specificate il nome di un nuovo file quando lanciate l'editor, **ed** non visualizza il numero di caratteri contenuti nel file, ma segnala che il buffer è vuoto con la notazione seguente:

```
$ ed new.file
?new.file
```

A questo punto vi trovate all'interno del programma **ed** e potete disporre di un nutrito insieme di comandi di scrittura.

Diversamente da **vi** o **emacs**, **ed** non visualizza alcuna parte del testo nel buffer se non ne fate richiesta, può quindi risultare difficile tenere sotto controllo il contenuto corrente del buffer; questo tuttavia permette l'uso di **ed** anche sui più semplici terminali scriventi.

MODALITÀ DI ESECUZIONE

Il programma **ed** dispone di due differenti modalità di esecuzione: "inserimento" e "comando". La modalità inserimento funziona come in **vi**, tutti i caratteri che introducete entrano nel contenuto del buffer. La modalità comando funziona come la modalità ultima linea di **vi**; viene usata per ricerca di espressioni regolari, per leggere e scrivere file, e altro. Notate che i comandi in **ed** non iniziano con **:** (due punti) e che non esiste in **ed** un equivalente del modo comando di **vi**.

Per default, **ed** non specifica la modalità corrente; al suo inizio si trova in modalità comando.

PROMPT E HELP

Se specificate il comando **P** (prompt), inducete **ed** a visualizzare un prompt quando siete in modalità comando.

```
$ ed new.file
?new.file
p
*
```

In questo modo **ed** visualizza il prompt * ogni volta che vi trovate in modalità comando e, per disattivarlo, battete ancora il tasto P.

Se commettete un errore quando specificate un comando, **ed** automaticamente visualizza solo il carattere **?**, come indicato qui di seguito.

```
$ ed new.file
?new.file
XXX
?
```

XXX non rappresenta un comando valido e, per avere maggiori informazioni sull'errore che avete commesso, lanciate il comando **H**.

```
$ ed new.file
?new.file
```

```
XXX
?
H
XXX
? illegal suffix
```

Questa spiegazione non è molto dettagliata, ma è sempre meglio di niente. Potete disattivare la funzione di aiuto premendo di nuovo il tasto H.

LETTURA DI UN FILE

Il comando **r** (read) permette di leggere nel buffer del file corrente un altro file. Per esempio:

```
r old.file
```

inserisce il file **old.file** dopo la linea corrente. Potete specificare l'indirizzo di linea, come in questo esempio:

```
0r old.file
```

dove la lettura del file procede a partire dall'inizio del buffer, in corrispondenza della linea zero.

SALVATAGGIO DEL BUFFER E TERMINE DI UNA SESSIONE

Quando siete in **ed**, le modifiche che apportate al file sono riportate in un buffer. Il programma **ed** non aggiorna il file originale fino a quando non lo richiedete esplicitamente tramite l'opzione di scrittura **w** (write).

```
$ ed old.file
260
p
*
w
260
*
```

L'opzione **w** visualizza il numero di caratteri che avete scritto nel file. Una volta che le modifiche sono state apportate al file, non è più possibile ripristinare la versione originale.

Quando state scrivendo, potete uscire da **ed** tramite l'opzione **q** (quit).

```
$ ed old.file
260
q
$
```

Il programma **ed** termina l'esecuzione e restituisce il controllo allo shell. Se cercate di uscire dopo che avete apportato delle modifiche, ma prima di avere aggiornato effettivamente il file, **ed** chiede una conferma delle vostre intenzioni. Se premete ancora **q**, **ed** termina senza riportare sul file le modifiche che avete indicato.

TRATTAMENTO DELLE LINEE DI TESTO

Molte operazioni in modalità comando interessano le singole linee di un file di testo. Potete spaziare all'interno di un file saltando di linea in linea, ma la maggior parte dei comandi agisce sulla linea corrente. In modalità comando, la linea corrente viene indicata con il carattere . (punto), mentre l'ultima linea del file è contrassegnata dal carattere $ (dollaro). In ambiente UNIX, durante la scrittura di testo, è raro disporre della numerazione di linea, e per averla dovete richiederla espressamente a **ed**, che tiene memoria in ogni istante della linea corrente.

VISUALIZZAZIONE DELLA LINEA CORRENTE

Per visualizzare la linea corrente, utilizzate il comando **p** (print), battendo il tasto p (minuscolo), a differenza del comando prompt che viene lanciato battendo il tasto P (maiuscolo).

```
$ ed old.file
260
p
ti vedo più tardi .....giorgio
```

Quando leggete un file in **ed**, la linea corrente è l'ultima linea del file.

```
$ ed old.file
260
p
ti vedo più tardi .....giorgio
.=
6
$=
6
```

L'operatore = (uguale) permette di conoscere il numero di una linea, in particolare la notazione .= restituisce il numero della linea corrente, mentre $= visualizza il numero dell'ultima linea del file. In questo esempio il file ha sei linee.

La maggior parte dei comandi di **ed** agisce su una linea o su un insieme di linee che precedono il comando stesso. Se non indicate esplicitamente il

campo di azione del comando, **ed** sottointende che si tratti della linea corrente. Invece del comando **p** è possibile scrivere **.p**, **$p** oppure **6p**. Poiché i caratteri punto e dollaro assumono lo stesso significato in questo esempio, cioè indicano entrambi la linea 6, le tre specifiche del comando **p** sono identiche. Per referenziare un insieme di linee, specificate il numero della prima e dell'ultima linea, separati da una virgola, come in **vi** modalità ultima linea.

```
$ ed old.file
260
1,2p
ciao. Leo cosa stai facendo? Ho studiato a fondo
il sistema UNIX e devo informarti che l'ho trovato
```

Il primo numero di linea deve essere minore del secondo. Potete anche utilizzare i caratteri punto o dollaro in questi indirizzamenti di linea.

```
$ ed old.file
260
3,$p
divertente. Il sistema operativo UNIX sembra essere molto più potente
di ogni altro sistema operativo per piccoli elaboratori e sono sicuro
che è di grande aiuto nella risoluzione dei tuoi problemi.
ti vedo più tardi.....giorgio
```

Analogamente, potete utilizzare semplici espressioni aritmetiche negli indirizzamenti di linea.

```
$-2,$p
di ogni altro sistema operativo per piccoli elaboratori e sono sicuro
che è di grande aiuto nella risoluzione dei tuoi problemi.
ti vedo più tardi.....giorgio
```

Per visualizzare l'intero file, potete utilizzare la forma contratta **,p.**

CAMBIO DI LINEA CORRENTE

Potete anche cambiare la linea corrente mentre operate in modalità comando. Il carattere newline permette di avanzare di una linea, visualizzando il contenuto della linea corrente, mentre il comando – (meno) seguito da un ritorno a capo vi fa retrocedere di una linea. Potete anche referenziare direttamente una linea, che **ed** assume come linea corrente.

```
.=
6
2 il sistema UNIX e devo informarti che l'ho trovato
.=
2
```

La linea corrente diventa la linea 2 che **ed** visualizza. È permesso anche utilizzare semplici espressioni aritmetiche.

MODALITÀ INSERIMENTO

Quando passate da modalità comando a modalità inserimento, potete posizionarvi su una linea che precede o segue quella corrente a seconda della opzione che specificate. Utilizzate infatti il comando **i** (input) per entrare in modalità inserimento prima della linea corrente e il comando **a** (append) per attivare la modalità inserimento dopo la linea corrente. Quando siete in modalità inserimento, il prompt non compare e tutti i caratteri che battete da tastiera vengono inseriti nel file. Per ritornare in modalità comando dopo che avete terminato di scrivere il testo, battete su una linea il carattere punto.

```
$ ed new.file
?new.file
a
salve gente
un'altra linea
una terza linea
.
,p
salve gente
un'altra linea
una terza linea
```

Se siete in modalità inserimento e non battete su una linea il carattere punto, potete inserire un numero qualunque di linee. Per correggere una linea che contiene un errore, dovete ritornare in modalità comando e fare le necessarie modifiche, nelle modalità indicate nei prossimi paragrafi. I comandi **i** e **a** possono anche essere preceduti da indirizzi di linea: **14a** induce **ed** ad aggiungere al file le successive 14 linee.

ELIMINAZIONE DI LINEE

Il comando **d** (delete) cancella una o più linee dal file. Se non specificate nessuna opzione, **d** cancella la linea corrente.

```
,p
salve gente
un'altra linea
una terza linea
2
un'altra linea
```

```
d
,p
salve gente
una terza linea
```

In questo modo viene cancellata dal buffer la linea 2. Il comando **d** può cancellare anche un insieme di linee.

ANNULLAMENTO DI ERRORI

Il programma **ed** ricorda l'ultimo comando che avete inserito, per cui potete annullare solo l'ultima modifica che avete apportato al file, come in **vi**. Il comando **u** (undo) ripristina il contenuto del buffer, come mostrato in questo esempio:

```
,p
salve gente
una terza linea
u
,p
salve gente
un'altra linea
una terza linea
```

Se battete ancora **u**, la linea ripristinata viene di nuovo cancellata, poiché l'operazione di ripristino tramite **u** è stata l'ultima modifica che avete eseguito.

```
,p
salve gente
un'altra linea
una terza linea
u
,p
salve gente
una terza linea
```

In questo modo è possibile apportare molte modifiche, anche se il comando **u** ripristina solo l'ultima operazione eseguita.

RICERCA DI STRINGHE

Come **sed** e **vi**, **ed** supporta operazioni di ricerca e sostituzione di espressioni regolari complete. Tramite l'operatore di ricerca / (barra) il programma **ed** può selezionare qualunque sequenza di caratteri sia presente su una linea, per esempio:

```
/stringa
```

A partire dalla linea corrente, il programma **ed** effettua una ricerca in avanti lungo il file fino a quando incontra la prima linea che contiene la stringa in esame. Notate che non viene usato :, diversamente dalla modalità prima linea di **vi**. Se la ricerca ha esito negativo, **ed** riparte dall'inizio del file e continua la ricerca fino a quando raggiunge la linea corrente. Quando la ricerca ha esito positivo, **ed** termina la ricerca e visualizza la linea contenente la stringa, come mostrato qui:

```
/sicuro
altro sistema operativo per piccoli elaboratori e sono sicuro che è
```

La linea visualizzata diventa la linea corrente; **ed** non visualizza il numero della linea. Se in tutto il file non esiste nessuna stringa del tipo indicato, **ed** visualizza l'indicatore di errore **?**.

Potete anche eseguire una ricerca a ritroso partendo dalla linea corrente tramite il carattere ? (punto interrogativo).

```
?stringa
```

A partire dalla linea corrente, il programma esegue una ricerca verso l'inizio del file e si arresta quando trova la stringa. Se la ricerca ha esito negativo, il programma si porta alla fine del file e continua la ricerca procedendo verso la linea corrente.

Poiché **ed** ricorda l'ultima espressione regolare, potete ripetere la ricerca della stringa semplicemente battendo il carattere / o ? (a seconda della direzione voluta), seguito da un ritorno a capo.

SOSTITUZIONE DI PARTI DI TESTO

Il programma **ed** dispone del comando **s** (substitute) che permette di sostituire una stringa con un'altra; la sintassi è la stessa del comando equivalente in **vi**. Il comando **s** accetta come argomenti un numero di linea o un ambito di numeri di linea se intendete effettuare la sostituzione su più linee. Per esempio:

```
p
di ogni altro sistema operativo per piccoli elaboratori e sono sicuro
4,6s/sicuro/convinto
di ogni altro sistema operativo per piccoli elaboratori e sono convinto
```

La linea aggiornata viene visualizzata per verificare se sono state riportate le modifiche preventivate.

Questo esempio mostra la sostituzione della *prima* occorrenza della stringa **sicuro** su ciascuna delle linee dalla 4 alla 6. Potete usare complesse espressioni regolari per la parte da cancellare nell'operazione di sostituzio-

ne, ma tenete presente di verificare il risultato, perché piccoli errori nell'espressione regolare possono provocare in **ed** l'identificazione di una stringa diversa da quella voluta.

Potete cancellare una stringa con questo comando; è sufficiente specificare una stringa nulla come stringa di sostituzione:

```
2,5s/la//
```

Aggiungete dunque un altro carattere / alla fine del comando di sostituzione. In generale, se è assente la barra di chiusura, **ed** visualizza la linea che è stata aggiornata, altrimenti non esegue alcuna visualizzazione.

Le espressioni regolari ^ e $ designano rispettivamente l'inizio e la fine della linea.

Per aggiungere una stringa alla fine di una linea, utilizzate il seguente comando:

```
s/$/nuovo testo alla fine
```

Per aggiungere una stringa all'inizio di una linea, utilizzate il seguente comando:

```
$/^/nuovo testo all'inizio
```

SOSTITUZIONI GLOBALI

Normalmente il comando **s** modifica solo la prima occorrenza di una stringa che si presenta su una linea. Potete aggiungere l'opzione **g** alla fine del comando di sostituzione per forzare il comando **s** a eseguire una sostituzione di carattere generale su ogni occorrenza della stringa presente sulla linea (o su un insieme di linee se specificate il campo di numeri di linea).

```
s/vecchio testo da cancellare/nuovo testo da aggiungere/g
```

In questo modo viene cambiata ogni occorrenza della stringa sulla linea corrente. Per eseguire la modifica su ogni occorrenza della stringa che compare nell'intero file, utilizzate il seguente comando:

```
1,$s/vecchio testo/nuovo testo/g
```

Potete utilizzare un *indirizzo di contesto* invece di un numero di linea; il carattere $ indicato nell'esempio precedente rappresenta un indirizzo di contesto.

```
/salve/,/ciao/s/vecchio/nuovo/
```

Il programma **ed** effettua la ricerca a partire dalla linea corrente ed esegue la sostituzione nella parte di file compresa tra le stringhe **salve** e **ciao**. Questa tecnica di indirizzamento è più complicata rispetto alla semplice notazione numerica di linea, ma conferma che in ambiente UNIX l'operazione di scrittura può essere realizzata senza una numerazione esplicita di linea. Per questo motivo **ed** viene denominato *editor di contesto*.

SPOSTAMENTO E COPIA DI LINEE

Il comando **m** (move) permette di trasferire una linea o un insieme di linee da una locazione del file a un'altra, questa operazione è nota anche come *cut and paste*. Per esempio, per trasferire la linea corrente alla fine del file, scrivete:

```
.m$
```

Il comando **m** riceve come argomento anche un indirizzo di linea o un insieme di linee. Questo comando:

```
3,5m1
```

trasferisce le linee 3, 4 e 5 immediatamente dopo la linea 1.

Per copiare una linea da una locazione a un'altra senza cancellarla, utilizzate il comando **t**.

```
2,4t$
```

In questo modo le linee comprese tra la 2 e la 4 vengono copiate alla fine del file.

APERTURA DI UNO SHELL: L'OPERATORE !

Infine, potete uscire da **ed**, per eseguire altri comandi in ambiente shell, inserendo da tastiera il carattere !, seguito dal comando che intendete eseguire.

```
!cat old.file
```

Quando il comando termina, **ed** visualizza il carattere ! e ritorna in azione. Per sospendere temporaneamente la sessione di **ed** e creare uno shell interattivo, potete scrivere:

```
!sh
```

Questo shell prende il nome di *subshell*, in quanto la vostra sessione originale di editing è ancora attiva e attende che terminino le operazioni gestite dal subshell per riprendere il controllo. Potete interrompere l'esecuzione del subshell premendo CTRL-D o **exit** e ritornare così alla sessione **ed**.

Abbiamo presentato finora solo alcune delle utility messe a disposizione da **ed**, ma avremo modo nei successivi capitoli di approfondirle ed esaminarne di nuove.

6.8 Approfondimenti

Insistiamo sul concetto che per apprendere l'uso di editor in ambiente UNIX conviene fare frequente pratica con l'editor da voi preferito. Inoltre osservate un esperto al lavoro e non temete di porgli delle domande quanto lo vedete eseguire operazioni troppo complicate, o troppo veloci, per voi. Infine, spendete quanto più tempo potete nel consultare la documentazione d'uso e applicare nuove idee.

FILE DI SCRIPT PER ed

L'editor **vi** è un programma completamente interattivo e come tale richiede che l'uscita sia diretta al terminale e l'ingresso provenga da terminale, per cui non utilizza come gli altri programmi i file standard input e standard output.

L'editor **ed**, invece, impiega normalmente standard input e standard output e permette di usare file con particolari procedure (*script*) per eseguire automaticamente operazioni di edit su file di cui conoscete il contenuto e per i quali possiate prefigurarvi la sequenza di comandi **ed** necessaria per le modifiche volute.

Il seguente file rappresenta una semplice procedura di **ed**:

```
$ cat ed.script
/findme/
a
salve
ciao
.
w
q
$
```

Se inserite interattivamente questi comandi in **ed**, si ottiene il seguente risultato: la linea corrente si inizializza alla linea contenente la stringa **findme** (/**findme**/); dopo questa linea vengono inserite, in modalità inserimento,

(**a**) le stringhe **salve** e **ciao**; si ritorna in modalità comando (.); il file viene scritto (**w**) e, infine, si esce dall'editor (**q**).

Il seguente comando

```
$ ed -s old.file < ed.script
$
```

esegue le operazioni che avete memorizzato nel file di **ed** e aggiorna il file **old.file** evitando un inserimento manuale delle modifiche. Questa tecnica di aggiornamento di file appare in diverse *procedure di shell* (shell script), ma è utile solo se conoscete gli esatti cambiamenti che intendete apportare al file prima di definire la procedura in **ed**.

Per eliminare l'informazione prodotta in uscita da **ed** durante l'elaborazione di una procedura, specificate l'opzione **-s** (silent) sulla linea di comando.

RICERCA CON ESPRESSIONI REGOLARI IN emacs

L'editor **emacs** supporta operatori addizionali di ricerca particolarmente efficienti per localizzare nel buffer corrente stringhe di testo specificate con espressioni regolari. Il comando **^M^s** (CTRL-**meta** CTRL-**s**) permette di effettuare la ricerca dell'espressione regolare la cui introduzione viene richiesta da **emacs**, utilizzando la stessa sintassi già vista per **ed**, con alcune estensioni di operatori particolari di **emacs**. Il comando precedente esegue la ricerca in avanti, per la ricerca a ritroso si usa il comando simile **^M^r**.

Quando la stringa richiesta viene trovata, potete continuare la ricerca per un'altra occorrenza della stringa stessa con i comandi **^s** (search) o **^r** (reverse) nel caso, rispettivamente, di ricerca in avanti o a ritroso; **emacs** riutilizza l'ultima espressione regolare definita.

Una procedura simile si applica per effettuare la sostituzione di stringhe che vengono referenziate tramite espressioni regolari. Utilizzando il comando

```
M-x replace-regexp
```

(seguito da newline) **emacs** chiede di specificare la stringa da cercare e quella che la sostituisce. Questo comando sostituisce *tutte* le istanze dell'espressione regolare con la relativa stringa di sostituzione; se volete che **emacs** chieda una vostra conferma prima di operare la sostituzione della stringa trovata, dovete usare:

```
M-x query-replace-regexp
```

Trovata la prima occorrenza dell'espressione regolare, **emacs** si arresta in attesa di una risposta: premete **y** se volete eseguire la sostituzione, altrimenti **n**; in ambedue i casi **emacs** passa a ricercare la successiva occorrenza della stringa, se esiste. Potete anche premere **R** per indurre **emacs** a sostituire senza ulteriori richieste di conferma tutte le restanti occorrenze della stringa contenute nel file, oppure potete battere **^g** per annullare l'operazione e ritornare nel normale ambiente **emacs.**

Capitolo 7

Altri comandi di uso generale

Finora abbiamo trattato le caratteristiche di base dello shell e dell'interfaccia utente in ambiente UNIX, oltre ai comandi più comunemente usati. UNIX supporta complessivamente più di 800 comandi eseguibili e in questo capitolo esamineremo in maggiore dettaglio quelli più noti e alcuni strumenti software indipendenti che possono facilitare l'uso del sistema UNIX e che vengono utilizzati nelle procedure di shell (shell script) per automatizzare particolari attività. Per avere maggiori dettagli sul significato e la definizione delle procedure di shell, leggete con attenzione il Capitolo 8.

7.1 L'ambiente in maggior dettaglio

Quando eseguite un comando, il sistema crea un ambiente di esecuzione a cui passa alcune delle variabili di ambiente, come nel caso degli editor descritti nel Capitolo 5 che utilizzano la variabile di ambiente **TERM.**

Non tutte le variabili di ambiente sono disponibili per il comando che eseguite, ma lo shell gli assegna un ambiente iniziale che può essere passato ai sottoprogrammi chiamati e solo quelle variabili di ambiente che sono dichiarate *exported* possono essere utilizzate anche ai livelli più interni. Si tratta di un sottoinsieme di tutte le variabili di ambiente che avete dichiarato. Se battete il comando

```
$ export
```

senza argomenti, il sistema visualizza un elenco di variabili di ambiente di tipo *exported* a cui potete aggiungere una nuova variabile di ambiente specificandola come argomento del comando **export**:

```
$ export TERM
$
```

In questo modo la variabile di ambiente **TERM** può essere utilizzata dai vostri comandi e sottoprogrammi.

Il comando **export** ammette variabili multiple nella linea di comando, come per esempio:

```
$ export TERM EDITOR HISTFILE
$
```

L'**export** funziona solo per i programmi sottostanti; in nessun caso una variabile di ambiente può essere esportata nell'ambiente del *padre* del comando.

LA VARIABILE PATH

Una delle più importanti variabili di ambiente è la variabile **PATH**. Quando eseguite un comando indicando il suo pathname completo, come per esempio

```
$ /usr/bin/vi
```

lo shell ricerca il comando da eseguire a partire dalla radice della gerarchia di directory, mentre, se non indicate un pathname completo, come per esempio

```
$ vi
```

lo shell non sa dove cercare il programma eseguibile. La variabile **PATH** memorizza una lista di directory in cui lo shell ricerca il comando che avete

specificato senza indicare il pathname completo. Esaminiamo un esempio tipico di variabile **PATH**:

```
$ echo $PATH
:/home/giorgio/bin:/sbin:/usr/sbin:/usr/bin:/usr/X/bin:/usr/ucb
$
```

La variabile **PATH** può differire, ma la sintassi è comune a tutti i sistemi UNIX. **PATH** corrisponde a un piccolo database, i cui elementi sono separati dal carattere : (due punti). Quando specificate un comando senza un pathname completo, lo shell esegue una ricerca in ogni directory di **PATH**, esaminandoli a turno da sinistra a destra, fino a quando incontra un comando con quel nome. Se un tale comando non risiede in alcuno dei directory specificati, lo shell visualizza un messaggio di errore; viceversa, se la ricerca si è conclusa positivamente, esegue il comando.

```
$ xxx
xxx: not found
$
```

Potete definire sulla vostra macchina più comandi con lo stesso nome, che però dovete inserire in directory diversi. La selezione dei directory in **PATH** avviene in ordine posizionale, per cui il tipo di comando prescelto a essere eseguito dipende dall'ordine con cui vengono esaminati i directory. L'esempio precedente indica che **/home/giorgio/bin** precede **/sbin** ed entrambi questi directory possono contenere comandi che possiedono lo stesso nome. È possibile che la variabile **PATH** di alcuni utenti non comprenda il directory **/home/giorgio/bin**, ma solamente **/usr/bin**, per cui può essere eseguita solo la versione di **vi** contenuta in questo directory. In definitiva, gli utenti possono controllare l'insieme dei comandi che hanno a disposizione cambiando la propria variabile **PATH**.

La maggior parte dei normali comandi eseguibili è allocata in uno dei directory **/sbin**, **/usr/bin** oppure **/usr/sbin** della vostra variabile **PATH**. Potete aggiungere altri directory, per ottimizzare l'esecuzione dei comandi o per memorizzare i vostri comandi privati.

Se avete installato X Window System o software applicativo aggiuntivo come un word processor o un sistema per gestione di database, anche il directory di collocazione di queste applicazioni può essere incluso nella vostra **PATH**. Parimenti, se volete usare gli strumenti BSD previsti in SVR4, dovrete aggiungere il directory **/usr/ucb** alla vostra **PATH**; posizionatelo all'inizio se volete che le versioni BSD dei comandi sostituiscano le equivalenti System V, alla fine se volete che i comandi System V abbiano la precedenza.

Come già notato, le singole componenti di **PATH** sono separate da un :.

La presenza di un ulteriore carattere : nella variabile **PATH**, come all'ini-

zio nell'esempio precedente, referenzia, nella posizione in cui si trova, il directory corrente. La differenza tra

```
$ PATH = /sbin:/usr/bin:/usr/sbin
```

e

```
$ PATH = :/sbin:/usr/bin:/usr/sbin
```

consiste nel fatto che solo la seconda variabile referenzia per primo il directory corrente, qualunque esso sia. Potete indurre lo shell a cercare il directory corrente dopo tutti gli altri directory con il seguente comando:

```
$ PATH = /sbin:/usr/bin:/usr/sbin:
```

Abbiamo in questo caso aggiunto un ulteriore carattere : alla fine di **PATH**. La variabile **PATH** viene di solito inizializzata e resa referenziabile quando entrate nel sistema e ne aggiornate il contenuto solo se volete modificare il suo valore rispetto a quello di default.

7.2 Il comando banner

UNIX mette a disposizione molti comandi di uso generale e, sebbene alcuni di questi siano di limitata utilità, ognuno ha una sua precisa collocazione nel sistema. Uno dei più semplici e immediati è il comando **banner** che riporta sullo standard output i suoi argomenti con caratteri ingranditi.

```
$ banner salve gente

XXXXXX   XXXXXX   X        X    X   XXXXXX
X        X    X   X        X    X   X
X        X    X   X        X    X   X
XXXXXX   XXXXXX   X          X X     XXXXXX
     X   X    X   X          X X     X
     X   X    X   X          X X     X
XXXXXX   X    X   XXXXXX       X      XXXXXX

XXXXXX   XXXXXX   X    X   XXXXXX   XXXXXX
X        X        XX   X     X      X
X        X        X X X     X      X
X   XX   XXXXXX   X   XX     X      XXXXXX
X    X   X        X    X     X      X
X    X   X        X    X     X      X
XXXXXX   XXXXXX   X    X     X      XXXXXX

$
```

Il comando **banner** tiene conto automaticamente delle dimensioni della pagina così da rendere sempre comprensibile il messaggio ingrandito.

Viene spesso utilizzato in alcune funzioni di stampa per distinguere tipi diversi di informazione. Per esempio, questo comando visualizza lo user id:

```
$ banner $LOGNAME
$
```

7.3 Il comando clear

Il comando **clear** è un semplice strumento per cancellare il video e ottenere il prompt $ sulla prima linea:

```
$ clear
```

Il comando non richiede argomenti, dovrebbe funzionare correttamente su quasi tutti i terminali.

7.4 Il comando date

I comandi di gestione di data e ora disponibili in ambiente UNIX hanno una risoluzione che è compresa tra un millisecondo e un anno. Quasi tutti i sistemi dispongono di un chip che svolge le funzioni di orologio/calendario e generalmente la data e l'ora vengono correttamente impostate in fase di caricamento. A ogni file è associata la data e l'ora dell'ultimo aggiornamento; esiste inoltre un comando specifico per visualizzare il giorno e l'ora correnti.

```
$ date
Wed May 27 18:12:56 EDT 1990
$
```

Per default, il sistema visualizza la data e l'ora del momento in base alle convenzioni locali, al fuso orario ed eventuale ora estiva; potete inoltre adattare alle vostre esigenze o comodità il formato di uscita di **date**. Il sistema può determinare l'inizio e la fine dell'ora estiva, ma può essere confuso in caso di cambiamenti nella legge di applicazione.

Il comando **date** viene spesso usato come datario dei lavori: può essere usato anche per cambiare la data e ora del sistema. Considerato che molte attività interne del sistema dipendono dall'esattezza di data e ora, dovranno essere sempre correttamente caricate. Il Capitolo 20 tratta del caricamento nel sistema della data e dell'ora e del modo di cambiare il formato dell'output del comando **date**.

7.5 Il comando cal

Il comando **cal** (calendar) rientra nella categoria dei comandi utili, ma non indispensabili. Senza argomenti, il comando riporta sullo standard output il calendario del mese corrente.

```
$ cal
   May 1990
 S  M Tu  W Th  F  S
                   1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
$
```

Se come unico argomento viene specificate l'anno, cioè un valore numerico compreso tra 1 e 9999, **cal** ne visualizza il calendario completo. Tenete presente che 92 si riferisce al 92 dopo Cristo e non al 1992. Il comando **cal** accetta anche due argomenti: nell'ordine, mese e anno. La seguente linea di comando

```
$ cal 9 1752
```

visualizza il calendario del mese di settembre dell'anno 1752, tenendo conto dell'applicazione della riforma del calendario.

7.6 Il comando calendar

Il comando **calendar** fornisce una semplice agenda che potete utilizzare per ricordare i vostri appuntamenti. Se il directory corrente contiene un file di nome **calendar**, il sistema ricerca nel file le linee che contengono l'indicazione della data di oggi o di domani. In pratica, potete inserire nel file **calendar** nuovi appuntamenti, uno per linea, ed eseguire il programma **calendar** almeno una volta al giorno per ricordare gli eventi più importanti della giornata.

```
$ calendar
Appuntamento con il medico in data May 28 per checkup
$
```

Il comando **calendar** accetta formati diversi per la data, oltre a quello mostrato nel precedente esempio, per cui risulta valida l'indicazione 5/28, a differenza del formato europeo, cioè 28/5, che non viene interpretato corret-

tamente. Potete predisporre che il sistema esegua **calendar** ogni volta che entrate nel sistema, per verificare così quali siano le attività in programma per la giornata. Il file **calendar** si trova generalmente nel vostro home directory.

7.7 I comandi more, tail, head

Abbiamo già parlato del programma **more**, che visualizza a blocchi l'uscita sullo schermo, passando da uno all'altro solo dopo che avete battuto il tasto di ritorno a capo. Il comando **pg** (pager) è simile a **more**, ma non così diffuso. Generalmente **more** rappresenta l'ultimo comando di un pipeline oppure, quando viene utilizzato singolarmente, presenta come argomenti un elenco di nomi di file.

```
$ more /etc/profile /etc/inittab
```

Dopo aver visualizzato una schermata piena, il programma si arresta visualizzando nell'ultima linea del video il prompt **– –More– – (%)**. Per passare alla schermata successiva, semplicemente premete la barra spaziatrice oppure **f** (effe); se prima di **f** introducete un numero, **more** avanza di altrettante schermate. Potete anche retrocedere nella visualizzazione di un file con CTRL-B (backward); introducendo un numero prima di CTRL-B potete retrocedere di quel numero di schermate. Con RETURN in luogo della barra spaziatrice potete avanzare una linea alla volta anziché di una schermata; un argomento positivo prima del RETURN specifica il numero di linee da scorrere in avanti. L'operatore CTRL-L visualizza di nuovo la schermata corrente.

In risposta al prompt, potete introdurre anche altri comandi che **more** interpreta come azioni da eseguire. Per esempio, nella sintassi utilizzata in ambiente **ed**, potete specificare una espressione regolare che inizia con il carattere / oppure ?: **more** esegue rispettivamente una ricerca in avanti o a ritroso nel file e visualizza la parte di testo che comprende l'espressione referenziata, se la trova. Se inserite un numero *n* prima del carattere /, **more** ricerca la *n*-esima occorrenza di quella espressione regolare. Per esempio

```
:6/[fe]grep/
```

ricerca la sesta occorrenza della stringa **fgrep** o della stringa **egrep** a partire dalla linea corrente.

L'operatore . (punto) richiede di visualizzare di nuovo la finestra corrente, mentre l'operatore **$** visualizza l'ultima finestra del file.

Se specificate più di un file nella lista di argomenti di **more**, l'opzione **:n** (next) posiziona il cursore all'inizio del file successivo rispetto a quello corrente, mentre **:p** (previous) lo posiziona all'inizio del file precedente. Sia **:n**

che **:p** possono essere preceduti da un numero che indica quanti file il comando deve saltare, rispettivamente in avanti o a ritroso, prima di referenziare il file che interessa, come in questo esempio:

```
:3p
```

Naturalmente, i comandi **:n** e **:p** non vengono eseguiti se lanciate **more** alla fine di un pipeline. Il comando **h** (help) visualizza l'elenco di comandi di **more**; per uscire da **more** e ritornare allo shell, battete il tasto q (quit) o il tasto DEL.

Il comando **more** utilizza la variabile **TERM** per adattare il formato dell'uscita al tipo di video che avete a disposizione, per cui il valore della variabile deve essere corretto.

Il comando **more** visualizza il contenuto di file a partire dall'inizio; per vedere la fine del file è necessario scorrerlo interamente o utilizzare l'operatore **$**. Per osservare la parte finale (o coda) del file usate il comando **tail** specificando come argomento il nome del file. Potete utilizzare **tail** alla fine o anche nella parte centrale di un pipeline. Il comando scrive sullo standard output le ultime 10 linee del file, come mostrato qui:

```
$ tail /etc/profile
EDITOR = /usr/bin/vi
FCEDIT = /usr/bin/vi
EXINIT = 'set ai'
ENV = "${ - : + $HOME/.ksh.aliases${ - # # *i*}}"
export HISTFILE EDITOR FCEDIT EXINIT ENV
>$HISTFILE

stty brkint echoe ixon ixany erase

trap 1 2 3
$
```

La presenza dell'opzione $-n$ con un argomento numerico n fa in modo che **tail** visualizzi un numero n di linee di file, in luogo del valore di default (10):

```
$ tail -1 /etc/profile

stty brkint echoe ixon ixany erase

trap 1 2 3
$
```

In aggiunta, l'argomento numerico $-n$ può essere seguito da una lettera per specificare un conteggio in unità diverse dalla linea: **-c** per contare in caratteri e **-b** per contare in blocchi.

Questo comando visualizza gli ultimi 20 caratteri del file, a partire dall'interno di una linea se necessario:

```
$ tail -20c /etc/profile
rt PATH;
trap 1 2 3
$
```

Il comando **tail** restituisce generalmente il controllo allo shell dopo aver letto la fine del file. In alcune situazioni, mentre un programma scrive in un file, può essere opportuno osservare il file mentre viene scritto. Potete ottenere lo scopo lanciando ripetutamente il comando **tail -3**, ma è più conveniente l'opzione **-f** (follow) in **tail**, che causa la ripetizione dell'esecuzione, ottenendo così la visualizzazione di ogni nuova linea del file mentre viene scritta, esempio:

```
$ tail -f aumento.file
```

Poiché il comando **tail -f** non termina mai, dovete interromperne l'esecuzione premendo il tasto DEL quando giungete alla fine del file.

Il comando **head** visualizza l'inizio di un file, per default le prime 10 linee; come argomento accetta una lista di nomi di file, in assenza di argomenti legge il suo standard input. Ovviamente non potete usare l'opzione **-f** per l'inizio di un file.

Potete visualizzare un numero di linee a piacere con un argomento numerico, per esempio:

```
$ head -4 /etc/profile
#ident "@(#)/etc/profile.sl 1.1 4.0 12/26/89 60576 AT&T-SF"
trap "" 1 2 3
umask 022 # predispone la maschera di creazione file di default
. /etc/TIMEZONE
$
```

7.8 I comandi cmp e diff

Spesso è utile confrontare due file per verificare se sono identici. Il comando **cmp** (compare) confronta due file qualsiasi, anche file binari, di cui avete specificato i nomi come argomenti.

```
$ cmp /sbin/sh /usr/bin/ksh
/sbin/sh usr/bin/ksh differ: char 5, line 1
$
```

Il comando visualizza in uscita solo il numero del primo byte, se esiste, che differisce nei due file. L'opzione **-l** induce invece **cmp** a specificare il nu-

mero di byte e i relativi valori per ogni differenza che riscontra tra i due file. In questo caso la dimensione dell'uscita può essere notevole, se esistono molte differenze tra i due file, ma sicuramente le informazioni che contiene risultano di maggiore utilità.

Il comando **cmp** accetta come argomento anche il carattere – (meno), invece del nome di un file, per indicare lo standard input.

```
$ cat /sbin/sh ¦ cmp -l - /sbin/sh
$
```

Se non si riscontrano differenze tra i file, **cmp** non genera nessun risultato. Per i file di testo, è disponibile uno strumento più efficiente, cioè il comando **diff** (difference) che produce un indice completo di tutte le linee che differiscono tra i due file, specificando i numeri di linea e le modifiche da apportare per rendere identici i file.

```
$ cat file1
questo è un file di esempio,
costituito da poche linee,
tutte molto concise.
Questo file permette di osservare l'operato di diff.
$ cat file2
questo è un file di esempio,
costituito da poche linee,
tutte molto semplici.
Questo file permette di osservare l'operato di diff.
$ diff file1 file2
3c3
< tutte molto concise
---
> tutte molto semplici
$
```

Il comando **diff** riporta in uscita innanzitutto i numeri delle linee che differiscono nei due file e li separa con il carattere **c** (changed); poi visualizza le linee in esame, contrassegnando quelle del primo file con il carattere < e quelle del secondo file con il carattere >. Il risultato in uscita diventa più complesso per file di dimensioni maggiori in quanto aumenta presumibilmente il numero di differenze, ma il comando **diff** non ha nessuna difficoltà a esaminarle tutte.

Se invece del nome di un file specificate il segno – (meno), il comando lo interpreta come file standard input.

Un numero diverso di caratteri spazio o eventuali caratteri di tabulazione presenti nei file fanno differire tra loro le linee, ma l'opzione **-w** (whitespace) induce il comando **diff** a non considerare distinte le linee che differiscono solo per il numero di caratteri spazio, per cui spazi e caratteri di tabulazione contenuti in un file possono essere trattati allo stesso modo.

```
$ cat file3 ¦ diff -w - file2
$
```

Potete anche usare l'opzione **-i** (ignore) per fare ignorare la differenza tra minuscole e maiuscole nella comparazione. Se i file sono identici, **diff** non genera alcun output.

L'output di **diff** ricorda il formato dei comandi di **ed** che convertono un file in un altro. In particolare, l'opzione **-e** (**ed**) produce una uscita che **ed** può utilizzare direttamente per convertire i file.

```
$ diff -e file1 file2
3c
tutte molto semplici.
.
$
```

È possibile utilizzare il risultato del seguente comando per convertire effettivamente **file1** in **file2**.

```
$ diff -e file1 file2 > ed.script
$ ( cat ed.script ; echo w ) ¦ ed file1
109
98
$ diff file1 file2
$
```

Questo esempio introduce molti concetti interessanti. Poiché **ed** legge i comandi dallo standard input, potete utilizzare un pipeline per inviare i comandi a **ed**, invece di batterli direttamente da terminale. Quando **ed** incontra la fine del file da pipeline, termina e restituisce il controllo allo shell. Così se visualizzate (**cat**) l'uscita prodotta da **diff -e** in **ed**, **ed** interpreta quei comandi e conclude la sua esecuzione quando il file **ed.script** finisce. Questo è il motivo per cui non è necessario battere **q** per terminare la sessione di **ed**. L'uscita prodotta da **diff -e** non contiene alcun comando **w** di scrittura di file in ambiente **ed**, per cui per default **ed** esegue le modifiche e poi esce senza riportarle. Per indurre **ed** a scrivere il file, dovete specificare l'opzione **w**, utilizzando il comando **echo w**.

Il comando **cat** che opera sul file **ed.script** è separato da **echo w** tramite l'operatore di shell ; (punto e virgola), che ha la funzione di mantenere distinti i comandi specificati su una linea. Il carattere finale di **ed.script** rappresenta il carattere di fine file, per cui **ed** esce prima di incontrare l'opzione **w** specificata dal comando **echo**. Potete evitare questa situazione delimitando con parentesi l'intera sequenza di caratteri che intendete sottoporre a **ed**. L'uso di parentesi come operatori di shell obbliga a eseguire tutti i comandi in esse contenuti in un unico subshell piuttosto che come comandi distinti. In questo modo l'uscita prodotta dai diversi comandi si concatena

e costituisce l'ingresso al comando **ed**, eliminando così l'effetto indesiderato di fine file altrimenti prodotto dal carattere ;.

Un altro modo per scrivere lo stesso comando può essere il seguente:

```
$ echo w ¦ cat ed.script - ¦ ed file1
```

I due numeri 109 e 98, indicati nell'esempio precedente, rappresentano il contenuto dello standard output di **ed** quando **ed** rispettivamente legge e scrive il file. Sperimentate queste soluzioni sulla vostra macchina.

7.9 Il comando dircmp

Il comando **dircmp** (directory compare) confronta il contenuto di due sottoalberi dei directory; segnala i file che esistono solo in un directory e confronta (usando **cmp**) i file che esistono in ambedue i directory. L'output ottenuto può essere voluminoso, tuttavia **dircmp** è utile quando occorre verificare due *versioni* di un directory. I due directory vengono forniti come argomenti al comando, per esempio:

```
$ dircmp $HOME /tmp/copied
```

Potete usare l'opzione **-d** (diff) per generare un output in formato simile a quello di **diff** in luogo di **cmp**.

7.10 I comandi sort e uniq

Il sistema UNIX definisce due ulteriori strumenti software che vengono spesso impiegati in una struttura a pipeline per filtrare file di testo secondo opportune modalità. Il comando **sort** riordina alfabeticamente un insieme di linee di testo. Può accettare una lista di nomi di file come argomenti, o leggere lo standard input, e scrivere le linee riordinate sullo standard output.

```
$ cat text.file
Ecco un breve file.
Ogni linea inizia con una lettera diversa.
ogni linea parte con una parola di quattro lettere.
Gran parte delle linee inizia con una lettera maiuscola.
$ sort text.file
Ecco un breve file
Gran parte delle linee inizia con una lettera maiuscola.
Ogni linea inizia con una lettera diversa.
ogni linea parte con una parola di quattro lettere.
$
```

Il comando **sort** ordina le linee di tutti i file specificati sulla linea di comando e include quindi le funzioni che in altri sistemi operativi sono denominate di **merge**. Questo comando può trattare file di notevoli dimensioni, anche se le sue prestazioni diminuiscono sensibilmente per file che hanno una dimensione maggiore di un migliaio di linee. Spesso risulta più rapido riordinare diversi file di grandi dimensioni separatamente e poi riordinarli tra loro con l'opzione **-m** (merge) del comando **sort**.

Il comando **sort** segue, durante la fase di riordinamento, l'ordine di precedenza definito dall'insieme dei caratteri ASCII (ASCII *collating sequence*) e lo applica a partire dall'inizio delle linee che confronta. Questa sequenza ASCII considera come primo carattere lo spazio, poi i caratteri di punteggiatura, i numeri, le lettere maiuscole e da ultimo le lettere minuscole. Quando il primo carattere delle due linee coincide, **sort** passa all'esame del carattere successivo, fino al carattere newline, se necessario, per stabilire l'ordine corretto. Quando riordinate i file alfabeticamente, il procedimento descritto si adatta perfettamente ma, quando l'ordinamento coinvolge dati numerici, i risultati non soddisfano le vostre attese.

```
$ cat dati.file
1
2
10
11
21
$ sort dati.file
1
10
11
2
21
$
```

L'ordinamento non risulta corretto in quanto il carattere 1 precede il carattere 2 nella notazione ASCII, per cui tutte le linee che iniziano con 1 precedono tutte le linee che iniziano con 2. L'opzione **-n** (numeric) specificata sulla linea di comando **sort** induce il comando a ordinare numericamente il file.

```
$ sort -n dati.file
1
2
10
11
21
$
```

Il comando **sort** permette di ignorare la differenza esistente tra caratteri minuscoli e maiuscoli durante l'operazione di confronto se utilizzate l'opzione **-f** (fold).

```
$ sort -f text.file
Ecco un breve file.
Gran parte delle linee inizia con una lettera maiuscola.
Ogni linea inizia con una lettera diversa.
ogni linea parte con una parola di quattro lettere.
$
```

In questo caso non esiste differenza tra **Ogni** e **ogni**, ma le linee differiscono in corrispondenza della colonna 12, poiché **inizia** precede **parte**. Analogamente, l'opzione **-d** (dictionary) annulla l'operazione di riordinamento sui caratteri di punteggiatura o altri caratteri speciali, per cui solo lettere, cifre e carattere spazio diventano significativi. L'opzione **-M** (month) induce il comando **sort** a considerare i primi tre caratteri della linea come indicatori del mese, per cui **Jan** precede **Feb**, anche se alfabeticamente lo segue.

Finora la chiave di ordinamento, cioè la parte di linea da cui il comando **sort** inizia l'ordinamento, veniva sempre considerata l'inizio della linea di ingresso. Il comando **sort** permette di indicare qualunque parte della linea come chiave di ordinamento, senza con questo suddividere in parti le linee, che sono invece considerate come record. Si può fare in modo che **sort** utilizzi come chiave di ordinamento qualche altra parte della linea di ingresso specificando gli argomenti **+n** e **-n**, dove *n* è un numero che identifica un campo della linea, mentre un campo è una sequenza di caratteri separati da un delimitatore. In particolare, il carattere spazio o il carattere di tabulazione costituiscono un delimitatore di campo e l'inizio della linea rappresenta il campo 0. In questo modo, l'opzione **+1** fa in modo che **sort** utilizzi il successivo campo invece dell'inizio della linea come chiave di ordinamento.

```
$ sort +1 text.file
Ogni linea inizia con una lettera diversa.
ogni linea parte con una parola di quattro lettere.
Gran parte delle linee inizia con una lettera maiuscola.
Ecco un breve file.
$
```

In questo esempio l'ordinamento delle linee procede dal secondo campo (**linea**, **parte** e **un**).

Utilizzate l'opzione **-n** per fare in modo che **sort** interrompa il confronto in corrispondenza del campo *n*-esimo.

```
$ sort +1 -3 text.file
```

In questo caso solo il secondo e terzo campo vengono utilizzati come chiave di ordinamento.

Quando l'ordinamento dei dati avviene per campi e le linee differiscono solo nei campi che non fanno parte della chiave di ordinamento, non potete

stabilire in quale ordine quelle linee appaiono in uscita. Comunque, potete controllare l'ordinamento dei campi del file specificando più di un campo nel comando **sort** come chiave di ordinamento.

```
$ sort +3 -4 +0 -1 text.file.
```

In questo caso, l'ordinamento delle linee procede considerando il campo 4 come chiave primaria. Per le linee in cui questo campo coincide, l'inizio della linea viene considerato chiave secondaria.

```
$ sort +3 -4 +0 -1 text.file
Ogni linea inizia con una lettera diversa.
ogni linea parte con una parola di quattro lettere.
Ecco un breve file.
Gran parte delle linee inizia con una lettera maiuscola.
$
```

Confrontate questa uscita con la seguente:

```
$ sort +3 -4 +7 text.file
ogni linea parte con una parola di quattro lettere.
Ogni linea inizia con una lettera diversa.
Ecco un breve file.
Gran parte delle linee inizia con una lettera maiuscola.
$
```

Potete cambiare il separatore di campo, che risulta per default il carattere spazio, tramite l'opzione **-t**, che accetta un carattere come argomento. Per esempio, per utilizzare il carattere : (due punti) come separatore di campo, scrivete:

```
$ sort -t: /etc/passwd
```

Inoltre, **sort** accetta l'opzione **b** (blank) per ignorare i caratteri spazio nella chiave di ordinamento e l'opzione **-r** (reverse) per invertire l'ordine di ricerca.

Potete combinare le diverse opzioni e chiavi di ordinamento per ordinare i file secondo le vostre intenzioni, ma la scelta corretta generalmente necessita di alcune prove.

Il comando **sort** viene generalmente utilizzato con **uniq** (unique) per calcolare il numero di occorrenze di una certa linea o campo presenti in un file. Il comando **uniq** esamina un file (lo standard input) contenente più occorrenze della stessa linea e riporta sullo standard output solo una istanza di ogni linea di ingresso che differisce dalle altre linee.

```
$ cat data2
abc
def
```

```
ghi
def
abc
$ sort data2 | uniq
abc
def
ghi
$
```

Viene scritta solo una copia di ogni linea, per cui **uniq** produce una lista di linee distinte. Inoltre, l'ingresso al comando **uniq** deve essere ordinato correttamente per non generare errori.

L'opzione **-c** (count) di **uniq** viene utilizzata spesso per calcolare il numero di occorrenze di linee distinte che sono presenti nel file e in uscita viene visualizzato prima della linea corrispondente.

```
$ sort data2 | uniq -c
   2 abc
   2 def
   1 ghi
$
```

Questo esempio indica che esistono nel file di ingresso due istanze delle linee **abc** e **def** e una sola istanza della linea **ghi**. L'opzione **-u** (unique) riporta solo le linee che non sono ripetute in ingresso, a differenza dell'opzione **-d** (duplicate) che riporta solo le linee che sono doppie.

Il comando **uniq** può anche limitare il suo campo di azione a particolari campi del file di ingresso. L'opzione **-n**, dove *n* è un numero, ignora durante il confronto i primi *n* campi della linea. Come accade per il comando **sort**, i caratteri spazio rappresentano i delimitatori di campo. L'opzione **+n**, dove *n* è un numero, ignora i primi *n* caratteri della linea, mentre, quando **-n** e **+n** vengono utilizzati insieme, i campi vengono trascurati prima dei caratteri.

7.11 I comandi cut e paste

Abbiamo visto il comando **cut** nel Capitolo 3. Questo comando viene utilizzato per frazionare le linee di ingresso in base a particolari campi o colonne esplicitamente indicati. Analogamente al comando **sort**, **cut** opera su specifiche parti della linea di ingresso; diversamente da **sort**, non riproduce in uscita l'intera linea di ingresso, ma permette di eliminarne alcune parti. Inoltre, oltre a **cut**, è disponibile un comando che riunisce in uscita le linee appartenenti a due file. Il comando **paste**, infatti, accetta due o più nomi di file in ingresso, legge una linea di ogni file, le combina tra loro generando una unica linea che riporta nello standard output. Ogni file di ingresso con-

tribuisce con un campo alla composizione della linea prodotta in uscita e il comando aggiunge un carattere di tabulazione come delimitatore di campo. L'opzione **-d** permette di specificare come delimitatore un carattere diverso da quello di default.

```
$ cat nuovo.d1
1111
2222
3333
$ cat nuovo.d2
AAAA
BBBB
CCCC
DDDD
EEEE
$ paste -d: nuovo.d1 nuovo.d2
1111:AAAA
2222:BBBB
3333:CCCC
:DDDD
:EEEE
$
```

Il comando **paste** accetta come argomento un numero qualunque di nomi di file e riunisce in uscita le relative linee seguendo un ordinamento da sinistra a destra. Se i file differiscono in lunghezza, **paste** inserisce semplicemente il delimitatore senza riportare alcun campo. Qualunque nome di file può essere sostituito dal segno - (meno), che **paste** interpreta come standard input.

Nell'esempio precedente abbiamo utilizzato il carattere : (due punti) come delimitatore, ma potete specificare più di un carattere dopo l'opzione **-d** e ogni elemento di questa lista viene impiegato a turno per delimitare una coppia di campi in uscita. Quando tutti i caratteri della lista sono stati utilizzati, **paste** riparte dal primo carattere della lista. In questo modo, delimitatori distinti separano i diversi campi e questo metodo risulta adatto soprattutto quando il numero di campi coincide con il numero di delimitatori.

```
$ paste -d123 file1 file2 file3 file4
```

In questo caso, il comando **paste** considera la prima linea di **file1**, a cui aggiunge la cifra 1, la prima linea di **file2**, la cifra 2, la prima linea di **file3**, la cifra 3 e infine aggiunge la prima linea di **file4**. La linea risultante termina con un newline e viene riportata sullo standard output. Il comando **paste** ripete queste operazioni su tutte le altre linee fino a quando raggiunge la fine del file.

7.12 Il comando join

Diversamente da **paste**, che combina intere linee corrispondenti prese da file diversi, il comando **join** combina linee di file che corrispondono su di un campo specificato. Per esempio, potreste avere due file che contengono un nome o identificatore di utente in un campo, ma differenti informazioni negli altri campi; il comando **join** vi consente di riunire, per ciascun nome o campo comune, le informazioni dei due file in una singola linea di output, per esempio:

```
$ cat file1
nome.a:ufficio:telefono
nome.b:ufficio:telefono
nome.c:ufficio:telefono
$ cat file2
nome.a:indirizzo:figlio
nome.b:indirizzo:figlio
nome.c:indirizzo:figlio
$ join -t: file1 file2
nome.a:ufficio:telefono:indirizzo:figlio
nome.b:ufficio:telefono:indirizzo:figlio
nome.c:ufficio:telefono:indirizzo:figlio
$
```

L'opzione **-t** (tab) definisce il delimitatore di campo; il default è il carattere di tabulazione. Il risultato di **join** viene inviato su standard output. Potete anche selezionare un sottoinsieme di campi in output con l'opzione **-o** (only), seguita da una lista di argomenti nel formato *m.n*, dove *m* è il numero di file (1 o 2) e *n* è il numero di campo.

Il comando **join** richiede che i due file in input siano preventivamente ordinati secondo il campo comune; i file possono anche contenere diverse linee con lo stesso campo di giunzione.

Per default, **join** usa come campo di giunzione il primo campo della linea in ambedue i file; l'opzione **-j** (join field) consente di cambiare questo criterio; per riunire i file usando il campo *n* nel primo file, e il campo *m* nel secondo, dovete dichiarare:

```
$ join -j1 n -j2 m file1 file2
```

7.13 Operazioni di database su file di testo

I comandi che abbiamo appena visto sono molto utili nella gestione di file di testo. Essi vengono frequentemente combinati in pipeline, utilizzando qualche volta file temporanei per essere in grado di memorizzare i risultati intermedi.

Con questi strumenti, potete realizzare operazioni complesse sui vostri dati, come per esempio operazioni di ricerca, di conteggio e di eliminazione

di parti di file, per produrre in uscita nuovi file di utile impiego. In realtà questi operatori costituiscono gli strumenti base per realizzare operazioni relativamente sofisticate su database.

Una volta che avete preso confidenza con i comandi base e le principali opzioni, diventa molto semplice creare database anche complessi, che si compongono generalmente di file di testo, ovvero di un insieme di linee strutturate a record, in cui ogni campo è separato da un delimitatore a vostra scelta. Il carattere : e i caratteri di tabulazione sono i più usati come delimitatori.

In pratica, potete creare diversi file contenenti i dati, prodotti da un editor o da qualche programma applicativo, che generalmente sono correlati in ogni singolo file, ma non lo sono se appartengono a file distinti. Se qualche campo dei file coincide o può essere messo in relazione con i campi di altri file, potete allora scrivere linee di comando per costituire database relazionali.

Quando definite nuove combinazioni di comandi e pipeline che soddisfano le vostre esigenze, potete eseguire specifiche domande sui dati senza utilizzare software scritto espressamente come per esempio un sistema di gestione di database.

Molti complessi database sono stati sviluppati in ambiente UNIX tramite questi comandi e le possibilità che si creano sono quasi illimitate.

Nel prossimo esempio cercheremo di valutare la consistenza delle possibilità offerte. Anche se questi strumenti lavorano più lentamente rispetto a un sistema professionale di gestione di database, i file presentano generalmente un formato di testo semplice da leggere, per cui potete aggiungere e riformattare i dati, fare domande e generare risultati complessi in modo molto naturale.

Per conoscere, per esempio, gli identificatori di tutti gli utenti della macchina, procedete come segue:

```
$ cut -f1 -d: /etc/passwd
root
daemon
bin
sys
adm
uucp
nuucp
slan
sync
lp
listen
sysadm
setup
powerdown
checkfsys
```

```
makefsys
mountfsys
umountfsys
jim
admin
msnet
pat
giorgio
$
```

Analogamente, potete determinare lo shell che è a disposizione degli utenti quando accedono al sistema.

```
$ cut -f1,7 -d: /etc/passwd > temp.dati
$ cat temp.dati
root:
deamon:
bin:
sys:
adm:
uucp:
lp:/sbin/sh
nuucp:/usr/lib/uucp/uucico
listen:
sync:/usr/bin/sync
install:
sysadm:/usr/sbin/sysadm
vmsys:/sbin/sh
oasys:/sbin/sh
giorgio:/usr/bin/ksh
pat:
jim:/usr/bin/ksh
$
```

Abbiamo ridiretto l'uscita nel file temporaneo **temp.dati**, per essere impiegata successivamente.

Si tratta di un esempio non banale: se l'ultimo campo della linea di ingresso è vuoto, **cut** non inserisce il delimitatore in uscita.

Analizzando il file **/etc/passwd**, potete notare che l'utente "root" e altri utenti non presentano nel settimo campo la specifica del loro shell. Se intendete aggiungere lo shell di default **/sbin/sh** alle linee che non contengono uno shell nell'ultima posizione, potete trarre vantaggio dal fatto che **cut** pone il suo delimitatore alla fine delle linee che non hanno un campo finale, per esempio:

```
$ grep ":$" temp.dati : sed 's/$/\/sbin\/sh/' > nuovi.dati
$ cat nuovi.dati
root:/sbin/sh
```

```
daemon:/sbin/sh
bin:/sbin/sh
sys:/sbin/sh
adm:/sbin/sh
uucp:/sbin/sh
slan:/sbin/sh
lp:/sbin/sh
liste:/sbin/sh
jim:/sbin/sh
admin:/sbin/sh
msnet:/sbin/sh
pat:/sbin/sh
$
```

Il comando **grep** analizza tutte le linee che contengono il carattere : (due punti) nel file temporaneo e poi **sed** aggiunge la stringa **/sbin/sh** alla fine di ogni linea.

Il comando **sed** presenta un'espressione di sostituzione interessante perché include il carattere / (barra) come semplice carattere e non come delimitatore e, per annullare lo speciale significato di /, gli abbiamo anteposto il carattere \ (barra rovesciata).

A questo punto il file **nuovi.dati** contiene solo l'elenco di utenti che non avevano specificato lo shell nel file **etc/passwd** e, per includere anche gli altri utenti, utilizzate il seguente comando:

```
$ grep -v ":$" temp.dati >> nuovi.dati
```

In questo modo vengono aggiunte alla fine del file tutte le linee che contengono il carattere : alla fine del file.

È possibile stabilire quali sono gli shell che gli utenti stanno usando sulla macchina:

```
$ cut -f2 -d: nuovi.dati | sort | uniq
/sbin/sh
/usr/bin/ksh
/usr/bin/sync
/usr/lib/uucp/uucico
/usr/sbin/sysadm
$
```

e quanti utenti stiano usando lo stesso shell:

```
$ cut -f2 -d: nuovi.dati | sort | uniq -c | sort -nr
  12 /sbin/sh
   2 /usr/bin/ksh
   1 /usr/sbin/sysadm
```

```
      1 /usr/lib/uucp/uucico
      1 /usr/bin/sync
$
```

L'ultimo comando, **sort −nr**, dispone i risultati in ordine numerico decrescente, ma esistono altri modi per ottenere gli stessi risultati direttamente dal file originale **/etc/passwd**.

La scrittura di complessi pipeline come quello precedente richiede una certa esperienza, per cui è preferibile costruire questi comandi progressivamente fino a quando siete sicuri che l'output generato da ciascuno di essi sia quello corretto.

Spesso è utile usare un piccolo database di prova, invece dei dati reali, specialmente quando questi risultano voluminosi e i comandi sono complessi; inoltre è anche conveniente mantenere la linea di comando memorizzata in un file e modificarla con l'editor, nel corso dei vostri esperimenti, invece di riscriverla ogni volta.

Se inserite da tastiera:

```
$ sh cmdfile
```

viene eseguita la linea di comando contenuta in **cmdfile**, ma per eseguire **cmdfile** potete anche scrivere:

```
$ chmod +x cmdfile
```

Inoltre potete eseguire il comando direttamente:

```
$ cmdfile
```

Questa tecnica, che può trasformare lunghi e complessi pipeline in comandi facilmente utilizzabili, costituisce anche la base della programmazione di shell, come vedremo in seguito.

7.14 Approfondimenti

I comandi che UNIX mette a disposizione dell'utente sono molti e svariati sono i modi per utilizzarli. Mentre fate esperienza sul vostro sistema e osservate altri utenti lavorare, potete imparare rapidamente le tecniche di programmazione, alcune delle quali sono argomento di trattazione dei prossimi capitoli.

IL COMANDO sleep

Il comando **sleep** svolge una funzione molto utile e spesso necessaria. Infatti ha il compito di ritardare di un certo numero di secondi, in accordo alle specifiche del suo argomento numerico, la restituzione del controllo dell'unità centrale allo shell, senza svolgere, nel frattempo, alcuna operazione.

```
$ date "+%r" ; sleep 100 ; date "+%r"
11:18:23 AM
11:20:04 AM
$
```

Il margine di errore del comando **sleep** è di circa un secondo, per cui se scrivete

```
$ sleep 1
```

non si ottengono risultati corretti.

Uno degli impieghi più semplici di **sleep** è quello di creare un sistema di allarme. La seguente linea di comando genera un suono sul terminale e visualizza un messaggio 10 minuti dopo aver battuto il comando:

```
$ sleep 600 ; echo "\007 Ora del pranzo! \007" &
```

Questo comando è attivo fino a quando non uscite dal sistema, ma viene eseguito in background tramite l'operatore **&**, per cui potete continuare a svolgere parallelamente qualunque altra attività. La stringa \007 nel comando **echo** rappresenta una forma alternativa di come visualizzare stringhe di caratteri: \007 è la codifica ottale ASCII della combinazione di tasti CTRL-G, che genera un suono (beep) al terminale. La stringa \007 in questo contesto ha un significato molto diverso da \07, 007, 7 o \7 e, per indicare al comando **echo** che si tratta di un unico carattere e non di tre cifre, anteponetegli il carattere \ (barra rovesciata).

Poiché UNIX è un sistema multiprocesso, il programma di un altro utente acquisisce il controllo della CPU quando il vostro programma interrompe la sua esecuzione in attesa di un evento esterno o della disponibilità di una risorsa. Questa situazione differisce dal comportamento che adottano i sistemi monoprocesso, come per esempio MS-DOS, in cui potete originare un ritardo utilizzando attivamente il sistema. Questo tipo di attesa, realizzato eseguendo un certo numero di iterazioni di ciclo, può risultare estremamente dannoso in ambiente UNIX, dove altri programmi possono richiedere l'uso del sistema durante il periodo in cui il vostro processo si trova in una condizione di attesa. Un principio fondamentale dei sistemi multiprocesso è che tutti gli utenti condividono le risorse del sistema, per cui, quando siete in una situazione di inattività, anche se il vostro comando è attivo

in background, dovete utilizzare i comandi che trasferiscono le risorse del sistema ad altri utenti. Il comando **sleep** esegue questo compito automaticamente poiché quando il vostro processo è in una situazione di attesa, non state utilizzando realmente una grande quantità di risorse del sistema. Un vantaggio ulteriore offerto da **sleep** è che il suo conteggio dipende dal clock interno e non dalla velocità di elaborazione e risulta arrotondato al secondo.

IL COMANDO find

Quando esaminate il contenuto di un directory con il comando **ls** o copiate i file utilizzando metacaratteri come *, come avviene in questo esempio:

```
$ cp * $HOME/directory
```

potete osservare che i comandi referenziano solo i file presenti in un particolare directory. Se, invece, vi interessa copiare o esaminare il contenuto di un insieme di directory, piuttosto che operare separatamente su ogni directory, utilizzate il comando **find** che, pur presentando una sintassi di linea di comando tutt'altro che banale, resta uno dei comandi più utili e potenti.

Il comando **find** è in grado di addentrarsi in una gerarchia di directory e localizzare tutti i file tramite particolari modalità indicate sulla linea di comando, in modo da intraprendere su di essi particolari azioni. La sua linea di comando presenta una forma del tipo: **find** *lista-pathname espressioni* dove *lista-pathname* rappresenta la lista di directory da ricercare, cioè uno o più pathname completi o relativi di directory. La parte *espressioni* contiene la lista di operatori che descrivono sia i criteri di selezione dei file che intendete referenziare sia le azioni che desiderate intraprendere quando un file soddisfa i criteri di selezione. Per esempio

```
$ find $HOME -print
```

rappresenta una tra le più brevi linee di comando **find** che implica la stampa dei nomi di tutti i file e directory presenti nel vostro home directory. Nella sezione *lista-pathname* della linea di comando potete specificare altri directory. Questo comando

```
$ find /home/giorgio /home/leo -print
```

visualizza i nomi di tutti i file e directory contenuti in **/home/giorgio** e **/home/leo**. È lecito anche specificare i pathname mediante i metacaratteri:

```
$ find /home/* -print
```

In una linea di comando **find**, tutti i pathname seguono il nome del comando e non devono essere preceduti dal segno – (meno), a differenza degli operatori che si trovano alla fine della linea.

Per default, **find** non intraprende azioni sui file che trova, per cui dovete specificare –**print** se volete listare tutti i file che **find** referenzia. L'opzione –**print** induce **find** a scrivere sullo standard output tutti gli elementi che trova, uno per linea. Questa proprietà vi permette di utilizzare il comando **find** per conteggiare tutti i file e i directory presenti.

```
$ find / –print ¦ wc –l
8642
$
```

In questo caso, veniamo a sapere che ci sono ben 8642 file e directory! Operazioni di ricerca a larga scala come questa, che esamina l'intera struttura a directory, possono avere bisogno di molto tempo assoluto e di CPU, specialmente su macchine di grandi dimensioni. Dovete dunque limitare l'impiego di **find** a una lista di pathname che realmente soddisfano le vostre esigenze.

Potete specificare alcuni operatori che restringono la ricerca a un particolare sottoinsieme di file, come per esempio –**name** che indirizza la ricerca solo ai file referenziati espressamente sulla linea di comando.

```
$ find / –name profile –print
/etc/profile
$
```

Il nome dei file che indicate sulla linea di comando può contenere anche metacaratteri di shell, protetti tra virgolette.

```
$ find / –name "*profile" –print
/etc/profile
/home/leo/.profile
/home/pat/.profile
/home/giorgio/.profile
/.profile
$
```

Tutti i metacaratteri di shell sono ammessi, purché protetti.

Quando **find** effettua la ricerca attraverso la gerarchia di directory, l'ordine in cui trova i file dipende dall'organizzazione interna del sistema e non dalle vostre scelte. Potete elaborare l'uscita prodotta dal comando **find** con **sort** o con altri strumenti, per riorganizzare l'elenco dei nomi secondo le vostre esigenze.

Quando specificate più opzioni sulla linea di comando, **find** le applica ai file procedendo da sinistra a destra. Il comando

```
$ find / –print –name "*profile"
```

non coincide con

```
$ find / -name "*profile" -print
```

perché il primo esempio stampa i nomi dei file e poi seleziona il nome che interessa, mentre il secondo esempio seleziona prima i nomi e poi li stampa. Il comando **find** referenzia ogni nome di file presente nella *lista-pathname*, esamina tutti i componenti della sezione *espressioni* per decidere se selezionare effettivamente o trascurare il file referenziato. Nella prima ipotesi le espressioni vengono eseguite sui file, mentre se una sola espressione impone la seconda ipotesi, le restanti espressioni non vengono analizzate. Nella terminologia **find** l'espressione valutata è *vera* se il nome di file viene selezionato e *falsa* in caso contrario.

Esistono molte altre opzioni che potete specificare al comando **find**, parecchie delle quali permettono di referenziare i file che soddisfano a particolari attributi: per esempio, **-user pat** seleziona i file che sono di proprietà dell'utente **pat**, mentre **-group sys** seleziona i file appartenenti al gruppo **sys**. Il comando

```
$ find / -user $LOGNAME -print
```

visualizza i nomi di tutti i file presenti nella macchina di cui siete proprietari.

L'opzione **-type c** seleziona i file di tipo *c*, cioè una tra le categorie di file che sono disponibili in ambiente UNIX, come evidenzia la prima colonna a sinistra delle informazioni sui file prodotte dal comando **ls -l** (**f**, per file, e **d**, per directory, sono le categorie di file più comunemente utilizzate). L'opzione **-mtime** *n* seleziona i file che avete aggiornato esattamente *n* giorni prima della data corrente, l'opzione **-mtime** *-n* seleziona i file che avete aggiornato negli ultimi *n* giorni, mentre **-mtime** *+n* seleziona i file che avete modificato *n* giorni prima della data corrente. L'opzione **-size** *n* seleziona invece i file che hanno una dimensione esattamente di *n* blocchi, l'opzione **-size** *-n* seleziona i file che hanno una dimensione inferiore a *n* blocchi, mentre **-size +n** seleziona i file che hanno una dimensione maggiore di *n* blocchi. Inserite da tastiera questo comando

```
$ find / -size +1000 -print
```

per verificare quali sono i file di grande dimensione che sono presenti sulla macchina.

Potete negare il significato di tutti questi operatori se anteponete al loro nome (per esempio **-size**) il carattere ! e un carattere spazio, per cui l'operatore assume un valore *falso* quando normalmente il suo valore è *vero* e viceversa. Per esempio, questo comando

```
$ find . ! -print
```

annulla l'operazione di stampa, ma è certamente ridondante. Una versione più utile del comando **find** permette di trovare tutti i file presenti nell'home directory che non sono di vostra proprietà.

```
$ find $HOME ! -user $LOGNAME -print
```

Dovete delimitare con caratteri spazio l'operatore ! che si applica solo al campo della linea di comando che lo segue immediatamente. Per avere un elenco di tutti i file presenti nell'home directory, che non sono di vostra proprietà e che hanno una dimensione maggiore di 50 blocchi, utilizzate il seguente comando:

```
$ find $HOME -size +50 ! -user $LOGNAME -print
```

Il comando può operare una scelta tra le sue componenti, come appare in questo esempio:

```
$ find $HOME -size +50 -o ! -user $LOGNAME -print
```

L'operatore **-o** (or), che connette le due espressioni **-size +50** e **! -user $LOGNAME**, permette si selezionare un file che soddisfa uno di questi due argomenti. I campi della linea di comando che si trovano in posizioni adiacenti, ma non sono collegati con l'operatore **-o**, sono logicamente connessi da un operatore **and**, per cui il file, per essere selezionato, deve soddisfare entrambi questi argomenti.

Il comando **find** può anche selezionare il tipo del file, per esempio:

```
$ find . -type f -print
```

Questo seleziona tutti i file normali; per selezionare directory usate **-type d**, per collegamenti simbolici usate **-type l**, potete anche selezionare file in base ad altri tipi di file, e anche in base al tipo di file system di appartenenza. Da notare che **find** non ricerca seguendo i collegamenti simbolici, a meno che sia usata l'opzione **-follow**.

Il comando **find** può utilizzare l'operatore **-exec** (execute) per eseguire un particolare comando sui file che seleziona. Questo operatore accetta dopo di sé una linea di comando che specifica il nome del file corrente che **find** ha selezionato e si serve, per questo, della sintassi speciale { } (parentesi graffe destra e sinistra). La linea di comando termina con l'operatore speciale \; (barra rovesciata e punto e virgola). Potete utilizzare questa struttura di comando per esempio per cambiare i diritti di accesso dei file che si trovano in alcuni directory:

```
$ find . -exec chmod -rw { } \;
```

Questo comando non visualizza i nomi dei file che vengono aggiornati, a differenza di:

```
$ find . -exec chmod -rw {} \; -print
```

L'operatore **-exec** assume il valore *vero* se il comando che esegue restituisce un valore zero, altrimenti assume il valore *falso*. È possibile definire, tramite **exec**, altri operatori per il comando **find** che viene esaminato da sinistra a destra fino a quando una delle espressioni valutate non è *falsa*.

IL COMANDO stty

L'ultimo comando che trattiamo in questo capitolo è **stty** (set tty), che viene utilizzato per esaminare e modificare i parametri di comunicazione che sono associati con il terminale nel corso della vostra sessione di lavoro. Il sistema UNIX garantisce un accurato controllo sulle modalità di elaborazione dei segnali dei tasti e dell'uscita al terminale. Il comando **stty** vi permette di inizializzare le opzioni per un particolare terminale o linea di comunicazione.

Utilizzate l'opzione **-a** (all) del comando **stty** per osservare i valori correnti delle opzioni del terminale.

```
$ stty -a
speed 9600 baud; line0; intr = DEL; quit = ^|; erase = ^h; kill = @; eof = ^d
-parenb -parodd cs8 -cstopb -hupcl cread -clocal -loblk
-ignbrk brkint ignpar -parmrk -inpck istrip -inlcr -igncr icrnl -iuclc
ixon ixany -ixoff
isig icanon -xcase echo echoe echok -echonl -noflsh
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel tab3
$
```

Le opzioni vengono visualizzate in una forma che può essere utilizzata per reintrodurre i parametri, se occorre. Ogni opzione del terminale possiede un identificatore, come per esempio **ixon**, e può essere attivata (set) o disattivata (clear). Se il nome dell'opzione è preceduto dal segno - (meno), l'opzione è disattivata, altrimenti è attivata. Per esempio, **ixon** si riferisce al controllo di flusso (XON/XOFF) da terminale, ma se il flusso è disattivato l'opzione assume la notazione **-ixon**.

Potete modificare il valore di una opzione tramite il comando **stty**. Utilizzate il segno - solo quando volete disabilitare l'opzione. Questo comando

```
$ stty ixon
$
```

abilita il controllo di flusso, mentre la seguente linea

```
$ stty -ixon
$
```

lo disabilita. Sulla linea di comando potete specificare più opzioni, come indicato qui di seguito:

```
$ stty -ixon 1200
$
```

Questo comando inizializza la velocità del terminale a 1200 baud e disabilita il controllo di flusso.

Potete predisporre l'estensione in caratteri, righe e colonne del vostro terminale (o finestra), per esempio:

```
$ stty rows 24 columns 80
$
```

La nuova estensione viene passata alle applicazioni a tutto schermo, tipo **vi**, quando sono lanciate, sovrapponendosi ai normali valori di default; se ridimensionate una finestra o vi collegate con un terminale non abituale, potete riadattare le dimensioni dello schermo con un comando manuale, se non appare corretto.

Il numero di opzioni utilizzabili è elevato, ma evitate di apportare modifiche che possono stravolgere la sessione. Quando il terminale non funziona correttamente è improbabile che si tratti di una situazione temporanea. Molto spesso non viene visualizzata alcuna uscita oppure lo shell non interpreta correttamente i segnali provenienti da tastiera, come nel caso del tasto di ritorno a capo che può essere interpretato come avanzamento di una interlinea senza ritorno carrello. Il procedimento di definizione di un identificatore di login inizializza correttamente le opzioni di **stty** e raramente dovete aggiornarne i valori di default. Comunque, se la variabile TERM non è inizializzata correttamente, oppure se intendete fare esperimenti con nuovi terminali o canali di comunicazione come per esempio reti locali, possono essere richieste modifiche alle opzioni di **stty**.

Se, quando entrate nel sistema, il terminale funziona bene ma poi inizia a comportarsi in maniera anomala per ragioni sconosciute, la cosa migliore è abbandonare il sistema ed effettuare un nuovo login. Il sistema allora inizializza le opzioni del terminale ai valori stabiliti in precedenza. Un comportamento meno drastico consiste nell'utilizzare il seguente comando:

```
$ stty sane
$
```

L'argomento **sane** definisce un insieme di opzioni del comando **stty**, utilizzate dalla maggior parte dei terminali, che non modificano la velocità del canale di comunicazione, ma hanno un effetto positivo quando il terminale si comporta stranamente, senza garantire però il perfetto funzionamento di applicazioni a tutto schermo, come per esempio **vi**. Se, invece, anche attivando l'opzione **sane**, non si riesce a ripristinare il normale comportamento del sistema, uscite e rientrate nuovamente. Se il terminale non funziona correttamente quando vi collegate, probabilmente l'errore si verifica durante questa operazione, mentre se le anomalie si verificano a seguito della errata interpretazione del tasto di ritorno a capo al termine delle linee di comando, premete CTRL-J per sostituire il ritorno a capo fino a quando eseguite il comando **stty sane** che potete lanciare con CTRL-J perché il sistema lo interpreti correttamente.

Le opzioni principali di **stty** sono le seguenti: **parenb** abilita la parità, mentre **–parenb** la disabilita. Le opzioni **parodd** e **–parodd** selezionano rispettivamente la parità dispari e pari (**parodd** viene utilizzata solo se è attiva l'opzione **parenb**). Le opzioni **cs5**, **cs6**, **cs7** e **cs8** assegnano la dimensione dei caratteri rispettivamente a 5, 6, 7 e 8 bit. Le opzioni **300**, **600**, **1200**, **2400**, **4800**, **9600** e **19200** stabiliscono la velocità di comunicazione del canale in baud, mentre **cstopb** e **–cstopb** stabiliscono rispettivamente uno o due bit di stop. L'opzione **tabs** induce il sistema a utilizzare i caratteri di tabulazione invece di sequenze di caratteri spazio per ridurre la dimensione dell'uscita, mentre l'opzione **–tabs** induce a utilizzare solo caratteri spazio e viene impiegata quando il terminale non interpreta correttamente il carattere di tabulazione.

Potete anche utilizzare **stty** per assegnare ai tasti alcune particolari funzioni di controllo. Per esempio, abbiamo assegnato a CTRL-D il significato di carattere di fine file, ma se preferite che CTRL-C svolga questo compito, eseguite i relativi assegnamenti con il comando **stty**:

```
$ stty eof c
```

dove *c* in questo caso viene rimpiazzato dal carattere di controllo che vi interessa.

Potete inserire direttamente i caratteri di controllo proteggendoli con il carattere \ (barra rovesciata) seguito dal carattere ^.
Questo comando

```
$ stty eof \^c
$
```

assegna a CTRL-C il significato di carattere di fine file. Potete utilizzare questa sintassi anche per modificare **erase** (rappresentata di solito da ←) e il carattere **intr** (interrupt, di solito rappresentato da DEL). Per assegnare **erase** a ←, utilizzate il seguente comando:

```
$ stty erase \^h
```

Potete anche impostare la sessione in modo che quando battete ←, il sistema risponda con la sequenza backspace-spazio-backspace che cancella il carattere al cursore; utilizzate anche il comando

```
$ stty echoe
```

perché questa caratteristica abbia effetto.

Lo *UNIX User's Manual* contiene l'elenco completo delle opzioni di **stty**, per cui consultate questo manuale e un esperto di comunicazioni prima di discostarvi troppo dalle indicazioni date in questo capitolo, soprattutto per quanto riguarda le modalità di esecuzione e gli strumenti che concorrono alla loro gestione in ambiente UNIX.

Capitolo 8

Programmazione sotto shell

Dopo avere esaminato i principali comandi di UNIX e trattato le modalità di composizione delle linee di comando, possiamo rivolgere la nostra attenzione alle caratteristiche fondamentali dello shell. Lo shell costituisce un ambiente di programmazione flessibile e altamente funzionale che favorisce lo sviluppo delle vostre applicazioni. La programmazione sotto shell ricorda concettualmente la programmazione *batch* MS-DOS, ma presenta caratteristiche più complesse che permettono di ottimizzare lo sviluppo e l'esecuzione di comandi e procedure di frequente utilizzo.

I programmi shell trovano abituale impiego nelle funzioni di utilità e nei comandi di amministrazione del sistema. I sistemisti normalmente scrivono applicazioni nel linguaggio shell quando la facilità di comprensione e di aggiornamento sono fattori più importanti dell'efficienza e della velocità di esecuzione dell'applicazione. In ogni caso, lo shell fornisce prestazioni eccellenti per l'esecuzione delle sue procedure (*shell script* è il termine con cui generalmente vengono denominati i file di comandi) e con questo sistema sono state sviluppate parecchie applicazioni.

In questo capitolo esamineremo innanzitutto alcune caratteristiche di shell non ancora trattate, poi parleremo di programmazione in ambiente shell e infine tratteremo utilizzi più progrediti di shell. Si noti infine che questo capitolo tratta di Bourne shell standard e delle caratteristiche fondamentali di Korn shell; la programmazione in C shell è molto differente. Il Capitolo 16 tratta di programmazione in shell Korn e C, e illustra molte caratteristiche di questi shell progrediti.

8.1 Comandi multilinea

Lo shell può accettare una linea di comando che ha una lunghezza che supera i limiti imposti dalle dimensioni dello schermo. Questa situazione si verifica spesso nel caso di complessi pipeline che si compongono di parecchi comandi e nomi di file molto lunghi. Finché non introducete un ritorno a capo, se superate i margini di linea, la maggior parte dei terminali dispone il cursore all'inizio della linea successiva per permettervi di completare il comando; quando battete un ritorno a capo, lo shell inizia la fase di elaborazione. Il comando viene trattato come una singola linea anche se occupa fisicamente più linee del terminale.

Lo shell interpreta correttamente i comandi multilinea, che potete anche creare su video in modo artificioso, suddividendo logicamente la linea di comando in più linee, per favorirne la leggibilità. Per realizzare questa suddivisione su linee distinte, anteponete al ritorno a capo il carattere \, che forza lo shell a ignorare il newline e consente di proseguire il comando sulla linea successiva. Un comando multilinea può verificarsi anche se aprite gli apici per un argomento del comando che lo richiede e non li chiudete prima del ritorno a capo.

In ambedue questi casi, lo shell memorizza la parte del comando che avete già inserito e visualizza un prompt diverso da $, per sollecitarvi a inserire la parte mancante del comando, per esempio:

```
$ cat /etc/profile \
>
```

Il nuovo prompt, che si chiama PS2 (prompt string di secondo livello, mentre il prompt $ ha come sigla PS1) può essere impostato e reso globale come qualunque altra variabile di ambiente, per esempio:

```
$ echo $PS2
>
$ PS2='altro:'
$ echo $PS2
altro:
$
```

In questo caso avevamo originariamente inizializzato il prompt PS2 a > ma, indipendentemente dal simbolo che lo rappresenta, PS2 segnala che il comando che avete inserito da tastiera è incompleto e che lo shell è in attesa di altri caratteri in ingresso. Potete annullare il comando se, in corrispondenza del prompt PS2, premete il tasto DEL. Questa operazione comporta la cancellazione del testo del comando fino a quel momento memorizzato e la visualizzazione del prompt PS1 (di solito $) che segnala la disponibilità dello shell ad accettare un altro comando.

INSERIMENTO DI DATI NELLA PROCEDURA DI SHELL (HERE DOCUMENT)

Lo shell interpreta correttamente i comandi multilinea, ma in due casi particolari interrompe l'analisi e visualizza il secondo prompt PS2. Primo, se alterate la reale funzione del ritorno a capo anteponendogli il carattere \ di escape o non delimitate correttamente un argomento, lo shell non esegue il comando che avete lanciato, ma visualizza il prompt PS2 per segnalarvi di completare opportunamente la linea di comando. Secondo, alcuni operatori di shell hanno un raggio di azione multilinea, per cui un comando può contenere alcuni caratteri di ritorno a capo che assumono il significato di divisori sintattici della linea di comando.

Un esempio di questo secondo caso è detto *here document*, inserimento diretto di dati nella procedura di shell, un processo di acquisizione immediata dei dati in ingresso che sono inseriti direttamente nella procedura di shell anziché memorizzati, preventivamente, su file. L'operatore di shell ≪*nome*, che non ammette al suo interno caratteri spazio, permette di fare accettare di seguito direttamente dalla procedura di shell i dati in ingresso: *nome* rappresenta una stringa di caratteri, mentre il simbolo ≪ ha una funzione simile a un normale operatore di ridirezione da un file. A partire dalla linea successiva in cui compare ≪*nome*, possono essere inseriti i dati che il comando deve leggere; la stringa *nome* termina la sequenza dei dati.

Per esempio, il comando

```
$ cat - ≪MARCATORE
> salve
> ciao
> MARCATORE
salve
ciao
$
```

visualizza sullo standard input i dati che inserite direttamente da tastiera dopo aver terminato la linea di comando con il carattere di ritorno a capo. Lo shell si rende conto che la linea di comando introduce a una operazione di inserimento diretto di dati, visualizza il prompt PS2 per ricevere e memo-

rizzare le linee di ingresso e, quando incontra il marcatore di chiusura, esegue il comando utilizzando i dati immessi come standard input. Al termine, visualizza il prompt PS1 e si mette in attesa del comando successivo.

In questo esempio, lo shell interpreta il carattere di ritorno a capo come marcatore di fine linea, ma non come marcatore di fine comando perché prima dovete inserire i dati da tastiera. Solo quando avete concluso questa operazione specificando la stringa di chiusura (cioè MARCATORE nell'esempio precedente), lo shell interpreta il successivo carattere di ritorno a capo come segnale di fine comando e visualizza quindi su linee distinte i risultati prodotti dal comando **cat**.

Esistono diversi altri casi di utilizzo di comandi multilinea in cui compare il prompt PS2, mentre in tutti i comandi normali la presenza di PS2 è sintomo di errore.

L'inserimento diretto dei dati si dimostra migliore rispetto alla ridirezione dello standard input, quando intendete utilizzare le variabili di shell nei dati, perché il comando legge direttamente un file ridiretto, mentre i dati inseriti nella procedura vengono prima esaminati dallo shell e poi consegnati al comando. Per esempio:

```
$ cat - <<HereDocuments
>Il mio identificatore di login è $LOGNAME
>HereDocuments
Il mio identificatore di login è giorgio
$
```

Oltre alle variabili di shell, nei dati che inserite nella procedura di shell potete anche utilizzare l'operatore (accento grave), per referenziare lo standard output di una linea di comando. Il seguente esempio:

```
$ cat - <<XYZZY
> echo "Ci sono `ls $HOME | wc -l` file in $HOME"
> XYZZY
```

rende possibile la combinazione in un file di testo di variabili di shell e di risultati prodotti da altri comandi.

8.2 File di comandi di shell

Qualunque comando, o sequenza di comandi, che potete direttamente inserire da terminale, può anche essere memorizzato in un file ed eseguito successivamente. Le modalità di esecuzione dei comandi memorizzati in un file coincidono con quelle adottate per i comandi che vengono battuti direttamente da tastiera, in questo caso lo shell fa in modo che sia connesso allo standard input il file piuttosto che il terminale. Un modo per eseguire i co-

mandi memorizzati in un file è di specificare il nome del file come argomento del comando **sh** che lancia un subshell, per esempio:

```
$ cat cmd.file
echo $LOGNAME
pwd
$ sh cmd.file
giorgio
/home/giorgio
$
```

Il file **cmd.file** – esempio di un breve file di comandi – contiene in questo caso i due comandi **echo $LOGNAME** e **pwd** che il comando **sh cmd.file** esegue, memorizzando i risultati prodotti nello standard output. Un altro modo per eseguire il comando è il seguente:

```
$ chmod u+x cmd.file
$ cmd.file
giorgio
/home/giorgio
$
```

Se il file di comandi è accessibile in esecuzione, è superfluo creare esplicitamente un subshell che lo esegua perché ci pensa per default lo shell di login. In questo modo un file di comandi può essere eseguito come un programma binario e difficilmente potete distinguere i due tipi di comandi.

In un file di comandi, ogni comando occupa una linea, mantenendo in ogni caso la stessa struttura che avreste usato inserendo da tastiera gli stessi comandi; potete utilizzare tutte le funzioni che sono disponibili da terminale. Per esempio, spesso conviene definire variabili di shell che hanno una vita pari alla durata dell'esecuzione:

```
$ cat cmd2.file
DIR=`pwd`
echo $DIR
$ cmd2.file
/home/giorgio
$
```

La variabile **DIR** viene impostata col nome del directory corrente e poi **echo $DIR** la visualizza. Le variabili di shell inizializzate in questo modo sono note solo all'interno del file di comandi in cui sono definite, ma non mantengono validità quando lo shell di login riacquista il controllo al termine dell'esecuzione del file.

Analogamente, anche i comandi che definite nei file non sono visibili all'esterno.

```
$ cat cmd3.file
pwd
echo Cambio di directory....
cd /home/leo
pwd
$ cmd3.file
/home/giorgio
Cambio di directory....
/home/leo
$ pwd
/home/giorgio
$
```

Potete cambiare directory all'interno del file di comandi, ma l'operazione **cd** non ha effetto sul resto della sessione.

Poiché il sistema non distingue un file di comandi da un programma binario eseguibile, le procedure di shell possono contenere al loro interno altre procedure, che vengono considerate veri e propri comandi. Non dimenticate l'uso di **export** per tutte le variabili locali che volete usare anche in sottoprogrammi o procedure gerarchicamente sottostanti.

I COMMENTI NEI FILE DI COMANDI

I file di comandi salvati su file, specialmente se sono di grande dimensione e di elevata complessità, vengono generalmente commentati per facilitare la loro lettura. In particolare, nella notazione shell, l'operatore # rappresenta il carattere di inizio commento ma, se impiegate questo carattere come simbolo di cancellazione, dovete annullarne la reale codifica anteponendogli il carattere \ prima di poterlo utilizzare come identificatore di commento. Lo shell ignora, ai fini esecutivi, qualunque cosa segua, sulla stessa linea, il carattere #:

```
$ #cat /etc/passwd
$ echo salve #ecco un esempio di commento al termine della linea
salve
$
```

8.3 L'operatore if

Lo shell dispone di adeguate strutture per il controllo del flusso di esecuzione di un file di comandi. Esaminiamo **if**, che consente di effettuare delle scelte. La sintassi di questo comando è semplice:

$ if *espressione*; **then** *comandi* ; **fi**

L'esempio appare su una sola linea, ma **if** può occupare più linee che sono concluse da **fi** (inverso di **if**). Inoltre, *espressione* rappresenta una qualunque espressione logica oppure un comando che restituisce un valore. Se l'espressione consiste in un valore di ritorno pari a zero, lo shell esegue i comandi che compongono la sezione **then**, fino a incontrare l'operatore di chiusura **fi**. Per esempio:

```
$ if true ; then echo salve ; fi
salve
$ if false ; then echo salve ; fi
$
```

Il carattere ; (punto e virgola) separa le diverse sezioni di questi comandi, che generalmente sono disposti su più linee, mentre il prompt PS2 vi ricorda l'incompletezza del comando.

```
$ if true
> then
>         echo salve
> fi
salve
$
```

I caratteri ritorno a capo e ; (punto e virgola) assumono in questo contesto lo stesso significato, cioè entrambi svolgono la funzione di delimitatori dei componenti del comando, per cui deve essere sempre garantita la presenza di almeno uno di essi.

Il precedente esempio introduce allo speciale operatore logico **true**, che potete utilizzare nei file di comandi. Questo operatore restituisce sempre il valore zero, per cui impone alla struttura **if** di eseguire solo i comandi che compongono la sezione **then**. È disponibile anche l'operatore logico **false**, che restituisce sempre un valore diverso da zero.

```
$ false
$ echo $?
1
$
```

Ricordate che **$** è una variabile di shell che contiene il valore di ritorno dell'ultimo comando eseguito.

Nella sezione **then** del costrutto **if** è possibile inserire quanti comandi volete e annidare altri comandi **if**.

```
$ if true
> then
>         echo salve
```

```
>         echo 'pwd'
> fi
salve
/home/giorgio
$
```

Versioni più complesse dell'espressione parte del costrutto **if** vengono trattate più avanti in questo capitolo.

Per indicare la biforcazione nel costrutto **if**, utilizzate l'operatore **else** che delimita una sequenza di comandi che vengono eseguiti quando la parte espressione restituisce un valore diverso da zero.

```
$ if false ; then
>         echo salve
> else
>         echo ciao
> fi
ciao
$
```

Anche la sezione **else** non ha limitazioni sul numero di comandi che possono comporla.

Altri costrutti **if** possono seguire l'operatore **else** se occorre e vengono identificati dalla forma contratta **elif** (else if).

```
$ if false ; then
>         echo salve
> elif true ; then
>         echo ciao
> fi
ciao
$
```

La presenza della sezione **then** dopo **elif** è obbligatoria e l'intero costrutto viene delimitato ancora con **fi**.

8.4 Il comando test

Finora abbiamo presentato una semplice versione della struttura **if**, così semplice che gli esempi trattati non sono stati probabilmente di grande utilità, per cui rivolgiamo ora la nostra attenzione a nuove strutture di controllo che realizzano operazioni logiche più complesse. Per esempio, risulta estremamente comodo disporre di una funzione che confronta due numeri e, se l'esito del confronto è positivo, esamina la sezione **then**, oppure disporre di una funzione che verifica l'esistenza di un file e, in base all'esito di questa verifica, esegue un insieme di comandi.

Per realizzare queste e molte altre funzioni utilizziamo il comando **test**, che segue sempre la struttura di controllo **if** e definisce i valori di verità (o codici di ritorno) su cui si basa il test condizionale. Esistono due versioni del comando **test** che potete utilizzare indifferentemente nei file di comandi, ma per uniformità e maggiore chiarezza sceglietene una e non cambiatela. La prima forma del comando **test** si compone del nome del comando, seguito da alcuni argomenti che ora analizzeremo.

```
$ if test $VAR
> then
>         echo salve
> fi
$
```

Se la variabile di ambiente **VAR** non è stata definita, l'espressione **test $VAR** fallisce e **if** considera il risultato falso. Se invece la variabile di ambiente è stata definita, allora il test dà esito positivo.

```
$ if test $HOME
> then
>         echo salve
> fi
salve
$
```

L'altra forma di **test** utilizza gli speciali operatori [e] (parentesi quadre). Gli argomenti di **test** sono racchiusi tra parentesi, ma il termine **test** non appare.

```
$ if [ $HOME ]
> then
>         echo salve
> fi
salve
$
```

Come per tutti gli operatori, dovete delimitare le parentesi quadre con caratteri spazio, altrimenti lo shell non interpreta correttamente la linea di comando. Tenete presente che il significato delle parentesi quadre in questo contesto differisce da quello trattato quando abbiamo parlato delle modalità di espansione dei nomi dei file tramite l'impiego dei metacaratteri, ma lo shell interpreta il significato corrente delle parentesi a seconda che seguano o meno l'operatore **if**.

La natura degli argomenti contenuti nelle parentesi quadre determina il tipo di test da effettuare. Il comando **test** esegue alcune operazioni sui file, esegue confronti numerici, confronti tra stringhe di caratteri e i valori delle

variabili di ambiente. In tutte le forme del comando, spazi devono separare le parentesi e ciascun operatore.

L'espressione **-f file** (file) restituisce il valore vero se il file esiste ed è un normale file; per esempio:

```
$ if [ -f /etc/passwd ] ; then echo il file esiste ; fi
il file esiste
$
```

Analogamente, l'espressione **-r file** (readable) restituisce il valore vero se il file esiste ed è accessibile in lettura; **-w file** (writable) risulta vera se il file esiste ed è accessibile in scrittura; l'espressione **-x file** (executable) restituisce il valore vero se il file esiste ed è eseguibile; **-d file** (directory) restituisce il valore vero se il file esiste ed è un directory; l'espressione **-s file** è vera se il file esiste e ha una dimensione maggiore di zero. Esistono altri operatori che potete utilizzare sui file, ma il loro impiego è abbastanza raro.

Gli operatori di confronto di stringhe verificano la presenza e il valore delle variabili di ambiente. Questo vi permette di eseguire alcune operazioni, di assegnare il valore a una variabile di ambiente e poi di verificare il suo valore per decidere le azioni da intraprendere. Per esempio, potete eseguire particolari comandi se vi trovate in uno specifico directory:

```
$ DIR=`pwd`
$ if [ $DIR = $HOME ]
> then
>         echo Nel directory preferenziale!
> fi
Nel directory preferenziale!
$
```

L'operatore = (uguale) verifica se le due stringhe di caratteri sono identiche e restituisce il valore vero se le stringhe coincidono, il valore falso in caso contrario. È possibile utilizzare l'operatore != che restituisce il valore vero se le stringhe non sono uguali. Tutti questi operatori devono essere separati dalle parentesi quadre e dai restanti argomenti che sono interni alle parentesi tramite caratteri spazio.

È possibile anche che le parentesi quadre contengano solo una stringa o una variabile di ambiente per stabilire se la relativa definizione è stata effettuata.

```
$ if [ "$NOVALUE" ] ; then echo salve ; fi
$
```

Se la variabile di ambiente **NOVALUE** non è stata definita, **test** restituisce un valore falso e quindi la sezione **then** non viene eseguita. Il costruttore

NOVALUE è protetto in questo esempio, perché se la variabile non fosse definita, non verrebbe eseguita alcuna sostituzione tra le parentesi quadre, che risulterebbero non contenere niente, con segnalazione di errore da parte di **test**. La protezione con virgolette causa la sostituzione del valore dell'espressione con una stringa vuota, invece di nessuna sostituzione, se **NOVALUE** non è definito.

Gli operatori = e != sono riservati per il confronto tra stringhe, ma non tra valori numerici. Il comando **test** può confrontare numeri interi, ma la programmazione in ambiente shell non è adatta alla risoluzione di problemi numerici, diversamente dalla maggior parte degli altri strumenti di programmazione. Il confronto numerico con il comando **test** assume la seguente forma:

```
if [ n1 -eq n2 ]
```

dove *n1* e *n2* sono espressioni numeriche. L'operatore **-eq** (equals) restituisce un valore vero se i due numeri sono uguali.

```
$ echo $VAL
2
$ if [ $VAL -eq 2 ]
> then
>       echo uguale
> fi
uguale
$
```

Altri operatori numerici sono **-ne** (not equal), **-gt** (greater then), **-lt** (less then) e **-le** (less then or equal). L'operatore di disuguaglianza **-lt** restituisce il valore *vero* se il primo numero è minore del secondo; anche gli altri operatori seguono questa sintassi.

Questi operatori numerici vengono spesso utilizzati per verificare il valore di ritorno prodotto da qualche comando o pipeline e intraprendere le relative azioni se il valore restituito soddisfa particolari requisiti. La maggior parte dei comandi restituisce il valore zero se termina l'esecuzione correttamente.

```
$ mkdir /tmp/SS
$ RET=$?
$ if [ $RET -eq 0 ]
> then
>       echo "mkdir ha terminato correttamente"
> else
>       echo "Il directory non poteva essere definito"
> fi
mkdir ha terminato correttamente
$
```

Tutti questi costrutti possono essere direttamente introdotti da terminale, tuttavia sono di maggiore utilizzazione pratica all'interno di procedure di shell.

Il comando **test** definisce altri strumenti che permettono di combinare gli operatori con espressioni complesse predefinite. L'operatore ! ha la funzione di negare il significato logico di un qualsiasi altro operatore, come indicato qui di seguito:

```
$ if [ ! -f file ]
> then
>       touch file          # questo crea il file
> fi
$
```

Questo comando crea un file se non esiste già. Altri operatori combinabili disponibili per **test** sono **-o** (or) e **-a** (and) che vengono impiegati per separare altri operatori.

```
$ F1=file1 ; F2=file2
$ if [ ! -f $F1 -a -f $F2 ]
> then
>       echo "$F1 non esiste, ma $F2 esiste"
> fi
$
```

Potete combinare questi operatori in espressioni molto complesse e raggruppare gli operatori complessi tra parentesi, se occorre. In pratica, comunque, è consigliabile mantenere i comandi **test** relativamente semplici.

8.5 Il comando exit

Potete utilizzare il comando **exit** nei file di comandi per terminare immediatamente la loro esecuzione. Il comando **exit** ha un argomento che diventa il valore che la procedura restituisce allo shell chiamante.

```
$ cat cmd4.file
if true
> then
>       exit 6
> fi
$ sh cmd4.file
$ echo $?
6
$
```

Quando utilizzate i valori di ritorno dai file di comandi, valgono le abituali convenzioni: i comandi che terminano correttamente restituiscono il valore

zero, mentre valori diversi da zero codificano possibili diverse condizioni di errore.

8.6 Il comando expr

Anche se il linguaggio di programmazione in ambiente shell non si adatta perfettamente alle elaborazioni numeriche, sono previsti anche comandi che possono eseguire calcoli. Tra questi, il più importante è il comando **expr** (expression) che ha come argomenti operatori aritmetici e numeri, li elabora e restituisce il risultato sullo standard output.

```
$ expr 4 + 5
9
$
```

Potete utilizzare comandi più complessi, separando ogni sezione dell'espressione da caratteri spazio, perché **expr** considera ogni argomento come una parte dell'espressione da valutare. È possibile specificare numeri interi, oltre agli operatori + (più), − (meno), * (moltiplicazione), / (divisione) e % (resto); gli operatori * e / devono essere preceduti dal carattere \ per impedire allo shell di interpretarli prima che **expr** li valuti.

```
$ expr 3 \* 4 + 2 \/ 2
13
$
```

Valgono le abituali regole di precedenza aritmetica, per cui le espressioni 3*4 e 2/2 vengono valutate prima dell'operatore +. Non potete alterare le regole di precedenza sulla linea di comando, ma potete facilmente suddividere calcoli complessi in operazioni separate.

```
$ VAL=`expr 3 \* 4 + 2`
$ expr $VAL \/ 2
7
$
```

Questo esempio mostra come il comando **expr** viene generalmente utilizzato nei file di comandi; le variabili di ambiente assumono valori numerici che successivamente altri comandi **expr** o anche operazioni di **test** impiegano.

In aggiunta, potete raggruppare operazioni dentro parentesi, purché protette, per esempio:

```
$ expr 3 \* \( 4 + 3 \) \/ 2
10
$
```

Il comando **expr** può anche eseguire *operazioni logiche*, ossia determinare se un argomento è *uguale a, maggiore di, minore di* un altro. Sono ammessi gli operatori = (uguale), != (diverso), > (maggiore di), < (minore di), >= (maggiore o uguale), <= (minore o uguale); questi operatori devono essere protetti perché molti contengono caratteri di interpretazione speciale in shell, per esempio:

```
$ X='expr 4 \< 5'
$ echo $X
1
$ X='expr 4 = 5'
$ echo $X
0
$
```

Se gli argomenti di questi operatori logici sono numerici, **expr** esegue comparazioni numeriche; se gli argomenti sono stringhe **expr** esegue comparazioni in base alla sequenza di ordinamento o *collating sequence* di stringhe nell'alfabeto ASCII.

8.7 L'operatore for

Altri operatori nel linguaggio di programmazione shell definiscono i costrutti iterativi che permettono di eseguire ripetutamente una parte del file di comandi. Tra questi operatori uno dei più rappresentativi è **for**, la cui sintassi è:

$ **for** *var* **in** *val1 ... valn* ; **do** *comandi* ; **done**

dove *var* è una variabile di ambiente così definita e *val1,valn* costituisce il generico elemento di una lista, ogni elemento della quale è separato dall'altro da un carattere spazio. Per esempio:

```
$ for VAL in 1 2 3 4 ; do comandi ; done
```

Quando lo shell interpreta questo comando, assegna a VAL il primo elemento della lista e poi esegue i comandi specificati tra **do** e il corrispondente **done**. Lo shell poi assegna a VAL il secondo elemento della lista ed esegue di nuovo i comandi indicati, ripetendo questo procedimento fino a quando termina di esaminare tutti gli elementi della lista. Il valore di VAL è disponibile all'interno della lista di comandi:

```
$ for VAL in 1 2 3 4
> do
>         echo $VAL
```

```
> done
1
2
3
4
$
```

Il numero di comandi che potete specificare tra i delimitatori **do..done** non è soggetto a particolari limitazioni e potete anche inserire strutture di controllo **if..fi** oppure altre strutture iterative **for**.

La costruzione **for** può essere usata per eseguire operazioni iterative su file; poiché il comando **for** viene interpretato dallo shell, viene eseguita la usuale espansione dei metacaratteri, per esempio:

```
$ for FILE in * ; do echo $FILE ; done
```

Poiché il metacarattere * viene rimpiazzato dai nomi di tutti i file, questo comando riporta su video il nome di tutti i file presenti nel directory corrente.

Un modo, particolarmente inefficiente, per contare i file contenuti in un directory, può essere il seguente:

```
$ COUNT=0
$ for FILE in *
> do
>         COUNT=`expr $COUNT + 1`
> done
$ echo $COUNT
14
$
```

La variabile di ambiente che segue il comando **for** deve essere presente alla sola funzione di specificare il numero di iterazioni che il comando **for** deve effettuare, non è obbligatorio usarla altrimenti.

Anche lo standard output di un comando può creare la lista di elementi per il comando **for**, per esempio:

```
$ for VAR in `ls`
> do
>         echo $VAR
> done
```

Il comando è accettabile perché lo shell esegue il comando **ls**, genera l'uscita all'interno del comando **for** e poi esegue l'intero comando.

Il comando **for** ha molti utilizzi, sia nei file di comandi sia direttamente da terminale, perché vi permette di realizzare un ciclo iterativo su una lista di oggetti, eseguendo un certo numero di comandi su ogni elemento della

lista. In conclusione, possiamo affermare che il comando **for** è uno degli operatori shell più frequentemente utilizzati.

8.8 L'operatore while e until

L'operatore **while** compendia alcune caratteristiche di **for** e alcune caratteristiche di **if**. Si compone di un comando **test** e poi di una sezione **do..done**. Se la sezione **test** fornisce esito positivo (true), viene eseguita la sezione **do..done**; se la sezione **test** fornisce esito negativo (false), non viene eseguita la sezione **do..done** e il ciclo termina. Al termine dell'esecuzione della sezione **do..done**, il comando **while** esamina nuovamente la sezione **test** e il ciclo termina quando l'esito di questa verifica è negativo (false), per esempio:

```
$ while [ ! -f file ]
> do
>         echo Creazione di un file
>         touch file
> done
```

Questo esempio rappresenta il tentativo di creare un file, ripetuto fino a quando l'operazione ha esito.

Dovete inserire all'interno della sezione **do..done** alcuni comandi per modificare il risultato del test, altrimenti il ciclo continua indefinitamente! Questo esempio crea dieci file, denominati **file1**, **file2** fino a **file10**.

```
$ VAL=1
$ while [ $VAL -lt 11 ]
> do
>         touch file$VAL
>         VAL=`expr $VAL + 1`
> done
```

Questo comando può essere scritto in un altro modo:

```
$ for VAL in 1 2 3 4 5 6 7 8 9 10
> do
>         touch file$VAL
> done
```

Potete inserire qualunque comando, tra cui altre strutture iterative **while**, all'interno di una sezione **do..done**.

Potete sostituire la parola chiave **while** con **until** per invertire il senso del test. In altre parole, la struttura **while** esegue ripetutamente i comandi specificati nella sezione **do..done** quando la parte condizionale è vera, mentre la struttura **until** esegue i comandi se la parte condizionale è falsa.

In un file di comandi, è buona norma indentare ogni sottolivello di comandi con un carattere di tabulazione, come abbiamo fatto nei precedenti esempi; ciò aiuta a strutturare logicamente il programma e favorirne la leggibilità rispetto alla scrittura con tutti i comandi allineati al margine sinistro, o subito dopo il prompt PS2. L'indentazione dei comandi non influenza il modo con cui lo shell interpreta il programma, ma aiuta notevolmente gli utenti a comprendere il senso del programma.

8.9 L'operatore case

Lo shell definisce l'operatore di controllo **case** che svolge una funzione analoga alla sequenza di comandi **if-elif-elif-...elif-fi**. Specificata una stringa di caratteri, **case** stabilisce a quale categoria, tra quelle indicate, la stringa appartiene e quindi esegue l'insieme di comandi che sono associati a quella categoria; le categorie non riconosciute sono ignorate. Il formato di una istruzione **case** è il seguente:

```
$ case $VAR in
>       caso1 )
>           lista-comandi
>           ;;
>       caso2 )
>           lista-comandi
>           ;;
>       caso3 )
>           lista-comandi
>           ;;
> esac
```

Il numero di casi e di comandi della struttura **case** non è soggetto ad alcuna limitazione, ma esistono precise regole sintattiche. Innanzitutto, la lista associata a ogni caso deve avere come delimitatore di chiusura lo speciale operatore ;;. Inoltre, ogni caso deve differenziarsi dagli altri e deve essere chiuso con l'operatore) (parentesi tonda chiusa). VAR rappresenta una variabile di ambiente o qualunque espressione che equivale a una stringa di caratteri; l'intera struttura di controllo deve terminare con la parola chiave **esac** (contrario di case). Come al solito, potete annidare in una struttura **case** qualunque altro operatore shell.

Il seguente comando visualizza un messaggio di saluto agli utenti della macchina.

```
$ case $LOGNAME in
>       leo)
```

```
>                   echo Ciao Leo, bentornato
>                   echo dalla tua vacanza
>                   ;;
>         pat)
>                   echo Pat, non dimenticare di leggere la posta
>                   ;;
>         giorgio)
>                   echo Cancella alcuni file, stai usando
>                   echo troppo spazio su disco.... Grazie!
>                   ;;
> esac
```

Una procedura di questo tipo generalmente verrebbe memorizzata in un file ed eseguita dagli utenti quando accedono al sistema; oppure potrebbe essere inserita come una parte del **profile** di login che tratteremo nel prossimo paragrafo.

La sezione *caso* della struttura di controllo **case** permette di utilizzare le espressioni regolari definite con i metacaratteri di shell. Nel precedente esempio potevamo specificare **gior***, ***r***, **?iorgio** o un'altra forma che rappresenta la stringa **giorgio**. Inoltre, potete utilizzare il carattere | (pipe) per indicare nella sezione *caso* un OR logico, come indicato qui di seguito:

```
$ case $LOGNAME in
>         giorgio|leo)
>                   echo non dimenticarti l'incontro di oggi
>                   ;;
>         *)
>                   echo Benvenuto nel nostro sistema UNIX
>                   ;;
> esac
```

Nel primo caso, il messaggio è diretto a **giorgio** o a **leo**, mentre nel secondo caso il metacarattere * individua qualsiasi stringa e viene utilizzato per inviare un messaggio di benvenuto a tutti gli altri utenti.

Il caso *) finale può essere usato in genere come caso di default, che raggruppa tutto quanto non associato ai casi definiti. Tenete presente che l'esecuzione del comando **case** implica l'esecuzione solo della sezione che per prima soddisfa ai criteri di associazione di **case**. Se nessun caso soddisfa questi criteri, l'istruzione **case** viene ignorata.

8.10 Il comando printf e l'output dei file di comandi

Il principale strumento per ottenere un output da una procedura di shell è il comando **echo**, usato frequentemente negli esempi precedenti. Costituisce un mezzo veloce e facile per inviare in output semplici stringhe di caratteri e valori di variabili di ambiente.

Per formati più complessi di output di caratteri è disponibile il comando **printf**, che fornisce un sottoinsieme orientato alla stringa delle possibilità del sottoprogramma di linguaggio C **printf(3)**.

Il comando **printf** prevede due tipi di argomenti. Il primo è una *specifica di formato*, una stringa di caratteri che specifica il formato generale dell'output e può includere speciali sequenze che verranno sostituite dai dati reali. Il secondo tipo di argomenti in **printf** è una lista di stringhe; devono esistere altrettante stringhe quante sono le sequenze speciali nella specifica di formato. Le sequenze speciali sono dette *specifiche di conversione* e definiscono il modo di stampa delle variabili; per esempio:

```
$ printf "questa è una stringa: %s\n" 1234567890
questa è una stringa: 1234567890
$
```

Il primo argomento (protetto con ") contiene la specifica di formato; i caratteri normali, come **questa è una stringa:** sono semplicemente stampati. La sequenza speciale **%s** rappresenta una specifica di conversione; ciascuna stringa (o comando che risulta in una stringa) che segue la specifica di formato sostituisce in ordine la corrispondente specifica di conversione all'interno della specifica di formato. Argomenti stringa non usati sono ignorati da **printf**, ma specifiche di conversione non usate producono risultati imprevedibili. Molte versioni di **printf(1)** sono orientate solo alla stringa, altri tipi di conversione, come **%d** e **%c** normalmente non sono previsti.

Nella specifica di formato sono utilizzabili alcune sequenze particolari: **\n** genera un ritorno a capo, **\t** inserisce un carattere di tabulazione.

L'esempio precedente di **printf** appare solo una forma più complicata del comando **echo**, ma **printf** risulta utile quando occorre troncare una stringa di caratteri a una lunghezza ridotta oppure quando occorre inserire la stringa in un campo definito nella linea in output. Nelle specifiche di conversione potete inserire dei numeri tra **%** e **s** per controllare il posizionamento della stringa in output e specificare un'ampiezza minima del campo, per esempio:

```
$ printf "stringa in campo di 20 car.: >%20s<\n" 1234567890
stringa in campo di 20 car.: >          1234567890<
$
```

Per giustificare a sinistra la stringa, inserite – (meno) prima del valore numerico; per esempio:

```
$ printf "stringa giustificata a sinistra: >%-20s<\n" 1234567890
stringa giustificata a sinistra: >1234567890          <
$
```

Potete troncare la stringa includendo nella specifica di conversione un . seguito dal numero dell'ampiezza del campo in output, per esempio:

```
$ printf "stringa troncata in 5 car.: >%.5s<\n" 1234567890
stringa troncata in 5 car.: >12345<
$
```

Il comando **printf** viene spesso usato per troncare una stringa nell'assegnarla a una variabile di ambiente, per esempio:

```
$ XXX='printf "%.4s\n" "1234567890"'
$ echo $XXX
1234
$
```

Potete usare questo metodo nei vostri programmi in shell per ricavare sottostringhe da stringhe di maggior lunghezza.

8.11 I file di comandi .profile ed /etc/profile

Un significativo esempio di file di comandi, che ogni utente generalmente definisce e gestisce personalmente, è un file che il sistema esegue durante la fase di login per definire il profilo del sistema e degli utenti. Questo file di comandi – denominato *profile* – permette di adattare la sessione di login alle vostre esigenze e di dichiarare le variabili di ambiente di impiego generale. Richiama concettualmente il file AUTOEXEC.BAT in ambiente MS-DOS, eccetto che ciascun utente può avere la sua particolare versione di **profile**.

In realtà, quando entrate nel sistema, vengono eseguiti due file distinti. Il primo di questi è del sistema e imposta l'ambiente di lavoro in base ai diritti di accesso per il sistema; questo file è contenuto in **/etc/profile** e tutti gli utenti possono accedervi in lettura. Il file **/etc/profile** viene eseguito per primo, subito dopo che siete entrati nel sistema. Non ne parleremo diffusamente in questo testo, ma vi consigliamo di esaminarne il contenuto sul vostro sistema macchina. Quando il gestore del sistema cambia l'ambiente di lavoro per tutti gli utenti, aggiungendo, per esempio, un nuovo directory alla variabile PATH, la modifica viene realizzata modificando **/etc/profile**.

Il secondo profilo viene eseguito dal sistema, prima che lo shell visualizzi il primo prompt **$**, ma dopo avere eseguito **/etc/profile**; questo è il file **.profile** che si trova nel vostro home directory. Il comando **ls** generalmente non evidenzia la presenza di **.profile** nel vostro home directory, perché il carattere . (punto) impedisce al comando di visualizzarlo. Se invece specificate l'opzione **–a** (all)

```
$ ls –a
```

il comando visualizza **.profile** insieme a tutti gli altri file che iniziano con il carattere . contenuti nell'home directory. Potete utilizzare il vostro file **.profile** per adattare l'ambiente di lavoro alle vostre esigenze più di quanto sia possibile con il file **/etc/profile**, perché in **.profile** potete memorizzare i comandi di carattere personale e farli eseguire dal sistema quando vi collegate.

UN ESEMPIO DI FILE .profile

Il gestore del sistema quando definisce il vostro identificatore di login, vi assegna un semplice file **.profile**, che raramente soddisfa le vostre esigenze. Non appena acquistate confidenza con il sistema UNIX, probabilmente vi interessa adattare l'ambiente di lavoro alle vostre più precise necessità. La Figura 8.1 mostra un tipico file **.profile** che suggerisce i tipi di modifiche che vengono generalmente apportate all'ambiente di lavoro. Anche se solo alcune di queste modifiche sono necessarie, potete rendere in ogni caso la sessione più piacevole, soddisfacendo le vostre esigenze ed evitando errori nel sistema.

Innanzitutto la variabile **PATH** viene ridefinita per identificare il directory corrente e un file binario privato con precedenza su quello dei comandi di sistema; questo vi permette di sostituire alcuni normali comandi con i vostri comandi personali. Tenete tuttavia presente che questo costituisce un rischio per la sicurezza, se a tutti gli utenti viene permesso di accedere in scrittura ai directory, in quanto qualcuno di essi potrebbe inserire nel vostro directory una versione errata di un comando, sostituendola a quella di default. La variabile **PATH** contiene, all'inizio o alla fine, anche il directory corrente. Successivamente alle operazioni descritte, il comando **stty** configura il terminale per abilitare il controllo di flusso (XON/XOFF), per sostituire ove possibile i caratteri spazio con caratteri di tabulazione e per associare al tasto ← la funzione di cancellazione. L'opzione **echoe** induce il sistema a eseguire la sequenza *backspace-spazio-backspace* quando premete ←. Inoltre **.profile** permette di assegnare ai prompt PS1 e PS2 i simboli che preferite.

Il file **.profile** della Figura 8.1 definisce i terminali che potete utilizzare. Innanzitutto, la variabile di ambiente locale **PORT** viene impiegata nel comando **test** per determinare se vi trovate su una console di sistema o su un terminale remoto. Sulla console, **.profile** inizializza correttamente la variabile **TERM**, per cui questa lunga sezione **if..fi** non viene eseguita. Invece, se non state lavorando su console il file **.profile** richiede l'introduzione del tipo del terminale per predisporre i parametri adatti. Il comando

```
read TERM
```

attende che battiate una linea da tastiera e poi la memorizza nella variabile di ambiente **TERM**. La struttura di controllo **case** permette di eseguire se-

```
# Un tipico esempio di .profile
# Può essere modificato secondo le esigenze

# Cambiamo PATH per collocare i directory privati prima dei bin di sistema
# Nota: cambiare PATH può comportare un rischio per la sicurezza
PATH = :$HOME:$HOME/bin:/sbin:/usr/sbin:/usr/bin

# configuriamo nuovamente il terminale con tab, XON/XOFF,
# e "spazio - carattere di backspace - spazio" per la cancellazione
stty tabs echoe erase ^h ixon ixoff ixany

# modifica dei prompt...
PS1 = "[!] 'uname'>"
PS2 = ">>"

# Esame del tipo di terminale e trattamento diverso per ogni
# terminale... Se non siamo su console, possiamo saltare
# questa fase...
PORT = 'tty'
PORT = 'basename $PORT'
if [ $PORT != console ] ; then

        # richiesta del tipo di terminale della sessione e memorizzazione
        # dell'identificatore in una variabile di ambiente
        echo "\nTERM = \c"
        read TERM

        case $TERM in
                s*)     # AT&T UNIX PC
                        if [ $TERM = ss ] ; then styy cr2 nl1 ; fi
                        TERM = ansi
                        stty -tabs
                        ;;

                pc|2621)        # HP 2621 o emulatore di terminale
                        TERM = 2621
                        tabs -Thp
                        ;;

                ansi|v*)        # DEC VT100, VT102 o VT200
                        TERM = vt100
                        tabs
                        ;;

                *)      # tutti gli altri vengono considerati
                        # terminali non intelligenti
                        TERM = dumb
                        stty -tabs
                        ;;
        esac
fi
```

Figura 8.1 Un esempio di file **.profile.**

```
# ripete l'identificatore di terminale utilizzato
echo "TERM = $TERM"

# questa istruzione riguarda l'editor vi
EXINIT = 'set ai'
# Specifica del nome dell'editor impiegato
EDITOR = /usr/bin/vi
EDIT = /usr/bin/vi

# Per i comandi "cd"
CDPATH = :$HOME:$HOME/bin

# È opportuno che siano tutte variabili globali
export PATH TERM TERMCAP PS1 PS2 EXINIT EDITOR CDPATH EDIT

# visualizziamo la data e l'ora
date

# visualizziamo le notizie non ancora esaminate
news

# indichiamo se esiste posta arretrata in "$HOME/mbox"
if [ -f mbox ] ; then echo "You have oldmail" ; fi

# a uso del Korn shell
FCEDIT = /usr/bin/vi
VISUAL = $FCEDIT
MAILCHECK = 60
ENV = "\${-:+$HOME/.ksh.aliases\${- # # *i*}}"
HISTFILE = $HOME/.ksh.history
export ENV HISTFILE FCEDIT VISUAL MAILCHECK

# alla fine, passiamo dallo shell di default a ksh
exec /usr/bin/ksh
```

Figura 8.1 Un esempio di file **.profile** (*continua*).

zioni di codice differenti a seconda di ciò che avete inserito da tastiera, per cui inizializza correttamente la variabile **TERM** per ogni differente terminale. Il comando **tabs** invia le relative sequenze di escape per inizializzare i caratteri di tabulazione al terminale, nel caso il terminale non possa ricordare i valori di tabulazione quando si toglie la tensione. Il comando **tabs**, impiegato nella linea di comando **stty tabs**, permette di ottimizzare la dimensione dell'uscita su terminale, sostituendo i caratteri spazio con caratteri di tabulazione.

La variabile **TERM** può assumere un valore diverso a seconda di quale caso della struttura di controllo **case..esac** viene eseguito; insieme a questa, vengono inizializzate altre variabili di ambiente: **EXINIT**, che viene utilizzata dall'editor **vi**, **EDITOR** e **EDIT**, che indicano quale sia l'editor più adat-

to a un particolare tipo di software. Tra i diversi tipi di software, anche la funzione di utilità help utilizza queste variabili di ambiente. Esiste inoltre **CDPATH**, cioè una progredita funzione shell di utilità che permette di definire una breve e intelligente forma del comando **cd**, trattata alla fine di questo capitolo. Infine **.profile** garantisce ai vostri subshell l'accesso a tutte queste variabili di ambiente, a cui assegna una visibilità globale con **export**. Dopo queste operazioni, il file **.profile** esegue un piccolo lavoro di tipo gestionale. Il comando **date** visualizza la data e l'ora correnti, mentre **news** elenca le ultime notizie non ancora visionate. La breve struttura **if..fi** che segue vi ricorda la posta che avete in giacenza; normalmente dovreste aver cancellato la posta che avete evaso, quindi la presenza del file **mbox** indica probabilmente la presenza di qualche attività in sospeso.

Infine, **.profile** inizializza altre variabili di ambiente associate a **ksh**, uno shell alternativo a quello di default, che la procedura lancia con il comando **exec /usr/bin/ksh**.

Il vostro **.profile** probabilmente differisce da quello che abbiamo mostrato in questo paragrafo, ma la maggior parte delle modifiche che apportate all'ambiente di lavoro e al processo di login, per adattarli alle vostre esigenze, fa parte del vostro file **.profile**.

L'OPERATORE .

Normalmente **.profile** viene automaticamente eseguito al momento della vostra operazione di login nel sistema, tuttavia, mentre state scrivendo e provando il vostro **.profile**, risulta estremamente tedioso entrare e uscire ripetutamente nel sistema per fare gli esperimenti necessari. Poiché i subshell eseguono particolari script di shell, i cambiamenti che avete apportato alle variabili di ambiente all'interno del subshell corrente non si mantengono a livello di shell padre, per cui non potete eseguire **.profile** come qualunque altro file di comando, nonostante sia un normale file eseguibile. Dunque, gli effetti che **.profile** può apportare, come per esempio la modifica della variabile **TERM**, non generano modifiche permanenti al vostro ambiente se viene eseguito con il seguente comando:

```
$ sh .profile
```

Lo shell definisce un altro operatore che permette di eseguire un file mantenendo attivo l'ambiente dello shell corrente, evitando di creare un subshell che esegua espressamente il file. Si tratta del comando . (punto o dot) che accetta come argomento il nome del file che volete eseguire.

```
$ . .profile
```

Qualunque file di comandi è un argomento valido, ma il comando . viene generalmente utilizzato con **.profile** per le prove, o per caricare un file di que-

sto tipo quando un subshell esegue un altro file di comandi. I cambiamenti apportati all'ambiente di esecuzione con l'operatore . aggiornano permanentemente l'ambiente corrente (e i subshell che possono essere creati successivamente), ma ancora non modificano gli shell esistenti di livello superiore.

8.12 Argomenti della linea di comando

Rimangono da discutere molti altri aspetti relativi alla programmazione di shell, tra cui gli *argomenti della linea di comando*. Potete scrivere programmi di shell che processano gli argomenti della linea di comando con le stesse modalità utilizzate dagli altri comandi, se negli argomenti dei vostri comandi mantenete la sintassi abituale. A questo scopo lo shell prevede diversi operatori per trattare gli argomenti di linee di comando che abbiano una struttura usuale.

I PARAMETRI $#, $* E I PARAMETRI POSIZIONALI

Abbiamo già parlato delle variabili di shell $# e $*, che lo shell interpreta rispettivamente come il *numero* di parametri della linea di comando per un file di comando e i *valori* di tutti i parametri. In questa procedura

```
echo $#
for VAR in $ *
do
        echo $VAR
done
```

lo shell espande la variabile $* per rappresentare tutti gli argomenti della linea di comando.

Se il nome del precedente file di comandi è ECHO.ARG, potete eseguirlo con un comando del tipo:

```
$ echo.arg primo secondo terzo
```

in cui gli argomenti **primo**, **secondo** e **terzo** vengono passati al file di comandi.

```
$ echo.arg primo secondo terzo
3
primo
secondo
terzo
$
```

Esaminando l'output, il 3 viene generato dal comando **echo $#**, mentre la parte restante dell'output prodotto dal comando **echo.arg primo secondo terzo** proviene dall'esecuzione del ciclo **for..do..done**.

Potete inoltre individuare un solo argomento della linea di comando mediante la notazione **$1**, **$2**, **$3**, ecc., fino a un massimo di nove argomenti. Per esempio, il file di comandi appena visto può assumere anche questa forma:

```
echo $#
echo $1
echo $2
echo $3
```

e i risultati che si ottengono sono gli stessi. È possibile inoltre utilizzare questi argomenti per passare, dalla linea di comando, nomi di file o altre informazioni in un file di comandi.

La loro utilità di impiego si riscontra di solito solo nei file eseguiti come comandi, poiché lo shell di login non ha argomenti, per cui

```
$ echo $#
0
$
```

Per il vostro shell di login non sono definiti argomenti di linea di comando, con una sola eccezione: **$0** è un argomento legittimo, come lo è **$2**, e restituisce il nome del comando corrispondente.

```
$ echo $0
sh
$
```

Il primo argomento della linea di comando è **$1**.

Inoltre, lo shell definisce un operatore per inizializzare questi argomenti della linea di comando.

```
$ echo $#
0
$ echo $2

$ set salve ciao terzo
$ echo $#
3
$ echo $2
ciao
$
```

L'operatore **set** viene spesso utilizzato nei file di comandi e assegna i suoi argomenti ai *parametri posizionali*, per cui i file di comandi possono utiliz-

zare gli identificatori **$2**, ecc., anche se non specificate alcun argomento della linea di comando. Per esempio, potete scrivere un file che utilizza i parametri della sua linea di comando, ma se non li specificate, il file può inizializzarli a valori di default.

8.13 Errori e messaggi d'errore nelle procedure di shell

Ovviamente lo sviluppo di un programma di shell richiede varie prove e correzioni per raggiungere il risultato voluto. Se i file di comandi contengono degli errori formali, lo shell fornisce brevi messaggi diagnostici.

```
$ for VAR in 'ls' do
> salve
syntax error: 'salve' unexpected
$
```

L'errore che abbiamo commesso in questo esempio riguarda l'omissione del carattere ; o del carattere ritorno a capo prima della parola chiave **do**, ma il messaggio di errore dello shell non fornisce indicazioni precise. Lo shell può comunque fornire una diagnostica più estesa, dato che procedure di shell vengono eseguite più frequentemente da file che da terminale. Potete seguire lo svolgimento di un programma di comandi da un file con il comando **sh** **−x**, come mostra la Figura 8.2. Viene fornita una traccia di esecuzione, che visualizza ogni comando eseguito, preceduto dal carattere **+**, dopo che lo shell ha eseguito le sostituzioni e trattato gli operatori, come **test** e altri operatori. L'esame della traccia dello svolgimento della prova rende molto più semplice esaminare il corso dell'esecuzione e individuare la condizione che conduce all'errore. Un altro aiuto per la messa a punto è offerto dall'opzione **−v** (verbose), che visualizza, sempre in fase di esecuzione, informazioni dettagliate su ogni comando presente nel file di comandi, per esempio:

```
$ sh −v shell.script primo secondo terzo
echo $#
3
for VAR in $* ; do
        echo $VAR
done
primo
secondo
terzo
$
```

Le opzioni **−x** e **−v** possono essere usate assieme, per ottenere un più completo output diagnostico; questo viene inviato in standard error, potete salvarlo quindi in un file separato dal normale output.

```
# cat shell.script
echo $#
for VAR in $* ; do
        echo $VAR
done
$ sh -x shell.script primo secondo terzo
+ echo 3
3
+ echo primo
primo
+ echo secondo
secondo
+ echo terzo
terzo
$
```

Figura 8.2 Traccia (**-x**) prodotta dall'esecuzione di una procedura di shell.

8.14 Approfondimenti

In questi primi capitoli, non abbiamo sicuramente trattato tutte le modalità per scrivere e utilizzare i programmi di shell. Per imparare la programmazione con lo shell è indispensabile fare molta pratica con file di brevi dimensioni inserite direttamente da terminale, ma è anche interessante e utile leggere e riadattare i file di programmi di altri utenti.

Alcuni comandi sono un valido esempio di procedure di shell e generalmente risulta difficile affermare con certezza se un comando è un programma binario compilato o una procedura di shell accessibile in esecuzione. Nella maggior parte dei sistemi i comandi **/usr/bin/calendar**, **/usr/bin/spell**, **/usr/bin/basename**, e molti altri, sono procedure di shell, che potete esaminare attentamente per capire il loro funzionamento.

shar: UN ESEMPIO DI FILE DI COMANDI

La Figura 8.3 mostra un esempio di moderata complessità, che genera in uscita un'altra procedura di shell. Si tratta di una semplice versione del comando **shar** (shell archive), la cui funzione è di creare un file che contenga diversi altri file al suo interno. Una utile applicazione di questo comando si ha nelle operazioni di trasferimento di file tra macchine diverse tramite posta elettronica o il comando **uucp**, in cui minore è il numero di file da trasferire, minori sono le difficoltà.

Il comando **shar** non ha le stesse potenzialità che si riscontrano in altri strumenti di archiviazione di file perché tratta solo file di testo e non file

```
#! /sbin/sh
# shar: shell archive shell script
# esegue in /sbin/sh
# shar: raggruppa file nel formato di distribuzione "shar"
#           adatto per estrazione in ambiente sh o ksh, non csh.

# argomenti della linea di comando non dichiarati
# ci deve essere almeno un argomento
if [ $# -lt 1 ] ; then
        echo "uso: shar file1 file2 file3 ... fileN"
        exit 1
fi

# ora iniziamo a creare il file di uscita in standard output.
# Innanzitutto una intestazione generale....
echo '# NON USARE csh per disimpaccare questo file'
echo '# Formato SHAR. Archivio creato "date'

# viene trattato ciascun file: "$*" è la lista di argomenti
for file in $* ; do

        # se il file non esiste, visualizziamo un messaggio,
        # ma non usciamo
        if [ ! \( -r $file -a -f $file \) ] ; then
                echo shar: "Non è possibile archiviare $file" > /dev/tty
                continue
        fi

        # il file viene trovato, scriviamo in standard output il nome
        echo "echo x - $file"

        # scriviamo in standard output la linea di comando "sed", non eseguiamo sed
        echo "sed 's/^X//' > $file << '"SHAR_MARK"'"

        # Qui eseguiamo il comando "sed" per avere 'X' come prefisso
        # su ogni linea inviata in standard output.
        sed 's/^/X/' $file

        # a fine file scriviamo in standard output il marcatore di fine dati
        echo "'"SHAR_MARK"'"

        # eseguiamo il comando "ls" sul file e serbiamo il risultato in un echo
        echo "echo "ls -l $file' (come inviato)"'

        # esaminamo i diritti di accesso in "ls -l", creiamo un
        # comando "chmod" e lo scriviamo in standard output
        ls -l $file | sed \
                -e 's/^.\(...\)\(...\)\(...\).*/u=\1,g=\2,o=\3/' \
                -e 's/-//g' \
```

Figura 8.3 Procedura di shell **shar**.

```
        -e 's/.*/chmod & "$file/"

# un comando "ls" per la procedura di uscita
echo "ls -l $file"

done # fine del ciclo

echo "exit 0" # comando exit in standard output
```

Figura 8.3 Procedura di shell **shar** (*continua*).

binari o compilati, ma è ampiamente utilizzato per scambiare dati tra sistemi UNIX. In pratica, tramite il comando **shar**, potete creare archivi che contengono diversi file sorgente di limitate dimensioni e inviarli poi ad altri utenti che li disimpaccano (*unshar*) ossia separano e ripristinano i file originali. Come tutti i buoni programmi di comunicazione, il programma **shar** esegue alcuni controlli per scoprire se una parte dei dati è stata persa nel corso delle operazioni.

Per eseguire il programma **shar**, utilizzate la seguente linea di comando:

$ shar *lista-nomifile* > *file-temp*

Il comando legge i file indicati nella lista e riporta il materiale archiviato nello standard output, che viene generalmente ridiretto a qualche altro file, probabilmente in un directory diverso da quello che state trattando con il comando **shar**.

Il file prodotto da questa procedura è in realtà un file di comandi eseguibile e potete ripristinare i file originali dall'archivio con il seguente comando:

```
$ sh /tmp/out.file
```

oppure con

```
$ chmod +x /tmp/out.file
$ /tmp/out.file
```

Il comando **shar** fornisce quindi un metodo per impaccare file in una procedura di shell e il programma indicato in Figura 8.3 richiede un accurato studio. Ne diamo brevemente una descrizione.

Le prime linee sono semplici commenti; dovete sempre inserire abbondanti commenti in tutti i programmi per favorirne la comprensione. Le linee successive verificano se il comando ha come argomenti nomi di file e in caso negativo **shar** visualizza un messaggio di errore ed esce.

Il vero programma inizia in corrispondenza dei comandi **echo**; la procedura di shell **shar** esegue alcuni comandi e ne scrive degli altri sullo stan-

dard output con le istruzioni **echo**. Dopo la visualizzazione di alcuni messaggi, inizia il ciclo principale, in cui l'operatore di shell **$*** definisce la lista di argomenti per la procedura, il cui numero coincide con quello specificato da **$#**. In particolare, **$*** rappresenta gli argomenti reali, separati da caratteri spazio, mentre **$#** indica solo il numero di argomenti. All'interno del ciclo, **shar** verifica se ogni file è realmente presente. La sezione **test** restituisce un valore positivo (true) se il file non è leggibile oppure se non esiste. Abbiamo utilizzato le parentesi per delimitare gli operatori di **test** e il carattere \ che le precede evita una errata interpretazione da parte dello shell. Se queste verifiche non trovano il file, **shar** visualizza un messaggio di errore; per evitare che entri nell'archivio che si sta costruendo, l'uscita di **echo** viene ridiretta direttamente al terminale con **> /dev/tty**.

Le linee successive non sono banali, in quanto generano alcuni comandi eseguibili nella procedura in uscita; infatti **shar** genera in uscita le informazioni per includere il file reale all'interno della procedura (vedi here document). Il comando **sed** appone **X** all'inizio di ogni linea della procedura prodotta che contiene dati di file; la sequenza di dati viene chiusa dalla linea **echo +SHAR+MARK+**.

Le successive poche linee rimanenti realizzano la verifica del corretto trasferimento dei file. Prima, **ls -l** invia in uscita la dimensione del file originale, poi la linea **echo ls -l $file** che si trova quasi in fondo alla procedura causa l'esecuzione del comando **ls** dalla parte ricevente. Deve essere cura del ricevente verificare se i due file hanno la stessa lunghezza. Il complicato comando **sed** trasforma l'uscita di **ls -l** in un comando **chmod** per inizializzare correttamente i diritti di accesso sul file ricevente, viene inviato infine in uscita anche un comando **exit**.

Quando **shar** viene eseguito, inserisce i file specificati nella sua lista di argomenti in una nuova procedura di shell eseguibile. Potete sperimentare sulla vostra macchina il funzionamento di questa procedura, magari cercare di migliorarla dal momento che non tratta file allocati nei subdirectory.

IL COMANDO getopts

Abbiamo trattato finora come argomenti delle procedure di shell i nomi di file o altri argomenti in input, tuttavia la normale sintassi di linea di comando ammette anche le opzioni, cioè flag preceduti dal segno - (meno), che modificano il comportamento abituale del comando. Anche le procedure di shell possono accettare flag di opzioni nella linea di comando. L'operatore di shell **getopts** (get options) facilita il lavoro di analisi delle opzioni, in quanto la collocazione di flag sulla linea di comando non deve essere obbligata da criteri posizionali, ed inoltre ad alcuni flag possono essere associati altri argomenti. L'operatore **getopts**, disponibile nei sistemi SVR3 e SVR4, costituisce l'evoluzione del precedente **getopt**, che deve essere ancora utilizzato nelle versioni meno recenti di UNIX. L'operatore permette di creare

procedure di shell che possono esaminare gli argomenti della linea di comando come altri comandi eseguibili. In particolare, dopo aver trattato opzioni e flag della linea di comando, alla fine, vi consente di utilizzare la funzione **set** per assegnare i rimanenti argomenti della linea di comando (non flag, non opzioni) alle usuali notazioni **$1**, **$2**, **$3**, ecc.

La Figura 8.4 mostra un esempio di procedura di shell che utilizza la funzione **getopts**. La seguente linea:

```
while getopts yz:x VAR
```

definisce un ciclo di programma che usa la funzione **getopts**, con associati due argomenti. Il primo rappresenta un elenco di lettere di opzioni accetta-

```
#! /sbin/sh
# Innanzitutto analizziamo le opzioni, assegnando le
# variabili di shell per indicare quali opzioni sono state
# definite.
while getopts yz:x c
do
        case $c in
                x)
                        XFLAG = true
                        ;;
                y)
                        YFLAG = true
                        ;;
                z)
                        ZFLAG = true
                        ZOPT = $OPTARG      # OPTARG è il valore corrente
                        ;;
                *)
                        echo $USAGE
                        exit 2
                        ;;

        esac
done

# ora abbiamo esaminato tutte le opzioni, riduciamo la lista di
# argomenti solo a quelli rimanenti
shift 'expr $OPTIND - 1'

# XFLAG, YFLAG o ZFLAG, possono essere state inizializzate secondo
# gli argomenti presenti sulla linea di comando.
# ZOPT viene inizializzata a "str" se è stata data l'opzione "-z str"
# Da qui può seguire il resto della procedura di shell voluta
```

Figura 8.4 Uso dell'operatore **getopts**.

bili, non contenente caratteri spazio. In questa lista, ogni lettera che richiede un argomento è seguita dal carattere : (due punti). Il secondo argomento – cioè VAR nell'esempio – è una variabile di ambiente temporanea contenente il valore dell'opzione corrente della linea di comando, che viene assegnato all'interno del ciclo **while**. Questo ciclo viene eseguito per ogni argomento che compare sulla linea di comando e, quando tutti gli argomenti sono stati esaminati, **getopts** restituisce il valore falso e il ciclo **while** termina.

Sono ammesse le opzioni **–x**, **–y** e **–z**, e l'opzione **–z** deve essere seguita da un argomento. Supponendo che il nome della procedura sia **script**, le seguenti sono linee di comando corrette, fra tante.

```
$ script
$ script -x
$ script -y -x
$ script -z salve
$ script -x -z salve
$ script -z salve -y
```

La funzione **getopts** definisce la variabile **USAGE** che riporta le opzioni ammesse nel comando **getopts** inserito nel ciclo **while**. La variabile di shell **OPTARG**, anch'essa definita dalla funzione **getopts**, viene inizializzata al valore dell'argomento che segue l'opzione corrente. In questo esempio, **OPTARG** viene definita solo quando viene trattata l'opzione **–z**, in quanto la stringa di controllo per **getopts** presenta il carattere : (due punti) solo dopo l'opzione **–z**.

All'interno del ciclo **while**, potete trattare ognuno degli argomenti della linea di comando, inizializzando alcune variabili oppure operando secondo le modalità opportune per ciascun argomento dato. Per le opzioni che accettano argomenti propri, come l'opzione **–z** nell'esempio, dovete utilizzare un'apposita variabile di ambiente per memorizzare il valore dell'argomento, perché la funzione **getopts** definisce **OPTARG** solo mentre una particolare opzione della linea di comando diventa l'opzione corrente nel ciclo **while**. Dopo che tutti gli argomenti della linea di comando sono stati trattati all'interno del ciclo **while**, il comando **shift** modifica la variabile **$*** per rimuovere le opzioni già esaminate, per cui **$1** diventa il primo argomento della linea di comando non ancora trattato. Una volta che tutte le opzioni sono state esaminate, la procedura di shell può continuare l'esecuzione delle operazioni previste.

IL COMANDO trap

Normalmente un segnale come *hangup*, CTRL-D (terminate), o alcuni segnali di errore del sistema, inviati a una procedura di shell in esecuzione, causano l'immediata fine del programma e il ritorno allo shell chiamante. Usual-

mente questo è ciò che si vuole, ma talvolta lo shell potrebbe essere in un processo di scambio di alcuni file (*swapping*) o in un'altra operazione critica. In questi casi, è preferibile intrappolare (*trap*) il segnale e lasciare che la procedura di shell continui fino a completamento; in altri casi potrebbe essere preferibile *notificare* il segnale e lanciare alcune operazioni di ripristino prima di uscire dallo shell. Il comando **trap** risponde a questi scopi: può essere usato ovunque in una procedura di shell, ma non deve assolutamente essere usato direttamente in linea di comando. Il comando **trap** richiede come primo argomento una linea di comando protetta con ", seguita da un elenco di numeri di segnali. Quando viene ricevuto uno dei segnali nell'elenco, viene eseguita la linea di comando, quindi la procedura prosegue nell'esecuzione; se usando **trap** volete uscire dalla procedura, dovete includere esplicitamente **exit** nel comando protetto; per esempio:

```
trap "echo ricevuto un segnale!; exit" 1 2 15
```

La lista dei segnali disponibili e i significati relativi è disponibile nella pagina di manuale **signal(5)**, del *Programmer's Reference Manual.*

Non potete avere accesso ai numeri di segnale nella lista protetta dei comandi, ma potete includere nella vostra procedura tanti comandi **trap** quanti volete. Per annullare una trappola di segnale usate un altro comando **trap** per lo stesso segnale; se la stringa di comando è nulla, il segnale viene ignorato. Per esempio:

```
trap "" 1 2 15
```

I tre segnali vengono ignorati, rendendo difficile interrompere la procedura.

IL COMANDO wait

Talvolta potreste voler lanciare diverse applicazioni in background, ma sospendere ogni altra attività fino a che una o tutte le applicazioni siano concluse. Normalmente i lavori in background ritornano immediatamente allo shell, anche se la loro esecuzione può proseguire per lungo tempo. Potete usare il comando **wait** per informare lo shell che pur avendo lanciato il lavoro in background, volete rimanere in attesa fino a che il lavoro non viene completato. Eseguite **wait** con argomento il process id del lavoro in background di cui volete attendere il termine, per esempio:

```
$ cat /unix >/tmp/ss &
10009
$ wait 10009
```

Quando il lavoro termina, **wait** vi fa ritornare allo shell per un altro comando. Il comando **wait** senza argomenti resta in attesa del termine di tutti i vostri lavori in background. Il comando **wait** fornisce alle procedure di shell alcune possibilità di *job control.*

LIVELLI DI SHELL E COMANDO shl

I sistemi SVR3 e SVR4 forniscono uno strumento che permette a più shell di essere attive su un unico terminale; non si tratta di un sistema a più finestre come X Window System, perché ogni shell occupa l'intero video. Esistono strumenti per definire nuovi shell, per passare da uno shell a un altro e per eliminarli dopo averli utilizzati. La definizione dei diversi livelli di shell (*shell layer*) permette al sistema di considerare ogni singolo shell come una sessione indipendente; il sistema agisce come se vi foste collegati più volte alla macchina da terminali diversi. Ogni sessione, denominata livello (layer) di shell, viene trattata come un terminale virtuale e i diversi livelli non interagiscono. Poiché ogni livello occupa l'intero video, sono disponibili alcuni strumenti che permettono il passaggio da un livello a un altro quando desiderato. In questo ambiente, potete eseguire contemporaneamente più programmi, ognuno dei quali ha pieno accesso al terminale. Potete utilizzare in diversi modi il sistema a livelli di shell, limitatamente ai terminali **tty**; non può essere usato su alcune reti locali o con X Window System.

Per console di sistema è preferibile il comando **vtlmgr** (virtual terminal layer manager), se il vostro sistema lo supporta.

Il comando **shl** (shell layers) attiva il gestore di livelli (layer manager), che controlla l'accesso ai diversi livelli che avete definito. Quando lanciate il programma **shl**, appare su video un nuovo prompt per segnalare che non siete più in una sessione normale, bensì sotto il controllo del sistema **shl**. Per esempio:

```
$ shl
>>>
```

A questo punto **shl** si mette in attesa di un comando. Potete creare un nuovo livello, eliminarne uno esistente oppure chiudere completamente il gestore di livelli.

Per creare un nuovo livello, battete **create** dopo il prompt ">>>".

```
$ shl
>>> create
(1)
```

Il livello così definito assume un nome, generalmente il numero di livello, da 1 a 7, essendo 7 il massimo numero di livelli permessi. Il prompt PS1 di

ogni livello viene cambiato nel nome del livello corrente, per distinguere su video il livello corrente. Nell'esempio precedente, una volta che avete lanciato il programma **shl** e avete definito il nuovo livello di shell, lo shell di login si sospende e lo shell (1) si mette in attesa dei vostri comandi. Potete utilizzare questo livello proprio come se fosse uno shell normale e potete eliminarlo con il comando **exit** oppure premendo la combinazione di tasti CTRL-D. Quando eliminate un livello, il controllo torna al gestore di livelli che visualizza il prompt >>>; potete allora specificare a **shl** ulteriori comandi.

Quando il programma **shl** è in esecuzione, indipendentemente da quale livello sia attivo, potete ritornare al prompt >>> in qualunque momento risvegliando **shl** con la combinazione di tasti CTRL-Z. Quando **shl** viene risvegliato, visualizza immediatamente il prompt >>> e si mette in attesa di comandi. Questo uso di CTRL-Z annulla l'uso di CTRL-Z per il *job control* (trattato nel Capitolo 16). Potete creare un secondo livello utilizzando i seguenti comandi:

```
$ shl
>>> create
(1) CTRL-Z
>>> create
(2)
```

Ora disponete di due livelli di shell, e quello attivo è il livello (2). Possono essere attivi fino a 7 livelli contemporanei.

Potete creare un livello, lanciare una applicazione che produce alcuni risultati in uscita e poi creare un secondo livello. In questa situazione, l'uscita prodotta dal primo livello (e da tutti gli altri eventuali livelli attivi) va direttamente su terminale e si sovrappone all'uscita prodotta dal livello corrente. In questo modo, le applicazioni continuano la loro esecuzione nei livelli, scrivendo i risultati prodotti al terminale, indipendentemente dal numero di livelli attivi. Solo il livello attivo può accettare i comandi che battete sulla tastiera, gli altri livelli non sono bloccanti perché continuano l'esecuzione anche quando non sono attivi (nonblocking). Il sistema **shl** prevede i comandi **block** e **unblock** per modificare questo comportamento e rendere bloccante un livello quando non è attivo, per cui l'uscita prodotta da quel livello si interrompe fino a quando diventa corrente. Utilizzate

```
>>> block 2
```

per rendere il livello (2) bloccante e

```
>>> unblock 2
```

per rendere il livello (2) non bloccante.

Quando ritornate al programma **shl**, in qualunque momento premendo la combinazione di tasti CTRL-Z, oltre a **create**, sono disponibili diversi altri comandi la maggior parte dei quali accetta come argomento l'indicazione numerica del livello su cui agisce. Introducete **?** al prompt >>> per ottenere un elenco di tutti i comandi disponibili.

Potete passare liberamente da un livello attivo a un altro; premete CTRL-Z, attendete il prompt >>> e battete il comando **resume** con il numero di livello che volete diventi attivo; oppure, più semplicemente, introducete il nome del livello, senza la parola chiave **resume**, per esempio:.

```
(1) CTRL-Z
>>> 2
(2)
```

A questo punto è attivo il livello (2).

Per eliminare un livello, premete CTRL-D all'interno del livello, oppure introducete al prompt >>> il comando **delete**, indicando come argomento il numero di livello da cancellare.

```
(2) CTRL-Z
>>> delete 2
>>>
```

Infine, il comando **quit** disattiva il gestore di livelli, ripristinando lo shell di login, e tutti i programmi che sono attivi nei diversi livelli vengono cancellati.

La combinazione di tasti CTRL-Z utilizzata per risvegliare il programma **shl** viene anche chiamata carattere di *switch*; potete modificarla per adattarla alle vostre esigenze mediante il comando **stty**.

```
$ stty swtch c
```

dove *c* è il carattere scelto per sostituire CTRL-Z.

CONSOLE VIRTUALI

La *console virtuale* rappresenta un significativo adattamento del concetto di **shl**. Alcune versioni di SVR3 e SVR4 permettono di utilizzare i tasti funzione per passare direttamente da un livello a un altro, evitando il richiamo del comando **shl** e le relative procedure CTRL-Z. La funzionalità di console virtuale è limitata alla console di sistema, mentre **shl** può essere messo in esecuzione sia su terminali remoti che sulla console.

In alcune versioni, la pressione di un tasto funzione cancella lo schermo e genera un prompt **login:** che consente di collegarvi di nuovo alla macchina; in altre versioni, il login corrente viene preservato e premendo il tasto

funzione si attiva un nuovo shell. Una volta instaurata la sessione, potete attivarla immediatamente premendo il relativo tasto funzione.

Generalmente avviene un riaggiornamento dello schermo quando scambiate sessione, risolvendo uno dei maggiori inconvenienti del comando **shl**.

In alcune release, la console virtuale è sempre disponibile e non dovete attivarla lanciando un comando come nel caso di **shl**; in altre release dovete attivare la caratteristica col comando

```
$ vtlmgr
```

(virtual terminal layer manager).

Per scambiare le schermate quando la caratteristica **vtlmgr** è attiva, dovete premere ALT-SYSREQ (tenete premuto ALT mentre premete SYSREQ), seguito da un tasto funzione. I primi tasti funzione sono associati ai livelli corrispondenti e la pressione del tasto funzione attiva il relativo livello. Se il livello di console virtuale non esiste già, la sequenza ALT-SYSREQ tasto funzione crea la nuova console virtuale; potete anche usare il comando **newvt** per creare console virtuali. Ancora, potete eliminare un livello sia con CTRL-D che con il comando **exit**.

Sulla console di sistema potete creare un terminale virtuale, lanciare X Window System oppure una sessione DOS in una finestra, scambiare con una nuova finestra per lanciare un'altra sessione DOS; quindi potete scambiare fra queste diverse sessioni tutte le volte che occorre.

Consultate la documentazione della vostra versione di SVR4 per verificare se supporta la funzionalità di console virtuale e quante sessioni permette di attivare simultaneamente.

FUNZIONI DI SHELL

Gli shell di SVR3 e SVR4 forniscono un altro meccanismo di definizione di procedure che risulta leggermente più efficiente dell'approccio alle procedure di shell trattato precedentemente. Tuttavia, queste *funzioni di shell* sono più difficili da controllare rispetto alle normali procedure di shell perché vengono definite direttamente nello shell e non hanno vita permanente in un file. Quando definite una funzione di shell, lo shell corrente legge la definizione e la funzione viene memorizzata all'interno dello shell che è attivo in quel momento. Potete eseguire una funzione di shell proprio come qualunque altro comando o procedura di shell, ma quando vi scollegate perdete la funzione. Potete anche inserire le definizioni delle funzioni di shell in un file ed eseguire il file con il meccanismo **.nomefile** già descritto, che permette la lettura e l'interpretazione del file da parte dello shell corrente, invece che da un subshell nuovamente creato. Le funzioni di shell non sono esportabili o automaticamente disponibili ai subshell, ma i subshell possono rileggerle se occorre.

Potete definire le funzioni di shell utilizzando il seguente formato:

```
$ nome ( ) {
>         lista-comandi
> }
```

dove *nome* è il nome che intendete assegnare alla funzione. Sono richieste le parentesi tonde ( ); la parentesi graffa sinistra definisce l'inizio della lista di comandi, in cui potete specificare qualunque comando eseguibile od operatore di shell. Inoltre, all'interno della funzione potete utilizzare i parametri posizionali **$#**, **$*** e **$1** fino a **$9**. Per esempio:

```
$ show ( ) {
>         echo $1
>         echo $2
>         exit 2
> }
$ show salve ciao
salve
ciao
$
```

Questo esempio definisce una funzione di shell denominata **show** che visualizza i primi due argomenti e poi restituisce il valore di uscita 2 allo shell chiamante. Le funzioni di shell non hanno limiti di complessità; potete definire come funzioni le procedure di shell utilizzate più di frequente, riducendo così il tempo di impegno di CPU. Comunque, se utilizzate ripetutamente un insieme di funzioni di shell, dovete fare in modo di inserirle nel vostro **.profile** o utilizzare qualche altro meccanismo per assicuravi che siano sempre disponibili quando le richiedete.

LA VARIABILE cdpath

Lo shell permette di specificare un percorso di ricerca per le operazioni **cd**, che agisce in modo molto simile alla variabile **PATH**. La variabile di ambiente **CDPATH** consente di dichiarare un elenco di nomi di directory da esaminare nella ricerca di un percorso relativo fornito come argomento al comando **cd**. Il comando **cd** ricerca il proprio argomento in ciascun directory elencato in **CDPATH** e assume come directory corrente il primo directory che contiene l'argomento percorso relativo. Il valore di **CDPATH** segue la sintassi di **PATH**, per esempio:

```
$ echo $CDPATH
:/home/giorgio:/home/giorgio/src
$
```

Il primo carattere : (due punti) significa che la ricerca deve essere fatta prima nel directory corrente; quindi la ricerca viene fatta in **/home/giorgio**, e infine in **/home/giorgio/src**. Se l'argomento del comando **cd** è un percorso assoluto completo, questo viene assunto come directory corrente; ciò risulta molto utile se cambiate frequentemente di directory.

SEQUENZE DI COMANDI

Un ulteriore costrutto di shell usato più raramente è la *sequenza di comandi*. Diversamente dall'operatore ;, che raggruppa semplicemente dei comandi, una sequenza di comandi viene costruita con gli operatori **&&** (*and logico*) oppure || (*or logico*) a separazione di comandi individuali. L'operatore **&&** esegue il secondo comando solo se il primo comando ha avuto successo (restituisce zero); l'operatore || esegue il secondo comando solo se il primo comando *non* ha avuto successo (restituisce non-zero), se il primo comando ha avuto successo il secondo non viene eseguito. Per le esempio:

```
$ touch file.mio || echo Non posso creare file.mio
```

è equivalente a:

```
$ touch file.mio
$ if [ $? -ne 0 ] ; then echo Non posso creare file.mio ; fi
```

Capitolo 9

La documentazione del sistema UNIX

Tutti i sistemi UNIX dispongono di una copia del manuale di documentazione, noto come *UNIX User's Manual*, o più semplicemente come *User's Manual* o *UNIX Manual*. Inoltre, molte versioni di SVR4 dispongono di una funzione di aiuto in linea che può facilitare la scelta del comando corretto per una particolare applicazione e selezionarne l'opzione più conveniente. In questo capitolo descriveremo entrambi questi strumenti di documentazione.

9.1 UNIX User's Manual

Lo *User's Manual* rappresenta la documentazione ufficiale del sistema UNIX e, nel corso degli anni, si è aggiornata e ampliata di pari passo al software di sistema. Tutti i comandi, i relativi argomenti, le librerie di funzioni, i formati dei file, le utility e gli strumenti software sono documentati esaustivamente nello *User's Manual*, a cui potete rivolgervi per risolvere i problemi che si presentano nel sistema UNIX, anche se la fonte ultima a cui ricorrere rimane sicuramente la verifica empirica delle caratteristiche del sistema da effettuare direttamente sulla macchina.

Sfortunatamente, anche se lo *User's Manual* risulta un documento di riferimento completo e sicuramente corretto, non sempre si dimostra di sem-

plice lettura e interpretazione. Non è raro che gli utenti debbano studiare attentamente una singola frase del manuale, discutendo, e verificando in pratica, il significato e gli sviluppi di una parola o di un'espressione.

Lo *User's Manual* è stato concepito per essere molto conciso, per riassumere le caratteristiche di UNIX in una forma da cui gli esperti e i principianti possano estrarre qualunque dettaglio specifico del sistema. È molto difficile, se non impossibile, imparare a usare il sistema avendo solo a disposizione lo *User's Manual*, ma è altrettanto improbabile diventare un utente esperto senza manuale.

Se il vostro sistema non dispone dello *User's Manual*, procuratevi immediatamente una copia del manuale che sia congruente con la vostra versione di UNIX e con il vostro hardware. Anche se lo *User's Manual* non differisce molto per le diverse versioni di UNIX, con ogni probabilità non riuscirete a sfruttare appieno le caratteristiche del vostro sistema se utilizzate una versione di manuale non adatta.

9.2 Organizzazione dello User's Manual

La prima pubblicazione dello *User's Manual* comprendeva un unico volume di piccole dimensioni che riassumeva tutte le caratteristiche di cui disponeva il sistema UNIX in quel momento. Il documento originale conteneva otto sezioni principali, che sono presenti ancora oggi:

1. Comandi e programmi applicativi
2. Chiamate di sistema
3. Subroutine
4. Formati di file
5. Varie
6. Giochi
7. File speciali
8. Procedure di gestione del sistema

Queste sezioni dello *User's Manual* vengono spesso individuate da un identificatore numerico. La Sezione 1 individua la sezione Comandi, la Sezione 3 rappresenta la sezione Subroutine e così via.

La Sezione 1 documenta la maggior parte dei comandi che vengono descritti in questo testo, come per esempio i comandi utente **cat** o **uux**, ma anche le Sezioni 4, 5, 6 e 7 contengono alcuni riferimenti ai comandi di UNIX.

Le Sezioni 2 e 3 interessano soprattutto ai progettisti software perché descrivono le procedure che un programma in linguaggio C o FORTRAN può utilizzare. Poiché qualche nome di funzione trattata nelle Sezioni 2 e 3 ricorda i nomi di alcuni comandi che vengono esaminati nella Sezione 1, dovete scegliere attentamente la sezione quando consultate il manuale. Mentre le

Sezioni 2 e 3 contengono informazioni o procedure che sono utili ai progettisti software, le sezioni in generale differiscono tra loro. Le chiamate di sistema trattate nella Sezione 2 sono riferimenti al kernel, simili come funzione alle interruzioni che si verificano in ambiente MS-DOS, mentre la Sezione 3 esamina le procedure o le funzioni che sono disponibili nel sistema. Molte procedure, ma non tutte, utilizzano internamente chiamate al sistema. Anche se non sembra esserci differenza tra le chiamate al sistema e le funzioni, la distinzione può essere significativa per i sistemisti e per i progettisti software.

La Sezione 4, "Formati di file", indica le modalità di memorizzazione dei dati sui file che il sistema utilizza. Potete trovare un paragrafo che descrive il file **/etc/passwd** o il vostro file **.profile**, ma anche paragrafi che trattano il formato dei file eseguibili e il formato dei file di descrizione dei terminali utilizzati dal sistema **terminfo**.

La Sezione 5, "Varie", contiene utili informazioni che non vengono trattate nelle altre sezioni. Per esempio, include la tabella di codifica dei caratteri ASCII, tratta delle variabili di ambiente di shell, descrive i codici dei caratteri specifici per nazionalità, definisce i caratteri di tabulazione per i terminali e molte altre caratteristiche del sistema.

La Sezione 6, "Giochi", descrive i giochi che potete eseguire in ambiente UNIX, ma alcuni sistemi non comprendono questa sezione se non forniscono i giochi. Potete comunque acquistare separatamente il software per i giochi con inclusa la documentazione relativa che viene trattata nella Sezione 6.

La Sezione 7, "File speciali", tratta i formati dei file speciali che sono contenuti nel directory **/dev**. Questo materiale è di grande interesse per i progettisti software e per chi intende utilizzare questi file speciali nei propri programmi. La Sezione 7 esamina anche la struttura dei dischi e dei nastri, le interfacce ai terminali sincroni e asincroni e le interfacce per le reti locali.

La Sezione 8, "Procedure di gestione del sistema", descrive le procedure di caricamento del sistema, di rilevamento di errori dell'hardware ed esamina altre attività di gestione a basso livello. Queste funzioni sono spesso obsolete e non risultano generalmente di grande utilità per sistemi SVR4 basati su microcalcolatore; attualmente alcune versioni dei manuali SVR4 non includono più la Sezione 8.

Il formato base del documento è rimasto invariato nel corso degli anni, ma la rapida crescita del sistema UNIX ha reso il manuale così ingombrante che ora si compone di tre parti distinte: *User's Manual*, *Programmer's Manual* e *Administrator's Manual*. Ogni parte tratta argomenti specifici che vengono esaminati nelle sue otto sezioni, per cui la Sezione 1 di *Administrator's Manual* può esaminare un comando non contenuto nello *User's Manual*. Sfortunatamente, non sempre è facile capire quale manuale può contenere la descrizione di un particolare comando, per cui spesso dovete ricercare il comando che vi interessa in ognuna delle parti di cui si compone il manuale.

In SVR4 usualmente il manuale è ulteriormente suddiviso in diversi volumi rilegati separatamente; in pratica l'intero manuale può essere composto di 25 volumi. Abitualmente i volumi sono organizzati per argomenti generali, cosicché ogni volume contiene tutte le informazioni relative allo sviluppo, alle reti, a X Window System, e così via. Spesso ognuno di questi volumi contiene parti di tutte le otto sezioni del manuale, tuttavia ciascun libro nel suo insieme potrà essere di maggior interesse per utenti X, esperti di reti, o altri gruppi.

A eccezione dell'indice strutturato (*permuted index*) e di una parte introduttiva, lo *User's Manual* è strutturato per argomenti (comandi e procedure sono documentati separatamente). Ogni voce viene denominata *man page* (pagina di manuale), per cui ogni argomento occupa una *pagina di manuale*, anche se fisicamente si estende su più pagine del documento. Questa organizzazione è valida per un manuale di riferimento, perché ogni pagina è indipendente dalle altre.

Man mano che il sistema si è evoluto e ampliato, la natura modulare dello *User's Manual* ha reso l'operazione di aggiornamento molto semplice. Potete sostituire, parallelamente alle modifiche di sistema, le pagine di manuale obsolete con quelle aggiornate e la ricerca di un argomento risulta estremanente rapida poiché tutti gli aspetti che lo riguardano sono raccolti in uno spazio ridotto.

I progettisti, quando sviluppano nuovi strumenti software e vogliono descriverne accuratamente le caratteristiche e le modalità di impiego, generalmente scrivono anche le relative pagine di manuale, che vengono poi utilizzate per sviluppare funzioni di aiuto interattivo.

9.3 Ricerca dei comandi sul manuale

Quando cercate un comando o altre voci nel sistema UNIX, è indispensabile sapere la sezione del manuale che contiene la descrizione del comando e questa indicazione viene specificata generalmente tra parentesi dopo il nome del comando. Per esempio, la notazione **uname(1)** si riferisce al comando utente **uname** che è documentato nella Sezione 1 del manuale, mentre **uname(2)** si riferisce alla chiamata di sistema (*system call*) a uso dei progettisti, documentata nella Sezione 2. Una lettera che segue il numero di sezione, come per esempio **df(1M)**, identifica la categoria del manuale in cui il comando è catalogato, per cui la notazione **1M** referenzia un comando della Sezione 1 associato al *System Administrator's Manual*, **3C** indica un sottoprogramma in linguaggio C, **3F** una procedura in FORTRAN, **3E** un sottoprogramma relativo al formato di file oggetto **elf**.

Ogni comando, procedura o tipo di file che possiede una propria pagina di manuale è catalogato alfabeticamente nella sezione, ma è possibile che alcune pagine di manuale si trovino nello *User's Manual*, mentre altre siano

in *Administrator's Manual* o nel *Programmer's Manual.* In ogni caso potete consultare un indice strutturato che permette di selezionare rapidamente l'informazione di vostro interesse e che esamineremo tra breve.

9.4 Un esempio di pagina di manuale

La Figura 9.1 mostra una breve, ma completa, pagina di manuale per il comando **df(1M)**. Questa pagina è estratta dalla Sezione 1 dello *User's Manual*, perché **df** rappresenta un comando utente di tipo generale ma, come specifica la notazione **1M**, interessa soprattutto i gestori di sistema.

Il nome del comando appare in cima alla pagina (Figura 9.1), su entrambi i lati e rappresenta la chiave per organizzare in ordine alfabetico le pagine del manuale.

La versione SVR4 di UNIX si compone di pacchetti software distinti, o unità caricabili separatamente, che potete anche acquistare e installare sulla macchina. Se il sistema non dispone di un particolare comando, probabilmente non è stato caricato il relativo pacchetto software. Altri esempi di pacchetti software sono il C Compilation set e il Text Preparation set.

La pagina di manuale specifica nella parte inferiore l'indicazione della data a cui risale l'ultima modifica oppure la stampa della pagina, mentre la numerazione di pagina si riferisce alla pagina di manuale e non al documento nel suo complesso, potete quindi sostituire una pagina di manuale obsoleta con una versione più aggiornata senza cambiare l'intero manuale. Una pagina di manuale si compone di diverse sezioni divise per argomento, la prima delle quali è *NAME* (nome), che specifica il nome del comando e ne riassume brevemente la funzione. Il comando qui indicato è quello *primario*, che compare anche in cima alla pagina di manuale e che si differenzia dai comandi *secondari*, cioè da quei comandi che vengono generalmente elencati insieme al comando primario e rappresentano i comandi derivati o comandi che sono in qualche modo legati al comando primario e non possiedono pagine proprie. Questi comandi non appaiono nell'intestazione di pagina, per cui non riuscite a trovarli eseguendo una ricerca alfabetica nelle intestazioni di pagina. Non tutti i comandi possiedono dunque una pagina di manuale propria e anche alcuni comandi comuni, come per esempio **mv(1)** e **ln(1)**, sono considerati comandi secondari e vengono elencati nella pagina di manuale associata a un altro comando, **cp(1)** nel caso dei due comandi precedenti. Per evitare questo problema, cercate il comando nell'indice.

Il nome del comando e la linea di riepilogo che specifica la sua funzione sono specificati parola per parola nell'indice strutturato del manuale, descritto in seguito.

df(1M) **df(1M)**

NAME

df (generic) – report number of free disk blocks and files

SYNOPSIS

df [–F *FSType*] [–begklntV] [*current_options*] [–o *specific_options*] [*directory* | *special* | *resource*...]

DESCRIPTION

df prints the allocation portions of the generic superblock for mounted or unmounted file systems, directories or mounted resources. *directory* represents a valid directory name. If *directory* is specified df reports on the device that contains the *directory*. *special* represents a special device (e.g., /dev/dsk/0s1). *resource* is an RFS/NFS resource name. If arguments to df are pathnames, df produces a report on the file system containing the named file.

current_options are options supported by the s5-specific module of df. Other FSTypes do not necessarily support these options. *specific_options* indicate suboptions specified in a comma-separated list of suboptions and/or keyword-attribute pairs for interpretation by the *FSType*-specific module of the command.

The generic options are:

–F Specify the *FSType* on which to operate. This is only needed if the file system is unmounted. The *FSType* should be specified here or are determined from /etc/vfstab by matching the *mount_point*, *special*, or *resource* with an entry in the table.

–b Print only the number of kilobytes free.

–e Print only the number of files free.

–g Print the entire statvfs structure. Used only for mounted file systems. Can not be used with *current_options* or with the –o option. This option will override the –b, –e, –k, –n, and –t options.

–k Print allocation in kilobytes. This option should be invoked by itself because its output format is different from that of the other options.

–l Report on local file systems only. Used only for mounted file systems. Can not be used with *current_options* or with the –o option.

–n Print only the *FSType* name. Invoked with no arguments this option prints a list of mounted file system types. Used only for mounted file systems. Can not be used with *current_options* or with the –o option.

–t Print full listings with totals. This option will override the –b, –e, and –n options.

–V Echo the complete command line, but do not execute the command. The command line is generated by using the options and arguments provided by the user and adding to them information derived from /etc/mnttab or /etc/vfstab. This option should be used to verify and validate the command line.

–o Specify FSType-specific options.

10/89 **Page 1**

Figura 9.1 Esempio di pagina dello *User's Manual*.

df(1M) df(1M)

-v Reports percent of blocks used as well as the number of blocks used and free.

If no arguments or options are specified, the free space on all local and remotely mounted file systems is printed.

NOTES

The -F option is intended for use with unmounted file systems.

This command may not be supported for all FSTypes.

FILES

/dev/dsk/*
/etc/mnttab mount table
/etc/vfstab list of default parameters for each file system

SEE ALSO

mount(1M), mnttab(4), vfstab(4).
statvfs(2) in the *Programmer's Reference Manual.*
Manual pages for the FSType-specific modules of df.

Page 2 10/89

Figura 9.1 Esempio di pagina dello *User's Manual.* (*continua*)

LA SEZIONE SYNOPSIS

La sezione *Synopsis* (sinossi) specifica brevemente le modalità di lancio del comando, in un formato conciso in cui ogni carattere assume un preciso significato, anche se raramente la linea di comando coincide nella struttura con le indicazioni date in questa sezione. Una volta che avete interpretato correttamente il formato, potete adattare la composizione della linea di comando alle vostre esigenze. Le indicazioni in carattere diverso, come per esempio **df** e **-o** nel caso del comando **df(1M)**, hanno una collocazione sulla linea di comando che corrisponde alle specifiche date in questa sezione oppure vengono tralasciate. I termini in corsivo rappresentano prototipi di argomento della linea di comando che potete adattare alle vostre esigenze, come nel caso della stringa *FSType* di Figura 9.1 che non dovete inserire nella forma in cui appare, ma interpretare come il nome del file system da specificare.

Una coppia di parentesi quadre indica l'opzionalità dell'argomento che delimitano, come[**-F** *FSType*], la cui presenza nella linea di comando è dettata solo da considerazioni di necessità. Per esempio, utilizzate l'opzione **-t** solo se volete la versione completa del listato. Una linea di comando può contenere un numero qualunque di argomenti separati dal carattere ¦ (pipe). Nel caso di **df(1M)** potete omettere come argomento il nome del directory perché è racchiuso in parentesi quadre, ma se lo inserite nella linea di comando, potete specificare solo un nome di file system, un nome di directory o qualche altra risorsa di questo tipo. Non inserite le parentesi quadre o il carattere ¦ (a meno che non vogliate richiedere allo shell la creazione di un pipeline).
I caratteri ... sottointendono una o più ripetizioni dell'argomento che li precede. Questa linea di comando, per esempio

```
cat [ -u] [ -s] [ -v [ -t] [ -e]] file ...
```

specifica uno o più file, ma soprattutto evidenzia la possibilità di annidare gli elementi della sinossi. Gli argomenti **-v**, **-t** e **-e** sono opzionali, ma **-t** e **-e** sono annidati all'interno di **-v**. Questo significa che **-v** è facoltativa, ma se specificate le opzioni **-t** e **-e**, dovete anche specificare l'opzione **-v**. Gli argomenti **-u** e **-s** sono indipendenti e possono essere utilizzati con o senza l'opzione **-v**. Questo annidamento di argomenti è tipico del sistema UNIX e della sezione *Synopsis* della pagina di manuale.

Potete anche inserire nella stessa linea di comando gli argomenti modificatori rappresentati dai flag.

```
ed [ -] [ -p stringa] [ -x] [ file]
```

Tutti gli argomenti sono opzionali in questo caso, ma se specificate l'argomento **-p**, dovete anche includere una stringa di prompt.

LA SEZIONE DESCRIPTION

La sezione *Synopsis* costituisce un rapido riferimento per comprendere a grandi linee il funzionamento del comando, ma specifica anche dettagliatamente i singoli componenti della linea di comando. Quando vi occorre una descrizione del comando leggermente più completa, dei suoi argomenti e delle sue funzioni, leggete la sezione *Description* (descrizione) che segue la sezione *Synopsis* e che spiega l'uso del comando e di tutte le sue opzioni. Questa sezione differisce molto da pagina a pagina e fornisce una completa descrizione del comando, ma dovete leggerla più volte per capire completamente tutti i dettagli e le implicazioni connesse al comando e ai suoi argomenti. Ogni frase che compone questa descrizione è corretta, ma spesso deve essere supportata dall'esecuzione pratica del comando, per capirne veramente il significato.

LE ALTRE SEZIONI

Dopo la descrizione del comando, seguono generalmente altre sezioni a concludere la pagina di manuale, che non sono però così ben definite e standardizzate come le sezioni che abbiamo descritto finora e possono anche mancare.

La sezione *Notes* (note) contiene una informazione a piè di pagina di grande importanza, che si discosta per qualche motivo dalla descrizione del comando.

La sezione *See Also* (riferimenti) referenzia comandi o informazioni rilevanti che sono presenti in altre sezioni e fornisce una guida all'esame di altri comandi che possono soddisfare meglio del comando corrente le vostre esigenze o che hanno associate alcune funzioni di potenziale utilità. In Figura 9.1, **mount(1M)** indica un comando la cui esecuzione modifica l'uscita prodotta da **df(1M)**.

La sezione *Files* (file) definisce i pathname dei database e di altri file che il comando utilizza o modifica, perché non sempre la sezione *Description* tratta le relazioni intercorrenti tra il comando e i file. Sarebbe utile se questi file fossero indicizzati cosicché quando incontrate un file durante l'attraversamento del file system, potete analizzarne la funzione, ma questo tipo di riferimento non è stato fatto.

La sezione *Diagnostic* (diagnostica) contiene una breve descrizione dei probabili messaggi di errore o valori di ritorno che il comando restituisce. Raramente si tratta di un elenco completo dei messaggi diagnostici che il comando può generare, ma spesso questa sezione fornisce la traccia di comportamento in caso di funzionamento anomalo del comando.

9.5 L'indice strutturato

L'*indice strutturato* rappresenta il metodo più efficiente per consultare il documento *User's Manual* quando avete un problema. Non cercate di referenziare un comando o una funzione in corrispondenza della relativa pagi-

Figura 9.2 Una pagina dell'indice strutturato.

na di manuale, poiché il nome del comando non contiene la chiave di ricerca. Fortunatamente, l'indice strutturato si presenta in una forma che quasi sempre permette di trovare l'informazione che vi interessa.

L'indice strutturato, noto anche come KWIC (keyword in context), elenca alfabeticamente tutte le parole che sono contenute nella sezione *Name* di ogni pagina di manuale e risulta uno strumento di ricerca molto efficace, poiché la sezione *Name* descrive brevemente la funzione del comando utilizzando principalmente *parole chiave*. Ognuno dei tre volumi di cui si compone il manuale dispone di un proprio indice strutturato, per cui qualche volta è necessario consultare più di un indice per trovare l'informazione che vi interessa. L'indice strutturato si trova all'inizio del volume, quasi a indicare la sua importanza, e non alla fine, come avviene nella maggior parte dei manuali.

La Figura 9.2 illustra una pagina dell'indice strutturato dello *User's Manual* in cui le voci, elencate in ordine alfabetico, occupano la parte centrale, mentre all'estrema destra appare l'indicazione della pagina di manuale che contiene l'informazione desiderata.

L'indice strutturato elenca tutte le parole chiave che sono contenute nella sezione *Name* di ogni pagina di manuale e le colloca al centro della pagina. Inoltre, dispone a destra della parola chiave corrente la parte di frase che la segue nella sezione *Name*, delimitando l'ultima parola con il carattere/, mentre a sinistra dispone la parte di frase che precede la parola chiave corrente nella sezione *Name*. Se la parte di frase che segue la parola chiave corrente supera i limiti di linea, l'indice la riporta all'inizio della linea, come si verifica in Figura 9.2 per la pagina di manuale **paste(1)**. Analogamente, se a sinistra della parola chiave corrente le parole superano i limiti di linea, l'indice ruota le parole in soprannumero a destra della linea.

Vediamo ora alcuni esempi. Per trovare il comando **ln**, è sbagliato cercare nella Sezione 1 la pagina di manuale **ln(1)** ma, se esaminate la voce **ln** dell'indice strutturato, scoprirete che il comando **ln** viene descritto nella pagina di manuale **cp(1)**, che compare nella Sezione 1 del manuale. Analogamente, se volete collegare insieme due file, potete cercare la parola chiave **link** e esaminare le voci elencate. Facendo molta attenzione al carattere / che definisce l'inizio della frase di descrizione del comando, potete verificare che la linea dell'indice strutturato

```
cp, ln, mv: copy, link or move files . . . . . cp(1)
```

soddisfa le vostre esigenze. Ancora una volta la ricerca di un comando coinvolge la pagina di manuale **cp(1)**.

Una volta che avete familiarizzato con il formato dell'indice e avete preso confidenza con la terminologia della sezione *Name* delle pagine di manuale, l'indice strutturato si dimostra estremamente utile. In ogni caso, se la parola chiave non è corretta e non riuscite a trovare il comando che cercate, con-

sultate la pagina di manuale di un comando che svolge funzioni analoghe e poi esaminate la sezione *See Also*, che contiene generalmente il comando di vostro interesse.

9.6 Il comando man in linea

Molti grandi sistemi offrono l'accesso on-line al contenuto dell'intero manuale che generalmente comprende le pagine di manuale per il software installato localmente e il documento di riferimento standard *User's Manual*. Il testo ha una dimensione che varia da 2 a 3 megabyte, per cui la maggior parte delle macchine di piccole dimensioni non ha questa opzione.

Potete ottenere sullo standard output le pagine del manuale battendo da tastiera il comando **man** (manual) seguito dall'indicazione della pagina richiesta. Questo comando

```
$ man diff
```

visualizza la pagina di manuale **diff(1)**. Se il manuale contiene più riferimenti all'argomento che avete specificato, appaiono su video sequenzialmente tutte le pagine di manuale corrispondenti. Potete anche ridirigere l'uscita a un file, se intendete effettuare un esame successivo. Poiché le pagine di manuale, prima di essere visualizzate, vengono generalmente impaginate con **nroff**, il comando **man** può impiegare un tempo di esecuzione relativamente lungo, che raggiunge valori di un minuto o due su macchine con notevole sovraccarico.

Il comando **man** accetta diverse opzioni. Per limitare l'uscita a una singola pagina di una sezione del manuale, specificate il numero della sezione prima dell'indicazione della pagina. Questo comando

```
$ man 1 man
```

visualizza la pagina di manuale solo per il comando **man(1)**.

Per adattare l'impaginazione dei risultati prodotti a un particolare terminale, inserite l'opzione **–T** (terminal), seguita dal tipo di terminale. Questo comando

```
$ man –Tvt100 1 man
```

adatta l'impaginazione di **man(1)** al terminale DEC vt100. Se non specificate l'opzione **–T**, il sistema interpreta il contenuto della vostra variabile di ambiente **TERM** come tipo di terminale.

Esistono molte altre opzioni: **–d** (directory) associa il cammino di ricerca di default al directory corrente piuttosto che al directory standard di sistema, mentre **–c** (col) genera un tipo di uscita senza avanzamento inverso di interlinee, adatto per la maggior parte delle stampanti semplici.

Quando sono presenti, le pagine di manuale si trovano nel directory **/usr/man** e nei relativi subdirectory, di cui i principali sono **u_man** (user's manual), **p_man** (programmer's manual) e **a_man** (administrator's manual). Ognuno di questi directory comprende altri subdirectory, per ogni sezione del manuale che fa parte di ognuno dei tre volumi: **man1** (i comandi della Sezione 1), **man2** (i comandi della Sezione 2) e così via fino a **man8** (i comandi della Sezione 8). Le pagine di manuale sono generalmente memorizzate nel formato originale **troff** e devono essere impaginate con i programmi **troff – man** oppure **nroff – man**, prima di assumere il formato di stampa definitivo per qualunque terminale o stampante.

Inoltre, alcuni sistemi mantengono l'insieme delle pagine di manuale in file preformattati, per cui è possibile accedere in linea senza incorrere nei ritardi e nei problemi, connessi all'utilizzo della CPU, che invece si verificano per formattare le pagine in modalità demand. In questo caso, i subdirectory di **u_man**, **p_man** e **a_man** vengono referenziati come **cat1**, **cat2**, fino a **cat8** e la risposta in termini temporali del comando **man** è molto più rapida quando le pagine sono preformattate, anche se il gestore di sistema generalmente adatta le pagine preformattate alla stampante o al terminale più lenti del sistema.

9.7 Il comando help in linea

Oltre all'indispensabile *User's Manual*, SVR4 fornisce una funzione di utility *help* in linea che può sostituire il manuale in casi di emergenza. Questa funzione occupa molto spazio su disco, per cui può anche mancare in qualche macchina di piccole dimensioni, dove risulta di estrema importanza ottimizzare lo spazio a disposizione.

Il termine "help" aveva in origine un significato diverso e indicava una piccola parte dello strumento di sviluppo software **sccs** (Source Code Control System). La funzione che descriviamo in questo paragrafo, invece, è un pacchetto completamente nuovo che ha poco in comune con le versioni dei sistemi UNIX precedenti a SVR4.

Attivate il sottosistema di aiuto con il comando **help**: apparirà su video una guida a menu che vi sollecita a selezionare la successiva azione.

```
$ help
help: UNIX System On-Line Help

    choices          description
      s              starter: general information
      l              locate: find a command with keywords
      u              usage: information about commands
      g              glossary: definitions of terms
```

```
        r           Redirect to a file or a command
        q           Quit

    Enter choice >
```

Siete ancora nel sottosistema di aiuto e potete intraprendere una azione selezionando una tra le opzioni elencate. Per ritornare allo shell, inserite **q** in corrispondenza del prompt **Enter choice >**. Se scegliete l'opzione **r**, potete invece copiare il contenuto dello schermo in un file o utilizzarlo come standard input di un comando.

```
Enter choice > r
Enter > file, ¦ cmd(s), or RETURN to continue >
```

A questo punto, potete inserire il carattere > seguito dal nome di un file o specificare il carattere ¦ seguito da un pipeline, per ridirigere il contenuto corrente dello schermo rispettivamente in un file o a un comando.

```
Enter choice > r
Enter > file, ¦ cmd(s), or RETURN to continue> >/home/giorgio/help.menu
Enter choice >
```

In questo esempio, abbiamo ridiretto l'uscita al file **/home/giorgio/help.menu** e, al termine di questa azione, il sistema di aiuto visualizza di nuovo la guida a menu del livello corrente. Ogni sottomenu di aiuto garantisce la ridirezione di una intera immagine di video e questo permette una semplice conservazione dei dati per eventuali impieghi futuri.

```
Enter choice > s

starter: General UNIX System User Information

    starter provides general information for system users. Enter
    one of the choices below to proceed.

        choices     description

            c       Commands and terms to learn first
            d       Documents for system users
            e       Education centers for UNIX System training
            l       Local UNIX System information
            t       Teaching aids available on-line

            r       Redirect to a file or a command
            q       Quit
            h       Restart help
```

Figura 9.3 Visualizzazione delle scelte del menu di help **starter**.

Il menu di livello superiore permette di accedere a tutte le utility di aiuto dei livelli inferiori. Selezionando l'opzione **s** (starter) per avere maggiori informazioni, il sistema visualizza un altro menu, come indicato in Figura 9.3. Se siete principianti, l'opzione **s** costituisce un metodo eccellente per prendere confidenza con il sistema UNIX. L'opzione **c** definisce i comandi più semplici e le caratteristiche meno complesse del sistema, mentre l'opzione **d** fornisce una bibliografia di documenti utili, anche se le fonti che

```
Enter choice > l
  locate: Find UNIX System Commands with Keywords

      Give locate one or more keywords related to the work
      you want to do. It will print a list of UNIX system commands
      whose actions are related to the keywords.

      For example, you enter the keywords print file

      locate could produce the list:    The cat (concatenate) command
                                        The ls (list) command
                                        The pr (print) command

  Enter a k to use locate.

      choices          description

         k             Enter a list of keywords

         r             Redirect to a file or a command
         q             Quit
         h             Restart help

Enter choice > k
Enter keywords separated by blanks > print

Commands found using print:

The cat (concatenate) command
The echo command
The line command
The ls (list) command
The pcat (concatenate packed file) command
The pr (print) command
The pwd (print working directory) command
The tail command
-------------------------------------------------------------------------
Choices: UNIX_command , k (new keywords), r (redirect), h (restart help),
q (quit)
-------------------------------------------------------------------------
Enter choice >
```

Figura 9.4 Visualizzazione di help del comando **locate**.

elenca possono comportare qualche problema di ricerca. L'opzione **l** fornisce alcune informazioni sulle caratteristiche e sulle utility locali della vostra macchina e il gestore di sistema può aggiungere questa sezione utilizzando il comando **helpadm**, che tratteremo tra breve. L'opzione **t** elenca altri aiuti che sono disponibili in linea, che si risolvono generalmente nel pacchetto *Instructional Workbench*, che non tutte le versioni SVR4 forniscono. L'Instructional Workbench è un sistema di istruzioni di supporto al calcolatore che definisce un insieme di strumenti efficienti per migliorare la vostra pratica su molte attività, non solo sul sistema UNIX. Potete ritornare al menu principale in qualunque momento battendo il tasto **h**, indipendentemente dal menu in cui vi trovate all'interno del sistema di aiuto.

Se selezionate l'opzione **l** del menu principale, il sistema di help vi permette di referenziare i comandi specificando le parole chiave che descrivono l'azione che intendete eseguire, come mostrato in Figura 9.4. Selezionate **k** da questo sottomenu se volete che il sistema esegua una ricerca sulla parola chiave che avete inserito. In questo esempio, abbiamo battuto **print** come parola chiave e il sistema di help ha indicato diversi comandi associati

```
Enter choice > u

usage: Information about Commands

    usage provides information about specific UNIX System commands.

        Within usage, double quotes " " mark options or literals,
    and angle brackets < > mark argument variables.

        You should see starter for basic UNIX system commands and
    terms before going on to anything else.

    Enter one of the choices below to proceed.

        choices              description

    UNIX_command         Obtain usage information for a command

            p            Print a list of commands

            r            Redirect to a file or a command
            q            Quit
            h            Restart help

    Enter choice > cat
    Enter d (description), e (examples), or o (options) >
```

Figura 9.5 Visualizzazione di help del comando **usage**.

con l'operazione di stampa di file. La funzione di aiuto e il manuale utilizzano parole chiave diverse, per cui questi esempi possono referenziare qualche comando che differisce da quelli indicati nel manuale. Comunque, non tutte le utility del sistema sono previste da help, come indica l'esempio, in cui la ricerca della parola chiave **print** non ha rivelato nulla sul sottosistema di stampa **lp**. Anche se si dimostra utile, il sistema di help non può certamente sostituire il documento di riferimento *User's Manual.*

Se selezionate l'opzione **u** del menu principale, ricevete informazioni su come utilizzare un comando, tra cui le specifiche sugli argomenti disponibili e le relative modalità di impiego, come indica la Figura 9.5.

Inserite il nome di un comando se lo conoscete; il sistema vi metterà in condizione di scegliere il tipo di informazione che vi interessa. Selezionate l'opzione **d** per avere una descrizione del comando, **e** per vedere qualche esempio oppure **o** per ottenere la lista delle opzioni del comando.

Se selezionate l'opzione **d**, il risultato può essere quello di Figura 9.6. Quanto indicato non coincide con il contenuto dello *User's Manual* e questo vi permette di arricchire le vostre conoscenze sul comando e sulle sue opzioni. Le opzioni di menu **o** (opzioni) ed **e** (esempi) forniscono altre utili informazioni.

Accade di frequente che il contenuto di queste finestre grafiche ecceda i limiti di schermo ma, se inserite **n** (next page) quando vedete indicato **MORE** in fondo al video, potete analizzare l'intera informazione a più riprese. È possibile anche ridirigere il contenuto del video in un file o in un comando per effettuare un esame a posteriori.

```
Enter d (description), e (examples), or o (options) > d
cat: Description

Syntax Summary: cat [ -u ] [ -s ] [ -v [ -e ] [ -t ] ] [ file_name ... ]

        where: file_name is the name of a file.

Description:
        cat is shorthand for "concatenate". Use cat to send the
        contents of a file to standard output. If more than one file name is used,
        cat prints each file in sequence on the standard output. cat
        echoes standard input if you do not list a file name or if you use "-"as an
        argument. See also: cp(1), pg(1) e pr(1) for commands with functions
        similar to cat.

Choices: o (options), e (examples), UNIX_command , p (print list),
r (redirect), h (restart help), q (quit)

Enter choice >
```

Figura 9.6 Visualizzazione di help del comando **description**.

L'opzione **g** del menu principale fornisce un glossario dei termini e caratteri speciali. Se specificate un termine o un carattere, il sistema di aiuto restituisce una breve, ma utile, definizione che non è disponibile nello *User's Manual*.

USO DEL SISTEMA DI HELP DA LINEA DI COMANDO

Invece di selezionare una opzione del menu principale del sistema di help, potete battere un comando direttamente dallo shell per introdurre uno dei sottosistemi di aiuto. Utilizzate il comando **locate** oppure **locate** [*parola-chiave*] per attivare direttamente il sottosistema. Se specificate il comando senza argomenti, appare il menu di locate a cui potete indicare una parola chiave. Se specificate la parola chiave come argomento del comando **locate**, il sistema di help fornisce i risultati solo per quella parola chiave.

Analogamente, potete utilizzare **glossary** oppure **glossary** [*termine*] per lanciare direttamente il sottosistema **glossary**, e **starter** oppure **usage** [**-d**] [-e] [-o] [*nome-comandi*] per le altre opzioni di livello superiore. Gli altri argomenti del comando **usage** inizializzano il sistema di aiuto rispettivamente al sottomenu **description**, **examples** e **options**.

9.8 Approfondimenti

Il sistema di help è uno strumento flessibile che può essere adattato o espanso. Nei prossimi paragrafi esamineremo gli strumenti che un gestore di sistema, collegatosi al sistema come *root*, può utilizzare per aggiornare i database, mentre alla fine parleremo di altri strumenti di SVR4.

STRUTTURA DEL DIRECTORY HELP

La maggior parte dei comandi del sistema di help risiede nel directory **/usr/bin**, mentre i suoi file di dati sono memorizzati nel directory **/usr/lib/help** e nei relativi subdirectory.

```
$ cd /usr/lib/help
$ ls -F
HELPLOG*    checklen*   default     ge          lib/        un
ad          cm          defnlen*    glossary*   list*       ut
admgloss*   cmds        delete*     he          locking*    vc
admstart*   co          editcmd*    helpclean*  prs
bd          db/         extract*    interact*   rc
cb          de          fetch*      keysrch*    replace*
$
```

I file con nomi brevi, come **cm**, **he** e **un**, appartengono alla parte meno recente di help associata al sistema **sccs**, mentre i comandi eseguibili rappre-

sentano i componenti base eseguibili che costituiscono la parte più recente di help.

La funzione di utility **HELPLOG** è associata con una attività di memorizzazione della storia dell'utilizzo del sistema di help. Il directory **lib** contiene i componenti eseguibili del sistema di help, mentre **db** rappresenta il directory principale che memorizza il testo di aiuto che appare su video.

```
$ ls -F db
descriptions.a    glossary.a    screens.a
examples.a        options.a     tables/
$
```

I file che terminano in **.a** contengono la descrizione del comando. Il subdirectory **tables** fornisce maggiori informazioni di aiuto, in una forma più compatta di quella presentata dai file ***.a**. Poiché questi file sono letti da software, non dovete scriverli direttamente.

MODIFICA DEL DATABASE DI AIUTO

La funzione di help in linea fornisce gli strumenti per ottimizzare l'uscita e aggiungere informazioni ai nuovi comandi e parole chiave, ma potete utilizzarli solo se siete supervisori. Il comando **/etc/helpadm** (help administration) permette di gestire il database di aiuto e vi suggerisce tutti i cambiamenti necessari per installare un riferimento completo. Quando rispondete ai prompt di **helpadm**, il sistema spesso lancia un editor per conto vostro. Per utilizzare il vostro editor abituale, assicuratevi di avere inizializzato correttamente la variabile di ambiente **EDITOR**.

```
$ echo $EDITOR
/usr/bin/vi
$
```

Se questa variabile non è inizializzata, **ed** risulta l'editor di default.

Il comando **/etc/helpadm** vi suggerisce le modalità che dovete seguire per aggiornare il database di aiuto, come mostrato in Figura 9.7. Il formato del menu è simile a quello utilizzato nelle normali finestre di help.

Potete selezionare le voci **starter**, **glossary** o **commands** per aggiornare le corrispondenti sezioni del sistema di aiuto. Inoltre, potete attivare o disattivare la memorizzazione della storia dell'utilizzazione del sistema di help; infine introducete **q** per ritornare allo shell. Quando passate a un menu di livello inferiore, venite informati sugli aspetti del sistema di aiuto che potete aggiornare, come indica la Figura 9.8. Se a questo punto fate una scelta, il comando **helpadm** esegue **EDITOR** utilizzando il testo come contenuto e potete apportare queste modifiche quando volete.

```
# /etc/helpadm
        helpadm: UNIX System On-Line help Administrative Utilities

These software tools will enable the administrator to change
information in the help facility's database, and to monitor use of
the help facility.

        choice      description

          1         starter

          2         glossary

          3         commands

          4         prevent recording use of help facility

          5         record use of the help facility

          q         quit

Enter choice>
```

Figura 9.7 Menu per la gestione di help.

Ognuna delle altre guide a menu del sistema **helpadm** vi permette di aggiungere o cambiare il contenuto corrente del database e le modalità di impiego si dimostrano molto semplici, anche se solo il supervisore ha il diritto di modificarlo.

LE ALTRE SEZIONI DEL MANUALE

Il formato delle pagine nelle restanti sezioni del manuale differisce leggermente dagli standard della Sezione 1 che abbiamo esaminato precedentemente. Le differenze si verificano principalmente nella *Synopsis* e la maggior parte degli utenti non ha validi motivi per referenziare queste sezioni del manuale. In ogni caso, analizziamone il formato di impaginazione.

Nelle Sezioni 2 e 3 la parte *Synopsis* specifica le convenzioni del linguaggio di programmazione C e i parametri che sono necessari per chiamare le funzioni documentate. Queste pagine includono anche una parte *Return Value* (valore di ritorno) che ha notevole importanza per il progettista. Nelle Sezioni 4 e 5 la parte *Synopsis*, se è presente, specifica al programmatore le modalità di sviluppo di software. Le pagine di manuale della Sezione 6 sono simili a quelle della Sezione 1 perché referenziano altri comandi utenti, mentre le pagine di manuale delle Sezioni 7 e 8 raramente includono una parte *Synopsis* perché non è possibile descrivere in modo conciso il contenuto di queste sezioni.

GENERATORE DI LINEE DI COMANDO

Alcuni sistemi SVR4 dispongono di un nuovo *front end* per l'esecuzione dei comandi denominato *generatore di linee di comando*. Questo strumento non è ancora ben conosciuto, ma fornisce tutte le utility di un sistema di aiuto, tra cui la ricerca delle parole chiave, i riassunti sulla sintassi dei comandi e una guida a menu. È in grado anche di eseguire i comandi da una interfaccia utente guidata con menu, favorendo così la stesura sintatticamente corretta dei comandi; visualizza prompt più espressivi ed esegue verifiche di errore prima di accettare i comandi. L'utility non è paragonabile a uno shell, ma si tratta di un programma applicativo che si interfaccia con lo shell, poiché legge i dati in ingresso e passa i comandi corretti allo shell di login perché li esegua. Il sistema può essere utilizzato con facilità anche senza avere acquisito una particolare esperienza e senza uso dei manuali, poiché comprende una funzione di aiuto in linea (che non tratta comunque tutti i comandi della Sezione 1 del manuale).

Esistono diversi strumenti di questo tipo, come il sistema AT&T Assist; alcuni possono funzionare sotto X Window System. Consultate la documentazione del vostro sistema per verificare la disponibilità di uno di questi prodotti, e conoscerne le caratteristiche dettagliate.

Capitolo 10

Calcolo ed elaborazione numerica

L'*elaborazione numerica* ha contribuito notevolmente alla diffusione dei calcolatori e rappresenta tuttora un'area di fondamentale importanza per chi si occupa di sistemi di elaborazione. Nonostante che nel campo dei piccoli computer le maggiori attenzioni siano state rivolte all'elaborazione di testi e di grafici, non è da trascurare il contributo attivo che l'elaborazione numerica fornisce nelle operazioni di calcolo.

Il sistema UNIX fornisce molti strumenti di calcolo: a livello più semplice, il comando **expr** (trattato nel Capitolo 8) permette di eseguire calcoli aritmetici di limitata complessità, mentre, a livello più sofisticato, il linguaggio di programmazione C non pone confini alle possibilità di elaborazione numerica. Tra i due estremi, si inserisce una serie di strumenti intermedi, atti a facilitare le operazioni di calcolo che, in ambiente SVR4, sono "line-oriented" e risultano più linguaggi di programmazione che simulazioni su video di calcolatrici tascabili, come in altri sistemi. X Window System, utilizzando terminali "bit-mapped", mette a disposizione simulatori a tutto video delle calcolatrici. In questo capitolo esamineremo gli strumenti di calcolo che il sistema standard mette a disposizione e che potete utilizzare su terminali remoti ASCII e console di sistema.

Se utilizzate X Window System potrete preferire le applicazioni **xcalc** (*X calculator*) oppure **hexcalc** (*hexadecimal calculator*) che vengono abitualmente previste in X; queste applicazioni forniscono all'utente una intuitiva interfaccia con mouse, ma perdono la tradizionale programmabilità degli strumenti di calcolo di sistema UNIX.

10.1 Alcune considerazioni sui fogli elettronici

La release standard UNIX non prevede alcun foglio elettronico come Lotus 1-2-3, Microsoft Excel o Multiplan. Nonostante siano stati sviluppati numerosi fogli elettronici per UNIX, nessuno è stato inserito nella versione standard del sistema; tuttavia sono disponibili pacchetti addizionali da varie sorgenti. In ogni caso, se avete pratica di un certo tipo di foglio elettronico potete probabilmente ottenere lo stesso prodotto in versione adattata per UNIX, e anche sotto X. Infatti, molti di questi diffusi strumenti provenienti dagli ambienti Apple o MS-DOS, sono stati trasportati sul sistema UNIX.

10.2 Strumenti di calcolo in programmi di shell

Avete visto che filtri e pipeline sono strumenti molto utili per elaborare grandi blocchi di caratteri, file di testo o database. Gli strumenti per la programmazione in shell, insieme al comando **expr**, permettono di eseguire calcoli anche complessi, ma risultano inefficienti per operazioni numeriche e possono risultare tediosi da usare senza un supporto di programmi eseguibili. Alcuni programmi, nati per essere utilizzati in pipeline di shell, sono stati sviluppati per risolvere i problemi di natura numerica, ma solo pochi risultano attualmente disponibili. Un esempio è costituito dal pacchetto software **stat**, orientato verso calcoli statistici, che non fa parte di molte versioni SVR4, ma può essere spesso acquistato assieme al Graphics Utilities Set.

10.3 Le calcolatrici dc e bc

La versione standard di UNIX dispone di due *calcolatrici* di elevate prestazioni: **dc** (desk calculator) e **bc**. Le due calcolatrici offrono le due diverse interfacce utente delle normali calcolatrici moderne. **dc** utilizza una *notazione postfissa*, più nota come notazione *polacca inversa*, in cui gli operandi precedono l'operatore, per esempio:

```
$ dc
2
3
+
p
5
```

In questo esempio abbiamo inserito da tastiera i numeri **2** e **3**, poi l'operatore aritmetico **+** e da ultimo **p** per visualizzare il risultato. Il calcolatore **dc** genera come risultato il valore 5.

Il comando **bc**, invece, utilizza la *notazione algebrica* o *infissa* in cui l'operatore si trova tra i due operandi, come nelle normali operazioni aritmetiche che si insegnano nelle scuole elementari, per esempio:

```
$ bc
2+3
5
```

In questo caso, abbiamo battuto **2+3** e **bc** visualizza il risultato. Potete utilizzare entrambi i comandi ma, poiché le loro caratteristiche differiscono leggermente, la scelta dipende dal tipo di operazione che volete eseguire. Non esistono limitazioni di *precisione* delle elaborazioni numeriche, che potete eseguire in qualunque base (*radice*), oltre a quella decimale. Il comando **dc** è "stack-oriented", cioè tutte le operazioni avvengono su un unico stack di variabili, mentre **bc** è "procedure oriented", cioè potete definire procedure locali e assegnare valori alle variabili in fase di elaborazione. Confrontando questi due strumenti, **bc** include istruzioni e operatori logici che lo rendono uno strumento di programmazione più completo di **dc**.

10.4 Il comando dc

In fase di esecuzione, il comando **dc** interpreta nel suo linguaggio interno le istruzioni contenute nello standard input. Potete anche specificare il nome di un file sulla linea di comando, che **dc** legge completamente, prima di passare alla lettura dei comandi eventualmente contenuti nello standard input. È indifferente specificare le istruzioni in un file o nello standard input, ai fini dell'interpretazione da parte di **dc**, ma il carattere di fine file inserito da tastiera (CTRL-D) interrompe l'esecuzione di **dc**, a differenza invece di quanto accade con la fine del file di ingresso. Il comando **q** (quit) pone termine all'esecuzione di **dc** da entrambe le origini di input.

Un'istruzione **dc** può essere un numero, che viene inserito nello stack di **dc**. I numeri possono essere espressi nella notazione decimale; quelli negativi sono preceduti non dal segno −, ma dal carattere di sottolineatura _. Per esempio, i seguenti numeri sono corretti per **dc**:

```
123
123.5
123.45678901234455
_23.4
```

Il comando **dc** riconosce gli operatori aritmetici + (addizione), − (sottrazione), / (divisione), * (moltiplicazione), % (resto) e ^ (esponente), che specificano l'operazione da compiere sui due operatori che si trovano in cima allo stack, e memorizzano il risultato dell'operazione nelle posizioni dello stack dei due operandi.

Esistono poi molti comandi speciali che eseguono operazioni di controllo, come per esempio il comando **p** (print), che visualizza il valore contenuto in cima allo stack, senza modificarlo.

```
$ dc
3
4
*
p
12
q
$
```

In questi esempi, ogni comando occupa una linea di input distinta, ma **dc** accetta anche linee che contengono più comandi. I numeri sono separati da un carattere spazio, ma altre combinazioni di comandi sono lecite. Un modo alternativo per ottenere gli stessi risultati è il seguente:

```
$ dc
3 4*pq
12
$
```

Inoltre, il comando **dc** può eseguire sequenzialmente più operazioni se lo stack contiene un sufficiente numero di valori. Poiché ogni operazione memorizza il risultato nelle stesse posizioni dello stack occupate dai numeri che ha utilizzato, una operazione successiva impiega quel risultato e il numero sottostante nello stack. Per esempio:

```
$ dc
3
4
7
+ - p
-8
```

In questo caso, l'operatore + addiziona i due numeri che si trovano in cima allo stack (7 e 4) e memorizza nello stack il risultato (11), mentre l'operatore - sottrae 11 da 3 ottenendo -8.

Il comando **dc** impiega una precisione di calcolo (numero di cifre decimali) che dipende dalle necessità contingenti. La relativa pagina di manuale specifica che tutti i calcoli vengono eseguiti su numeri interi, a meno che non utilizziate il comando **k** per modificare il *fattore di scala*, ma questa affermazione non è del tutto esatta. Gli operatori di addizione e sottrazione operano su tutte le cifre che compongono i numeri in ingresso e solo gli operatori di moltiplicazione e divisione restituiscono un risultato che dipende dal fattore di scala.

Altri comandi di tipo "stack oriented" sono **c**, per ripulire lo stack, **d**, per duplicare l'elemento contenuto in cima allo stack, e **f**, per stampare in ordine tutti gli elementi dello stack.

VARIABILI IN dc

Nessuna operazione permette di estrarre l'elemento contenuto in cima allo stack e di scartarlo. Il comando **dc**, comunque, definisce le variabili *registro* che possono contenere un valore, per cui potete estrarre dallo stack un elemento e memorizzarlo in un registro e, analogamente, inserire nello stack il contenuto delle variabili registro. I nomi dei registri sono indicati da caratteri minuscoli, per cui è possibile definire fino a 26 variabili di registro. Il comando **s** (save) accetta un nome di registro come argomento e trasferisce il contenuto della cima dello stack nel registro specificato. Il comando **l** (lettera *l*) esegue l'operazione inversa: trasferisce il contenuto del registro indicato in cima allo stack, per esempio:

```
$ dc
2.34
4.56
p
4.56
st
p
2.34
lt
p
4.56
```

Per eliminare un valore dalla cima dello stack, utilizzate il comando **s**, che lo trasferisce in un registro inutilizzato; questo registro intermedio permette anche di invertire, se occorre, l'ordine degli elementi contenuti nello stack.

I comandi **s** e **l** possono operare anche con stack ausiliari. Se il nome del registro è una lettera maiuscola, il registro viene considerato come stack, per cui il comando **s** trasferisce il valore contenuto in cima allo stack principale nello stack ausiliario; per esempio:

```
2.34
sT
p
empty stack
5.44
sT
p
empty stack
```

```
IT
p
5.44
IT
p
2.34
```

Se lo stack è vuoto o non contiene un numero sufficiente di elementi per completare una operazione, appare su video il messaggio "empty stack".

Una sequenza di caratteri delimitata da [e] (parentesi quadre) viene considerata una stringa ASCII e viene inserita nello stack:

```
[salve gente]
p
salve gente
```

Questa capacità di memorizzare stringhe ASCII è inusuale per una calcolatrice. Non potete utilizzare queste stringhe per operazioni numeriche, ma il comando **x** (execute) estrae una stringa dalla cima dello stack e la esegue come un comando **dc**; è necessario rimuovere esplicitamente la stringa dallo stack dopo l'esecuzione del comando. L'operatore ! richiama lo shell e permette a **dc** di eseguire il resto della linea in un subshell; al termine dell'esecuzione, **dc** riacquisisce il controllo.

La calcolatrice **dc** mette a disposizione molti altri comandi, tra cui **v** che esegue l'operazione di estrazione di radice quadrata dell'elemento in cima allo stack e memorizza il risultato al suo posto, e diversi comandi che permettono di modificare la base numerica (*radice*) per le successive elaborazioni. Il comando **i** (input) induce **dc** a utilizzare il valore contenuto in cima allo stack come radice per rappresentare i numeri specificati in ingresso, mentre **o** (output) definisce il valore in cima allo stack come nuova radice con cui rappresentare i numeri in uscita. Questi comandi permettono dunque di eseguire conversioni di base ed elaborazioni numeriche con una base diversa da quella di default (base 10); per esempio:

```
2
i
1001
p
9
2
o
p
1001
```

Dal momento in cui cambiate la radice in ingresso, tutti i dati che inserite successivamente devono essere espressi nella nuova base numerica.

Gli utenti esperti di calcolatrici tascabili Hewlett-Packard o dei linguaggi di programmazione FORTH o PostScript, riconosceranno che questo relativamente limitato insieme di comandi è più che sufficiente per le abituali elaborazioni numeriche, e anche per soluzioni iterative di integrali e funzioni trascendenti. Comunque, se non siete pratici della notazione postfissa, **dc** può mettervi in confusione, mentre il programma **bc** può fornirvi un ambiente più familiare.

10.5 Il comando bc

Anche se **bc** è solamente un preprocessore per **dc**, offre molte funzionalità in più come, per esempio, funzioni, operazioni logiche, funzioni matematiche quali **sqrt** (square root o estrazione di radice). Il linguaggio di comando nell'insieme assomiglia al linguaggio C, ma è molto semplificato data la sua natura di interprete. Il comando **bc** è molto semplice da usare e le sue interazioni con **dc** sono invisibili all'utente; viene richiamato con:

```
$ bc
```

La calcolatrice **bc** si mette in attesa silente dei dati di ingresso; termina la sua esecuzione con **quit** oppure con la sequenza di tasti CTRL-D, che segnala la fine del file in input. **bc** legge il contenuto dello standard input e scrive i risultati sullo standard output, per cui è possibile effettuare l'operazione di ridirezione dei file; quando il file termina, **bc** esce, come nell'esempio seguente:

```
$ cat cmd.file
/*i commenti iniziano con barra-asterisco, terminano con asterisco-barra*/
6+5
$ bc < cmd.file
11
$
```

Inoltre, **bc** può accettare come argomento il nome di un file.

```
$ bc cmd.file
```

Questo procedimento differisce dalla ridirezione dello standard input di **bc** da un file. Quando specificate il nome di un file come argomento, **bc** legge il file, interpreta i comandi e poi richiede ulteriore input da terminale. Questo consente di memorizzare comandi e funzioni complesse in un file e quindi utlizzarli direttamente da tastiera.

LA NOTAZIONE DI bc

La calcolatrice **bc** utilizza la notazione *infissa* e la fine della linea di ingresso segnala che il comando deve essere valutato.

Nella notazione di **bc**, i numeri possono avere una lunghezza qualsiasi, contenere il punto decimale (invece della virgola) e, se negativi, il segno – (meno), per esempio:

```
-3.45667
```

Potete utilizzare gli abituali operatori aritmetici, per esempio:

```
$ bc
3.45 + 2
5.45
```

Potete usare variabili assegnando loro nomi di un solo carattere minuscolo; il valore viene assegnato con l'operatore = (uguale), come indicato qui di seguito:

```
w = valore
```

Il carattere **w** indica una qualunque lettera minuscola. Le variabili permettono di memorizzare temporaneamente i numeri che saranno utilizzati in seguito e possono essere referenziate direttamente nelle istruzioni numeriche:

```
$ bc
y=4
3+y
7
```

Le variabili mantengono il loro valore fino a quando non vengono riutilizzate in altre istruzioni di assegnamento.

Inoltre, il comando **bc** tratta gli array di numeri; i delimitatori dell'indice di array sono gli operatori [e]:

```
s[2]=3.3
```

La numerazione degli indici di array parte da 0, per cui in questo esempio abbiamo inizializzato il terzo elemento dell'array **s** al valore 3.3. L'indice di array può essere qualunque espressione che **bc** interpreta come numero. La calcolatrice **bc** per default esegue molte elaborazioni come se i numeri fossero interi, ma potete modificare il fattore di scala assegnando alla variabile *scale* un valore che specifichi il numero di cifre a destra della virgola che **bc** deve valutare nelle sue operazioni. Per esempio:

```
$ bc
6.456/5.678
1
scale = 3
6.456/5.678
1.137
```

Potete utilizzare le variabili *ibase* e *obase*, con le stesse modalità descritte per il comando **dc**, per inizializzare rispettivamente la base numerica dei valori in input e output.

```
$ bc
ibase=2
1001
9
obase=8
1001
11
```

In questo modo potete eseguire conversioni di base o realizzare elaborazioni numeriche in sistemi di numerazione diversi da quello decimale.

La calcolatrice **bc** permette di utilizzare molti altri operatori, oltre ai normali operatori aritmetici che sono riconosciuti da **dc**, come per esempio + + (più più) e − − (meno meno) che, rispettivamente, incrementano e decrementano di una unità il valore di una variabile.

```
s=4
s
4
− −s
3
s
3
```

Questi operatori cambiano il valore della variabile e restituiscono il nuovo valore, per cui in questo esempio:

```
s=4
t[− −s]=3.3
t[3]
3.3
```

t[4] non è stato definito e **s** è pari a 3 dopo l'operazione. Gli operatori + + e − − possono precedere o seguire il nome della variabile; nel primo caso, aggiornano la variabile e restituiscono il nuovo valore, mentre, nel secondo caso, restituiscono il vecchio valore della variabile e poi la aggiornano. Per esempio:

```
s=4
t[s--]=3.3
t[4]
3.3
```

In entrambi gli esempi precedenti il valore finale di **s** è 3, ma l'array **t** è diverso.

ISTRUZIONI E OPERATORI DI bc

Il comando **bc** fornisce ulteriori operatori di *assegnamento* che modificano direttamente una variabile specificata secondo il significato dell'operatore:

```
=+
=-
=*
=/
=%
=^
```

Per esempio, la seguente istruzione di assegnamento

```
s =- 3
```

equivale a

```
s = s - 3
```

Sequenze di operazioni raggruppate all'interno dei delimitatori { e } (parentesi graffe) vengono interpretate da **bc** come un unico oggetto e possono essere usate come una singola istruzione. Per esempio:

```
{ s=3 ; y=4 }
```

L'istruzione nel suo insieme restituisce qualsiasi valore che viene restituito dalla sua ultima parte componente. Per esempio:

```
{ s = 3 + 2 ; s }
5
```

Il raggruppamento di operazioni si dimostra particolarmente utile nell'impiego degli *operatori logici*, che valutano o meno il contenuto delle parentesi in base all'esito di precedenti esami di condizioni.

Il comando **bc** supporta gli operatori logici **if**, **for** e **while** che hanno lo stesso significato degli analoghi operatori utilizzati nel linguaggio di programmazione shell, ma differiscono nella sintassi. Per esempio, la sezione

condizionale di una istruzione **if** è racchiusa tra le parentesi (e) e, se la condizione è vera, vengono valutate le istruzioni contenute tra le parentesi graffe, per esempio:

```
s=3
if ( s == 3 ) {
        s = + 2
}
s
5
```

Questo esempio ci permette di fare alcune osservazioni interessanti. In primo luogo, se viene aperto un operatore di raggruppamento come (o { , le istruzioni per il comando **bc** possono occupare più linee, fino a che l'operatore di raggruppamento non viene chiuso con) o } . Il comando **bc** non richiede di completare l'istruzione, come invece fa lo shell visualizzando il prompt PS2 ma, se l'istruzione non viene completata in modo corretto, **bc** evidenzia l'errore con un breve messaggio.

```
syntax error on line 4, teletype
```

Il messaggio indica il numero di linea che contiene l'errore e il terminale da cui avete inserito i dati in ingresso, che **bc** identifica, in questo caso, come "teletype" (telescrivente).

In secondo luogo, gli operatori logici == (uguale), <= (minore o uguale), >= (maggiore o uguale), != (diverso), > (maggiore) e < (minore), possono essere utilizzati all'interno della sezione condizionale della struttura **if**, per specificare la condizione che deve essere esaminata. Se la condizione esaminata risulta vera, il comando esegue le istruzioni contenute nelle parentesi graffe; al contrario, se la condizione esaminata risulta falsa, il comando non esegue le istruzioni contenute nelle parentesi graffe.Tenete presente che, se la struttura **if** non comprende la sezione **else** e volete prendere qualche altra azione in caso di esito negativo, dovete riscrivere la sezione condizionale invertita (cioè con significato negativo). La condizione viene esaminata una sola volta, così come una sola volta vengono eseguite le istruzioni contenute nelle parentesi graffe, se la condizione è vera. Quando utilizzate una struttura di questo tipo, vi consigliamo di indentare le linee interne alle singole sezioni, per favorire la leggibilità delle istruzioni, come abbiamo fatto in questi esempi.

La struttura **while** presenta un formato simile a quello della struttura **if**, con la differenza che inizia un ciclo che esegue ripetutamente le istruzioni contenute nelle parentesi graffe fino a quando la condizione risulta falsa (cioè l'esito del test è negativo). All'interno del ciclo devono essere presenti istruzioni che modificano la variabile esaminata nella sezione condizionale, altrimenti il ciclo continua indefinitamente.

```
s=4
while ( s > 0 ) {
        s
        s = - 1
}
4
3
2
1
```

La struttura **for** è una versione più completa della struttura **while**; all'interno delle parentesi tonde comprende tre distinte parti, separate tra loro dal carattere ; (punto e virgola). La prima parte inizializza la variabile condizionale, la parte centrale rappresenta la sezione condizionale identica a quella utilizzata nel comando **while**, la terza parte modifica la variabile condizionale. Possiamo scrivere il comando precedente come

```
for ( s = 4 ; s > 0 ; - -s ) {
        s
}
```

Il risultato è lo stesso, ma il raggruppamento logico delle operazioni sull'indice all'interno dell'istruzione **for**, rende questo comando più immediatamente comprensibile di **while**. L'operatore **for** viene di solito utilizzato quando la variabile indice è interessata solo nel ciclo stesso, mentre la struttura **while** viene preferita quando la variabile condizionale viene modificata a seguito di cause complesse o dipende da fattori esterni al ciclo. Non esiste alcuna limitazione nel numero di istruzioni contenute all'interno delle parentesi { e } , e nell'annidamento delle strutture **while** e **for**.

Potete interrompere l'esecuzione prima della condizione d'uscita del ciclo tramite l'istruzione **break**. Per esempio:

```
for ( s = 4 ; s > 0 ; - -s ) {
        if ( s < = 2 ) break
        s
}
4
3
```

In questo caso, **break** determina semplicemente la conclusione del ciclo e l'esecuzione dell'istruzione successiva alla fine del **for**. Se esistono cicli multipli annidati, **break** interrompe l'esecuzione di quello più interno; nessun operatore è in grado di interrompere l'esecuzione di tutti i cicli annidati. All'interno dei programmi **bc** non potete utilizzare il comando **quit**, perché il sistema lo esegue immediatamente quando lo legge in ingresso.

FUNZIONI IN AMBIENTE bc

Il comando **bc** permette di definire le *funzioni*. Una funzione è una procedura che possiede un nome, accetta alcuni argomenti e restituisce un valore quando viene chiamata. Potete definire una funzione tramite il comando **define**, seguito dal nome, dagli argomenti racchiusi tra parentesi tonde e dal corpo della funzione racchiuso tra parentesi graffe.

```
define x ( a, b ) {
        for ( s = a ; s < b ; + +s ) {
                s
        }
        return ( 22 )
}
```

Le funzioni vengono identificate con nomi di un carattere, proprio come le variabili. Gli *argomenti* che compaiono sulla linea di definizione della funzione, rappresentano i parametri formali che potete utilizzare come variabili all'interno della funzione; il loro numero non è soggetto a limitazioni, anche se è meglio non eccedere. All'interno della funzione potete impiegare anche variabili che sono state dichiarate esternamente. Nel precedente esempio, **s** rappresenta una variabile esterna che cambia il suo valore dopo l'esecuzione della funzione. Le variabili locali, visibili solo nell'ambito della funzione, devono essere *dichiarate* all'interno della funzione tramite l'istruzione **auto**. Il prossimo esempio utilizza una variabile locale **s**:

```
define x ( a, b ) {
        auto s
        for ( s = a ; s < b ; + +s ) {
                s
        }
        return ( 22 )
}
```

La variabile **s** definita esternamente non modifica il suo valore originale quando si esce dalla funzione.

Per eseguire una funzione, richiamatela specificando il nome di funzione e il corretto numero di variabili racchiuse tra parentesi e separate da una virgola. I parametri attuali – cioè i valori che gli argomenti possiedono al momento della chiamata – sostituiscono i corrispondenti parametri formali specificati in fase di definizione della funzione. Per esempio, la funzione dell'esempio precedente deve essere richiamata con:

```
x( 3, 6 )
```

e il risultato è

```
x( 3, 6 )
3
4
5
22
```

I numeri 3, 4 e 5 sono ottenuti dalla valutazione ripetuta dell'espressione **s** all'interno di ogni iterazione del ciclo **for**, mentre il numero 22 rappresenta il valore restituito dalla funzione.

Potete memorizzare in una variabile il valore di ritorno della funzione, se intendete impiegarlo successivamente:

```
w = x( 3, 6 )
3
4
5
```

Ora la variabile **w** contiene il valore 22. Da questi esempi si deduce che una sola istruzione di chiamata di funzione sostituisce una parte di programma che può risultare complessa o che viene impiegata ripetutamente. Le funzioni costituiscono generalmente parti di programmi di notevole dimensione memorizzati su file, che vengono letti in ambiente **bc** quando sono necessari.

Esistono alcune funzioni matematiche predefinite che possono essere incluse quando lanciate il comando **bc**; se specificate per esempio l'opzione **-l** (library), caricate la libreria matematica:

```
$ bc -l
```

La libreria matematica include le seguenti funzioni: **s(x)** calcola il seno di x, **c(x)** calcola il coseno di x, **e(x)** restituisce l'esponente di x, **l(x)** restituisce il logaritmo di x, **a(x)** calcola l'arcotangente di x e **j(n,x)** restituisce la funzione di Bessel di x. Quando la libreria matematica è caricata, dovete fare attenzione a non utilizzare i nomi delle sue funzioni per definire nuove funzioni.

La calcolatrice **bc** si può ritenere estremamente utile sia nell'esecuzione di somme e prodotti che per altri calcoli più complessi e, poiché può eseguirli con la precisione desiderata, il suo impiego è polivalente. Il comando **bc** legge dallo standard input le espressioni da interpretare, pertanto può essere utilizzato in procedure di shell e, con l'operatore ` (accento grave), negli assegnamenti alle variabili di shell. Tuttavia, né **bc** né **dc** sono strumenti ottimizzati per la lettura di dati dai file, anche se possono leggere con facilità programmi memorizzati su file. L'ingresso per **bc** e **dc** proviene dal terminale o può essere specificato in fase di esecuzione come argomento di funzioni chiamate.

10.6 Il comando awk

Esiste uno strumento che si dimostra molto utile sia nelle operazioni di calcolo che nell'analisi sintattica, si tratta del comando **awk** e del linguaggio di controllo associato. Il termine **awk** è un acronimo delle iniziali dei tre sviluppatori (Aho, Weinberger e Kernighan). Nonostante alcune critiche per una certa difficoltà nel suo utilizzo, **awk** è di fatto un linguaggio di programmazione molto potente ed elegante, che si dimostra molto efficiente e versatile in confronto agli altri strumenti di elaborazione di testi.

Un libro recente su **awk** da parte dei suoi tre implementatori dovrebbe consentire una migliore comprensione dello strumento, la cui accettazione è stata certo ostacolata dalla documentazione estremamente concisa.

I sistemi SVR3 e SVR4 offrono spesso due versioni di **awk**, chiamate **nawk** (new awk) e **oawk** (old awk). Uno o l'altro di questi strumenti viene connesso al nome **awk** nel directory **/usr/bin**; la scelta tra i due varia tra le diverse implementazioni; dovete controllare il vostro sistema per questo aspetto. Lo strumento **oawk** è la venerabile vecchia versione che fu inclusa in UNIX molti anni fa, è molto lento e inefficiente nelle operazioni, ma oramai molte procedure sono state scritte per questa versione di **awk**. Lo strumento **nawk** è una versione grandemente migliorata e ottimizzata inclusa per la prima volta in SVR3; è molto più veloce e include più potenti operatori e comandi interni di **oawk**, ma la vera ragione della sua esistenza è il supporto che fornisce per le configurazioni di caratteri internazionali. Sfortunatamente, **nawk** non risulta compatibile al cento per cento con **oawk**, quindi alcuni programmi **oawk** non funzionano correttamente con **nawk**. La trattazione di **awk** in questa sezione si applica ad ambedue le versioni; alcuni miglioramenti di **nawk** sono trattati alla fine del capitolo.

CONCETTI BASE DI awk

Il comando **awk** esegue una scansione di un elenco di file in input alla ricerca di linee che corrispondono a un insieme di *modelli* specificati; per ciascun modello riconosciuto viene eseguito un insieme specificato di *azioni*. Queste azioni possono riguardare trasformazioni di alcuni campi all'interno di una linea oppure operazioni aritmetiche sui valori dei campi. Lo strumento **awk** è un linguaggio di programmazione che riassume le caratteristiche della programmazione di shell, del linguaggio **bc** e del linguaggio di programmazione C. È completamente interpretato come **bc**, fornisce variabili per i campi di ogni linea in input da **$1** a **$n**, come per gli argomenti di una linea di comando in ambiente shell, e dispone di alcuni operatori di controllo e di stampa simili a quelli del linguaggio C.

Per usare il comando **awk**, dovete creare un *programma* che specifica una lista di sezioni di *modelli* di ricerca e *corrispondenti* azioni da intraprendere. Il comando **awk** legge i file in ingresso e, per ogni linea in ingresso che

soddisfa al criterio di selezione di un modello, esegue l'azione a esso associata. La linea di comando per lanciare **awk** è dunque:

```
$ awk -f programma lista-nomifile
```

Al comando **awk** l'insieme dei modelli di ricerca e delle azioni viene specificato in un file il cui nome compare dopo l'opzione **-f** (file). Altri nomi di file indicati sulla linea di comando referenziano i file di testo che **awk** legge come input; se non appare alcun nome di file, **awk** legge lo standard input. Potete anche includere lo standard input in un elenco di nomi di file utilizzando il carattere speciale - (meno). Per esempio:

```
$ awk -f programma file1 - file2
```

In questo caso, **awk** legge il programma dal file **programma**, legge e processa **file1**, legge lo standard input fino a quando arriva al carattere di fine file e poi processa **file2**.

Potete inoltre inserire direttamente nella linea di comando il programma, se non utilizzate l'opzione **-f**; in questo caso, il programma appare letteralmente dopo il nome **awk**, ma prima della lista dei nomi di file. Comunque, anche agli esperti occorrono diversi tentativi prima di ottenere un programma **awk** corretto, per cui il formato in linea di comando viene raramente utilizzato tranne in file di shell accuratamente preparati e provati. L'inserimento in linea di comando del programma consente di risparmiare un file in linea nell'applicazione, ma risulta meno gestibile nel caso di elaborazioni complesse, per cui accertatevi che il programma sia corretto prima di abbandonare il formato **-f file**.

LETTURA DELLE LINEE DI INGRESSO DA PARTE DI awk

Il comando **awk** considera ogni linea che legge dai file o dallo standard input suddivisa in campi, separati da caratteri spazio, o da qualunque altro carattere scelto come delimitatore di campo, specificando sulla linea di comando **awk** l'opzione **-F** (field), seguita dal nuovo delimitatore. Per esempio, per avere come delimitatore di campo il carattere : (due punti), scrivete la seguente linea di comando:

```
$ awk -F: -f programma file
```

Potete far riferimento ai campi della linea in ingresso mediante la notazione **$1**, **$2**, **$3**, ecc. Il primo campo sulla linea è **$1**, mentre la variabile speciale **$0** fa riferimento all'intero record (linea), non suddiviso in campi.

MODELLI E AZIONI DI awk

Le coppie modello-azione definiscono le operazioni che **awk** esegue sulle linee e sui campi che legge. Il formato di queste coppie modello-azione è il seguente:

modello { *azione* }

La parte *azione* viene separata dalla parte *modello* racchiudendola fra parentesi graffe. Un modello senza corrispondente azione seleziona le linee che lo soddisfano e le emette in standard output; un'azione senza modello associato viene eseguita su tutti i record del file. In altre parole, un modello mancante viene soddisfatto da tutte le linee del file. La parte azione può includere una serie complessa di operatori, tra cui variabili e operatori logici, simili a quelli di **bc**, e le variabili di campo **$1**, **$2**, ecc., usate come dati in input.

Un operatore utilizzato abitualmente è **print**, che riporta i suoi argomenti sullo standard output. L'azione

```
{ print $2, $1 }
```

inverte i primi due campi in ingresso e li trascrive in output. Se il dato in ingresso è

```
$ cat in.file
salve ciao ancora
111 222
trenta quaranta
$
```

e il programma **awk** è

```
$ cat awk.prog1
{ print $2, $1 }
$
```

l'uscita risulta

```
$ awk -f awk.prog1 in.file
ciao salve
222 111
quaranta trenta
$
```

In questo esempio, gli argomenti da stampare sono separati da una virgola, che fa inserire a **print** il delimitatore di campo tra i dati in uscita. Se la virgola viene omessa, **$1** e **$2** vengono inviati in uscita senza alcun separatore.

Ricordate che il comando **awk** esegue l'azione se la linea di ingresso soddisfa il modello di ricerca e, se il modello manca, l'azione viene intrapresa su ogni linea di ingresso, come nell'esempio precedente.

I modelli sono espressioni regolari, o sequenze di espressioni regolari separate dagli operatori **!** (not), **||** (or logico), **&&** (and logico), oppure da parentesi di raggruppamento. Dovete racchiudere ogni espressione regolare fra caratteri / (barra) come in **ed**. Per esempio, il programma **awk**

```
/salve/ { print $2, $1 }
```

agisce sul file **in.file** che abbiamo creato prima, come in questo esempio:

```
$ awk -f awk.prog2 in.file
ciao salve
$
```

Solo una linea del file in ingresso soddisfa il modello di ricerca **/salve/**, per cui il comando **awk** visualizza solo quella linea, come imposto dalla parte azione. Poiché il programma non specifica nessuna azione per le altre linee, il comando **awk** termina la sua esecuzione. In ogni caso, se specificate più istruzioni modello-azione, **awk** permette di trattare casi diversi:

```
/salve/ { print $2, $1 }
/trenta/ { print $1, $2, "e altro" }
```

Questo programma produce risultati differenti.

```
$ awk -f awk.prog3 in.file
ciao salve
trenta quaranta e altro
$
```

L'intero programma **awk**, cioè tutte le coppie modello-azione che sono specificate, viene eseguito per ogni linea di ingresso. Questo esempio illustra che il comando **print** accetta anche come argomento stringhe di caratteri delimitate da virgolette e le riporta invariate in uscita.

Potete collegare i modelli con operatori logici per ampliare i criteri di selezione. Per esempio, il programma

```
/salve/||/111/ { print "buongiorno", $1, $2 }
```

produce

```
$ awk -f awk.prog4 in.file
buongiorno salve ciao
buongiorno 111 222
$
```

L'operatore || (or) fa in modo che **awk** esegua l'azione se la linea di ingresso soddisfa uno dei due modelli, l'operatore **&&** fa in modo che **awk** esegua l'azione solo se la linea di ingresso soddisfa entrambi i modelli e l'operatore ! fa in modo che **awk** esegua l'azione solo se la linea di ingresso non soddisfa il modello specificato. L'operatore ! precede l'espressione regolare che indica il modello e non deve essere impiegato come separatore di due espressioni. Per esempio:

```
!/salve/ {
        print "no salve"
}
```

Naturalmente sono permesse espressioni regolari più complesse.
Questo comando

```
/^[Ss1]/ { print "buongiorno", $0 }
```

produce

```
$ awk -f awk.prog5 in.file
buongiorno salve ciao ancora
buongiorno 111 222
$
```

Notate l'uso di **$0** per visualizzare l'intera linea di ingresso originale.

Potete specificare l'azione su più linee, ognuna delle quali contiene una singola istruzione, e racchiudere l'intera azione in parentesi graffe. Per esempio:

```
$ cat awk.prog6
/salve/ {
        print $2
        print "altra"
        print $1
}
$ awk -f awk.prog6 in.file
ciao
altra
salve
$
```

Ogni istruzione **print** genera una linea in output, ma tutto l'output viene prodotto dalla sola linea in ingresso che soddisfa il modello **/salve/**.

OPERAZIONI NUMERICHE CON awk

Come mostrano questi semplici esempi, **awk** dimostra eccellenti capacità di analisi e trasformazione di stringhe di caratteri e risulta particolarmente efficiente nell'elaborazione programmata di testi, rispetto ad altri strumenti come per esempio l'editor **sed**. La sua reale potenzialità risiede soprattutto nella possibilità di definire azioni complesse mediante gli operatori logici o aritmetici; **awk** impiega intelligentemente le variabili numeriche e possiede strumenti adatti alle operazioni di conversione fra stringhe di caratteri e numeri. Il comando **awk** consente di usare l'aritmetica in maniera diversa da **bc**, poiché, una volta selezionato un sottoinsieme di linee di un file che soddisfa il modello di ricerca, può identificare alcuni campi che contengono l'informazione chiave e operare su altri campi che contengono il dato. Inoltre, **awk** può convertire l'input dal formato carattere al formato numerico e viceversa; può formattare meglio l'uscita rispetto a **bc**, che si dimostra però più efficiente, più preciso e più semplice da apprendere. Ognuno di questi due strumenti assume comunque in UNIX una adeguata collocazione.

Il comando **awk** interpreta automaticamente i campi di una linea in ingresso come stringhe di caratteri o valori numerici in relazione al contesto corrente. Per esempio, la funzione **length** restituisce il numero di caratteri che compongono un campo della linea di ingresso, ma **awk** può anche interpretare il campo come numero e assegnare il suo valore alle variabili numeriche. Per esempio:

```
$ cat awk.prog7
{
        s + = $2
        print $2, "lunghezza = " length($2), "s = " s
}
$
```

Questo programma somma i valori numerici del secondo campo di ogni linea di ingresso. Il comando **awk** esegue automaticamente la *conversione di tipo*, senza che sia necessario dichiarare anticipatamente la variabile *s* e il suo tipo. I risultati prodotti in uscita sono:

```
$ awk -f awk.prog7 in.file
ciao lunghezza = 4 s = 0
222 lunghezza = 3 s = 222
quaranta lunghezza = 8 s = 222
$
```

Le stringhe che non possono essere convertite in numeri assumono il valore zero, per cui non si verificano errori né interruzioni del programma. La stringa **quaranta** non può essere convertita in numero, a differenza della

stringa **222**, che il comando **awk** interpreta come numero, a meno che non sia racchiusa tra apici.

Il comando **awk** permette di inizializzare le variabili con un valore, come nel caso di **s=0**, e tratta le variabili come **bc**, con la differenza che il nome non è limitato a un singolo carattere, ma può essere di lunghezza qualunque purché il primo carattere sia una lettera. Potete utilizzare variabili di tipo array, racchiudendo l'indice tra parentesi quadre come in **bc**. Per esempio, queste sono variabili corrette per **awk**:

```
s
S
SS
S1
array1[42]
```

Non siete obbligati a dichiarare o inizializzare le variabili prima di utilizzarle. Il comando **awk** inizializza una variabile a una stringa vuota, ma potete memorizzare in una variabile sia una stringa che un numero. Inoltre, una stessa variabile può modificare il suo tipo quando viene utilizzata in una azione. Per esempio,

```
/salve/ {
        SSS = 34
        print "SSS è", SSS
        SSS = salve
        print "SSS è", SSS
}
```

produrrebbe i seguenti risultati:

```
$ awk -f awk.prog8 in.file
SSS è 34
SSS è salve
```

La definizione automatica del tipo delle variabili semplifica il loro utilizzo, tuttavia **awk** segnala un messaggio d'errore se ne fate un uso improprio.

MODELLI PREDEFINITI PER INIZIO E FINE PROCESSO

Oltre ai modelli descritti da espressioni regolari, che stabiliscono se eseguire le azioni sulla linea di ingresso, il comando **awk** dispone di due modelli predefiniti (built-in), sempre eseguiti, denominati **BEGIN** ed **END**. Se la parola **BEGIN** appare come modello, questo modello viene soddisfatto all'inizio del programma **awk**, prima della lettura di qualsiasi linea in input, e consente quindi di avere il controllo delle elaborazioni prima che **awk** abbia

iniziato qualunque azione. In modo analogo, **END** è un modello che è soddisfatto dalla parte finale del programma, dopo l'ultima linea in ingresso, e consente di avere il controllo dell'elaborazione al termine delle azioni di **awk**. **BEGIN** viene di solito utilizzato per inizializzare le variabili e per altre operazioni inziali, mentre **END** viene impiegato per calcoli finali e generare in uscita informazioni riassuntive. Né **BEGIN** né **END** sono obbligatori in un programma **awk**, ma possono essere inclusi se occorre. Per esempio, questo programma calcola la media di una lista di numeri:

```
BEGIN {
        print "Inizio della elaborazione di dati di ingresso..."
      }
      {
        s + = $1
        n + +
      }
END   {
        print "la media di questi", n, "valori numerici è", s/n
      }
```

L'azione associata con il modello **BEGIN** viene eseguita all'inizio del programma e consiste solo nella stampa di un messaggio, dato che **awk** provvede automaticamente ad azzerare il contatore *n* e il totale *s* in assenza di inizializzazione esplicita. Quindi, viene letta ciascuna linea in ingresso e, poiché l'azione non è associata a nessun modello, l'azione viene eseguita per tutte le linee in input. L'azione somma il primo campo di ogni linea alla variabile *s* e incrementa il contatore di una unità. Terminata la fase di lettura e trattamento di tutte le linee in ingresso, viene eseguita l'azione indicata dopo il modello **END**, che consiste nella visualizzazione dei risultati dei calcoli effettuati.

ISTRUZIONI DI awk

Il comando **awk** fornisce una grande varietà di operazioni consentendo di scrivere programmi complessi che possono combinare elaborazioni numeriche con elaborazioni di stringhe di caratteri. Il formato di base è l'istruzione, che corrisponde a una singola operazione di **awk**, che termina con un ritorno a capo o un punto e virgola. Ad esempio, queste sono istruzioni di **awk**:

```
s + = $1
print $2 $1
```

Inoltre, **awk** considera qualunque sequenza di istruzioni racchiusa tra parentesi graffe come una singola istruzione, per cui anche le azioni associate ai modelli di ricerca sono interpretate come istruzioni.

Anche altre strutture logiche sono istruzioni, per esempio **awk** utilizza la struttura condizionale **if** nello stesso modo in cui la impiega il comando **bc**:

```
if ( parte condizionale ) istruzione
```

Questa struttura causa l'esecuzione dell'istruzione se (**if**) il valore della parte condizionale è vero (diverso da zero). Naturalmente, la parte istruzione della struttura **if** può contenere un numero qualunque di istruzioni **awk**, purché sia delimitata da parentesi graffe. Per esempio:

```
{
        if ( s < 2 ) {
                + +s
                print s
        }
}
```

Questo intero esempio consiste di una sola istruzione **awk** che si compone di altre istruzioni.

È possibile utilizzare diverse forme più semplici di istruzioni, di cui le più comuni risultano gli *assegnamenti* di espressioni aritmetiche alle variabili. Il formato ricalca quello impiegato dal comando **bc**; potete utilizzare gli abituali operatori matematici. Per esempio,

```
x = 3
n + = 3
n+ +
w = 14 / 4 + 32 - ( 14 * 6 ) / 5.2
```

sono tutte valide istruzioni di assegnamento. Il comando **awk** converte i numeri nella notazione in virgola mobile prima di eseguire le elaborazioni, per cui le istruzioni possono trattare sia numeri interi che numeri in virgola mobile.

Il comando **awk** dispone anche di strutture iterative, come per esempio **while** che assume la forma

```
while ( parte condizionale ) istruzione
```

dove *parte condizionale* è una qualsiasi espressione che assume il valore zero (falso) o un valore diverso da zero (vero). Se la parte condizionale è vera, allora viene eseguita la parte istruzione. Per esempio:

```
{
        while ( s < 10 ) {
                s = $1 / 32.3
                + +m
        }
}
```

Le due istruzioni interne alle parentesi graffe vengono eseguite se *s* è minore di 10.

Il comando **for** è simile, ma si compone di tre parti.

```
{
        for ( s = 0 ; s < 10 ; + +s ) {
                s = $1 /32.3
                + +m
        }
}
```

Ogni parte all'interno delle parentesi è separata dalle altre dal carattere ; (punto e virgola). La prima parte può essere usata per inizializzare qualsiasi variabile locale, la seconda è la parte condizionale e l'istruzione successiva al **for** viene eseguita se la parte centrale assume il valore vero. La terza parte viene eseguita una volta terminata l'esecuzione dell'istruzione, ma prima che la parte condizionale sia verificata ancora per il nuovo passaggio nel ciclo, e permette di aggiornare il valore di alcune variabili che compaiono nella parte condizionale. L'istruzione viene eseguita ripetutamente fino a quando la parte condizionale assume il valore falso; ognuna di queste strutture può annidare al suo interno altre strutture condizionali o iterative. Un esempio di quanto detto è il seguente:

```
{ for ( s = 2; s < 10; + +s ) {
        print "ciclo esterno: s è ora", s
        if ( s > 3 && s < 6 ) {
                print "dentro 'if': s è ora", s
        }
    }
}
```

Potete utilizzare l'istruzione **break** all'interno di una struttura **for** o **while** per uscire anzitempo dal ciclo, anche se la parte condizionale rimane vera. Dopo l'esecuzione dell'istruzione **break**, il programma continua dalla prima istruzione che segue la struttura **for** o **while**. Per esempio:

```
{ for ( s = 2; s < 10; + +s ) {
        print "Nel ciclo: s è ora", s
        if ( s = = 4 ) break
  }
  print "ciclo terminato, s è ", s
}
```

In questo modo, il ciclo **for** termina quando **s** è uguale a 4, anche se la sua parte condizionale (**s<10**) è ancora vera. L'istruzione **break** determina la fine immediata del ciclo; se esistono cicli annidati, conclude solo il ciclo più interno, mentre i cicli più esterni continuano.

Analogamente, l'operatore **continue** prosegue l'esecuzione dall'inizio della successiva iterazione del ciclo, anche se non tutte le istruzioni del ciclo sono state eseguite. Per esempio, il seguente programma **awk**

```
$ cat awk.prog9
BEGIN { for ( s = 2; s < 6; + + s ) {
            if ( s = = 4 ) continue
            print "s è ora", s
        }
  }
```

genera questa uscita:

```
$ awk -f awk.prog9
s è ora 2
s è ora 3
s è ora 5
$
```

In questa operazione condizionale

```
if ( s = = 4)
```

l'operatore logico = = (se uguale a) restituisce un valore vero o falso. L'operazione condizionale

```
if ( s = 4)
```

contiene nella parte condizionale un'istruzione di assegnamento che restituisce il valore 4, cioè un valore diverso da zero e quindi vero, e modifica la variabile **s**, a differenza di quanto accade nel primo esempio. L'uso improprio degli operatori = e = = all'interno delle espressioni condizionali si rivela molto frequente e di difficile diagnosi, per cui prestate molta attenzione quando utilizzate le istruzioni condizionali.

Due altri operatori di controllo del flusso di programma in **awk** sono **next** ed **exit**. L'operatore **next** determina la fine dell'azione corrente, per cui **awk** passa a esaminare la successiva linea in ingresso. Analogamente, **exit** induce **awk** a terminare l'elaborazione dell'azione corrente, ma a ignorare anche tutte le rimanenti linee in ingresso e passare a una eventuale sezione **END**. In quest'ultimo caso, se non esiste una sezione **END**, **awk** restituisce il controllo allo shell.

FORMATTAZIONE DELL'USCITA CON awk

Il comando **print** di **awk** può essere molto utile per visualizzare semplici risultati in uscita, come abbiamo mostrato negli esempi precedenti, mentre,

per stampe più complesse, è disponibile il comando **printf** (print function). Questo comando differisce da **print** in quanto consente un pieno controllo sul formato di uscita, implementa l'insieme completo di funzioni che **printf(3)** mette a disposizione nel linguaggio C, e fornisce una funzione simile a quella del comando **printf(1)** descritto nel Capitolo 8. La versione di **awk** supporta un nutrito elenco di potenti e complesse specifiche di conversione dei dati in uscita; considereremo solo le più importanti.

Il comando **printf** accetta due tipi di argomenti: il primo è una *specifica di formattazione*, consistente in una stringa di caratteri che specifica il formato generale dell'uscita e contenente alcune posizioni particolari ove verranno inseriti i dati reali; il secondo argomento di **printf** è un elenco di variabili, il cui numero è pari al numero di posizioni particolari nella specifica di formattazione. Queste posizioni di inserimento sono denominate *specifiche di conversione* e definiscono le modalità di stampa delle variabili. Per esempio,

```
printf "questo è il numero %d e questa è la stringa %s\n", 333, "333"
```

produce

```
questo è il numero 333 e questa è la stringa 333
```

Il primo argomento del comando **printf** è la specifica di formattazione; si compone di diverse parole, che vengono riportate invariate su video, e di due specifiche di conversione. Gli altri argomenti sono le variabili corrispondenti alle specifiche di conversione; gli argomenti sono separati da virgole.

Le specifiche di conversione sono contraddistinte da una stringa che inizia con il carattere % (percentuale), i caratteri immediatamente seguenti definiscono le conversioni di formato. Le specifiche di formato più significative sono **d** (decimale), **s** (stringa), **c** (singolo carattere), **o** (ottale), **x** (esadecimale) e **f** (virgola mobile). Nel precedente esempio, **%d** implica la visualizzazione della variabile come intero decimale, mentre **%s** impone di visualizzare la variabile come stringa di caratteri. Le variabili che appaiono dopo la specifica di conversione vengono visualizzate in accordo con la specifica di formato, una volta eseguita la conversione di tipo.

Il comando **printf** esegue la conversione di tipo in maniera intelligente, impiegando il numero di cifre indispensabile, tuttavia potete alterare la lunghezza di conversione, specificando il numero di cifre desiderato dopo il carattere % e prima dell'indicazione del tipo di conversione. Per esempio,

```
printf "uno = >%1d<, due = >%2d<, tre = >%3d<\n", sss, sss, sss
```

utilizza *almeno* una, due, tre cifre per formattare la variabile **sss** ma, se occorre, **printf** impiega un numero di cifre superiore. In altre parole, se **sss** vale 42, il comando **printf** genera in uscita il seguente risultato:

```
uno = >42<, due = >42<, tre = > 42<
```

Quando il numero di cifre indicate nella specifica di conversione supera le reali necessità, **printf** riempie a sinistra il formato di uscita con caratteri spazio e allinea a destra le cifre. Se il carattere – (meno) segue il carattere % e precede l'ampiezza di campo, l'uscita viene allineata a sinistra all'interno del campo. Per esempio,

```
printf "uno = >%1d<, due = >%2d<, tre = >%-3d<\n", sss, sss, sss
```

genera questa uscita

```
uno = >42<, due = >42<, tre = >42 <
```

Queste specifiche della dimensione di campo permettono di controllare l'incolonnamento dell'uscita, tuttavia tenete presente che il comando **printf** non tronca un numero se la specifica di campo è troppo piccola per il valore che viene stampato; al contrario, espande la specifica di campo in base alle reali necessità.

Per i numeri in virgola mobile, che conterranno un punto decimale in output, potete definire una specifica di campo più complessa, composta da due numeri separati da un carattere . (punto). Il numero a sinistra del punto decimale indica l'ampiezza totale richiesta per il campo in output, mentre il numero a destra del punto indica il numero di decimali che devono comparire in uscita a destra del punto decimale. Per esempio,

```
printf "breve= >%6,1f<, lungo= >%6.4f<\n", 6.345678, 6.345678
```

genera questo risultato:

```
breve= >  6.3<, lungo= >6.3457<
```

I risultati vengono arrotondati correttamente in base all'ampiezza del campo. Il comando **printf** esegue questo tipo di troncamento in uscita solo a destra del punto decimale; viceversa, incrementa il numero di cifre, se occorre, per rappresentare correttamente la parte alla sinistra del punto decimale. Potete specificare il segno – dopo il carattere % anche coi numeri in virgola mobile per allineare a sinistra il numero all'interno del campo. Ancora una volta viene prima rispettata la precisione, quindi vengono inseriti spazi bianchi a destra. Questo comando

```
printf "breve= >%-6.1f<, lungo= >%-6.4f<\n", 6.345678, 6.345678
```

genera in uscita:

```
breve= >6.3  <, lungo= >6.3457<
```

Inoltre, **printf** riconosce altri due speciali caratteri di controllo: **\n** che invia in uscita un ritorno a capo, **\t** che invia in uscita un carattere di tabulazione. Questo comando

```
printf "salve 12\t%c%c\nciao", 3, 4
```

genera in uscita

```
salve 12    34
ciao
```

Il comando **printf** supporta molti altri operatori e specifiche di conversione, ma quelli che abbiamo discusso sono sufficienti per permettere un esteso utilizzo del comando **awk**.

10.7 Approfondimenti

Esistono molti altri strumenti di elaborazione numerica, includendo i numerosi linguaggi di programmazione che possono essere usati per sviluppare applicazioni. Generalmente questi non fanno parte del software fondamentale incluso nel sistema UNIX, e richiedono l'acquisto del pacchetto C Compilation System. Tuttavia, programmi complessi sono quasi sempre scritti con uno di questi linguaggi di programmazione; tratteremo brevemente i più importanti strumenti di sviluppo di software nel Capitolo 26.

IL PROGRAMMA new awk

Il comando **nawk** include molti miglioramenti rispetto a **oawk**, e in un futuro potrà rimpiazzare completamente la versione più vecchia, che potrà essere eliminata quando tutte le vecchie applicazioni saranno state trasportate in **nawk**. Questo probabilmente non avverrà prima di un certo numero di anni, tuttavia i nuovi programmi dovranno essere sviluppati in **nawk**, a meno che debbano essere installati in un sistema che possiede solo **oawk**. Tenete conto che è molto facile scrivere programmi in **nawk** che risultano incompatibili con **oawk**.

Il maggiore miglioramento del linguaggio **nawk** è la disponibilità di un maggior numero di funzioni e variabili predefinite, che consentono l'accesso agli argomenti della linea di comando, al nome del file in ingresso corrente (o standard input), al numero di campi nella linea corrente, al contatore di linea corrente e altre variabili. Presi insieme, questi nuovi operatori compendiano molte delle caratteristiche di **oawk** e **sed**.

In **nawk**, l'argomento della linea di comando **–F** (field), in aggiunta alla dichiarazione di un delimitatore di campo di un solo carattere, consente di

specificare un'espressione regolare, nel formato di **egrep**. Quando l'espressione regolare così specificata identifica una stringa corrispondente nel file in input, **nawk** tratta quella stringa come delimitatore di campo. Questo consente di scrivere operazioni più potenti, anche se più complesse, sui campi di file complessi in input, per esempio:

```
$ nawk -F "^[Hh]ello" -f programma listanomifile
```

Non trascurate di proteggere l'espressione regolare se contiene qualunque metacarattere di shell.

Il comando **nawk** include molte più variabili predefinite di **oawk**, migliorando grandemente la programmabilità. La più importante di queste variabili è **FILENAME**, che contiene sempre il nome del file in trattamento corrente; potete includere il nome di ciascun file nel vostro output, se vi occorre, per esempio:

```
$ cat prog8
/Hello/ {
        print "file", FILENAME, "contiene la stringa", "Hello"
}
$
```

La variabile **RS** contiene il *separatore di record* corrente, usualmente il newline, come negli esempi precedenti. Ogni linea di file in input viene trattata come un record separato, ma se cambiate il valore della variabile **RS** potete trattare file che contengono record dati multilinea, a condizione che i vostri file in input contengano il nuovo valore di **RS** alla fine di ogni record. Se **RS** viene modificato, **nawk** legge le linee multiple dall'input e le concatena prima di cominciare a trattare il record. Solitamente la sezione **BEGIN** del programma provvede a inizializzare **RS** al valore richiesto. Per produrre record multilinea in output potete assegnare un valore alla variabile **ORS** (output record separator), **nawk** scriverà quel valore alla fine di ciascun record in uscita. Abbiate cura di scrivere correttamente i comandi **print** o **printf**, altrimenti il valore **ORS** non sarà corretto.

La variabile **NR** (number) contiene il contatore dei record trattati dall'inizio del programma **nawk**; esso accumula tutti i record anche su più file, se ne sono stati specificati diversi nella linea di comando. Anche diverse altre variabili sono disponibili in **nawk**.

Inoltre, un notevole numero di funzioni predefinite consente operazioni di indicizzazione di caratteri e di sottostringhe sui record di ingresso. Un'operazione **getline** consente di abbandonare il record corrente e leggere il successivo record in input, potete leggere (o abbandonare) record dati senza lasciare la corrente operazione modello-azione. Potete anche scrivere in file anziché nel normale standard output; potete magari tenere una storia delle operazioni di **nawk** nel trattamento dei file, senza inserire questo output nel

vostro normale file in output. A questo scopo, dovete solo scrivere un semplice comando **print** o **printf** come al solito, aggiungendo *>nome-file* alla fine. Per esempio:

```
print "Output in un altro file" > "/tmp/file.storia"
```

Questo scriverà la stringa nel file **file.storia** senza interferire col normale I/O del programma. Se il nome del file contiene un nome di percorso assoluto, deve essere protetto, altrimenti **nawk** segnala un messaggio d'errore. Potete combinare questa caratteristica con la funzione **getline** per leggere da un qualsiasi altro file in input, per esempio:

```
getline < altro.in.file
```

Questa operazione cambia il valore corrente di **$0** con quello della prima linea del file **altro.in.file**, scartando il valore precedente; usi successivi di questa operazione di input leggeranno successive linee dal file **altro.in.file**. Se l'intero file viene letto prima della fine del programma **nawk**, l'operazione **getline** non ha successo e il record corrente non viene cambiato, tuttavia **nawk** non fornisce alcuna segnalazione di anormalità. Questa è una causa comune di errori: tenetene conto nell'uso del comando **getline**.

Infine, **nawk** consente la definizione di funzioni all'utente. Queste funzioni possono essere usate come le regolari funzioni predefinite, a condizione che vengano definite all'inizio del programma, rimpiazzando una coppia modello-azione, per esempio:

```
$ cat awk.prog9
function putout( x, y ) {
        print "x è >>" x "<<"
        print "y è >>" y "<<"
}
{
        putout ( $0, $1 )
}
$
```

Dovete iniziare la definizione di una funzione con la parola chiave **function**, seguita dal nome, e dagli argomenti formali (tra parentesi), nel numero che vi occorre. La funzione stessa deve essere racchiusa tra parentesi graffe. Dopo la definizione di tutte le vostre funzioni potete continuare con la sezione **BEGIN** oppure con la normale coppia modello-azione; la parte azione può includere il richiamo delle funzioni che avete definito.

Dopo avere appreso le basi della programmazione **awk**, potrete usare con notevoli vantaggi le nuove caratteristiche di **nawk**. Consultate la *man page* di **nawk** per avere maggiori informazioni.

Capitolo 11

I processi

Prima di analizzare gli aspetti multiprocesso del sistema UNIX, chiariamo che con il termine *processo* (task) si intende una istanza di un programma in esecuzione. Lo shell di login è un esempio di processo, che si mette in attività quando entrate nel sistema e cessa la sua attività quando uscite; nel periodo in cui sono in esecuzione, anche i comandi che lanciate sono processi. I processi hanno molte proprietà e per controllarle sono disponibili diversi comandi. In questo capitolo tratteremo i problemi associati con la multiprogrammazione e daremo alcune direttive su come controllare l'ambiente di esecuzione.

Per esempio, il comando

```
$ cat /etc/passwd
```

genera un processo che rimane attivo fino a quando l'operazione **cat** non termina. Se definite un pipeline di shell mediante l'operatore ¦ , ogni com-

ponente del comando rappresenta un processo separato. La linea di comando

```
$ cat /etc/passwd ¦ wc
```

genera due processi, uno per ogni comando, che comunicano tra loro attraverso la struttura pipeline.

Verifichiamo quanti processi vengono creati dal seguente comando:

```
$ NUM = 'cat /etc/passwd ¦ wc -l'
```

La risposta è sempre due, uno per **cat** e uno per **wc**. Lo shell esegue alcuni comandi (in questo caso l'istruzione di assegnamento della variabile di ambiente **NUM**) senza ricorrere a nessun altro processo di supporto. Esistono comandi di shell predefiniti, per esempio **cd** ed **echo**, che non coinvolgono nella loro esecuzione un processo distinto, ma la maggior parte dei vostri comandi, così come molti altri comandi che UNIX crea per esigenze proprie, sono processi.

11.1 Timesharing nel sistema UNIX

Poiché molti sistemi UNIX dispongono generalmente di un'unica CPU (Central Processing Unit o *unità centrale di elaborazione*), solo un programma può di fatto essere in esecuzione in un dato istante. Una delle funzioni principali del *kernel*, la parte del sistema UNIX che gestisce il sistema nel suo complesso, è di fornire il controllo e il supporto per i programmi che richiedono di utilizzare la CPU. Se condividete la macchina con altri utenti, lo shell di ogni utente è un processo, ed è un processo ogni applicazione e ogni comando usato da ogni utente.

Più programmi possono richiedere contemporaneamente il controllo della CPU e il kernel deve garantire l'accesso solo a uno di essi, o mantenere egli stesso il controllo della CPU. Un processo acquisisce il controllo della CPU per un breve periodo di tempo e poi lo lascia a un altro processo; poiché questi cambi di contesto di processo, o *process-switching*, si verificano almeno una volta ogni secondo, e anche più di frequente, ogni utente ha l'impressione di essere l'unico utilizzatore della macchina. Questo è il motivo principale per cui i sistemi operativi multiprocesso sono chiamati sistemi *timesharing*: la CPU diventa una risorsa condivisa nel tempo da tutti gli utenti e da tutti i processi. Il kernel provvede a gestire le attività dei singoli processi dando l'impressione che ciascun processo disponga completamente della macchina, anche se ogni processo può di fatto controllare la macchina per meno di un secondo ogni volta ma, poiché riacquista il controllo dopo brevissimo tempo, raramente un utente si rende conto della presenza di altri utenti.

Il sistema può aumentare il proprio carico rispetto alla situazione iniziale di installazione; quando questo accade, i tempi di risposta aumentano in maniera sensibile. Una volta che conoscete la velocità media di risposta del sistema UNIX, potete considerare il ritardo maggiore come *tempo di risposta* e scoprire quindi gli eventuali problemi ed errori di sistema. Non associate, comunque, ogni ritardo che si verifica a un errore di sistema, perché i comandi impiegano tempi diversi di esecuzione, in relazione alla loro complessità elaborativa. Solo se il sistema ha un comportamento che si discosta sensibilmente dalle abituali condizioni di funzionamento, è possibile che esista qualche anomalia.

Ricordiamo che un processo è una istanza di un programma in esecuzione e descrive lo stato *corrente* della macchina. Potete lanciare la versione eseguibile dei comandi tramite il nome che lo identifica su disco ma, fino a quando i comandi non sono in esecuzione o attendono l'accesso alla CPU, non sono processi.

11.2 Classi di priorità dei processi

La maggior parte dei processi ha la stessa *priorità*, cioè lo stesso diritto a utilizzare la CPU, ma il sistema UNIX dispone di strumenti che permettono di modificare la priorità di un processo. In SVR4 esistono due classi del tutto separate di processi, la cui priorità può essere manipolata separatamente. La priorità *timesharing* è la classe normale delle applicazioni degli utenti, e trattiamo questa classe per prima, ma viene gestita anche una nuova classe *real time*. La priorità real time consente finalmente l'utilizzazione del sistema UNIX per applicazioni che richiedono tempi di risposta contenuti e prefissati. Potete manipolare separatamente la priorità dei processi di ciascuna classe, ma i processi real time ottengono sempre l'uso del tempo di CPU con priorità sui processi timesharing. Tratteremo i processi real time e il comando **priocntl** (priority control) alla fine di questo capitolo; fino ad allora tratteremo un "normale" sistema UNIX che gestisce in pratica tutti processi timesharing.

11.3 Controllo della priorità dei processi in timesharing

L'incremento di priorità comporta generalmente una maggiore rapidità di esecuzione del processo, a spese di altri processi presenti nella macchina, che ritardano l'esecuzione perché aumentano i loro tempi di attesa per la disponibilità della CPU. Al contrario, decrementare la priorità di un processo comporta la diminuzione della sua autorità nel richiedere la CPU e quindi l'inevitabile rallentamento della sua esecuzione; gli altri processi presen-

ti nella macchina traggono beneficio da questa operazione perché aumentano la loro parte nell'acquisizione delle risorse del sistema.

Se non siete il supervisore del sistema, non potete incrementare la priorità dei processi; infatti potrebbe risultare abbastanza rischioso assegnare al vostro processo una elevata priorità tale da impedire o ridurre al minimo l'utilizzo della CPU da parte di altri processi. Al contrario, potete facilmente diminuire la priorità di un comando facendo precedere la linea del comando stesso dal comando **nice**, per esempio:

```
$ nice cat /etc/passwd
```

L'argomento di **nice** costituisce di fatto il comando che intendete eseguire. Nel precedente esempio, **nice** riduce la priorità del comando **cat /etc/passwd** di 10, che rappresenta l'unità di misura predefinita. Il valore di priorità è un parametro arbitrario di selezione della risorsa CPU compreso tra 0 e 19.

Potete ridurre (o incrementare) la priorità di un valore diverso da 10, specificando come argomento di **nice** il decremento (o incremento) numerico che intendete apportare. Per esempio:

```
$ nice -14 cat /etc/passwd
```

Questo comando riduce la priorità della linea di comando **cat** di 14 unità. Se la vostra applicazione impegna una grande quantità di tempo di CPU, potete eseguire il comando a bassa priorità in modo che utilizzi i tempi morti di CPU senza interferire con i processi a priorità maggiore che sono attivi nella macchina.

Il supervisore può incrementare la priorità del processo, ma questo privilegio deve essere utilizzato con accortezza. Un comando **nice** con un argomento di valore numerico negativo incrementa la priorità di un comando, per esempio:

```
$ nice --14 cat /etc/passwd
```

incrementa la priorità del comando **cat** di 14 unità.

Il comando **nice** in realtà è un'interfaccia di uso semplificato alle funzionalità generali di SVR4 delle priorità dei processi; il comando più complesso **priocntl** viene trattato alla fine del capitolo.

11.4 Processi a bassa priorità (background)

Il comando **nice** viene generalmente associato ai comandi che lanciate da shell, ma che operano in *background*. Questo significa che, mentre un vostro programma è in esecuzione, lo shell può accettare ed eseguire contem-

poraneamente altri programmi. In particolare, lo shell prevede l'operatore **&** che, specificato al termine della linea di comando, permette di lanciare i comandi in background.

```
$ cat /etc/passwd &
```

In questo esempio, il comando **cat** viene eseguito in background, ma l'uscita compare ancora su terminale perché non è stata ridiretta.

Quando lanciate un comando tramite **&**, lo shell torna immediatamente a richiedere un altro comando, anche se il processo che avete creato è ancora in esecuzione nel sistema, e restituisce un valore numerico che corrisponde al *numero di processo* o *pid* (process id) con cui potete referenziare successivamente il processo. Lo shell visualizza il prompt per segnalare che è in attesa di un altro comando.

```
$ cat /etc/passwd &
1536
$
```

Tratteremo in seguito il significato del numero di processo.

Generalmente, l'ingresso e l'uscita dei comandi eseguiti in background vengono ridiretti, per cui non si verifica interferenza con l'output della sessione di lavoro interattiva.

```
$ cat /etc/passwd > file.copy &
1540
$
```

Ricordate di ridirigere anche l'uscita dello standard error, che altrimenti viene visualizzata su terminale anche se lo standard output è ridiretto. Questo esempio ridirige sia standard output che standard error:

```
$ cat /etc/passwd > file.copy 2> error.out &
1544
$
```

D'altra parte, può essere preferibile inviare lo standard error su terminale per ricevere notifica immediata degli errori nei processi di background.

Il sistema UNIX non definisce un limite al numero di processi in background, ma le prestazioni del sistema tendono a degradare in presenza di un eccessivo numero di processi. Quando un processo background termina l'esecuzione, il sistema non fornisce alcuna segnalazione; potete controllare lo stato dei processi nel sistema con il comando **ps** e, se avete ridiretto l'uscita di questo comando a un file, esaminare con comodo queste informazioni.

11.5 Chiusura del colloquio con processi background in corso

I processi background che avete creato durante la sessione al terminale terminano l'esecuzione quando uscite dal sistema perché sono associati con lo shell di login. Il sistema UNIX fornisce lo strumento **nohup** (no hang up) che permette ai processi background di continuare l'esecuzione dopo che vi siete scollegati, per cui potete utilizzare anche i periodi di inattività (per esempio le ore notturne) per eseguire lavori di routine. Le modalità di impiego di **nohup** sono analoghe a quelle di **nice**; ambedue devono essere collocati all'inizio della linea di comando.

```
$ nohup cat /etc/passwd &
```

Il comando **nohup** fa in modo che **cat** ignori la vostra operazione di chiusura colloquio e completi l'esecuzione. Generalmente, **nohup** viene utilizzato con comandi background perché non è possibile disconnettersi dal sistema, se non appare su video il prompt di shell.

Quando utilizzate **nohup** con un pipeline, dovete specificarlo all'inizio di ogni elemento del pipeline:

```
$ nohup cat /etc/passwd ¦ nohup wc > out.file &
```

Se non seguite queste regole, l'elemento del pipeline non preceduto dal comando **nohup** viene eliminato quando uscite dal sistema e il pipeline nel complesso fallisce.

Se non ridirigete l'uscita, il comando **nohup** genera automaticamente un file di uscita (nohup.out) perché, se vi scollegate dalla macchina, non esiste più un terminale che può raccogliere l'output prodotto, per esempio:

```
$ nohup cat /etc/passwd &
1565
Sending output to nohup.out
$
```

In questo caso, lo shell specifica il numero di processo perché avete utilizzato l'operatore **&**, mentre il comando **nohup** visualizza il messaggio **Sending output..** per informarvi che sta ridirigendo l'uscita. Il file **nohup.out** nel directory corrente contiene i risultati del comando che è stato eseguito sotto il controllo di **nohup**. Procurate di ridirigere su un file separato l'uscita dello standard error se non volete che vada incluso nello stesso file **nohup.out**.

11.6 Processi padre e processi figlio

Quando entrate in UNIX, il sistema operativo assegna un processo di shell alla vostra sessione e lo disattiva quando uscite, per cui, in qualunque momento, avete alle vostre dipendenze almeno un processo.

Un processo nasce quando va in esecuzione e muore quando termina; molti processi seguono questo corso all'interno del sistema, ma la loro vita dipende da come utilizzate il sistema. Mentre alcuni processi nascono quando il sistema viene acceso e vivono fino al momento dello spegnimento della macchina, generalmente un processo ha un corso di vita relativamente breve che corrisponde alla durata dell'esecuzione del comando richiamato da terminale.

Un nuovo processo può nascere solo a opera di un altro processo e, nella terminologia dei sistemi operativi, il processo creatore è denominato *processo padre*, mentre quello creato è detto *processo figlio*. Un padre può generare più figli, ma un processo può avere solo un padre. Analogamente, un processo può generare un figlio, che a sua volta può generare un altro figlio, e così via. Anche se non si parla di "processo nonno", ogni processo possiede un albero genealogico, che permette di risalire, attraverso i processi padre, allo shell di login e fino all'accensione del sistema. I processi intermedi possono anche cessare di esistere, ma ogni processo ha un padre.

Se un processo genera un figlio che a sua volta genera un altro figlio, il processo intermedio può morire; generalmente quando un padre muore, tutti i figli muoiono, ma in certe situazioni questo può non accadere. Quando si verifica una di queste situazioni, il processo padre originario eredita i figli del processo morto, così un processo ha sempre un padre.

Quando usate **nohup** per eseguire un comando, nel momento in cui uscite dal sistema, **nohup** riassegna il processo 1 come padre al vostro processo che esegue il comando specificato a **nohup**, e lo libera quindi dalla dipendenza con lo shell di login. Normalmente, i processi che create da shell sono figli dello shell e muoiono quando lo shell muore.

Ogni utente possiede processi privati associati alla propria sessione di login, ma esistono anche altri processi creati dal sistema a uso del sistema stesso, denominati *processi di sistema*. Alcuni di questi processi il sistema li crea per una funzione particolare e li uccide subito dopo avere concluso quella funzione. Per esempio, il sistema crea un processo per spedire i vostri messaggi di posta elettronica a un'altra macchina e lo mantiene in vita solo per la durata di questa operazione. Viceversa, altri processi, come per esempio lo *spooler* per il sottosistema di stampa **lp**, rimangono in vita anche per l'intero arco di tempo in cui la macchina è in attività.

11.7 Il comando ps

Per avere informazioni sui processi che sono attivi nella macchina, utilizzate il comando **ps** (process status). Se lanciate questo comando più volte, i risultati visualizzati nelle varie occasioni probabilmente differiscono, perché **ps** produce un'istantanea dell'attività corrente della macchina. Se la linea di comando **ps** non contiene nessun argomento, su video appaiono solo le informazioni relative ai processi che sono associati con la vostra sessione di login:

```
$ ps
  PID   TTY   TIME COMMAND
 6756   tty00 0:01 -sh
 6760   tty00 0:02 ps
$
```

Dalle informazioni visualizzate, risulta che disponete di due processi, uno è lo shell di login, che è nato quando siete entrati nel sistema e muore quando uscite, mentre l'altro è il processo che sta eseguendo il comando **ps**. Per ogni processo viene indicato un tempo di esecuzione (in questo esempio, è pari a 2 secondi per il comando **ps**) che non corrisponde a un clock o all'indicazione del tempo trascorso, ma totalizza il tempo totale di CPU che il processo ha utilizzato da quando è stato creato. I processi che sono associati alla vostra sessione utilizzano di solito, ma non sempre, un terminale da cui leggono e in cui scrivono, come potete facilmente verificare esaminando la colonna TTY nei dati prodotti dal comando **ps**. Se invece, in corrispondenza di una riga e della colonna TTY, appare il carattere ? (punto interrogativo), il processo che occupa quella riga non è collegato ad alcun terminale. Inoltre, ogni processo ha un unico *pid* che lo identifica all'interno del sistema. Questi identificatori di processo assumono un valore che parte da 0, quando il sistema viene caricato, e si incrementa di una unità a ogni creazione di un nuovo processo, fino a raggiungere un limite prefissato, che corrisponde di solito a 32767. In questo esempio, **ps** coincide con il 6760-esimo processo che è stato creato dal momento in cui abbiamo caricato il sistema. Quando si raggiunge il limite massimo, il conteggio riparte ancora da 0; tuttavia i valori numerici assegnati ai processi ancora in vita non vengono ovviamente riassegnati ad altri processi. I figli di solito hanno un pid più alto dei loro padri, ma se il pid del padre è vicino al valore massimo, un figlio può anche avere un numero di identificazione più basso.

Esistono di fatto altre informazioni associate ai processi oltre a quelle che abbiamo esposto; alcune sono rese disponibili dall'opzione **-f** (full) del comando **ps**:

```
$ ps -f
     UID    PID  PPID   C    STIME TTY      TIME COMMAND
 giorgio   6756     1   6 13:04:57 tty00   0:01 -sh
 giorgio   6761  6756  23 13:05:19 tty00   0:01 ps -f
$
```

Oltre al nome del comando (COMMAND), tempo di esecuzione (TIME), terminale (TTY) e identificatore di processo (PID), il comando **ps** visualizza altre informazioni. All'estrema sinistra compare l'identificatore di utente (UID); poiché ogni processo è di proprietà dell'identificatore di login che lo ha creato, la colonna PPID indica il pid del padre del processo. Poiché il sistema assegna all'utente uno shell quando si collega alla macchina, l'identificatore del processo padre dello shell è un numero molto basso che corrisponde al pid associato a una parte del kernel. L'identificatore del padre della maggior parte dei processi creati dal sistema corrisponde generalmente al pid 1. Quando lanciate il comando **ps**, lo shell crea il relativo processo, per cui il pid del processo padre corrisponde all'identificatore di processo dello shell. Potete risalire nell'albero genealogico di un processo esaminando i corrispondenti pid e ppid.

La colonna C indica il numero di risorse di CPU che il processo ha utilizzato recentemente e il kernel analizza questa informazione per decidere a quale processo assegnare il controllo della CPU. In particolare, il kernel permette a un processo con un basso valore di C di avere il controllo della CPU prima di un processo che ha un valore elevato di C. Il comando **nice** agisce cambiando l'algoritmo interno in base a cui vengono calcolati questi valori numerici. Generalmente, la colonna C può essere utile solo ai progettisti di software di base che lavorano a miglioramenti nel kernel o nei driver.

Infine, STIME indica la data e l'ora di inizio del processo; potete utilizzare questa informazione per rintracciare vecchi processi che sono rimasti erroneamente nel sistema per troppo tempo.

11.8 Le attività degli altri utenti

Il comando **ps** può anche fornire informazioni relative alle attività degli altri utenti che sono collegati alla macchina. Se siete a conoscenza che un utente si trova nel sistema, potete visualizzarne lo stato dei processi di quell'utente con:

```
$ ps -u utente
```

dove "utente" è l'identificatore di login dell'utente di cui volete conoscere le informazioni.

Su un piccolo sistema risulta generalmente più semplice esaminare tutta insieme l'attività di tutti gli utenti, specificando l'opzione **-a** (all) sulla linea di comando **ps**:

```
$ ps -af
     UID   PID  PPID   C    STIME TTY      TIME COMMAND
    root    82     1   0     Apr 9 console  0:05 -sh
giorgio6756     1   3 13:04:57 tty00   0:01 -sh
giorgio6762  6756  21 13:05:28 tty00   0:00 ps -af
$
```

Questo esempio evidenzia alcuni concetti interessanti. Innanzitutto, possiamo notare che un altro utente si trova nel sistema e che si tratta del supervisore, riconoscibile dall'identificatore di login root, che si può collegare solo dalla console principale di sistema, come indica la colonna TTY. Apparentemente l'utente root è entrato nel sistema poco dopo l'accensione della macchina, come potete dedurre dal pid relativamente basso che è associato a quel processo. Se esaminate la colonna STIME, potete avere ulteriori informazioni: l'ora di inizio di un processo è espressa nel formato hh:mm:ss (ore:minuti:secondi) riferito al giorno corrente, mentre, per processi nati nei giorni precedenti, il sistema specifica solo il mese e il giorno. Questo formato viene adottato unicamente allo scopo di ridurre la dimensione dei dati in uscita; il sistema UNIX memorizza sempre al completo la data e l'ora di ogni evento.

11.9 Processi di sistema

Abbiamo esaminato i processi associati agli utenti, ma esistono anche processi permanenti, con un arco di vita molto esteso, che supportano le attività del sistema, e processi temporanei che nascono e poi muoiono mentre il sistema svolge attività proprie, indipendenti da quelle dei singoli utenti. L'opzione **-e** (every) di **ps** fornisce alcune informazioni su tutti i processi che sono attivi nella macchina. I risultati prodotti da **ps -e** aiutano a capire le attività interne della macchina e risultano molto utili per diagnosticare eventuali problemi che si verificano nel sistema. Il formato dei dati prodotti in uscita dipende molto dal software installato nella macchina e dalle periferiche di I/O collegate al sistema, così come l'organizzazione dei processi di sistema dipende dalla versione di sistema UNIX che utilizzate. Se installate una nuova versione di sistema UNIX, cercate di distinguere subito i processi che sono normali, allo scopo di identificare problemi o errori che risultano nell'output di **ps**, in caso di anomalie nel funzionamento del sistema. Lanciate di frequente questo comando sul vostro sistema per acquisire la sensibilità necessaria ad avvertire se le attività procedono correttamente.

Se eseguite il comando

```
$ ps -ef
```

su un sistema SVR4 basato sul microprocessore Intel 80×86, i dati prodotti assumono un formato di questo tipo:

```
$ ps -ef
UID   PID  PPID  C    STIME  TTY      TIME COMMAND
root    0     0  0     Apr 9  ?       0:00 sched
root    1     0  0     Apr 9  ?       0:01 /sbin/init
```

```
root     2     0   0     Apr 9  ?         0:00  pageout
root     3     0   0     Apr 9  ?         0:00  fsflush
root     4     0   0     Apr 9  ?         0:00  kmdaemon
root   173     1   0     Apr 9  ?         0:00  /usr/lib/saf/sac -t 300
root   245   173   0     Apr 9  ?         0:03  /usr/lib/saf/ttymon
root   174     1   2  13:04:57  console   0:01  -sh
root   184   174   8  13:05:28  console   0:00  ps -ef
root   163     1   0  13:05:01  ?         0:00  /usr/sbin/cron
$
```

Questo output è relativo a un sistema di base, configurato senza pacchetti di software opzionali.

Il primo processo che il sistema esegue quando la macchina viene caricata è lo *scheduler*. Lo scheduler è la base delle caratteristiche di timesharing (suddivisione di tempo) del sistema UNIX, ha il compito di selezionare il processo più prioritario che ha effettivamente diritto di accedere alle risorse della macchina, e gli cede il controllo della CPU. Il nome di questo processo "selezionatore" è **sched**, ha il pid 0; lancia a sua volta il processo **init** (initialization), con pid 1, che mette in attività i processi di sistema che sono fondamentali. Tratteremo **init** nel Capitolo 21. Il processo lanciato successivamente è **pageout**, che gestisce la memoria virtuale della macchina e scambia (*swap*) parte dei processi attivi tra disco e memoria reale quando vengono messi in esecuzione o vengono temporaneamente accantonati. Il processo **pageout** è responsabile della maggior parte dell'attività di gestione dei processi nell'ambiente multiprocesso. Il processo **kmdaemon** (kernel memory demon), svolge la stessa funzione per le parti del sistema operativo stesso. I processi **sched**, **kmdaemon** e **pageout**, svolgono strettamente assieme l'attività di sistema, e costituiscono il cuore del kernel. Il processo **pageout** ha pid 2, **kmdaemon** ha pid 4.

Il successivo processo, **fsflush** (file system flush), gestisce le operazioni di I/O del sistema su disco. Il sistema UNIX contiene molti buffer dati interni che hanno una funzione analoga a quella che ricopre un disco di memoria RAM in altri sistemi operativi, per cui, nel caso di un'improvvisa caduta del sistema (*crash*), i dati nei buffer possono non coincidere con quelli memorizzati su disco. Per evitare ciò, il processo **fsflush** riscrive periodicamente su disco il contenuto di tutti i buffer mediante la funzione **sync**; solo i buffer che hanno subito cambiamenti vengono riscritti. La frequenza con cui ha luogo questa operazione di sincronismo dipende da un parametro di sistema; su molte macchine di piccole dimensioni si verifica una volta ogni 20 secondi. Il processo **fsflush** ha pid 3. Ad eccezione di **init**, questi processi di sistema sono memorizzati su disco nel directory **/stand/unix**, che rappresenta il kernel; nessun file su disco contiene gli oggetti eseguibili dei processi **sched**, **pageout**, **fsflush** o **kmdaemon**.

Questi processi vengono creati quando il sistema viene caricato e rimangono attivi fino a quando la macchina viene spenta. Per questa ragione, la

colonna TIME, nelle informazioni prodotte in output da **ps –ef**, può indicare per questi processi un tempo apparente di utilizzo di CPU molto elevato; ma non sempre il conteggio è accurato, poiché alcune implementazioni di UNIX assegnano arbitrariamente a **sched**, o a un altro dei processi di kernel, il tempo in cui la CPU è inattiva. In ogni caso, questo non causa problemi; in effetti, se questi processi di kernel non sono realmente in esecuzione, il sistema probabilmente non è neanche in grado di eseguire il comando **ps**.

I restanti processi di sistema possono risalire nella loro discendenza fino a **init**, che è responsabile della gestione dei processi di sistema in accordo al contenuto del file **/etc/inittab**. Il sottosistema di controllo dei terminali e altri processi fondamentali derivano da **init**. Mentre i processi nascono e muoiono, il contatore pid si incrementa, ma tutti i processi che non hanno un padre ancora attivo vengono alla fine ereditati da **init**, come dimostra l'elevato numero di processi con ppid 1 presenti nel sistema.

Gli altri processi nell'output del comando **ps –ef** appartengono agli utenti o ad applicazioni particolari in esecuzione, come, per esempio, il sottosistema di stampa **lp**, se attivo nella macchina. Un processo di sistema che opera senza esplicita richiesta degli utenti, come il processo per il sottosistema **lp**, viene denominato *demon* (o daemon); può essere un importante processo di sistema, come **fsflush**, un processo applicativo sempre in esecuzione, come **lpsched** (e **lpNet**) per il sottosistema **lp**, o un processo in esecuzione sotto il controllo dei sottosistemi di temporizzazione, come **uuxqt**. Questi processi in esecuzione normalmente hanno un breve tempo di vita, ma potete trovarli nell'output del comando **ps**, e ovviamente incrementano il valore di pid.

Con SVR4, diversi nuovi processi sono associati con la gestione delle porte dei terminali, sotto *Service Access Facility*. Il processo **sac** (service access controller) è il processo principale per servizi d'entrata, provvede a creare **ttymon** (tty monitor) per gestire le porte seriali (tty), ed eventualmente diversi altri processi come **listen**, che tratta l'attività di rete locale (local area network).

Se state usando X Window System e il pacchetto per rete Ethernet (TCP/IP), sono presenti anche diversi altri demon, come mostrato nella Figura 11.1. Il processo **xntad** controlla l'accesso alla rete, **mousemgr** controlla il mouse, questi processi sono demon che restano in vita anche se X Window System non è attivo. Altri processi associati con X sono **olinit**, che attiva il sistema OPEN LOOK; **olwm**, il gestore di finestre di OPEN LOOK; **olwsm**, il gestore di Workspace; **olfm**, il gestore di file OPEN LOOK. Anche altre applicazioni attive sotto la sessione X, come **xclock**, possono apparire nell'output di **ps**.

Altri demon sono associati alla gestione della rete; il processo **inetd** è il ricevitore basilare per l'attività della rete; **rpcbind** attende le richieste di *Remote Procedure Call*; i processi **rpc.walld**, **rpc.rusersd** e **rpc.sprayd** supportano varie funzioni a livello utente in un sistema in rete. Se utilizzate la

```
$ ps -ef
     UID   PID  PPID  C    STIME TTY      TIME COMMAND
    root     0     0  0 21:02:15 ?       0:00 sched
    root     1     0  0 21:02:15 ?       0:01 /sbin/init
    root     2     0  0 21:02:15 ?       0:01 pageout
    root     3     0  0 21:02:15 ?       0:01 fsflush
    root     4     0  0 21:02:15 ?       0:00 kmdaemon
    root   256     1  0 21:03:32 ?       0:00 /usr/lib/saf/sac -t 300
    root   257     1  0 21:03:32 console 0:01 ksh
    root   161     1  0 21:03:18 ?       0:00 /usr/sbin/rpcbind
    root   185   175  0 21:03:23 ?       0:00 lpNet
    root   157     1  0 21:03:15 ?       0:00 /usr/sbin/cron
    root   163     1  0 21:03:18 ?       0:01 /usr/lib/netsvc/rwall/rpc.rwalld
    root   165     1  0 21:03:18 ?       0:01 /usr/lib/netsvc/rusers/rpc.ru-
                                                sersd
    root   167     1  0 21:03:19 ?       0:01 /usr/lib/netsvc/spray/rpc.spra-
                                                yd
    root   175     1  0 21:03:22 ?       0:01 /usr/lib/lpsched
    root   259   256  0 21:03:36 ?       0:01 /usr/sbin/inetd
    root   251     1  0 21:03:31 ?       0:00 /usr/X/lib/xntad
    root   267   266  0 21:03:43 vt01    0:19 X :0
    root   253     1  0 21:03:31 ?       0:00 /usr/lib/mousemgr
    root   266     1  0 21:03:43 pts/1   0:01 olinit
    root   270   266  0 21:03:47 ?       0:02 olwsm
    root   309   308  0 21:06:31 pts/9   0:02 ps -ef
    root   273     1  0 21:03:53 ?       0:04 olwm
    root   285   277  0 21:04:28 pts/5   0:00 ksh
    root   277     1  0 21:03:53 ?       0:01 xterm -geometry 80x24+10+0
    root   278     1  0 21:03:53 ?       0:01 xterm -geometry 80x24+1-1
    root   286   278  0 21:04:28 pts/9   0:00 ksh
$
```

Figura 11.1 Ambiente con grande numero di processi per **init state 2**.

macchina come server in LAN, possono essere presenti anche molti altri demon per la rete.

Esistono molte altre opzioni del comando **ps**, che vengono principalmente utilizzate dai progettisti di sistema e dagli sviluppatori di software applicativo. Verificate per esempio sulla vostra macchina le informazioni fornite dalla linea di comando **ps -l**.

11.10 Diagnosi di problemi nei processi

Se studiate frequentemente l'output di **ps -ef**, potete acquisire la capacità di controllare il corretto sviluppo delle attività della vostra macchina. Un

piccolo sistema generalmente dispone di un numero relativamente basso di processi attivi, anche se questo numero varia durante la giornata. Se nella colonna TIME risulta che qualche processo ha accumulato una inconsueta quantità di tempo, se un processo è padre di molti altri processi, o se il tempo di risposta della macchina cresce improvvisamente, allora potete sospettare che un processo non funzioni correttamente.

Dovete comprendere bene il comportamento normale della vostra macchina, prima di poter riconoscere qualcosa di anormale, e questo comportamento varia ampiamente da una macchina a un'altra; inoltre il diagnosticare l'origine di un problema può non essere di grande aiuto nel risolvere il problema. La maggior parte dei problemi deriva da un comando o da un'applicazione particolari, spesso dalla mancanza di file e directory o dalla scorrettezza dei diritti di accesso dei file esistenti. Poiché UNIX è un sistema operativo multiutente e multiprocesso, un singolo problema può influenzare il funzionamento di tutte le attività della macchina, può causare un abbassamento significativo delle prestazioni e può causare comportamenti anormali del sistema. Raramente si verificano danneggiamenti ad altri programmi, utenti, o file nel sistema, tuttavia questo non può essere escluso in caso di malfunzionamenti eccezionali; per questa eventualità è sempre opportuno salvare regolarmente dati e file su dischetto o su nastro, specialmente quando si esperimenta sul sistema.

Un'applicazione che non funziona come previsto generalmente evidenzia una delle seguenti condizioni di errore:

1. *Il processo muore prematuramente.* Se si tratta di un comando, il sistema restituisce il controllo improvvisamente allo shell; se si tratta di un processo di sistema, **init** può cercare ripetutamente di rilanciare l'applicazione. Nel primo caso, il comando **ps** non evidenzia nessun processo associato all'applicazione, la cui esecuzione appare molto più rapida del solito. Nel secondo caso, il sistema rallenta sensibilmente, per cui occorrono generalmente diversi minuti per lanciare un comando, mentre il disco di sistema appare sovraccarico di lavoro. Qualche volta l'operazione di ricaricamento del sistema risolve questo problema, ma più spesso occorre disinstallare l'applicazione o anche rimuovere dal file **/etc/inittab** l'elemento che referenzia il programma.

2. *Il processo genera troppi figli.* Talvolta si verifica una situazione d'errore tale per cui un processo genera un grande numero di processi figlio. Questa anomalia viene evidenziata dal sensibile rallentamento della macchina e dalla presenza in output di **ps** di un numero di processi notevolmente superiore al normale. Questi processi non previsti possiedono generalmente lo stesso *pid* e probabilmente il processo padre è la causa di tutti i problemi. A volte un errore di questo tipo provoca la creazione di un singolo processo erratico ogni volta che entra in esecu-

zione l'applicazione che origina il problema; si rintracciano più copie dello stesso processo erratico, ma generalmente il padre di questi processi non è più attivo, per cui il suo ppid non è significativo. Questa situazione è difficile da diagnosticare; potete provare a eseguire il programma sospetto e verificare se compare un nuovo processo erratico. Le colonne TTY e STIME visualizzate dal comando **ps −ef** possono talvolta aiutare a stabilire da chi e quando il processo erratico è stato creato.

3. *Il processo consuma eccessivo tempo di CPU.* Eccezionalmente può accadere che un processo esegua inconsuete sequenze di istruzioni e impegni tutte le risorse di CPU che la macchina può concedere. Potete rilevare questa situazione anomala se la macchina incrementa considerevolmente i tempi di risposta e se l'output del comando **ps −ef** indica che un processo ha accumulato una notevole quantità di tempo di CPU, generalmente diversi minuti o più. Se eseguite ripetutamente il comando **ps −ef**, noterete che il processo colpevole acquisisce quasi tutto il tempo in cui la CPU dovrebbe altrimenti rimanere inattiva. Si tratta di un problema difficile da individuare, perché dovete essere certi che un'applicazione real time non abbia il legittimo uso di tutto il tempo di CPU. Usualmente, i processi timesharing che funzionano correttamente impiegano più tempo per concludersi, se necessitano di grande quantità di risorse di CPU, ma non rallentano la macchina in maniera apprezzabile.

Concludendo, sintomi chiave per scoprire un comportamento anomalo nei processi sono: un incremento apprezzabile del tempo di risposta del sistema o una inusuale attività su disco senza motivi apparenti. Mentre acquistate maggiore familiarità con la macchina, potete rendervi conto delle variazioni di prestazioni del sistema, e spesso stabilire l'origine del problema utilizzando il comando **ps.**

11.11 Interruzione dell'esecuzione dei processi

Se individuate un processo "uscito di programma", o se avete semplicemente lanciato un lavoro che ha un lungo tempo di esecuzione e non vi interessa aspettare la sua conclusione, potete interromperne l'esecuzione, ovvero ucciderlo. Il supervisore può interrompere l'esecuzione di qualsiasi processo presente nel sistema UNIX ad eccezione di quelli i cui pid valgono 0, 1, 2, 3 e 4; tutti gli altri utenti possono interrompere l'esecuzione solo dei processi di cui sono proprietari.

Per interrompere l'esecuzione di un processo, utilizzate il comando **kill** specificando come argomento l'identificatore del processo che intendete eliminare:

```
$ kill 4314
$
```

Se l'operazione ha successo, il comando **kill** termina senza commenti e il processo non compare più in output di **ps**. Sfortunatamente, il comando **kill** può non eseguire il compito per molti motivi, per cui dovete lanciare sempre il comando **ps** per verificare che il processo sia stato effettivamente cancellato e che non sia stato subito rilanciato dal sistema, sotto altro pid. Quando eliminate un processo, **init** può ereditare i suoi figli, se non li avete eliminati esplicitamente. Pertanto, prima di eliminare un processo, dovreste determinare se possiede dei figli, ed eliminare tutti i figli assieme al genitore. Potete specificare più identificatori di processo al comando **kill**, se volete eliminare contemporaneamente più processi:

```
$ kill 4320 4326 4356
```

L'interruzione dell'esecuzione dei processi, specialmente quando siete collegati come **root**, può risultare dannosa perché potete interrompere un'importante funzione del sistema; in questo caso, l'unico rimedio possibile è generalmente reinizializzare il sistema.

11.12 Segnali

L'interruzione dell'esecuzione di un processo consiste di fatto nell'invio da parte del sistema di un *segnale* al processo. I segnali sono usati come comunicazione tra processi; possono essere scambiati molti differenti segnali. Il sistema SVR4 prevede normalmente 31 segnali distinti, la lista completa viene data nella man page **signal(5)**. La maggior parte dei segnali si riferisce a diverse condizioni di errore che si possono verificare nel sistema; per esempio, se un processo cerca di accedere alla memoria di sistema fuori dalla sua area autorizzata, il sistema UNIX gli invia il segnale numero 11, tentativo di violazione di un segmento di memoria (memory segmentation violation).

Il sistema invia segnali quando viene tolta la tensione al terminale, quando premete il tasto DEL, quando un processo figlio muore, quando scade un clock interno. Spetta all'applicazione rispondere in modo appropriato al segnale, terminando l'esecuzione o prendendo misure correttive in modo che la condizione che ha causato l'invio del segnale non si ripresenti.

Quando eseguite il comando **kill**, il sistema invia il segnale 15, di default, al processo o ai processi di cui avete specificato il pid. Si tratta di un segnale di terminazione software che generalmente conduce all'eliminazione del processo, ma che tuttavia il processo può anche ignorare. Per questo caso, è previsto un segnale kill incondizionato, che funziona sempre immediatamanete; il segnale 9 elimina il processo immediatamente e incondizionata-

mente. Potete specificare sulla linea di comando **kill** il numero di segnale come opzione:

```
$ kill -9 4367
```

Potete anche usare il nome logico del segnale, come dall'elenco in **signal(5)**, senza il prefisso SIG, come nell'esempio seguente:

```
$ kill -HUP 4369
```

Ripetiamo ancora: abbiate cura di non eliminare nessun processo se non in caso di necessità.

11.13 Approfondimenti

I processi e la loro gestione sono una parte del sistema UNIX che può disorientare, ma un esame attento degli output prodotti dal comando **ps -ef** può aiutare molto a risalire alle cause di un problema del sistema. Inoltre, vi sono molti altri aspetti da considerare rispetto ai processi.

RIATTIVAZIONE DEI PROCESSI

Il sistema UNIX prevede la possibilità di riattivare (*respawn*) i processi che muoiono. Occasionalmente un processo andato "fuori programma" che avete eliminato può ripartire automaticamente; se questo avviene dovete richiedere al sistema di non rilanciare quella applicazione.

Il file **/etc/inittab** contiene l'elenco dei processi che devono essere riattivati, oltre a molte altre importanti informazioni, per cui se deve essere modificato occorre la certezza di non commettere nessun errore. Occorre comunque la qualifica di superuser per modificare il file **/etc/inittab**. Nella Figura 21.5 troverete un esempio di file **/etc/inittab**, uno dei più importanti file di supporto per le attività di sistema; ne esamineremo in dettaglio le caratteristiche principali nei Capitoli 21 e 25. Qui tratteremo solo l'argomento relativo alle modifiche da apportare in **/etc/inittab** per controllare processi andati "fuori programma" che vengono riattivati da **init**.

Tenete presente che le modifiche fatte direttamente in **/etc/inittab**, come quelle indicate in questa sezione, possono essere annullate se aggiungete o cancellate pacchetti software, perché questo file viene ricreato dalle procedure *build* di kernel. Le informazioni presentate qui sono da utilizzare solo per casi di emergenza.

Il file **/etc/inittab** è un semplice database in cui ogni record è una linea, ogni linea ha quattro campi e ogni campo è separato dal successivo dal carattere : (due punti). Il programma **init** esamina ogni linea alla ricerca di un

processo da riattivare in base a regole prefissate per il file: se il secondo campo del record contiene le cifre 2 o 3 (per esempio, 1245 contiene 2 ma 145 no) e il terzo campo contiene la stringa *respawn*, **init** automaticamente fa ripartire il processo ogni volta che ne viene interrotta l'esecuzione. L'ultimo campo della linea contiene il comando da eseguire e gli argomenti relativi.

Se individuate un problema in un particolare processo, basandovi sul comportamento della macchina e sull'output del comando **ps –ef**, e notate che il programma riparte immediatamente quando ne interrompete l'esecuzione, ricercate il suo nome nell'ultimo campo delle linee del file **/etc/inittab**. Se la ricerca ha successo controllate se il secondo campo della linea contiene le cifre 2 o 3 e se il terzo campo contiene la parola **respawn**. Se queste due condizioni sono presenti, allora potete disattivare quel processo, mentre determinate le cause che generano il problema. Non potrete usare l'applicazione, ma il resto del sistema dovrebbe rimanere disponibile ed efficiente.

Per disattivare un'applicazione, modificate con l'editor il file **/etc/inittab**, sempre che siate nel sistema da supervisore (cioè come **root**): sostituite nel terzo campo della linea interessata del file la parola **respawn** con la parola **off**, riscrivete il file e uscite dall'editor. Ora dovete segnalare al processo **init** che avete cambiato le sue istruzioni per cui deve rileggere il file **/etc/inittab** per trattare le nuove informazioni. A questo scopo utilizzate il comando **telinit** (tell init what to do), con l'argomento **q** (Q minuscolo), come nell'esempio seguente:

```
# telinit q
#
```

In questo modo, si evita la riattivazione automatica del programma associato alla linea modificata nel file **/etc/inittab**; tuttavia, se verificate con **ps –ef** che un'istanza del programma è ancora in esecuzione, dovete interromperla manualmente mediante il comando **kill**, con argomento il pid del processo incriminato. La procedura **telinit** infatti non elimina un processo esistente, ma ne impedisce la riattivazione.

Per ripristinare la situazione originale, cambiate il contenuto del terzo campo della linea in precedenza modificata nel file **/etc/inittab** da **off** a **respawn** ed eseguite di nuovo **telinit q**. Dovrete naturalmente aver eliminato la causa dell'errore per evitare che il problema originale si ripresenti.

TEMPO DI ATTESA SUL PRIMO COMANDO ps

Per svolgere le sue funzioni, il comando **ps** ha bisogno di esaminare alcune informazioni che fanno parte della memoria interna del kernel. Per elaborare i dati in output occorre leggere le tabelle interne di ciascun processo.

Quest'operazione richiede un tempo notevole: per un piccolo sistema possono essere necessari da 7 fino a 10 secondi di tempo di CPU, fino a tempi molto superiori per un sistema di grandi dimensioni che gestisce un centinaio di processi. Per minimizzare questo tempo di risposta, il comando **ps** del sistema SVR4 mantiene una tabella con una parte dei dati che gli occorrono, memorizzata in forma adatta nel file binario **/etc/ps_data**; la presenza di dati intermedi in questa tabella permette a **ps** di velocizzare notevolmente la sua risposta.

Tuttavia, le informazioni di kernel contenute nella tabella possono differire a seguito di un'operazione di ricaricamento del sistema, per cui il comando **ps** deve ricostruire automaticamente la tabella quando viene eseguito dopo un'inizializzazione, e comunque quando il file **/etc/ps_data** non è presente. Conseguentemente, il comando **ps** ha un tempo di esecuzione che può risultare molto lungo, o molto breve, a seconda che il comando sia stato eseguito più o meno di recente. Per verificare quanto abbiamo detto, provate a lanciare il comando **ps** dopo avere cancellato il file **/etc/ps_data**.

PROCESSI IN STATO DI ATTESA E IN STATO DEFUNCT

Se un processo genera un altro processo figlio, il processo padre attende di norma che il figlio abbia terminato l'esecuzione prima di riprendere la sua attività. Di norma, anche lo shell di login segue questo comportamento quando lanciate un comando da terminale. Il kernel di UNIX riconosce il momento in cui il processo figlio termina l'esecuzione e invia al processo padre il segnale 18 (morte di un processo figlio). Il processo padre riconosce in questo modo la morte del figlio e il sistema continua nella sua attività. Lo scenario differisce leggermente quando eseguite un processo in background con l'operatore **&**, perché, in questo caso, lo shell non aspetta che il figlio termini, e può anche accadere che non sia lo shell il padre del processo background.

Se il padre del processo che viene eliminato è impegnato in altre attività, può non essere in condizioni di avvertire la morte del figlio. In questo caso, il sistema non elimina definitivamente il processo figlio, ma gli assegna uno stato di "morte apparente" (*zombie* o *defunct*), in cui il processo non è realmente morto, ma non è neanche in esecuzione né sta utilizzando tempo di CPU. Non appena il padre riconosce la morte del figlio, il processo in stato defunct scompare dal sistema.

Quando lanciate il comando **ps**, in uscita possono occasionalmente apparire processi marcati con la segnalazione *<defunct>*; questa situazione non costituisce un problema, a meno che i processi in stato di morte apparente rimangano nel sistema oltre un'ora o quando il numero di questi processi cresce con il passare del tempo. Poiché il padre del processo in stato di morte apparente deve riconoscere la morte del figlio, la presenza di processi in questo stato è sintomo di qualche problema nel processo padre.

Probabilmente il processo padre non può riconoscere la morte dei figli perché è rimasto bloccato per qualche motivo; per riportare il sistema alla normalità, di regola dovete eliminare il processo padre che ha questo comportamento anomalo.

GRUPPI DI PROCESSI

Esaminando la creazione di processi in ambiente UNIX, avete visto che il processo numero 1, **init**, è responsabile della generazione della maggior parte dei principali processi di sistema, che a loro volta generano altri processi figli che sono i veri esecutori del lavoro. Lo shell di login è un esempio di processo di primo livello o "superpadre", come molti altri processi che hanno come padre il processo 1. Questi processi generano spesso *gruppi di processi* figli, di cui sono i *process group leader* o *session leader*. Non tutti i padri sono leader di un gruppo di processi perché, per esserlo, devono avere un requisito particolare: un leader di un gruppo di processi è di solito figlio del processo 1, ma non obbligatoriamente, perché un programma può cambiare il suo gruppo di processi senza cambiare ppid. Per esempio, gli shell spesso fanno questo durante procedure di *job control*.

Il comando **ps** può visualizzare non solo i leader di gruppi di processi, ma può visualizzare tutti i processi che non sono leader di gruppi. Lanciate il comando **ps -df** per visualizzare tutti questi processi per distinguere tra i processi quali sono i leader e quali no.

Potete utilizzare **kill** per inviare un segnale kill a tutti i processi che fanno parte di un gruppo invece che a un solo processo. Questo comando

```
$ kill 0
```

invia il segnale 15 a tutti i processi che appartengono alla sessione, ovvero gruppo di processi, del vostro terminale. Ovviamente, se non siete un superuser, non potete eliminare in questo modo nessuno dei gruppi di processi di altri utenti.

Lo shell di login viene generalmente predisposto per ignorare il segnale 15 (il segnale di terminazione software), per cui il comando **kill 0** può non avere effetto se non esistono processi background in esecuzione. Provate l'effetto del lancio del comando **kill -9 0**, che invia il segnale incondizionato di terminazione a tutti i processi presenti nella vostra sessione. Quando uscite dal sistema, il sistema in realtà esegue il comando **kill -1 0** per inviare il segnale di hangup a tutti i processi che appartengono alla vostra sessione. Tutti i processi vengono terminati, a meno che abbiate lanciato alcuni programmi con il comando **nohup**.

IL DIRECTORY /proc

SVR4 include un nuovo directory che contiene il nome di ciascun processo in esecuzione nel sistema: il directory **/proc**. Il contenuto di **/proc** cambia come cambiano i processi; potete listare i processi correnti attivi nella macchina, con:

```
$ ls -l /proc
total 7824
----------   1 root     root           0 Apr  5 18:49 00000
----------   1 root     root       77824 Apr  5 18:49 00001
----------   1 root     root           0 Apr  5 18:49 00002
----------   1 root     root           0 Apr  5 18:49 00003
----------   1 root     root           0 Apr  5 18:49 00004
----------   1 root     root      274432 Apr  5 18:49 00153
----------   1 root     root      671744 Apr  5 18:49 00161
----------   1 root     root      557056 Apr  5 18:49 00171
----------   1 root     root      659456 Apr  5 18:49 00237
----------   1 root     root      229376 Apr  5 18:49 00239
----------   1 root     root      450560 Apr  5 18:49 00242
----------   1 giorgio  other     323584 Apr  5 18:49 00243
----------   1 root     root      512000 Apr  5 18:49 00245
----------   1 giorgio  other     249856 Apr  5 18:49 06514
$
```

In realtà, i "file" in **/proc** sono le immagini di memoria virtuale dei processi attivi in macchina, mappati in una rappresentazione file system. Il campo size nell'output di **ls** indica la dimensione reale di memoria usata dal processo, il campo time indica il tempo corrente di esecuzione, non l'ora di lancio del processo; i campi user e group contengono le informazioni abituali. Strumenti speciali, come i debugger, possono accedere a queste immagini di memoria; tuttavia, attualmente **/proc** non ha grande importanza per gli utenti finali. L'innovazione di **/proc** è molto potente e potrà essere sfruttata pienamente quando l'idea di file system mappati in memoria si sarà pienamente diffusa nel mondo UNIX.

PROCESSI REAL TIME

In aggiunta ai "normali" processi in timesharing trattati fino a ora, SVR4 introduce un nuovo concetto di *classi di priorità*. Potete configurare una macchina includendo classi multiple di processi; i processi in ciascuna classe saranno messi in esecuzione secondo una politica di selezione (*scheduling policy*) specifica per la loro classe. Le differenti classi possono avere differenti politiche di selezione, facilitando l'adeguamento della personalizzazione della macchina a scopi speciali come l'automazione di produzione o l'acquisizione di dati ad alta velocità.

La configurazione SVR4 di default include tre classi: una classe sistema, una classe timesharing (TS) e una classe real time (RT). La classe timesharing riceve il "normale" trattamento del sistema UNIX: il sistema assegna imparzialmente il tempo di CPU a ciascun processo, in modo che nessuno manchi di quella risorsa. La classe real time si comporta in modo del tutto differente: innanzitutto, implementa una politica di scheduling a priorità fissa o a prelazione (*fixed-priority, preemptive*); vale a dire che a ciascun processo viene assegnato un livello di priorità, e che un processo con priorità più alta viene sempre eseguito prima di un processo con più bassa priorità. Questo vale solo se il processo è pronto (*ready*), cioè "chiede" di essere eseguito; talvolta un processo può essere in attesa (*wait*) di un evento esterno, come un carattere battuto su tastiera, o come la fine di un altro processo. In ogni caso, *tutti* i processi di classe real time pronti vengono eseguiti prima di *qualsiasi* processo in classe timesharing. La classe di priorità sistema viene usata solo dal kernel e non è disponibile agli utenti.

Con questa nuova possibilità di priorità, potete specificare completamente il comportamento del sistema; tuttavia è facile cadere in errori e compromettere il buon funzionamento del sistema. Solo il superuser può configurare e manipolare le classi di priorità, e queste operazioni devono essere compiute sempre con cura estrema. È preferibile far trattare da esperti la gestione delle priorità se volete sperimentare voi stessi in quest'area, preparatevi a dover reinizializzare la macchina di frequente.

Il comando **priocntl** (*priority control*), consente di assegnare i processi a una classe di priorità e di definire i livelli di priorità dei processi all'interno della classe. Il comando prevede diversi modi, corrispondenti alle differenti azioni che volete compiere; dovete selezionare uno di questi modi nella linea di comando di **priocntl**. Dopo aver selezionato un modo, potete selezionare un processo o un gruppo di processi per l'esecuzione del comando. L'opzione **-l** lista le classi di priorità configurate correntemente nel sistema, per esempio:

```
$ priocntl -l
CONFIGURED CLASSES
= = = = = = = = = = = = =
SYS (System Class)
RT (Real Time)
        Maximum Configured RT Priority: 59
TS (Time Sharing)
        Configured TS User Priority Range: -20 through 20
$
```

Oltre all'elenco delle classi, l'output evidenzia l'estensione disponibile in ciascuna classe per una regolazione fine delle priorità relative a ciascun processo. Tanto più basso è il numero di priorità, quanto più bassa sarà la priorità del processo nel sistema.

L'opzione **-d** visualizza la classe di priorità di un processo:

```
$ priocntl -d
TIME SHARING PROCESSES:
      PID  TSUPRILIM     TSUPRI
     6620         -1         -1
$
```

In realtà, questa linea di comando visualizza solo i parametri associati al comando **priocntl** stesso; ciò non è particolarmente utile, ma può servire per illustrare il formato di output del comando. Per ogni processo elencato viene indicato il pid, la priorità di esecuzione del processo (TSUPRI o RTPRI), e la massima priorità ammessa per processi TS (TSUPRILIM). Per i processi RT, la colonna TQNTM indica il numero di unità di tempo di CPU (*clock tick*) che il sistema concede al processo per l'esecuzione, prima di trasferire il controllo a un altro processo RT con la stessa, o più alta, priorità. Ricordate che i processi RT (se pronti) vengono sempre eseguiti prima dei processi TS, e che i processi RT con valori di RTPRI più alti vengono sempre eseguiti prima dei processi RT con valori di RTPRI più bassi.

Per selezionare la visualizzazione di alcuni processi attivi, dovete usare l'opzione **-i**, seguita dalla parola chiave **pid**, e di seguito specificare una lista di pid di processi, oppure **all** per selezionare tutti i processi nel sistema, per esempio:

```
$ priocntl -d -i pid all
```

Questo visualizza informazioni per tutti i processi nel sistema.

L'opzione **-i** accetta diverse altre parole chiave, oltre a **pid**. Usate **-i class** seguita da RT o TS per conoscere tutti i processi della classe selezionata, per esempio:

```
$ priocntl -d -i class RT
REAL TIME PROCESSES:
   PID    RTPRI       TQNTM
  6714        0        1000
$
```

Per conoscere tutti i processi associati a un determinato utente, usate **-i uid** seguito da un user id numerico, per esempio:

```
$ priocntl -d -i uid 104
```

Sono disponibili anche altri tipi.

L'opzione **-s** assegna la classe di priorità a un processo esistente; può essere associata con i formati dell'opzione **-i**, per assegnare simultaneamente la priorità a diversi processi. Gli utenti normali possono ridurre la priori-

tà dei propri processi, ma solo il superuser può aumentare la priorità ai processi, o cambiarla ai processi appartenenti ad altri utenti. Inoltre, **priocntl** **-s** consente diverse opzioni specifiche per le classi RT o TS; dovete avere molta cura nell'utilizzo di queste opzioni. Per esempio, potete cambiare un processo TS esistente in processo RT:

```
# priocntl -s -c RT 6714
```

Questo cambia il processo con pid 6714 da processo TS a processo RT, con i valori di default della priorità RT. Potete anche modificare le priorità all'interno di una classe.

Infine, potete eseguire un nuovo processo a un livello di priorità specificato, con l'opzione **-e**; la linea di comando da eseguire segue le opzioni di **priocntl**, per esempio:

```
# priocntl -e -c RT cu -s9600 -l/dev/tty00s
```

Questa linea di comando esegue un comando **cu** in classe RT, usando i valori di scheduling RT di default. Per usare in maniera effettiva le opzioni **-s** e **-e** dovete avere i privilegi di superuser.

Capitolo 12

Gestione del sistema UNIX

Per *gestione del sistema* (system administration) si intende l'insieme delle operazioni ripetitive di manutenzione sosftware che il sistema richiede mentre cresce e cambia. In altre parole, aggiungere e cancellare identificatori di login di utente, installare nuovo software, cancellare file di servizio, formattare dischetti ed eseguire copie di salvataggio sono tutte attività che rientrano nella gestione del sistema. Un utente di una macchina UNIX viene generalmente designato quale gestore di sistema e il suo compito è di mantenere efficiente il sistema, a vantaggio degli altri utenti. La gestione di si-

stema ha una importanza vitale anche in una macchina UNIX monoutente. Anche se il lavoro di gestione risulta sensibilmente più impegnativo su un sistema multiutente che su un sistema monoutente come MS-DOS, le recenti versioni di sistema UNIX dispongono di strumenti notevolmente migliorati per facilitare la gestione del sistema. Molte implementazioni, e molti sistemi SVR3 e SVR4, forniscono un'interfaccia utente per uso gestionale (*user agent*); generalmente uno strumento che con menu guida il gestore nelle sue varie funzioni, mentre lo strumento stesso compie una grande parte del lavoro più complicato senza richiedere particolare attenzione. La maggior parte di queste interfacce comprende menu di video e schemi per l'introduzione dei dati in ingresso, con un buon controllo degli errori.

Questi strumenti consentono una gestione del sistema UNIX molto più facile che in passato, e anche gestori esperti tendono a usare questi strumenti in luogo dei metodi manuali finora usati. Tuttavia, molti di questi "agenti" sono progettati per utenti che conoscono l'idea e la terminologia proprie delle procedure manuali. Anche se gli strumenti di interfaccia facilitano il compito con menu di scelte e con controlli di errore nei vari passaggi, non eliminano la necessità di conoscere i concetti che stanno dietro alla procedura.

In questo capitolo, esamineremo i principi base della gestione del sistema, le differenti interfacce utente e alcune delle procedure di gestione usate comunemente; riprenderemo in maggiore dettaglio questi argomenti nei prossimi capitoli. Dopo che avrete acquisito una discreta conoscenza del sistema UNIX dovreste sperimentare sia le interfacce facilitate che i metodi manuali, per imparare a gestire efficientemente la vostra macchina.

12.1 Il superuser

Il gestore del sistema è generalmente un utente responsabile del funzionamento corretto del sistema. L'esperienza ha dimostrato che una singola persona può garantire il buon funzionamento del sistema molto meglio di parecchie persone. In una piccola macchina UNIX il proprietario è probabilmente il gestore del sistema, dà accesso agli altri utenti, salva i file su disco e così via, mentre in un sistema multiutente il gestore deve anche avere funzioni di controllo della sicurezza del sistema, e mantenere efficiente la macchina, a vantaggio degli altri utenti. La persona del gestore diventa anche il punto di contatto e raccolta delle richieste degli altri utenti.

Il sistema UNIX definisce uno speciale identificatore di login, **root**, con cui individua il gestore del sistema. Questi dispone di password personale e ha privilegi speciali, tra cui pieno accesso a tutti i file e risorse del sistema; per questo motivo, il login **root** viene chiamato superuser. Questo identificatore di login deve avere sempre una password, che deve essere a conoscenza solo del gestore del sistema e possibilmente di una persona che può

sostituirlo. L'utente **root** è in grado di causare grave danno al software e ai file di sistema per cui deve operare con la massima cautela. Anche se voi siete il gestore del sistema, dovete riservare l'identificatore di login **root** solo per funzioni gestionali e utilizzare un normale identificatore di login per tutte le altre operazioni. Questo garantisce una certa misura di protezione perché il sistema non consente a un utente normale di apportare modifiche che sono di competenza esclusiva del gestore.

L'AMBIENTE DEL SUPERUSER

Lo shell visualizza un prompt speciale, il carattere # (cancelletto), per segnalare che siete il superuser e ricordarvi che avete accesso completo al sistema.

Anche l'ambiente di sistema per un utente con l'identificatore di login **root** differisce notevolmente da quello definito per un utente normale. Il directory home è il directory / (barra o radice), e anche il contenuto di default della variabile **PATH** è probabilmente differente. Inoltre, alcuni comandi si comportano in modo diverso quando sono usati dal superuser, e offrono quasi sempre maggiori possibilità, per cui tutte le linee di comando che potevate utilizzare come utente normale sono accettate anche quando entrate nel sistema come **root**. L'identificatore di login **root** ottiene i servizi di **.profile** come un utente normale e quindi potete creare o cambiare il file **/.profile** per personalizzare l'ambiente del superuser.

Sulle macchine UNIX attuali l'impiego dell'identificatore di login **root** è limitato alla console di sistema, per cui non potete utilizzarlo da un terminale remoto, a meno di speciali adattamenti in **/etc/default/login**. Ogni volta che intendete eseguire compiti gestionali da console, dovreste terminare la sessione di lavoro da utente normale ed entrare nuovamente nel sistema da superuser (**root**), per ritornare poi al vostro iniziale identificatore di login quando il lavoro di gestione è concluso.

12.2 Il comando su

Il sistema UNIX fornisce anche un mezzo che permette di cambiare temporaneamente l'identificatore di login, senza uscire dal sistema; si tratta del comando **su** (switch user, o superuser). Il comando **su** è accessibile anche da un terminale remoto, per cui in situazione di emergenza potete entrare nel sistema utilizzando un normale identificatore di login e poi acquisire i privilegi di **root** con il comando **su**. Quando avete finito, dovete uscire appena possibile dall'ambiente privilegiato **su**.

Potete utilizzare **su** per passare in qualunque altro identificatore di login oltre che in **root**, ma **root** è predefinito, o default. Il comando **su**:

```
$ su
Password:
#
```

Dopo che avete introdotto la password corretta per l'identificatore di login **root**, il prompt diventa quello del superuser (#). Come al solito, il sistema non visualizza la password; se non è corretta, il comando **su** genera un messaggio di errore:

```
$ su
Password:
Sorry
#
```

In qualche sistema, il comando **su** richiede di specificare il pathname completo:

```
$ /sbin/su
```

Non usare il pathname completo può essere un rischio per la sicurezza, per i motivi che saranno esaminati nel Capitolo 22, perciò è buona regola utilizzare il pathname completo ogni volta che lanciate **su**.

Quando il comando **su** è in esecuzione, disponete dei pieni privilegi del gestore di sistema. Per ritornare al vostro normale identificatore di login, premete CTRL-D o utilizzate il comando **exit**:

```
$ /sbin/su
Password:
# exit
$
```

Il comando **su** di fatto definisce uno shell di livello inferiore (subshell) per la vostra normale sessione e, quando terminate **su**, ritornate al vostro normale ambiente di lavoro.

L'AMBIENTE su

Il comando **su** definisce un ambiente di shell uguale al vostro normale ambiente di lavoro, con in più disponibili i privilegi di superuser, o root. In questo modo conservate la vostra variabile **PATH**, il vostro home directory e così via. Per passare completamente in ambiente **root** e abbandonare temporaneamente il vostro normale ambiente di lavoro, lanciate il comando **su** con l'opzione – (meno).

```
$ /sbin/su
Password:
# echo $HOME
/home/giorgio
# exit
```

```
$ /sbin/su -
Password:
# echo $HOME
/
# exit
$
```

Solo quando compare l'opzione – sulla linea di comando **su** mette a disposizione i servizi di **.profile** del superuser. Questa differenza può talvolta essere causa di confusione, per cui abbiate cura di specificare il – quando volete passare a un completo ambiente **root**.

PASSAGGIO IN UN'ALTRA SESSIONE

Potete anche utilizzare **su** per passare a un altro utente diverso da **root**, specificando l'identificatore di utente come ultimo argomento della linea di comando.

```
$ /sbin/su - leo
Password:
$
```

Come al solito, il comando **su** chiede la password corretta dell'utente su cui volete passare, e l'opzione – permette di creare l'ambiente di lavoro per quell'utente. Viene anche presentato il PS1 del nuovo utente, mentre il prompt # rimane riservato per il superuser.

Potete anche utilizzare **su** per lanciare un singolo comando, come se foste l'altro utente; quando il comando termina, il privilegio di superuser viene revocato e ritornate immediatamente al vostro abituale ambiente di lavoro.

```
$ /sbin/su - root -c "chgrp sys /etc/passwd"
Password:
$
```

Con questo formato dovete specificare l'identificatore di utente e il comando da eseguire delimitato da virgolette; l'argomento – attiva l'ambiente di lavoro di quell'utente. L'argomento **-c** (command), prima del comando che volete eseguire, richiede a **su** di attivare un subshell per eseguire il comando. Potete utilizzare **-r** (restricted) invece di **-c** se volete eseguire il comando in una versione limitata di shell piuttosto che in uno shell completo. Le caratteristiche della versione limitata di shell (restricted shell) sono trattate nel Capitolo 22.

Quando siete nel sistema come superuser e poi lanciate il comando **su** per entrare nell'ambiente di lavoro di un altro utente, **su** non richiede la password, perché come utenti **root** possedete già pieni privilegi. Se l'utente non

ha una password assegnata, **su** non la richiede, ma in tutti gli altri casi dovete conoscere la password corretta prima di entrare nell'ambiente di lavoro di un altro utente.

La linea di comando **su** con un comando da eseguire è usata spesso nelle procedure di shell quando sono richiesti privilegi speciali.

12.3 Notizie e messaggio del giorno

Uno dei compiti del gestore di sistema è di informare gli altri utenti sulle novità del sistema, preavvertire dell'orario di chiusura del sistema, annunciare il nuovo software e fornire altre notizie di interesse generale. Il comando **news** viene spesso usato per questi scopi perché consente agli utenti di leggere un avviso con notizie di vario genere quando si collegano alla macchina.

Per informazioni di maggiore priorità viene usato spesso il *messaggio del giorno* (**motd** o message of the day), che viene automaticamente visualizzato dal sistema a ciascun utente quando si collega al sistema. Il gestore del sistema deve solo provvedere a scrivere il file **/etc/motd** per creare il messaggio. Tenete conto che quando un utente si collega, il messaggio non può essere evitato, quindi è bene usare questa caratteristica con discrezione e cancellare i messaggi non più di attualità.

Per avvisi della massima priorità, come per esempio una imminente chiusura del sistema, potete utilizzare il comando **/usr/sbin/wall** (write all), che legge un messaggio in standard input e lo invia direttamente ai terminali di tutti gli utenti collegati.

Tratteremo queste caratteristiche in maggior dettaglio nel Capitolo 14.

12.4 Messaggi del sistema al gestore

Molti processi demon di sistema e altre speciali funzioni amministrative sono configurati per inviare messaggi per posta elettronica al gestore di sistema, durante il funzionamento. Normali demon di servizio di **uucp**, inconvenienti alla stampante, e anche l'installazione di nuovo software, causano la creazione e l'invio di messaggi per posta a un gestore. Usualmente questi messaggi vengono indirizzati a utenti diversi, a seconda del programma che li origina. Per esempio, messaggi riguardanti il sistema **uucp** vengono inviati all'utente **uucp**, mentre i messaggi relativi all'installazione di software vengono inviati all'utente **root**. Altri messaggi ancora vengono comunemente inviati agli utenti **lp**, **adm** e **sysadm**.

In molti casi, questi messaggi sono solo informativi di una certa azione intrapresa dal sistema, ma in certi casi forniscono utili notizie riguardo parziali malfunzionamenti del sistema, come problemi di stampante, o an-

che evidenziano alcuni tipi di violazione della sicurezza. In ogni caso, questi messaggi sono importanti mezzi per seguire il corretto funzionamento della macchina, e non devono essere trascurati. Dopo che una macchina è stata in attività per una settimana o due, esaminate il contenuto del directory della posta **/var/mail**, e determinate la rispedizione della posta (trattata nel Capitolo 14) per i login di sistema che hanno ricevuto questo tipo di posta. Normalmente il gestore di sistema rispedirà questi messaggi al login **root**, o al login personale del gestore, se **root** viene usato raramente.

Ancora una volta, abbiate cura di comprendere e trattare questi messaggi non appena ricevuti; sono realmente molto utili per mantenere il corretto funzionamento della macchina.

12.5 Soluzione di problemi inusuali

Dopo l'iniziale configurazione del sistema, il sistema UNIX normalmente funziona senza problemi, e molte operazioni di gestione sono banale routine. I problemi più comuni consistono nella perdita di file, cancellati per errore, o perduti in conseguenza di malfunzionamenti hardware. Altri problemi possono essere causati dalla carenza di spazio sui dischi rigidi, se sono troppo occupati, da inconvenienti con la stampante e dalla gestione degli identificatori di utenti. La posta elettronica e la definizione dei parametri di **uucp** possono richiedere maggiore attenzione del solito, se cambia l'elenco dei corrispondenti della posta. Tuttavia queste attività devono essere considerate normali attività di gestione, ma non problemi del sistema.

Problemi reali, comunque rari, possono presentarsi inaspettatamente a causare malfunzionamenti della macchina. Le porte seriali possono smettere di funzionare, i processi apparire e sparire, e altri imprevisti fenomeni possono presentarsi nel sistema; in questi casi la vostra immediata reazione deve sempre essere quella di reinizializzare la macchina. Molti comportamenti anomali del sotfware del sistema UNIX scompaiono dopo la reinizializzazione; se ciò non avviene, e non vi sono guasti hardware, la vostra prossima mossa deve consistere nel ricaricare il software di sistema (dopo aver salvato tutti gli altri file, locali o privati). Risulta difficile anche per gli esperti individuare e risolvere i malfunzionamenti maggiormente complessi e sconcertanti.

12.6 Interfacce utente per la gestione del sistema

Le case costruttrici di software forniscono alcune interfacce utente (user agent) che facilitano le operazioni di routine per la gestione del sistema. La release standard SVR4 include il sistema OA&M (Operations, Administration, and Maintenance), ma dovete consultare la documentazione specifica

della vostra macchina per determinare come lavora il vostro sottosistema amministrativo.

Solo il superuser dovrebbe utilizzare questi strumenti, ma alcune versioni di sistema UNIX permettono di delegare ad altri utenti i privilegi di gestione del sistema. Quando definite un nuovo identificatore di utente, potete scegliere se assegnargli o meno questo potere, tuttavia, la sicurezza è meglio garantita quando solamente uno o al massimo due utenti sono autorizzati a eseguire attività di gestione.

In molte release di UNIX, lo strumento di gestione è denominato *sysadm* (system administration); se disponete di questo comando, dovete sempre lanciarlo dal directory / (**root**), l'home directory per il gestore di sistema.

```
# cd /
# sysadm
```

Se il vostro identificatore di login non è **root**, o se il directory corrente non è /, **sysadm** può non essere utilizzabile.

Altre release possono usare il nome **sysviz** (visual system administration), oppure **adm** in luogo di **sysadm**. I comandi **sysadm** e **adm** sono strettamente per gestione, mentre i menu di **sysviz** includono alcune funzioni di utente, come la visualizzazione di file e directory, la spedizione di posta e l'esecuzione di applicazioni installate.

Queste interfacce utente sono generalmente grandi sistemi a menu con molte scelte e sottosistemi distinti per le diverse operazioni di gestione. Differiscono sostanzialmente nella presentazione e nei dettagli fra i diversi produttori, ma tutte dispongono di funzioni simili perché tutti i sistemi UNIX richiedono le stesse operazioni base di gestione. Tratteremo in questo capitolo le funzioni più comuni, ma le richieste specifiche e le scelte di menu disponibili sul vostro sistema possono differire da quelle mostrate in questi esempi. Potete facilmente abbinare le funzioni con le voci dei menu disponibili nella vostra macchina; dedicate del tempo a sperimentare con attenzione le caratteristiche delle interfacce utente che avete a disposizione, perché possono facilitare sensibilmente l'utilizzo di UNIX.

La Tabella 12.1 elenca un insieme di funzioni che generalmente l'interfaccia utente mette a disposizione. Tratteremo queste funzioni una per una nei paragrafi seguenti.

CONTROLLO DELLE INTERFACCE DI GESTIONE

L'interfaccia standard di gestione dei sistemi SVR4 di base è il sistema OA&M (Operations, Administration, and Maintenance); a sua volta basato sugli strumenti FMLI (Form and Menu Language Interpreter). Per usare questi strumenti, dovete installare l'appropriato pacchetto di software.

Tabella 12.1 Le funzioni di gestione nelle interfacce utente SVR4.

Operazione principale	Sottofunzioni
Backup and restore	Backup history
	Schedule backup
	Personal backup
	System backup
	Personal restore
	System restore
Disk operations	Add device
	Remove device
	Erase disk
	Floppy-to-Floppy copy
	Format disk
File system operations	Check file system
	Create file system
	Display disk usage
	Find files
	List mounted file systems
	Mount file system
	Unmount file system
Networking	**uucp** management (mail)
	Distributed file system configuration
	Networked addresses of machines
Printer operations	Printer setup
	Filters setup
	Forms setup
	Printer restart
	Printer status
	Remote printing
Schedule jobs	Add
	Delete
	Display schedule
Service access facility	Port monitors
	Port services
	Terminal setup
Software management	Check
	Install
	Remove
	Display
	Application dependent
System configuration	Display configuration
	Machine name
	Date and time
	Root password
	Reboot
	Shutdown
	List users
User Logins	Add
	Delete
	Display
	Reset passwords
	Set defaults

Se disponete del sistema OA&M, dopo avere eseguito il comando **sysadm**, lo schermo apparirà all'incirca come nella Figura 12.1; il menu principale compare nella parte sinistra in alto. Potete spostare la banda ad alta intensità attraverso il menu, con il tasto TAB, oppure battendo alcuni dei primi caratteri di una voce del menu. I menu possono contenere più voci di quelle contemporaneamente possibili sullo schermo; se tabulate oltre la fine del menu, possono apparire voci addizionali, se esistono, altrimenti la barra ad alta intensità ritorna all'inizio del menu.

Alla fine dello schermo compare una fila di diciture che assegnano delle funzioni ai primi otto tasti funzione della tastiera; per esempio, F1 è il tasto per Help. Premendo il tasto F1 in qualunque momento, compare una breve descrizione in una finestra; abbandonate la finestra Help premendo il tasto Cancel, F6. Il tasto F6 viene usato per abbandonare qualunque finestra, eccetto il menu principale.

Per uscire dal sistema OA&M e ritornare a shell, premete CMD-Menu (F7), che richiama un menu di comandi; in questo selezionate Exit per ritornare a shell. Gli altri tasti funzione forniscono funzioni addizionali, ma sono correntemente disponibili solo le funzioni che compaiono nelle diciture nella linea finale.

Potete selezionare una voce nel menu principale, se è in alta intensità, premendo il tasto Save o Enter, F3. Anche il tasto RETURN può avere lo stesso effetto, tuttavia l'uso del tasto F3 è il modo corretto di selezionare una voce in un menu o di salvare il contenuto di una schermata. Quando viene

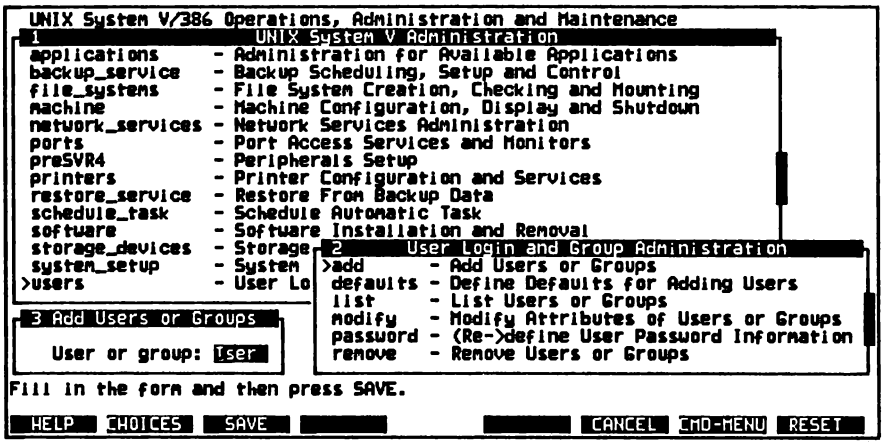


Figura 12.1 Semplice schermata dell'interfaccia di gestione OA&M.

selezionata una voce, il sistema presenta un sottomenu o uno schema, a seconda della funzione corrente; la nuova finestra apparirà sovrapposta al menu principale, come mostrato nella parte in basso a destra della Figura 12.1.

Alla fine della serie di menu, usualmente compare uno schema con campi per informazioni come nome utente, nome directory o device, e simili. Potete passare da un campo all'altro con TAB, ma alcuni campi richiedono obbligatoriamente l'introduzione di un dato per consentire di passare a un altro campo. Usualmente il sistema vi guida con istruzioni e messaggi d'errore nella linea dello schermo immediatamente sopra alla linea delle diciture dei tasti funzione.

Gli schemi spesso mostrano prespecificate le voci che ciascun campo può accettare; in questo caso sarà disponibile la funzione Choices (scelte), F2. Se potete premere F2 e selezionate una voce nello schema, quella voce verrà introdotta nel campo corrente; potete vedere in basso a sinistra nella Figura 12.1 uno schema molto semplice, con una sola voce! Per far eseguire al sistema un'azione su uno schema, premete F3; per abbandonare lo schema e l'operazione in corso, premete F6.

Usualmente, nel sistema OA&M sono disponibili molte altre funzioni; altri sistemi possono differire nelle funzioni e nell'interfaccia utente; dovrete verificare le possibilità del vostro sistema.

VALORI DI DEFAULT

La maggioranza delle interfacce d'utente fornisce l'accesso a parti molto complesse del sistema UNIX, per questo vi sono spesso molte voci di menu e schemi associati a ciascuna funzione. Per contro, questi schemi e menu usualmente propongono valori di default per molti dati; se non comprendete esattamente il significato e l'uso di questi dati, lasciate scegliere al sistema. Nella trattazione seguente, si presume che le voci non menzionate assumano corretti valori di default.

12.7 Gestione di dischetti

Le voci di menu **Disk Operations** o **Storage Devices** sono dedicate alle operazioni di *gestione dei supporti*, cioè la formattazione e la ricopiatura di dischetti.

Generalmente, per usare un dischetto, lo dovete: *formattare*, definire su di esso un *file system* e poi *montare* (mount) in una precisa locazione del file system; dopodiché potete utilizzare il dischetto proprio come un normale directory. Potete lanciare quindi i comandi **mv** o **cp** sui file memorizzati sul dischetto, cambiare le modalità di accesso ai file o utilizzare il pathname dei directory nei comandi e nelle procedure di shell.

Quando avete terminato le operazioni sul dischetto, potete *smontarlo* (unmount) e quindi rimuoverlo dal drive. Un secondo metodo per accedere ai dischi è disponibile con il comando **cpio** (copy in/out), che viene utilizzato per le funzioni **backup** e **restore**. Nel Capitolo 18 tratteremo in dettaglio i supporti di memoria.

FORMATTAZIONE DI DISCHETTI

Selezionate la corretta opzione dal menu **Storage Devices** per formattare un dischetto. Il sistema può formattare dischetti da 360 kB, a doppia faccia e doppia densità, e da 1.2 MB ad alta densità, per drive da 5 pollici e 1/4; e dischetti a 1.4 MB per drive da 3 pollici e 1/2. Il sistema vi chiede di inserire il dischetto e di chiudere lo sportello del drive prima di lanciare la procedura di formattazione. Non potete formattare dischetti protetti in scrittura, per cui accertatevi che l'etichetta sia stata rimossa dall'apposita tacca sul dischetto prima di inserirlo nel drive.

Il sistema generalmente visualizza messaggi di errore se l'operazione di formattazione non ha successo. Se questo accade, ripetete l'operazione di formattazione di nuovo, ma scartate il dischetto se l'operazione dà esito negativo dopo due tentativi. È sicuramente meno costoso acquistare nuovi dischetti che perdere i dati che avete memorizzato, per cui non correte rischi con dischetti rovinati.

Per cancellare un dischetto, formattatelo di nuovo.

CREAZIONE DI UN FILE SYSTEM SU UN DISCHETTO FORMATTATO

Dopo l'operazione di formattazione, dovete definire un file system sul dischetto prima di montarlo. Se utilizzate il dischetto in una operazione di salvataggio, non avete bisogno di definire un file system: è sufficiente la sola formattazione. Se, invece, intendete utilizzare il dischetto come un normale directory all'interno del file system, dovete utilizzare la funzione di creazione del file system sul dischetto formattato. La procedura fa parte del menu **File Systems**; inserite il dischetto formattato e selezionate dall'interfaccia utente l'opzione **Create File System**.

In SVR4 dovete scegliere un tipo di file system; la scelta non è importante se montate sempre i dischetti tramite interfaccia utente. Il tipo **ufs** comunque è preferibile a **s5**, a meno che prevediate di montare il dischetto su SVR3 o sistemi più vecchi, che richiedono il tipo **s5**.

Dopo aver selezionato il tipo, dovete specificare il numero di *blocchi* per il nuovo dischetto; se l'interfaccia utente non vi indica un valore di default, dividete la capacità del disco per 512, per esempio, un dischetto da 1.2 MB avrebbe 1440 blocchi disponibili. Se indicate un valore troppo grande per il disco, l'operazione viene rifiutata; se indicate un valore troppo basso, lo spazio rimanente non verrebbe utilizzato.

Viene anche richiesto di introdurre il nome del file system, che serve a determinare la collocazione nel file system del dischetto quando viene montato. Questo significa che potete passare (**cd**) a quel directory per accedere al contenuto del dischetto. I dischetti montati sono trattati proprio come normali directory; ciascuno assume una collocazione nel file system, quando viene montato, quando viene smontato scompare dal file system. Generalmente, il dischetto si trova alla locazione di directory **install** o **/mnt**; qualsiasi file o subdirectory di quei directory è contenuto sul dischetto. Potete utilizzare qualunque locazione di directory per montare dischetti, ma, se non esiste, dovete crearla esplicitamente.

Potete anche assegnare a un dischetto un nome che non superi i sei caratteri, come vostro identificativo, ma questa operazione non è indispensabile. Alcune versioni di UNIX richiedono di stabilire anche il massimo numero di file e directory che il disco può contenere, ma il valore di default è di solito sufficiente.

Dopo aver selezionato l'opzione **Create File System**, non rimuovete il dischetto dal drive fino a quando il sistema non vi richiede di farlo. Molti sistemi automaticamente montano il dischetto, nella stessa operazione di creazione di file system.

MONTARE UN DISCHETTO

Potete montare un dischetto ogni volta che intendete utilizzarlo; tuttavia per poter essere montato, il disco deve contenere un file system. L'opzione *mount* si trova generalmente nel menu **File Systems**. Dopo aver montato il disco, potete uscire dall'interfaccia utente e utilizzare gli abituali comandi per referenziare il disco sotto la sua collocazione di directory (generalmente **/install** o **/mnt**) e impiegarlo come un normale directory. Prima di rimuovere il dischetto dal drive rientrate nell'interfaccia di gestione del sistema per smontare il dischetto; danneggereste seriamente il contenuto rimuovendo il dischetto dal drive prima di averlo scaricato.

Quando montate un dischetto, dovete specificare correttamente il tipo di file system scelto all'atto della creazione; tuttavia molti sistemi individuano automaticamente il tipo di file system da montare.

Dovete inoltre indicare la collocazione dove montare il contenuto del dischetto, generalmente il nome di file system da voi specificato alla creazione del file system sul dischetto, ma se occorre potete montare un dischetto in una collocazione differente.

Potete scegliere se montare un dischetto accessibile solo in lettura (read only), oppure sia in lettura che in scrittura. Nel primo caso, non potete riscrivere nessuno dei file, e potete utilizzare un dischetto protetto in scrittura; al contrario, il dischetto deve essere abilitato alla scrittura se è montato accessibile in lettura e scrittura.

Un disco che viene montato per la prima volta, dopo aver creato il file system, è vuoto. I file o i subdirectory che create nel directory in corrispon-

denza del punto di collocazione sono sul dischetto, e saranno nuovamente disponibili in quella collocazione la volta successiva che montate il dischetto. Se in una successiva utilizzazione montate il dischetto modificando il punto di collocazione, i file appaiono sotto il nuovo punto di collocazione e non più in corrispondenza di quello vecchio. Per copiare un file dal disco rigido a un dischetto, o viceversa, potete utilizzare il comando **cp** o **mv**; il comando **ln** non può avere effetto.

Per smontare un dischetto, dovete indicare il device o il punto di collocazione usato per montarlo; curate di non sbagliare, perché potreste smontare qualche importante file di sistema! Prima di smontare, dovete anche concludere ogni applicazione che usi file su quel dischetto, e rimuovere l'effetto di **cd** a ogni directory al punto di collocazione, o sottostante; il dischetto deve essere completamente inutilizzato prima di poterlo smontare.

12.8 Copia di un dischetto

Per copiare un dischetto selezionate il menu **Storage Devices** dell'interfaccia utente di gestione del sistema. Prima di effettuare la copia, il dischetto destinazione deve essere formattato nello stesso modo del dischetto sorgente, ma non è necessario che contenga anche un file system poiché l'utility lo copia automaticamente dal sorgente.

La funzione vi chiede quale drive volete usare (*diskette1* o *diskette2*), quindi di inserire nel drive il dischetto sorgente e, dopo averlo letto, chiede di rimuoverlo. Infine, viene chiesto di inserire il nuovo dischetto, e l'operazione di copiatura viene eseguita. Potete eseguire generalmente più copie dello stesso dischetto sorgente; l'interfaccia utente vi chiede di inserire un altro dischetto nuovo formattato dopo che è stata completata una copia.

Ogni copia risulta identica al dischetto sorgente, con tutti i file, i file system e i directory, con le relative date e orari invariati.

Potete copiare indifferentemente qualsiasi tipo di dischetto, che può essere un disco di supporto, di salvataggio o di inizializzazione.

12.9 Salvataggio e ripristino di dischi

Utilizzate le opzioni **backup** e **restore** per salvare i file di sistema e i directory dal disco rigido della macchina a un disco rimovibile o un nastro. In questo modo, potete preservare i dati se il disco rigido ha delle anomalie di funzionamento o se per sbaglio cancellate qualche file importante. Generalmente conviene mantenere i file che avete salvato aggiornati alla situazione del sistema, per evitare che un guasto di sistema possa farvi perdere dati importanti. È vero che le probabilità che si verifichino guasti nel sistema sono minime, ma quei pochi capitano spesso nelle situazioni meno opportu-

ne e il prezzo che si paga è sempre molto alto, per cui vi consigliamo di salvare frequentemente i vostri dati.

Esistono di solito quattro diverse procedure di salvataggio. La prima è il salvataggio completo (*full backup*), che potete eseguire subito dopo l'installazione del software del sistema. La seconda è il salvataggio regolare e incrementale (*incremental backup*), con cui potete salvare solo i file che hanno subito aggiornamenti rispetto all'ultima operazione di salvataggio; dovete avere eseguito almeno un iniziale salvataggio completo, prima di poter eseguire un salvataggio incrementale. La terza procedura è il salvataggio personale (*personal backup*), che permette di salvare tutti i file che appartengono a un home directory. L'ultima, è il salvataggio selettivo (*selective backup*), che salva un elenco di file da qualsiasi posizione nel file system. Inoltre, potete chiedere al sistema un elenco degli ultimi salvataggi, con indicato il momento in cui sono stati effettuati e quali dati sono stati inclusi (*backup history*).

Le operazioni di salvataggio hanno luogo generalmente su dischetti, ma potete anche utilizzare nastri magnetici se la vostra macchina possiede una unità a nastro. Formattate anzitempo un numero sufficiente di dischetti, o di nastri, che effettivamente occorrono, per eseguire senza problemi l'operazione di backup perché la procedura di salvataggio vi chiede di cambiare il supporto quando ha riempito quello corrente. Formattate un elevato numero di supporti prima di lanciare la procedura e, con l'esperienza, imparerete presto a valutare quanti supporti sono necessari per salvare i vostri file.

Se un'operazione di salvataggio coinvolge più supporti, numerateli progressivamente quando li estraete dal drive, perché dovete inserirli nello stesso ordine quando li ricaricate sul disco rigido della macchina.

Selezionate il file system che vi interessa tra i file system disponibili nel vostro sistema. Il numero di file system di un sistema dipende dalla configurazione e dal numero di dischi installati nella macchina. Salvate ogni file system separatamente, perché meno dischetti utilizzate in una operazione di salvataggio, minore è la probabilità che qualche difetto di dischetto crei problemi nel ripristino dei dati. Le operazioni di salvataggio coinvolgono una serie di dischetti che possono essere letti solo nell'ordine in cui sono stati creati. Di solito è possibile tralasciare un dischetto difettoso, durante il ripristino dei file, ma il procedimento non è esente da problemi; in ogni caso una parte di dati viene sempre peduta. È raccomandabile eseguire salvataggi brevi e ripetuti, anziché rari e voluminosi.

Inizialmente dovete effettuare un salvataggio completo, poi sono sufficienti solo salvataggi incrementali, ma senza riutilizzare il set dei dischetti del salvataggio completo. Un salvataggio completo copia tutti i file di sistema oltre ai vostri file; potete arrivare a dovere utilizzare anche 40 dischetti da 1.2 MB, in un sistema tipico. Infine, dovete anche selezionare il dispositivo da usare, che deve corrispondere al tipo di drive installato.

Potete anche salvare una lista di specifici file e directory; dopo avere selezionato dal menu la voce appropriata, di solito *Selective Backup*, potete introdurre un elenco di file e directory da salvare. Potete modificare e correggere l'elenco, aggiungere, eliminare e riesaminarlo in ogni momento. Quando specificate un nome di directory, tutti i file e subdirectory di quel directory vengono inclusi nell'elenco; quando l'elenco soddisfa le vostre esigenze, potete salvare i file.

Ripetiamo che il salvataggio può richiedere numerosi supporti, perciò siate certi di avere a disposizione un adeguato numero di supporti formattati prima di iniziare l'operazione.

RIPRISTINO DI FILE DOPO IL SALVATAGGIO

Se occorre ripristinare i dati salvati, selezionate la funzione **Restore Service**, che vi chiede di caricare i supporti nell'ordine in cui sono stati creati. Assicuratevi di selezionare lo stesso dispositivo che avete utilizzato nella fase di salvataggio.

In caso di un disastroso malfunzionamento del sistema, potete ripristinare l'esatta immagine del vostro sistema al momento dell'ultimo salvataggio. Ricaricate il software di sistema dai dischetti originali, eseguite il ripristino del salvataggio completo e poi ripristinate ciascun salvataggio incrementale nello stesso ordine in cui li avevate effettuati. Si tratta sicuramente di una procedura dispendiosa in termini di tempo, che può garantire però l'integrità del vostro sistema. Comunque, i più esperti gestori di sistema eseguono il salvataggio dei dati directory per directory, utilizzando una procedura manuale o le procedure di salvataggio selettivo dell'interfaccia di gestione. Questo metodo risulta di solito più veloce e richiede molti meno supporti, ma si dimostra più difficile da eseguire rispetto all'impiego dell'interfaccia utente.

12.10 Dati sull'uso del disco rigido

La maggior parte delle interfacce utente dispone di una utility per visualizzare la quantità di spazio utilizzato nel disco rigido e nei file system che sono presenti quando il comando viene eseguito. Queste informazioni generalmente fanno parte del messaggio **System Information** o si trovano nel sottosistema **File Systems**.

La memoria su disco viene generalmente misurata in megabyte di spazio totale disponibile e di spazio libero; spesso vengono forniti i valori anche in percentuale. Alcuni sistemi, invece, specificano la quantità di spazio in termini di blocchi fisici di 512 byte contenuti nel file system.

I dati sull'utilizzo del disco sono molto importanti per una accurata gestione del sistema, perché abituali operazioni del sistema UNIX creano file

temporanei che occupano spazio su disco, senza darne avviso e poi lo liberano appena possono. Se nel vostro sistema la quantità di spazio disponibile è inferiore al 10% del totale, il vostro disco si può rapidamente riempire, e questo può causare il malfunzionamento del sistema. I dischi tendono sempre a riempirsi con una rapidità superiore alle previsioni, per cui è conveniente esaminare regolarmente i dati relativi all'utilizzo del disco, per salvare e cancellare i file non necessari quando lo spazio occupato si avvicina al 90% del totale.

12.11 Inizializzazione della data e dell'ora

L'interfaccia utente fornisce anche una funzione per inizializzare l'hardware di clock e di calendario del sistema con la data e l'ora correnti. La funzione è nota generalmente come **Date and Time** oppure **datetime**, e spesso fa parte del menu **System Setup**. Il sistema UNIX utilizza l'informazione data e ora in diverse attività, per cui è indispensabile inizializzare correttamente nel sistema la data e l'ora. Molti piccoli calcolatori non mantengono un'informazione oraria esatta, anche se dispongono di un hardware di clock interno, per cui conviene controllarla regolarmente e reinizializzarla se l'errore è di qualche minuto. Il sistema può gestire anche le differenze dovute al fuso orario e all'ora legale.

12.12 Spegnimento della macchina

L'interfaccia utente generalmente include un'opzione per spegnere la macchina; questa operazione nel sistema UNIX differisce dal sistema MS-DOS e da altri sistemi operativi monoprocesso. Non dovete *mai* togliere direttamante la tensione alla macchina, perché questa operazione non garantisce l'integrità dei file che si trovano sul disco rigido; utilizzate *sempre* la procedura di spegnimento dell'interfaccia utente oppure lanciate il comando **shutdown** quando siete collegati come superuser (**root**).

Prima di far partire la procedura di spegnimento, verificate che non sia in atto un'operazione critica. Dopo la sua partenza, la procedura di spegnimento avverte gli altri utenti in linea che la macchina verrà spenta tra breve tempo e li mette in condizione di uscire dal sistema, e poi mette il sistema in una situazione che permette di togliere la tensione. In alcuni sistemi anche l'operazione di togliere tensione viene controllata dal software della procedura **shutdown**. Quando compare

```
Reboot the system now.
```

oppure

```
Reset the CPU to reboot.
```

potete togliere tensione, ma *non prima* che questo messaggio venga visualizzato. Le operazioni di inizializzazione e spegnimento sono trattate in dettaglio nel Capitolo 21.

12.13 Aggiunta e rimozione di login id di utente

Il menu **Users** permette di aggiungere o cancellare dal sistema gli identificatori di utente, o login id. Tutti gli utenti del sistema devono possedere un identificatore login id personale che non devono condividere con altri utenti. Questo è importante in termini di sicurezza del sistema, e anche per evitare agli utenti di danneggiare per errore file appartenenti ad altri utenti. Gli utenti possono appartenere a uno stesso unico gruppo o essere membri di gruppi diversi; il gruppo interessa la parte mediana dei diritti di accesso ai file, come il comando **ls −l** li visualizza.

Quando create un nuovo identificatore login id, dovete assegnare all'utente una password iniziale, che l'utente può cambiare quando si collega al sistema. Inoltre, quando gli utenti cessano definitivamente l'uso del sistema, dovete immediatamente rimuovere o disabilitare i loro identificatori di utente per evitare in seguito problemi di sicurezza.

Il menu **Users** include molte opzioni, anche se solo alcune vengono utilizzate regolarmente. **List** visualizza le informazioni associate agli utenti noti al sistema, **Add** consente di aggiungere nuovi utenti, **Remove** permette di cancellare utenti.

Per aggiungere un nuovo utente al sistema, dovete specificare l'identificatore di login e il nome dell'utente. L'identificatore di login deve essere differente da ogni altro nella macchina, e l'interfaccia utente verifica se la condizione è rispettata. La maggior parte degli utenti preferisce scegliere il proprio identificatore di login, per cui dovreste consultare l'utente prima di assegnare l'identificatore. Introducete correttamente il nome personale dell'utente nel campo *Comments*, perché parte del software usa questa informazione, leggendo il file **/etc/passwd**, dove viene memorizzata.

Su molti sistemi dovete anche selezionare il numero di identificazione dell'utente (*id number*) e il nome del gruppo (*group name*); sono rappresentazioni numeriche degli identificatori di utente e di gruppo, usate dal software di sistema. Quasi sempre è meglio lasciarli scegliere al sistema per garantire che non ci siano errori. Altri sistemi non chiedono di scegliere questi identificatori e quindi non potete cambiare quelli predefiniti. Per default, tutti gli utenti normali vengono inclusi in un unico gruppo; questo va bene per per piccoli sistemi. Soltanto se esiste più di un distinto gruppo di interesse tra i vostri utenti, dovete sceglierne esplicitamente uno. In tal caso, dovete creare un nuovo gruppo usando l'opzione **Add Group**, se disponibile, oppure dovete modificare in edit **/etc/group**. Il Capitolo 22 tratterà dettagliatamente il concetto di gruppo.

Successivamente, dovete selezionare il pathname dell'home directory dell'utente e, a meno che non esista qualche necessità particolare, dovete lasciare al sistema questo compito.

Talvolta esiste la possibilità di assegnare una data di scadenza (*expiration date*) del login; questo può essere utile per utenti temporanei, ma gli utenti abituali possono essere disorientati dall'improvvisa scomparsa del loro login id.

Infine, su qualche sistema, potete concedere al nuovo utente i privilegi che appartengono al gestore di sistema e, in questo caso, l'utente è in grado di accedere all'interfaccia gestionale, aggiungere o cambiare i nuovi identificatori di login e concedere ad altri utenti i privilegi di gestione del sistema. Poiché si tratta di un rischio per la sicurezza, solo pochi utenti fidati devono possedere i privilegi di gestione del sistema.

Quando inserite un nuovo utente, di solito dovete assegnare una password: potete anche stabilire una data di validità (*password aging*), per costringere gli utenti a cambiare periodicamente la loro password. Potete specificare un intervallo minimo e massimo, in giorni, di validità della password; un minimo di pochi giorni e un massimo di 180 sono termini accettabili. Non create *mai* un login id di utente senza password; assegnate una password iniziale a ciascun utente ed esigete che venga cambiata al più presto possibile. Potete assegnare alla password lo stato *lock*, per impedire all'utente di collegarsi al sistema, ma non in questo caso.

Dopo aver eseguito tutte le operazioni descritte, il sistema visualizza tutte le informazioni introdotte e vi lascia l'opzione tra installare l'utente o annullare l'operazione. Evitate di installare l'utente se avete commesso qualche errore perché potreste introdurre rischi per la sicurezza nel sistema. Le altre opzioni del menu **Users** hanno un comportamento analogo. Per rimuovere un identificatore di utente, selezionate **Remove**; l'opzione **Modify** permette di cambiare gli attributi di login di un utente senza rimuovere e ricreare l'identificatore di login; il menu Password permette di cambiare la password di un utente o inibire temporaneamente a un utente l'uso del sistema, senza rimuovere tutti i suoi file e directory. In seguito potete ancora cambiare password, e consentire all'utente di nuovo l'uso del sistema. L'opzione *lock* in **Password Status** consente queste operazioni; abbiate cura di riabilitare o cancellare il login id non appena la condizione cambia.

12.14 Installazione di pacchetti software

Il sistema generalmente dispone di strumenti per installare e rimuovere pacchetti applicativi: questi strumenti abitualmente risiedono sotto il menu **Software**.

Queste funzioni non supportano tutto il software aggiuntivo perché chi sviluppa l'applicazione deve adattarla allo schema, tuttavia, se il pacchetto

software è compatibile, potete installarlo, rimuoverlo e visualizzarne gli attributi. Seguite le istruzioni di installazione allegate al software applicativo.

La voce **Package Location** seleziona la sorgente del pacchetto da installare; l'opzione *spool* viene usata per pacchetti distribuiti tramite posta elettronica, più comunemente vengono usate le opzioni *Ntape1* o *diskette1* per cartucce di nastro magnetico o dischetti, rispettivamente.

In aggiunta, abitualmente potete listare i pacchetti installati, verificare i pacchetti per alcuni tipi di coerenza interna e per accuratezza di installazione, rimuovere pacchetti installati. Il Capitolo 25 fornisce informazioni più estese sulla installazione di software e sulla configurazione del sistema.

12.15 Impostazione del nome della macchina

Potete cambiare **nodename**, o **uname** (UNIX name), della macchina con il menu **System Setup. uname** è il nome di identificazione della macchina; deve essere unico, almeno nel *dominio* delle macchine che comunicano con la vostra tramite **mail**. Una volta scelto, il nome non può essere cambiato facilmente perché altrimenti dovete cambiarlo anche in tutte le macchine remote in collegamento con voi. Quando installate per la prima volta il sistema, assegnategli un **nodename** e mantenetelo, evitando di assegnare il nome di default, che certo non è unico.

Se l'interfaccia utente dispone dell'opzione **Mail Setup**, la voce **Mail Name of This System** cambia nome alla macchina, altrimenti cercate l'opzione nel menu **System Setup**.

Il nome della macchina non deve avere lunghezza superiore a otto caratteri, deve iniziare con una qualsiasi lettera minuscola e può contenere al suo interno anche numeri, ma non caratteri speciali.

12.16 Predisposizioni per la posta elettronica

Come una macchina remota deve avere precise informazioni sulla vostra macchina per potervi spedire dei messaggi tramite la posta elettronica, così accade per il vostro sistema quando deve colloquiare con gli altri sistemi. All'installazione il sistema SVR4 è di solito configurato in maniera che il sistema rifiuta la posta in arrivo da macchine sconosciute. Se la vostra macchina è collegata a un modem e linea telefonica, dovete elencare tutte le macchine che possono inviarvi posta; in alternativa, potete esaminare il Capitolo 15 e configurare il vostro sistema in maniera da rilasciare la restrizione se avete dei corrispondenti su macchine sconosciute. In generale, dovete distinguere le macchine che usano linee telefoniche con modem, o connessioni dirette, da quelle connesse tramite LAN. Qui tratteremo le macchi-

ne che possono collegarsi alla vostra tramite linee telefoniche; per ulteriori informazioni sull'accesso tramite LAN consultate il gestore della vostra rete. In molti sistemi, le funzioni di gestione della rete sono suddivise fra una voce del menu **Mail Setup** e un distinto menu **Network Services**; in altre release, la configurazione della posta fa parte della voce del menu **Basic Networking**, sotto il menu **Network Services**.

Per trasmettere dalla vostra macchina a un'altra macchina, dovete indicare al sistema UNIX l'identificatore univoco di ciascuna macchina remota che intendete contattare, o che deve contattarvi. Dovete indicare l'identificatore di login e la password di accesso alla comunicazione, che dovete richiedere al gestore di sistema di ogni macchina con cui intendete stabilire un colloquio. Infine, dovete specificare alla vostra macchina la modalità di chiamata delle altre macchine, che comprende l'indicazione del dispositivo e la velocità di comunicazione da utilizzare; potete anche limitare la chiamata a determinate ore del giorno o giorni della settimana. Se questi dati non sono correttamente indicati non potrete inviare della posta a un altro sistema; altri sistemi potranno ugualmente chiamarvi, anche se voi non potete farlo.

Il login id usato dalle altre macchine per chiamarvi e inviare la posta è di solito **nuucp** (new uucp); non lo dovete cambiare se non esiste un motivo particolare. Potete anche specificare una password; questa non è indispensabile e potete non definirla in questo contesto, in quanto gli strumenti di comunicazione dati garantiscono una eccellente sicurezza interna. Se assegnate una password, tutte le macchine che comunicano con voi la debbono conoscere; questo implica che dovete distribuire la password a tutti i corrispondenti, sia vostri che loro. È preferibile non avere password per il login id **nuucp**, a meno che non abbiate una ragione molto particolare per limitare la posta a un ristretto numero di macchine. Alcune release non consentono di definire il login id e la password tramite l'interfaccia di gestione.

Il menu **Add System** di solito richiede l'introduzione del nome univoco dell'altro sistema, il suo login id (default **nuucp**), e della password del login id, se necessaria.

Se la connessione viene effettuata tramite un modem e una linea telefonica commutata, dovete conoscere il numero di telefono che collega il modem chiamato all'altra macchina.

A ciascun sistema è associato un device per chiamare quella macchina: di solito potete scegliere tra *modem* e *direct*; dopo questa scelta dovete selezionare il tipo di modem in uso con quel device.

Alcune interfacce di gestione supportano solo sistemi che possono essere chiamati via linea telefonica (device *modem*), e non quelli accessibili tramite LAN o con una connessione diretta, mentre altre interfacce rendono possibile la scelta tra diversi tipi di collegamento. Gli strumenti di controllo delle connessioni nei sottosistemi di comunicazione di dati sono molto potenti, ma possono apparire complessi. Esaminate attentamente il Capitolo

15 prima di collegare in rete sistemi che utilizzano un tipo di collegamento diverso dalla rete telefonica commutata.

Dovete poi selezionare la velocità di trasmissione dei dati; per rendere possibile la comunicazione il vostro modem deve essere adattato alla stessa velocità del modem della macchina ricevente. La maggior parte dei modem può essere predisposta a 300, 1200 o 2400 baud; consultate il gestore di sistema della macchina remota per accordarvi sulla velocità di comunicazione.

Quando selezionate un device dovete specificare la *porta* di connessione del device, di solito **/dev/tty00s** o **/dev/tty01s**; alcune interfacce non richiedono il prefisso **/dev/**.

Potete limitare le chiamate a giorni e/o ore particolari del giorno, in questo caso il software di comunicazione chiama l'altra macchina solo nei periodi permessi; di solito i messaggi vengono spediti solo quando è disponibile la macchina remota o durante le ore notturne, quando il carico di lavoro sulla macchina e la tariffa telefonica sono ridotti. Potete anche specificare **Never**, che obbliga l'altra macchina a chiamarvi per raccogliere la posta accumulata. Se volete una rapida trasmissione della vostra posta alle altre macchine, non dovete limitare i periodi di chiamata; queste limitazioni comunque non impediscono alle altre macchine di chiamarvi anche nelle ore di limitazione per la vostra macchina.

Nei menu **Basic Networking** e **Mail Setup** sono disponibili anche molte altre opzioni, come polling e altre caratteristiche; studiate il Capitolo 15 prima di applicare queste opzioni.

12.17 Avvio automatico di applicazioni

Il menu **Schedule Task** consente di scadenziare l'esecuzione automatica di un lavoro a una determinata ora di un determinato giorno della settimana o dell'anno. Di solito, potete creare una procedura di shell che dovete provare attentamente; quindi lo schema dell'interfaccia di gestione vi permette di scadenziare l'ora e la data di esecuzione dell'applicazione. Indicate il pathname completo della vostra procedura. Se la macchina è spenta all'ora e nel giorno scadenziati, l'esecuzione scadenziata viene ignorata.

Lo scheduling di attività può essere origine di errori, come nel caso che la procedura di shell venga spostata o cancellata senza cancellare il lavoro dallo scadenzario; abbiate comunque cura di provare accuratamente la procedura prima di immetterla nello scadenzario. Lo scheduling può essere inibito ad alcuni utenti; consultate il Capitolo 20 per maggiori informazioni.

12.18 Gestione delle unità di stampa

Il sistema UNIX dispone di un potente *print spooler* che gestisce le unità di stampa installate sul sistema. Potete inviare l'uscita di un lavoro di stampa a una o più stampanti installate specificando il loro *nome*. Per installare una stampante, dovete indicare un nome, assegnarla a una porta seriale o parallela e specificare il tipo. Molti sistemi SVR4 permettono il riferimento a una stampante di un altro sistema collegato alla vostra macchina, tramite posta o LAN.

Una volta che la stampante è installata, potete gestire la coda di lavori che sono in attesa di essere stampati, annullare quelli che non intendete più stampare, cambiare la stampante associata a un lavoro oppure togliere una stampante dal servizio.

INSTALLARE UNA UNITÀ DI STAMPA

Il menu **Printer Services** permette di installare una nuova stampante su una porta *parallela* (cioè LPT1 o LPT2), o *seriale* (cioè COM1 o COM2), ma non su entrambe. I file device associati con queste porte fisiche sono specificati in Tabella 13.1. Il Capitolo 13 tratta dettagliatamente l'operazione di stampa e deve essere consultato prima di installare una stampante.

Prima di tutto, dovete specificare il tipo e il nome della stampante che intendete installare; il nome viene usato per specificare la stampante da usare, nelle richieste di lavori di stampa. Di solito il menu include un ampio elenco delle stampanti disponibili, in modo da permettervi di selezionare quella che vi interessa; se la vostra stampante non è in lista, selezionate la voce **Dumb**. Se la vostra stampante non funziona con nessuna voce, dovete predisporre la stampante manualmente.

Il tipo e il nome della stampante assieme, determinano varie predisposizioni standard, o default, di altri attributi della stampante; potete accettare i default, o cambiarli mediante sottomenu addizionali. Di solito, potrete accedere ai sottomenu solo se rifiutate i default nel menu principale **Printer**. Cambiate questi attributi con molta cura; spesso una stampante cessa di funzionare se gli attributi nel sottosistema **lp** sono in conflitto con le caratteristiche della stampante.

Se utilizzate stampanti seriali, dovete specificare la velocità di comunicazione; la scelta di default è generalmente appropriata, ma dovete posizionare gli switch sulla stampante alla stessa velocità selezionata nel menu. Le stampanti parallele non richiedono di selezionare la velocità di comunicazione.

Infine, rimane da assegnare a una (e una sola) stampante il ruolo di stampante di default a cui vengono automaticamente assegnati tutti i lavori per i quali non è stata specificata l'unità di stampa.

Conviene ignorare i menu **Filters** e **Forms**, a meno che non abbiate una ragione speciale per usarli, e abbiate ben compreso il loro scopo.

Il menu **Printer** contiene spesso una voce per destinare il lavoro di stampa a una stampante remota connessa a un'altra macchina in una rete LAN. Consultate il gestore della vostra rete prima di tentare una connessione di stampante remota, perché il modo di utilizzo della stampa remota è determinato dalla politica della rete. Potete anche selezionare una stampante collegata tramite la rete telefonica pubblica commutata; nel campo **Basic Networking Address** dovete indicare un nome remoto conosciuto al vostro sottosistema **cu**. Si tratta di solito di una stampante standalone con un modem a risposta automatica, non del sottosistema **lp** di un sistema UNIX.

GESTIONE DELLE UNITÀ DI STAMPA

Una volta che la stampante è installata e funziona, potete inviare l'uscita su di essa con il comando **lp**. È possibile accodare diversi lavori di stampa; il sottosistema **lp** gestisce la coda di spool.

Il menu **Printer Operations** prevede opzioni per analizzare lo spool corrente dei lavori da stampare; se intendete eliminare un lavoro non più desiderato, utilizzate l'opzione **Printer Queue** o **Printer Requests**.

Il menu **Printer Status** elenca le stampanti configurate nel sottosistema **lp**, specificando se sono in grado di accettare richieste di stampa. In altre parole, potete accodare nuove stampe, anche se la stampante non si trova in condizione di eseguirle. Per esempio, se la stampante manca di carta oppure viene spenta e riaccesa, non è più in grado di eseguire l'operazione di stampa, ma viene considerata in grado di accogliere richieste; il contenuto dello spool si incrementa di nuovi lavori, ma la stampante non è in funzione. Quando si verifica una situazione di questo tipo, dovete far ripartire o abilitare la stampante con l'opzione **Printer Restart** o **Enable**, disponibili nel menu **Printer Operations**. Dopo questa operazione, la stampante immediatamente inizia a stampare il primo lavoro che si trova in coda. Se invece la stampante viene indicata non in grado di accogliere richieste di stampa, dovete rimuoverla e poi reinstallarla nuovamente col menu **Peripherals Setup** per rimetterla in grado di accettare richieste. Molti sistemi SVR4 includono nel menu una voce **Accept** che permette di fare accettare lavori a una stampante, anche se non abilitata. Tuttavia, per funzionare normalmente, una stampante deve essere sia abilitata che in grado di accettare richieste.

12.19 Servizi della rete

Potete configurare, oltre alla stampante, anche altri servizi della rete, come l'accesso a file remoti e i parametri di indirizzamento di rete, tramite altre funzioni dell'interfaccia utente di gestione. Anche in questi casi con-

viene consultare il gestore della vostra rete prima di avventurarvi in questo campo. Predisposizioni scorrette della rete fatte per tentativi possono avere effetti deleteri sulle prestazioni dell'intera rete. Solo dopo avere ben compreso la configurazione della vostra rete locale, potete usare le voci del menu di servizi della rete per configurare la vostra macchina nella rete.

Se occorre stabilire dei servizi come parte della configurazione della rete, troverete le funzioni necessarie sotto **Service Access Facility**, trattata in seguito.

12.20 Gestione delle porte

La funzione di utility più complessa nell'interfaccia gestionale è la predisposizione delle porte o unità periferiche. Questa funzione può essere distribuita in più voci di menu, o raggruppata sotto una sola voce, a seconda della versione di sistema. Le principali azioni interessate sono: *terminal setup*, *port monitors* e *port services*; alcuni sistemi includono opzioni per la predisposizione di altri dispositivi periferici come per esempio un secondo disco rigido o unità nastro. Questa voce di menu cambia automaticamente se aggiungete nuovi dispositivi hardware alla vostra macchina, per consentirvi di gestire questi nuovi dispositivi.

La configurazione delle porte è completamente nuova in SVR4, e molte precedenti informazioni sui processi **getty** non sono più applicabili. La nuova organizzazione viene chiamata *Service Access Facility*, in quanto rispecchia un concetto generale di servizio per tutti i tipi di porte che accettano chiamate in entrata. Questo concetto riguarda sia porte seriali connesse a un modem o terminali locali, che porte a elevate prestazioni connesse a una LAN; le porte solo in uscita, come le porte di stampante, non sono associate con un servizio.

Dovete stabilire nel sistema una dichiarazione per ciascun tipo di porta che volete usare per chiamate in entrata; le porte seriali e le porte LAN sono le più comuni, ma sono previsti molti altri tipi di connessioni in entrata. Ciascun tipo di porta ha un proprio *listener* o *port monitor* per trattare le chiamate in entrata; questi processi di sorveglianza "ascoltano" il traffico su più porte simultaneamente. Per esempio, diverse porte seriali possono essere sorvegliate da un singolo processo monitor, chiamato **ttymon** (tty monitor) in SVR4. Per ogni processo monitor dovete dichiarare le porte che deve sorvegliare, e quali azioni intraprendere quando arriva una chiamata.

Quando a una porta compare una chiamata, il monitor "risponde" alla chiamata ed esegue il servizio appropriato per la chiamata; il monitor esce di scena fino alla fine della chiamata, quando il servizio restituisce il controllo al monitor che si mette in ascolto di una prossima chiamata.

Dovete definire tutte queste procedure con la voce **Port Management**.

PREDISPOSIZIONE VELOCE DEI TERMINALI

Il menu **Quick Terminal Setup** fornisce uno strumento semplice per aggiungere servizi su porte seriali. Questo strumento tratta i monitor e i servizi in tutta la loro complessità, ma è molto limitato, e gestisce solo la semplice predisposizione delle porte seriali; potete selezionare la porta da aggiungere tra le porte della configurazione del vostro sistema, e scegliere la velocità. Potete scegliere la velocità più alta possibile; per esempio, se alla porta è collegato un modem a 1200 o 2400 bps, selezionate 2400; le chiamate in arrivo a velocità più bassa verranno comunque accettate se il chiamante invia un segnale *break* prima di scambiare i messaggi di collegamento (logging in). I terminali in connessione diretta vengono normalmente collegati a 9600 bps. Il menu **Quick Terminal Setup** non consente di configurare le porte per computer in connessione diretta.

GESTIONE DEI PORT MONITOR

La procedura completa di predisposizione consente molta maggiore flessibilità, ma è anche molto più complessa. Il primo passo consiste nello scegliere i port monitor; ciascun port monitor viene identificato da un nome, o etichetta. Potete scegliere questo nome, che deve descrivere il tipo della porta; la cosa migliore è di usare lo stesso nome del tipo della porta. Il tipo di port monitor è di solito **ttymon** per una porta seriale, **inetd** per un collegamento Ethernet, **listen** per un collegamento StarLan. Il comando associato al port monitor è indicato col pathname per eseguire il monitor: **/usr/lib/saf/ttymon** per porte seriali, **/usr/sbin/inetd** per Ethernet, **/usr/lib/saf/listen** per StarLan. Consultate la vostra particolare documentazione per altri tipi di porte supportate sotto SAF.

Potete anche scegliere se il monitor debba essere *started* e se debba essere *enabled*. Se occorre fermare un port monitor potete farlo in qualunque momento con un'altra voce del menu così che sia più facile lanciare il monitor immediatamente. Aggiungete un commento che descrive il monitor e infine salvate lo schema; a questo punto il nuovo monitor dovrebbe essere in servizio.

Come per i servizi di stampante, se il monitor non risulta essere in funzione, dovete verificare che sia *started* e anche *enabled*; usate la voce **List Port Monitors** per verificare queste condizioni.

Selezionando la voce **Terminal Setup** oppure **LAN Setup** anziché **Port Setup**, il pacchetto **sysadm** lancia per voi il corretto port monitor; voi potete richiedere l'elenco dei monitor esistenti per esaminare i dati esatti di predisposizione.

GESTIONE DI PORT SERVICE

Avendo messo in servizio un port monitor, dovete selezionare le porte specifiche che quel monitor deve sorvegliare, e il nome del comando o servizio da eseguire quando arriva una chiamata in entrata. Spesso dovrete specificare diversi parametri di comunicazione e flag come parte della procedura di predisposizione, per determinare lo specifico comportamento della porta quando risponde alle chiamate. Questi compiti sono svolti dal menu **Port Services**; in questo menu iniziate con introdurre l'etichetta usata nel passo precedente come nome del port monitor; questo nome seleziona il monitor al quale volete aggiungere nuovi servizi. Quindi, definite una nuova etichetta per il servizio, e assegnate un identificatore di login id (*service invocation identity*) che esegue il servizio, di solito **root**, in quanto molti servizi, come il login, scambiano il corretto identificatore, se occorre. Infine, dovete abilitare il servizio, e aggiungere un commento descrittivo.

A questo punto potete entrare in un secondo schema, che include i dati effettivi per il servizio. Questo schema differisce probabilmente a seconda del port monitor. Dovete sempre selezionare l'identificatore della porta con il percorso completo del device nel directory **/dev**; di solito **/dev/tty00s** o **/dev/tty01s** per le porte seriali 1 e 2, rispettivamente. Altri tipi di porte e connessioni della rete avranno altri percorsi.

Per le porte seriali, dovete selezionare l'etichetta corrispondente alle informazioni di configurazione del terminale; di solito potete selezionare l'etichetta che coincide con la massima velocità del dispositivo connesso a quella porta. Per esempio, usate 9600 o 19200 per un terminale o un computer in connessione diretta, e 2400 o 1200 per un modem. Queste etichette sono nomi nel file **/etc/ttydefs**, trattato più avanti in questo capitolo. Sono di solito disponibili numerose etichette, molte delle quali per necessità speciali. Quando la velocità di una chiamata in arrivo non corrisponde a quella dell'etichetta, il sistema si dispone per una velocità più bassa; in questo modo una predisposizione a 2400 accetta chiamate sia a 2400 che a 1200 bps.

Poi, dovete selezionare il comando (*service command*) che tratta la chiamata; di solito **/usr/bin/login**, dato che il servizio di login fornisce la protezione di password usando il file **/etc/passwd**. Selezionate *bidirectional* se volete condividere la porta fra le chiamate di login in arrivo e il traffico in uscita di **uucp** o **mail**. Nei restanti campi dello schema accettate i valori di default.

Dopo avere salvato lo schermo, la porta dovrebbe essere disponibile per le chiamate in arrivo, in caso contrario accertatevi che l'etichetta di velocità sia corretta, di avere lanciato il monitor e di avere abilitato il servizio.

12.21 Approfondimenti

L'attività di gestione in ambiente UNIX è complessa e i restanti capitoli lo confermeranno. Una volta che avrete acquisito familiarità con il sistema

UNIX, vi renderete conto che l'interfaccia utente di gestione si dimostra molto utile nella maggior parte delle situazioni, anche se le procedure manuali risultano più veloci. Impiegate gli strumenti automatici dove è possibile, in quanto sono meno inclini agli errori rispetto ai metodi manuali. Nella parte restante di questo capitolo parleremo di terminali e menzioneremo qualche altro problema di carattere generale.

IL COMANDO uname

L'interfaccia e i programmi di configurazione del sistema forniscono gli strumenti per inizializzare il nome univoco di identificazione della macchina, conosciuto come *uname*. Inoltre, è disponibile un semplice comando UNIX chiamato **uname** (UNIX name) con cui potete conoscere o inizializzare il nome della macchina da shell, per esempio:

```
$ uname
my-sys
$
```

Per default, **uname** visualizza solo il nome di sistema. Utilizzate l'opzione **-a** (all) per visualizzare altre informazioni sulla macchina:

```
$ uname -a
my-sys my-sys 4.0 1 i386 386/AT
$
```

I sei dati visualizzati indicano in sequenza il nome di sistema; il nome di nodo (*nodename*), che può differire dal nome di sistema se la macchina svolge la funzione di server in qualche rete; il numero di release del sistema operativo (4.0); il numero di versione della release; il tipo di hardware di CPU che la macchina utilizza (i386); infine il tipo di macchina. Il comando **uname** dispone di altre opzioni che permettono di visualizzare ognuno di questi campi singolarmente; viene spesso utilizzato nelle procedure di shell il cui comportamento dipende da alcuni di questi dati.

Il superuser può anche utilizzare il comando **uname** per inizializzare il nome del sistema ma, quando è possibile, cercate di adoperare l'interfaccia utente. Utilizzate l'opzione **-S** (set) per inizializzare il nome della macchina, seguita dal nuovo nome che intendete assegnare:

```
# uname -a
my-sys my-sys 4.0 1 i386 386/AT
# uname -S giorgio
# uname -a
giorgio giorgio 4.0 1 i386 386/AT
#
```

Ricordatevi che altri utenti possono utilizzare il nome della vostra macchina per spedirvi la posta, per cui informateli se decidete di cambiarlo.

CARATTERIZZAZIONE DEI TERMINALI

Come tutte le funzioni di utility di cui dispone l'interfaccia gestionale, i menu **Ports** hanno corrispondenza in analoghe procedure manuali. Quando una porta di comunicazione è abilitata ad accettare chiamate in ingresso, un processo di sistema, denominato **/usr/lib/saf/ttymon** (*tty monitor*), è in ascolto in attesa del *carrier detect signal* su una serie di porte RS232. Quando il segnale viene asserito su una porta, **ttymon** predispone quella linea a certi valori iniziali **stty** (*initial line settings*) e passa il controllo al programma **login**, che visualizza il messaggio **login:** all'utente. Quando l'utente termina la sessione e si scollega, **ttymon** riprende il controllo e si rimette in ascolto per un altro login.

Le caratteristiche di linea, che **ttymon** legge per determinare come predisporre la porta RS232, sono memorizzate nel file **/etc/ttydefs** (tty definitions). In SVR4 questo file **ttydefs** sostituisce l'equivalente vecchio **gettydefs**, tuttavia in alcune release il file **gettydefs** può essere ancora presente. Il file **/etc/ttydefs** si compone di una serie di caratteristiche di linea, una per ogni linea del file **ttydefs**. Il seguente esempio mostra due linee del file **ttydefs**:

```
9600: 9600 opost onlcr tab3 ignpar ixon ixany parenb istrip echo echoe
echok isig cs7 cread : 9600 opost onlcr sane tab3 ignpar
ixon ixany parenb istrip echo echoe echok isig cs7 cread ::4800
4800: 4800 opost onlcr tab3 ignpar ixon ixany parenb istrip echo echoe
echok isig cs7 cread : 4800 opost onlcr sane tab3 ignpar
ixon ixany parenb istrip echo echoe echok isig cs7 cread ::2400
```

Si tratta di fatto di sole due linee molto lunghe di **ttydefs**, anche se ogni linea del file può memorizzare più di 80 caratteri e occupare più righe del video.

Ogni linea contiene cinque campi separati, delimitati dall'operatore : (duepunti). Il primo campo contiene il nome delle caratteristiche di linea; le due linee dell'esempio precedente si chiamano rispettivamente 9600 e 4800. Il secondo campo contiene tutti i valori iniziali **stty**, mentre il terzo campo contiene i valori finali **stty**. Ogni parametro **stty** ha un nome ed è separato dal successivo da un carattere spazio all'interno del campo. I valori iniziali vengono attribuiti alla linea nella risposta della chiamata e finché **ttymon** esegue il servizio di login; i valori finali vengono attribuiti appena prima dell'inizio del servizio di login. La distinzione fra i due casi è sottile; di solito ambedue i campi contengono gli stessi valori. Il quarto campo è vuoto. L'ultimo campo specifica un riferimento a un'altra caratteristica di linea in **ttydefs**. Se il processo **ttymon** rileva un segnale di break sulla linea, pri-

ma che l'utente risponda al messaggio **login:**, il processo attribuisce alla linea le caratteristiche corrispondenti all'ultimo campo, e rilancia il servizio. Questo ciclo può essere ripetuto ogni volta che viene ricevuto un break. Questa procedura permette a una singola porta in ingresso **tty** di ricevere chiamate a diverse velocità o differenti valori **stty** senza un costante intervento da parte del gestore. Se volete evitare che il processo ricicli su una nuova caratteristica di linea, potete ricopiare il nome dal primo al quinto campo; in questo caso **getty** riutilizza la stessa caratteristica di linea al ricevimento del break.

AVVIO DI UN PROCESSO PORT MONITOR

I supervisori di porte (port monitor) come **ttymon**, che controlla il login sulle porte seriali, sono gestiti dal processo **/usr/lib/saf/sac** (service access controller). Il processo demon **sac** parte all'inizializzazione e deve essere sempre in funzione; ricerca il file **/etc/saf/_sactab** (service access controller table) per ottenere l'elenco dei monitor da gestire, mostrato come esempio:

```
# cat /etc/saf/_sactab
# VERSION=1
inetd:inetd::0:/usr/sbin/inetd #internet daemon
ttymon:ttymon::0:/usr/lib/saf/ttymon #
#
```

I contenuti di ciascuna linea sono i dati chiave originati dal menu *Port Monitor* dell'interfaccia di gestione. Potete interpretare questi dati con il comando **sacadm** (sac administration), per esempio:

```
# sacadm -l
PMTAG  PMTYPE FLGS RCNT STATUS   COMMAND
inetd  inetd    -    0  ENABLED  /usr/sbin/inetd #internet daemon
ttymon ttymon   -    0  ENABLED  /usr/lib/saf/ttymon #
#
```

L'opzione **-l** (list) elenca i monitor correntemente attivi.

Il comando **sacadm**, con le sue numerose opzioni, è il programma che viene eseguito dall'interfaccia di gestione per configurare i supervisori delle porte.

Potete usare **sacadm** per attivare e fermare i port monitor, riconfigurare **_sactab**, e altre operazioni.

AVVIO DI UN SERVIZIO IN UN PORT MONITOR

Dopo che un port monitor è stato configurato, legge il file **/etc/saf/pmtag/_pmtab** (port monitor table), in cui *pmtag* è l'etichetta o nome attribuito

al port monitor. Come **_sactab**, anche **_pmtab** contiene un elenco dei servizi sotto il controllo del suo port monitor; ecco un esempio:

```
# cat /etc/saf/ttymon/_pmtab
# VERSION=1
00s:u:root:reserved:reserved:reserved: /dev/tty00s:bhr::/usr/bin/login
::9600::tty00 login: :: # this is a comment
01s:u:root:reserved:reserved:reserved: /dev/tty01s:bhr::/usr/bin/login
::9600::tty01 login: :: # COM2 setup
#
```

Queste sono solo due lunghe linee in **_pmtab**: la linea 00s è COM1 o tty porta 0, la linea 01s configura il servizio login su COM2 o tty porta1.

Usate il comando **pmadm** (port monitor administration) per comprendere il significato dei diversi campi:

```
# pmadm -t ttymon -l
PMTAG  PMTYPE SVCTAG  FLGS ID    <PMSPECIFIC>
ttymon ttymon 00s     u    root  /dev/tty00s bhr - /usr/bin/login
- 9600 - tty00 login:   - #  this is a comment
ttymon ttymon 01s     u    root  /dev/tty01s bhr - /usr/bin/login
- 9600 - tty01 login:   - #  COM2 setup
#
```

Il comando **pmadm** ha molte opzioni; usate **-t** (type) per specificare il tipo di monitor e **-l** per elencare i monitor correntemente attivi.

L'output è ancora di sole due lunghe linee; i primi cinque campi sono generici del monitor **ttymon**, gli altri cinque sono specifici delle porte seriali. Questi ultimi indicano il pathname di device (/dev/tty00s), gli indicatori o flag (bhr), il pathname del servizio (/usr/bin/login), l'etichetta in **ttydefs** (9600), l'etichetta di device (tty00), il messaggio di login (login:) e il commento. I vari caratteri - (meno) indicano campi non utilizzati in questo esempio.

Gli indicatori (flag) definiscono il comportamento della porta in certe circostanze; **b** definisce una porta *bidirezionale*, che consente sia chiamate in entrata che traffico in uscita (mail); **h** sopprime un *hangup* o chiusura automatica della linea subito dopo il ricevimento della chiamata; **r** forza **ttymon** ad attendere di ricevere un carattere dalla porta prima di trasmettere il messaggio **login:**. L'indicatore **r** è necessario per connettere due computer che hanno ambedue il servizio di login; se un computer vuole connettersi con l'altro, trasmette un carattere o un newline, e questo induce l'altro computer a trasmettere **login:** sulla porta. In assenza di questo indicatore, ciascuna macchina tenterebbe costantemente di inviare il **login:** all'altra e ognuna delle due riterrebbe che l'altra stia richiedendo il login.

Se viene dato un messaggio *inattivo* nel campo precedente il commento, e il servizio per quella porta è stato fermato, le chiamate in entrata riceve-

ranno il messaggio *inattivo* invece del messaggio **login:**; questo può essere a volte utile per segnalare che la connessione non è disponibile.

Le linee **_pmtab** sono generate da comandi specifici di ciascun monitor; esaminate la man page **ttyadm(1)** per maggiori informazioni circa i campi ammissibili per i servizi **ttymon**.

INSTALLAZIONE DI NUOVE DESCRIZIONI DI TERMINALI

Quando un programma come **vi** utilizza la vostra variabile di ambiente **TERM**, il suo valore definisce una *descrizione di terminale*. Queste descrizioni di terminali sono contenute nel directory **/usr/share/lib/terminfo** e nei suoi subdirectory. Le caratteristiche dei terminali sono mantenute in una forma compilata in questo albero di directory e non sono direttamente leggibili dalle persone. Nelle versioni meno recenti di sistema UNIX (prima di SVR2), le descrizioni di terminali erano mantenute in forma di testo nel file **/etc/termcap**. Inoltre, il formato delle descrizioni è cambiato tra le versioni **termcap** e **terminfo**. Per supportare questi cambiamenti, i sistemi forniscono tre nuove funzioni di utility.

Il programma **tic** (terminfo compiler) accetta un nome di file come argomento:

```
$ tic termdesc.ti
```

Il file contiene una descrizione di terminale nel formato sorgente. Il programma compila la descrizione e memorizza il risultato nel database **terminfo**.

Il programma **infocmp** (terminfo compare) può confrontare descrizioni **terminfo** compilate o visualizzare il formato sorgente dal formato compilato; usato senza argomenti visualizza la forma sorgente delle caratteristiche del terminale definito nella variabile **TERM**. Questo potente strumento dispone di numerose opzioni e può generare un formato sorgente che potete modificare in edit e poi compilare tramite **tic** per ottenere una versione modificata.

Infine, il programma **captoinfo** permette di convertire nel formato **terminfo** le descrizioni dei terminali scritte nel vecchio formato **termcap**.

Non potete utilizzare correttamente questi strumenti sofisticati senza avere qualche esperienza di terminali e del pacchetto software **curses**, per cui esaminate attentamente le *man page* di questi comandi prima di utilizzarli.

USO DI MONITOR A COLORI

Molti terminali e console di sistema delle macchine odierne supportano video a colori, tuttavia per le macchine della classe AT la variabile **TERM** as-

sume spesso il valore di default 386AT-M per monitor monocromo. Per avere console a colori assicuratevi di attribuire **TERM=386AT**.

Il sistema UNIX non definisce nessun comando standard per gestire il colore delle immagini sullo schermo (eccetto che per X Window System), ma molte versioni forniscono speciali comandi per cambiare il colore di fondo e i colori dei caratteri ASCII visualizzati. Inoltre, gli utenti esperti impiegano spesso l'utility **terminfo** (terminal information) per evidenziare con colori particolari alcuni output, come per esempio il PS1, le linee di comando, o i messaggi provenienti dalle procedure di shell. I risultati possono essere differenti tra console e terminali remoti, ma generalmente potete definire file di descrizione **terminfo** con cui ottenere una gamma di colori per i due casi.

Molte versioni SVR4 dispongono del comando **setcolor** per predefinire sulla console i colori di sfondo e di testo. Questo comando accetta generalmente due argomenti che specificano rispettivamente i colori di sfondo (background) e di testo (foreground). Gli argomenti possono essere espressi con nomi di colori o numeri; consultate in dettaglio la *man page* del comando della vostra versione di UNIX. Per esempio, il comando

```
$ setcolor blue white
$
```

definisce caratteri bianchi su fondo blu. In qualche versione di UNIX, **setcolor** può accettare altre opzioni, che specificano una sfumatura di chiaro o di scuro, o può utilizzare numeri codificati invece di nomi per specificare i colori. Potete prima fare degli esperimenti e poi rendere permanente la scelta di colori preferita inserendo il comando nel file **.profile**.

In aggiunta al database **terminfo**, potete usare *sequenze di escape* per cambiare il colore di messaggi particolari. In altre parole, il terminale o la console interpretano speciali sequenze di caratteri che iniziano con un carattere ESC e contengono la codifica dei diversi colori; potete usare questa caratteristica per visualizzare a colori i vostri messaggi. Potete anche visualizzare a colori la linea di comando, inserendo sequenze di escape in PS1 o in procedure di shell. Se avete un monitor a colori provate questo comando:

```
$ echo "\033[34mSalve gente\033[0m"
```

Potete anche modificare il database **terminfo** del vostro terminale in modo che le applicazioni a tutto schermo come **vi** visualizzino messaggi a colori. I codici di colore possono variare tra differenti implementazioni di sistema UNIX, e tra differenti monitor, per cui occorre sempre procedere per tentativi.

Capitolo 13

Il sottosistema di stampa

UNIX dispone di ottimi strumenti di controllo delle stampanti e di gestione delle operazioni di *spool* dei risultati delle elaborazioni alle unità di stampa. Il software può essere configurato per usare una sola stampante oppure per assegnare alla macchina UNIX la funzione di server di stampa che controlla decine di differenti stampanti a disposizione di tutti gli utenti del sistema. È possibile che più utenti richiedano contemporaneamente l'uso della stampante; in questo caso il software di sottosistema accoda i dati in uscita e li invia sequenzialmente in stampa, separandoli con pagine intestate a caratteri giganti (*banner*), in modo che i singoli utenti riconoscano facilmente i propri lavori.

Il supporto di stampa è particolarmente efficiente in ambiente UNIX grazie alla natura multiprocesso del sistema; a differenza di altri sistemi operativi, il software di stampa è eseguito come una applicazione a livello utente, e non necessita di speciali driver di periferica o di buffer in memoria visibili all'utente. Gli strumenti di stampa UNIX risolvono in maniera eccellente la dipendenza dall'hardware.

Gli strumenti di stampa generalmente consistono nel sottosistema *lp* (*line printer subsystem*) e sono così completi e potenti che di solito costituiscono l'unico software di stampa nel sistema UNIX. Poiché la filosofia UNIX pri-

vilegia il concetto di affidare ai programmi la risoluzione di singoli problemi, funzioni come l'impaginazione dei dati in uscita e le operazioni di formattazione dipendenti dall'hardware sono svolte da applicazioni dedicate e da filtri, piuttosto che dal sottosistema **lp**. Questa scelta consente a **lp** di essere dedicato alla gestione della coda di stampa e al controllo efficiente dell'hardware.

Un sottosistema addizionale di stampa, molto simile, originato nella release BSD, viene incluso nel BSD Compatibiliy Package di SVR4. Questo sottosistema **lpr** agisce in molti aspetti in modo molto simile ad **lp**, tuttavia quest'ultimo sottosistema standard è più funzionale per grandi centri di stampa. Se la vostra macchina è configurata con **lpr**, potete usare il comando **lpr** per accodare lavori di stampa e il comando **lpq** (lp queue) per ottenere un rapporto sullo stato della coda di spool. Per maggiori informazioni sul sottosistema **lpr** dovete contattare il gestore del vostro sistema.

13.1 Il comando lp

Nel sottosistema standard di stampa di SVR4 le stampe vengono ottenute usando il comando **lp**. Il programma **lp** colloca il file argomento della linea di comando, o il suo standard input, nella coda di lavori di una stampante; **lp** restituisce il controllo allo shell dopo che il lavoro è stato accodato e non quando la stampa è stata completata.

Il comando **lp** è solitamente usato al termine di un pipeline di shell, per esempio:

```
$ cat /usr/spool/lp/model/dumb | lp
```

ma può anche accettare un pathname come argomento:

```
$ lp /usr/spool/lp/model/dumb
```

Il comando **lp** gestisce esclusivamente la stampa; pertanto se vi interessa impaginare i risultati in uscita con speciali intestazioni nelle pagine, dovete utilizzare strumenti addizionali nella linea di comando:

```
$ pr /usr/spool/lp/model/serial | lp
```

Sono disponibili diverse opzioni che adattano il processo di stampa alle esigenze dell'utente. L'opzione **-m** (mail) provoca l'invio nella mail-box dell'utente di un messaggio che segnala l'avvenuta stampa del file; risulta utile, poiché la coda di stampa può avere la dimensione dell'intero spazio libero su disco e l'operazione di stampa può impiegare molto tempo. L'opzione **-n** (number) permette di stampare un numero predefinito di copie con un unico comando **lp**. Per esempio, se intendete stampare sei copie di un file e ave-

re dalla posta elettronica la comunicazione che l'operazione è stata ultimata, lanciate il seguente comando:

```
$ lp -m -n6 /usr/spool/lp/model/1640
```

Il comando **lp** inserisce una pagina intestata a caratteri giganti (banner) all'inizio di ogni lavoro; per gestire il contenuto di questa intestazione, utilizzate l'opzione **-t** (title):

```
$ lp -t"Questo file appartiene a $LOGNAME" /usr/spool/lp/model/prx
```

Questo titolo appare solo sulla pagina di intestazione di **lp** e non su ciascuna delle successive pagine in uscita. Potete eliminare totalmente la pagina di banner con l'opzione **-o nobanner**; potete invece ottenere un'intestazione su tutte le pagine utilizzando l'opzione **-h** del comando **pr**.

Quando il comando **lp** accoda un lavoro di stampa, restituisce allo standard output un identificatore di richiesta (*request id*):

```
$ lp /usr/spool/lp/model/prx
Request id is ATT470-78 (1 file)
$
```

Questo identificatore rappresenta il numero che il sistema assegna al lavoro e risulta molto utile per rintracciare o annullare il lavoro. Alcune versioni di **lp** non restituiscono l'identificatore di richiesta e, per rintracciare il vostro lavoro, dovete lanciare il comando **lpstat**, che tratteremo tra breve. Se avete diverse stampanti collegate alla macchina, potete chiedere a **lp** di utilizzare una stampante piuttosto che un'altra, utilizzando l'opzione **-d** (destination), seguita dal nome della stampante:

```
$ lp -d ATT470 file1 file2 file3
```

Il comando **lp** dispone di altre opzioni che permettono di copiare i file prima di stamparli (**-c**), visualizzare un messaggio di fine stampa direttamente sul vostro terminale (**-w**) e trasmettere opzioni specifiche di stampante direttamente al programma che controlla la stampante (**-o**).

ANNULLAMENTO DI UN LAVORO DI STAMPA

Potete annullare una richiesta di lavoro di stampa, mediante il comando **cancel**. Il comando accetta come argomenti un identificatore di richiesta di stampa, o un elenco di identificatori di richiesta, e rimuove i lavori indicati dalla coda di stampa:

```
$ cancel ATT470-78
request "ATT470-78" cancelled
$
```

La stampa del lavoro viene annullata anche se si trova in corso di stampa, per cui potete interrompere la stampa di lavori di grandi dimensioni, richiesti per errore o non più necessari, evitando di tenere impegnata fino alla fine la stampante.

Il comando **cancel** può anche accettare come argomento il nome di una stampante; in questo caso annulla solo il lavoro che si trova in corso di stampa.

```
$ cancel ATT470
request "ATT470-78" cancelled
$
```

Questo comando comunque non ha nessun effetto sui lavori accodati che non sono ancora andati in stampa.

13.2 Stampa su moduli prestampati

In grandi centri di elaborazione possono esistere diverse stampanti predisposte con speciali moduli prestampati, come carta da lettere intestata o fatture. Se la vostra installazione utilizza diversi moduli, potete specificare che la vostra richiesta venga inviata alla stampante che contiene quel particolare modulo, oppure potete specificare il nome del modulo nella richiesta di stampa; il sottosistema **lp** invierà il lavoro alla stampante adatta. La seconda procedura è senz'altro preferibile, perché il gestore del sistema può spostare i moduli tra le varie stampanti a seconda delle necessità, mentre per gli utenti è più facile rintracciare un modulo col nome anziché con la stampante.

Per potere avvantaggiarsi di questa procedura, prima di tutto il gestore del sistema deve stabilire dei nomi di moduli conosciuti dal sistema (il procedimento viene descritto più avanti), quindi deve rendere pubblico l'elenco di questi nomi di moduli. Per stampare su di un determinato modulo, dovrete semplicemente usare l'opzione **-f** (form) seguita dal nome del modulo, per esempio:

```
$ cat lettera | lp -f cartadalettere
$
```

Se nella stessa linea di comando assieme all'opzione **-f** usate l'opzione **-d** per una determinata stampante, e su quella stampante il modulo non è disponibile, la vostra richiesta viene rifiutata. Questa è un'intelligente proce-

dura, perché consente di verificare che il modulo corretto sia montato sulla stampante desiderata. In alcuni casi, il sottosistema invia il lavoro nello spool, quindi invia un messaggio al gestore chiedendo il caricamento del modulo su una delle stampanti disponibili; quando il modulo necessario viene montato, il lavoro viene stampato.

13.3 Tipi di contenuto e filtri di stampa

Il sistema **lp** è in grado di produrre molti diversi tipi di file in stampa. Per esempio, se una stampante accetta l'output in linguaggio PostScript, un file PostScript preformattato può essere stampato direttamente; al contrario, un file di testo deve essere convertito al formato PostScript prima di essere stampato. In usuali lavori di stampa sono necessarie anche altre conversioni di formato; il sottosistema di filtri nel pacchetto **lp** fornisce le funzioni necessarie a queste conversioni. Nella linea di comando **lp** potete specificare un formato di conversione desiderato (o filtro), per differenti *tipi di contenuto* (*content-type*) dei file. Il file verrà trasmesso attraverso il filtro e il contenuto del file convertito nel formato appropriato prima di essere stampato.

Molti sistemi supportano vari filtri standard, altri filtri addizionali possono essere installati e configurati dal gestore del sistema; chiedete la lista dei tipi di contenuto (e filtri) supportati dal vostro sistema, oppure usate il comando **lpfilter** (trattato più oltre nel capitolo) per ottenere l'elenco dei filtri disponibili.

Nella linea di comando **lp** potete indicare il nome del tipo di contenuto del file, dopo l'opzione **-T** (type), per esempio:

```
$ lp -T postscript miofile.ps
$
```

Se una stampante è configurata per accettare direttamente quel tipo di contenuto, il lavoro viene messo in spool per quella stampante; altrimenti, **lp** invoca un filtro per trattare il file e produrre un output nel formato accettato dalla stampante. Dopo che il file è stato convertito dal filtro, viene inviato a una stampante che possa accettare il formato in output del filtro.

Molti file di testo sono considerati di tipo di contenuto **simple**, e possono essere stampati direttamente dalle usuali stampanti di testo, come le stampanti a matrice di punti, o le stampanti a margherita. Se nella linea di comando **lp** non viene specificato il tipo di contenuto, **lp** tenta di determinare il tipo di contenuto esaminando il file; se non riesce a determinare il tipo, assume il tipo **simple**. Poiché **lp** riesce a determinare solo pochi tipi di contenuto comuni, è sempre preferibile indicare esplicitamente il tipo di contenuto, se lo conoscete.

13.4 Opzioni addizionali di stampa e default

Il sottosistema **lp** prevede molte altre opzioni e controlli. Il gestore può autorizzare o proibire a singoli utenti, le richieste di particolari stampanti, moduli o filtri. Nelle richieste di stampa sono ammessi vari tipi di trattamenti speciali (*special handling*); un utente, se autorizzato, può richiedere che un lavoro sia immediatamente stampato su una stampante, oppure trattenuto nella coda di spool. Può essere richiesta la stampa di pagine selezionate di un documento, e possono essere richieste non abituali dimensioni delle pagine o speciali modi di stampa.

In aggiunta, un utente può specificare controlli addizionali per specifiche stampanti, a seguito dell'opzione **-o** (option) sulla linea di comando **lp**, per esempio:

```
$ pr file | lp -o nobanner
$
```

Il file viene stampato senza la normale pagina iniziale di intestazione; questo può essere necessario per stampare con una stampante PostScript. Altre opzioni controllano la larghezza della pagina, la lunghezza della pagina, specifiche **stty** usate dalla procedura di interfaccia per configurare il collegamento della stampante, e altri attributi. Consultate le man page **lp(1)** e **lpadmin(1)** per i dettagli di queste caratteristiche progredite.

13.5 Determinazione dello stato della stampante

Il comando **lpstat** fornisce alcune informazioni sullo stato globale del sottosistema **lp**. Se lo lanciate senza argomenti, **lpstat** vi informa sui vostri lavori in coda di spool.

```
$ lp /etc/profile
Request id is ATT470-79 (1 file)
$ lpstat
ATT470-79       giorgio       4290       Apr 27 19:07
$
```

I dati sono, nell'ordine: l'identificatore di richiesta, il nome dell'utente che ha fatto la richiesta, la dimensione dell'uscita in byte e infine la data e l'ora della richiesta. Potete utilizzare **lpstat** in questo modo per determinare l'identificatore di richiesta di tutti i vostri lavori.

Se un lavoro è già in corso di stampa, **lpstat** fornisce anche questa informazione:

```
$ lpstat
ATT470-79       giorgio       4290       Apr 27 19:07   on ATT470
$
```

Al termine dell'operazione di stampa, il lavoro viene rimosso dalla coda e non esiste la possibilità di rintracciarlo, tuttavia se sapete che il lavoro era stato accodato, la sua scomparsa dalla coda indica che è stato stampato. L'opzione **-u** seguita dall'identificatore di un utente permette di esaminare le richieste di stampa di altri utenti:

```
$ lpstat -u luigi
ATT470-79        luigi        4290        Apr 27 19:07
$
```

OPZIONI DELLA LINEA DI COMANDO DI lpstat

Il comando **lpstat** dispone di altre opzioni che possono darvi indicazioni precise sulla configurazione della stampante. L'opzione **-d** (default printer) restituisce il nome della stampante di default.

```
$ lpstat -d
system default destination: ATT470
$
```

L'opzione **-r** (request) indica se è attivo il sistema di stampa:

```
$ lpstat -r
scheduler is running
$
```

Se il sistema **lp** non è disponibile, **lpstat -r** lo specifica:

```
$ lpstat -r
scheduler is not running
$
```

Per attivare il sistema **lp** è indispensabile intraprendere alcune azioni di gestione, come vedremo tra breve.

Potete determinare lo stato di una determinata stampante con l'opzione **-p** (printer), seguita dal nome della stampante:

```
$ lpstat -p ATT470 -l
```

Assieme all'opzione **-p**, l'opzione **-l** produce una elencazione estesa delle informazioni sulla stampante; l'opzione **-D** produce informazioni sommarie. L'opzione **-t** (total) di **lpstat** fornisce tutte le indicazioni sul sistema di stampa:

```
$ lpstat -t
scheduler is running
```

```
system default destination: ATT470
members of class Parallel:
        ATT470
device for ATT470: /dev/lp0
ATT470 accepting requests since Aug 19 18:58
Parallel accepting requests since Aug 19 18:58
printer ATT470 now printing ATT470-85. enabled since Apr 27 21:03.
        available.
ATT470-85       giorgio      4290       Apr 27 21:01 on ATT470
ATT470-86       root          526       Apr 27 19:11
$
```

In questo output vengono indicati: la stampante di default del sistema (ATT470 in questo caso), il pathname del dispositivo hardware (**/dev/lp0**) per quella stampante, l'indicazione se la stampante accetta lavori da stampare, se la stampante è abilitata e la coda dei lavori correnti.

Le stampanti possono essere raggruppate in *classi* (che tratteremo in dettaglio tra breve), in modo che stampanti dello stesso tipo sono raggruppate insieme, ma distinte da quelle di un altro tipo. In questo modo il carico di lavoro può essere ripartito tra più unità di output dello stesso tipo, e tuttavia potete indirizzare l'uscita a un tipo o a un altro secondo le vostre esigenze. Il comando **lpstat -t** elenca gli appartenenti a una classe di stampanti e indica se la classe nel suo complesso accetta richieste di stampa. Tratteremo le classi di stampanti in dettaglio più oltre nel capitolo.

L'opzione **-t** si rivela una delle più utili per un piccolo sistema, in quanto tutta l'uscita risulta di solito contenuta in una sola schermata. Quando un sistema include molte stampanti di classi diverse e molti sono gli utenti che richiedono l'uscita in stampa, è più indicato utilizzare una opzione di **lpstat** più selettiva.

13.6 lpsched: il demon del sottosistema lp

Il sistema **lp** è controllato da un *demon* (processo di sistema) denominato **lpsched** (**lp** scheduler), che rimane in esecuzione per tutto il tempo che il sistema **lp** è attivo. Quando eseguite il comando **lp** per inviare un file alla stampante, **lp** comunica al processo che un nuovo lavoro è pronto per la coda di stampa. Il comando **lpsched** gestisce la coda dei lavori in stampa, per evitare che più lavori creati nello stesso momento entrino in competizione per le risorse di stampa, e controlla le unità di stampa, rilevando quando la stampante è inattiva o non funziona come dovrebbe.

Se volete qualche informazione sul programma **lpsched**, lanciate il comando **ps -ef**.

```
$ ps -ef
     UID   PID  PPID  C    STIME TTY      TIME COMMAND
    root     0     0  0     Apr 9 ?        0:00 sched
    root     1     0  0     Apr 9 ?        3:03 /sbin/init
    root     2     0  0     Apr 9 ?        0:00 pageout
    root     3     0  0     Apr 9 ?        2:46 fsflush
    root     4     0  0     Apr 9 ?        0:16 kmdaemon
    root    82     1  0     Apr 9 ?        0:05 /usr/lib/saf/sac -t 300
    root    89    82  0     Apr 9 ?        0:05 /usr/lib/saf/ttymon
    root    78     1  0     Apr 9 ?        0:45 /usr/sbin/cron
    root    69     1  3     Jan 1 ?        1:20 /usr/lib/lp/lpsched
  giorgio  6756  6755  3  13:04:57 tty00    0:01 ksh
  giorgio  6762  6756 21  13:05:28 tty00    0:00 ps -ef
$
```

Il processo **lpsched** è di proprietà dell'utente **root**, il suo processo padre è **init**. In alcuni sistemi il processo **lp** ha associato un particolare identificatore di utente **lp**, che potete usare come login per stampare lavori accodati da macchine remote. Di solito, nessuno dovrebbe cercare di collegarsi come **lp**, tuttavia è conveniente disabilitare la password di **lp** in **/etc/shadow** per ragioni di sicurezza.

ATTIVAZIONE E DISATTIVAZIONE DELLO SCHEDULER

Per stabilire se il processo **lpsched** è in esecuzione sulla vostra macchina, potete utilizzare il comando **lpstat -r**; oppure il comando **lp** vi avverte se lo scheduler non è attivo. Se il processo non è in esecuzione, non potete accodare lavori per la stampante. Il comando

```
# /usr/lib/lp/lpsched
```

fa partire lo scheduler se non è già in esecuzione, ma di solito lo scheduler parte automaticamente all'inizializzazione del sistema. Il comando **lpsched** e altri comandi di gestione **lp** sono riservati al superuser; se **lpsched** non è in esecuzione, è segno che esiste qualche problema nel vostro sistema **lp**.

Potete disattivare **lpsched** con

```
# /usr/sbin/lpshut
```

Questo metodo deve essere usato in luogo di un comando **kill** del processo **lpsched**, perché non interrompe la stampa bruscamente, ma conclude ordinatamente i lavori in corso.

13.7 Connessione di una stampante

Esistono due tipi fondamentali di stampante: *seriale* e *parallela*. Questi termini si riferiscono al modo con cui i dati vengono trasmessi attraverso il cavo che collega l'elaboratore alla stampante. Le stampanti seriali generalmente si collegano al calcolatore attraverso una porta RS-232, ed eventualmente un modem seriale, mentre le stampanti parallele possono collegarsi al calcolatore attraverso un unico connettore di interfaccia parallela, o qualche volta con un connettore DB-25 molto simile a un connettore RS-232. Non potete collegare una stampante parallela a una porta seriale o viceversa, per cui è importante definire il cavo per la stampante con la casa costruttrice, o col vostro venditore di hardware. Poiché il cablaggio dipende dal tipo di stampante, collegare a un elaboratore stampanti "sconosciute", anche se ne conoscete la natura seriale o parallela, può risultare un'operazione difficile e fastidiosa. È quindi sempre preferibile acquistare assieme alla stampante il cavo adatto alla connessione.

Le stampanti PostScript spesso consentono solo accesso seriale, mentre le stampanti a matrice di punti e stampanti *letter-quality* possono essere seriali o parallele. La serie HP LaserJet (e compatibili) può includere ambedue le interfacce. Le stampanti PostScript connesse a una porta seriale richiedono di solito un cavo *null-modem*.

La maggior parte delle stampanti dispone di un grande numero di switch (interruttori) che definiscono il comportamento della stampante; per l'impiego consultate il manuale d'uso, o la casa costruttrice della stampante, oppure il rivenditore.

CONFIGURAZIONE DELLA STAMPANTE

Con il sistema **lp** potete impiegare senza problemi sia stampanti seriali che stampanti parallele; dovete specificare a quale porta (*port*) viene collegata la stampante. La Tabella 13.1 elenca le porte e i *device file* associati disponibili nei sistemi UNIX della classe AT. Molte macchine di piccole dimensioni hanno una sola porta parallela; se ne hanno più di una, esiste anche un numero corrispondente di *device file* associati **/dev/lp1**, **/dev/lp2**, ecc. Poiché è più facile e più economico aggiungere porte seriali che porte parallele, avrete più probabilità di configurare sistemi con stampanti seriali multiple, utilizzando i device file **/dev/tty02s**, **/dev/tty03s**, ecc.

Le stampanti seriali usualmente comunicano di default alla velocità di 9600 bps; dovete accettare il default se la stampante prevede il controllo di flusso dei dati (*xon-xoff*). Molte stampanti seriali (anche stampanti PostScript) possono comunicare a 9600 bps, ma possono stampare solo a velocità inferiore a 1200 bps; senza il controllo di flusso dei dati la stampante non funzionerebbe correttamente. Predisponete la velocità di comunicazione della stampante con gli switch hardware interni, e predisponete la velocità nei default di stampante dal lato del sistema UNIX, come spiegato più avanti in questo capitolo.

Tabella 13.1 Dispositivi di stampa per piccoli sistemi SVR4.

Nome PC	Device file	Tipo
LPT1	/dev/lp /dev/lp0	Parallelo
LPT2	/dev/lp1	Parallelo
LPT3	/dev/lp2	Parallelo
COM1	/dev/tty00s /dev/term/tty00s	Seriale
COM2	/dev/tty01s /dev/term/tty01s	Seriale
COM3	/dev/tty02s /dev/term/tty02s	Seriale
COM4	/dev/tty03s /dev/term/tty03s	Seriale

Se avete una stampante seriale, accertate che sulla porta della stampante non sia configurato anche un supervisore di porta per terminale.

Se volete collegare una stampante di tipo nuovo o poco conosciuto, dovete avere una conoscenza molto approfondita del comportamento di quella stampante. Il sistema **lp** tratta i file di uscita attraverso un programma di interfaccia che generalmente è una procedura di shell; questa procedura prepara la pagina di testata, configura la porta di I/O (utilizzando **stty**) e scrive i dati nel *device file* opportuno. Diversamente dalle release precedenti, SVR4 usa una singola procedura di interfaccia, **standard**, per i più comuni tipi di stampante. Potete avere necessità di aggiornare il contenuto di questa procedura per adattarla a stampanti inusuali; talvolta i costruttori di stampanti forniscono una procedura di interfaccia propria, se quella standard non risulta adatta.

13.8 Installazione di una stampante nel sistema lp

Per aggiungere una nuova stampante al sistema occorre seguire vari passi. Primo, dovete verificare che la stampante sia collegata correttamente alla macchina e che funzioni. Secondo, dovete esaminare la procedura d'interfaccia verificando che si comporti come necessario; in rari casi può essere necessario modificarla per adattarla alle esigenze della stampante. Poi, dovete definire le caratterizzazioni di default per la stampante, in modo che **lp** possa dialogare anche se l'utente non specifica opzioni speciali in linea di comando. Potete anche stabilire dei filtri per convertire i file al formato in output corretto per la stampante. Poi, dovete informare il sistema **lp** che la stampante è disponibile. Finalmente, dovete abilitare la stampante perché il software **lp** inizi ad accodare i lavori di stampa. Nelle sezioni seguenti esaminiamo dettagliatamente questi passi.

VERIFICA DELLA CONFIGURAZIONE DELLA STAMPANTE

Potete verificare la connessione della stampante ridirigendo l'uscita direttamente al device file a cui ritenete sia collegata la stampante.

```
$ cat /usr/spool/lp/model/serial > /dev/lp
```

Se in stampa non appare niente del tutto, il cablaggio o gli switch di configurazione sono impostati in modo errato oppure la stampante stessa non funziona; se appare una stampa, ma alterata, l'errore è da ricercare nell'impostazione degli switch.

La verifica precedente funziona con molte stampanti che accettano flussi di normali caratteri (tipo di contenuto **simple**), come le stampanti a matrice di punti e simili. Invece, stampanti che dialogano in PostScript o in altro *page description language*, prima di potere stampare qualunque testo devono ricevere un programma eseguibile nel loro linguaggio di controllo. Se avete una stampante PostScript potete provare con il testo seguente.

```
$ cat testfile.ps
%!
/Times-Roman findfont 14 scalefont setfont
300 400 moveto
(salve gente) show
showpage
$ cat testfile.ps > /dev/tty00s
```

Se il messaggio *salve gente* compare su stampante, la connessione funziona. Se questa procedura non funziona con una stampante PostScript, tentate di connettervi alla porta della stampante alla velocità configurata (normalmente 9600 baud) usando il comando **cu**, e ponete la stampante in modo interattivo. Se non potete ancora dialogare con la stampante, qualcosa non è configurato correttamente.

MODELLI DI INTERFACCIA CON LE STAMPANTI

Il directory **/usr/lib/lp/model** contiene alcune procedure di shell che interfacciano il comando **lp** con il device file che di fatto controlla la stampante. Queste procedure, chiamate *modelli*, sono responsabili della caratterizzazione degli attributi dell'unità di stampa, come per esempio la velocità di comunicazione e il controllo di flusso dati; della formattazione e della stampa del messaggio di testata (banner) che separa i differenti lavori in uscita; della preparazione di copie multiple dei lavori inviati in stampa, ecc. Generalmente, quando aggiungete una nuova stampante, dovete selezionare uno di questi modelli per utilizzarlo con la stampante. In molti sistemi SVR4 il singolo modello *standard* è adatto per quasi tutti i comuni tipi di stampante, sia seriale che parallela.

I modelli sono procedure di shell, quindi potete leggerli per comprendere come interfacciano il demon **lpsched** con l'hardware della stampante. Accettano numerosi argomenti in linea di comando, e producono l'uscita sul loro standard output. Il programma **lpsched**, quando è in azione, esegue la procedura, o modello, con la sua uscita ridiretta al device file appropriato.

Se avete necessità di creare un nuovo modello di interfaccia, copiate il modello standard in un directory separato, e modificate la copia. Dovete proprio collocare la vostra interfaccia in un directory separato, perché gli strumenti di gestione di **lp** hanno una limitazione che richiede che distinguiate i vostri modelli dal modello standard. Caratterizzate i permessi e la proprietà del nuovo modello nella stessa identica maniera del modello standard.

CONFIGURAZIONE DEL SOFTWARE lp

Il comando **/usr/sbin/lpadmin** serve per inizializzare e cambiare la configurazione della stampante. Con questo comando potete collegare nuove stampanti, definire il tipo di stampante per il sistema **lp**, assegnare le stampanti alle classi, caratterizzare i default di stampante, abilitare e disabilitare logicamente le stampanti. Potete utilizzare il comando **/usr/sbin/lpadmin** solo dopo esservi assicurati che la stampante è configurata correttamente e che la procedura di modello è corretta. Il comando **lpadmin** è un sofisticato programma che gestisce numerose opzioni, il suo uso è riservato al superuser. In SVR4 dovete essere sicuri che il programma **lpsched** sia in esecuzione quando usate **lpadmin**; verificate questa condizione con **lpstat** **–t** prima di avviare operazioni di gestione del sistema **lp**.

Potete assegnare a una stampante un nome qualsiasi, ma trattandosi del nome pubblico usato nei vostri comandi, di solito è bene scegliere un nome che richiami il tipo della stampante. Negli esempi che seguono, la nuova stampante viene chiamata PS; nel sistema esiste già una stampante di nome ATT470. Per referenziare una stampante, utilizzate il comando **lpadmin** seguito dall'opzione **–p** (printer).

```
# lpadmin –p PS ....
```

Altre opzioni per **lpadmin** forniscono informazioni per le operazioni di gestione della stampante.

Quando aggiungete una nuova stampante, **lpadmin** si aspetta opzioni che definiscono la procedura di modello e il dispositivo, oltre al nome della stampante.

```
# lpadmin –p PS –m standard –v /dev/tty01s
```

L'opzione **–m** (model) è seguita dal nome della procedura di modello, mentre l'opzione **–v** è seguita dal pathname completo del device file che inten-

dete utilizzare. Il comando di questo esempio aggiunge una stampante di nome PS al sistema. L'opzione **-m** tratta correttamente solo i modelli predefiniti che vengono consegnati con il sistema; se dovete cambiare il modello per un nuovo tipo di stampante, dovete utilizzare l'opzione **-i** (interface), specificando il pathname completo della vostra nuova procedura di interfaccia.

```
# lpadmin -p PS -i/home/giorgio/lpscript -v/dev/tty01s
```

IL TIPO DI STAMPANTE

Dovete specificare anche il *tipo* di stampante, usualmente il nome del modello di stampante, o un equivalente. In SVR4 i tipi di stampante sono memorizzati nel database **terminfo**, assieme all'elenco dei terminali accettati, e con le informazioni specifiche di ogni stampante, come le dimensioni "naturali" della pagina, o come generare un salto pagina. Può essere difficile a volte determinare il tipo corretto di una stampante, ma se scorrete il file **terminfo** (col programma **infocmp**), di solito potete trovare un nome che rappresenta la vostra stampante. Per esempio, il tipo della stampante ATT-470 è **att470**, il tipo di Apple LaserWrite è **lw** o **postscript**. Definite il tipo di stampante a seguito dell'opzione **-T** (type) del comando **lpadmin**, dopo avere installato la stampante; oppure usate il comando con questo formato:

```
# lpadmin -p PS -T lw
```

Per un corretto funzionamento del sistema **lp** è essenziale che il tipo di stampante sia esatto.

IL TIPO DI CONTENUTO

Aggiungendo una nuova stampante, dovete anche specificare il formato dei dati che può stampare direttamente; il default è il tipo di contenuto **simple**. Questo è adatto per le stampanti a matrice di punti e letter-quality, ma non è valido per le stampanti PostScript, che normalmente non stampano correttamente file di semplice testo. Per definire il tipo di contenuto accettato da una stampante dovete fornire un elenco di tipi di contenuto dopo l'opzione **-I** (Interface), sulla linea di comando **lpadmin**; per esempio:

```
# lpadmin -p PS -I postscript
#
```

I tipi di stampante accettabili coincidono con i tipi di contenuto permessi, cosicché se la stampante può trattare solo un tipo di contenuto, potete omettere l'opzione **-I**.

Sono consentiti più tipi di contenuto, separati da virgola, nell'opzione **-I**; dovete includere esplicitamente **simple**, se la stampante può trattare file di semplice testo.

Se una stampante può trattare direttamente un tipo di contenuto, e se l'utente in una richiesta di stampa ha specificato un tipo di contenuto compreso fra i tipi di contenuto di quella stampante, il sistema **lp** stamperà direttamente il lavoro sulla stampante; altrimenti sarà necessario un filtro, come spiegato più avanti nel capitolo.

OPZIONI ADDIZIONALI E DEFAULT

Se l'utente nella linea di comando **lp** non specifica alcuna opzione con **-o**, per quel lavoro di stampa verranno usati valori di default. Di solito, questi default sono tratti dall'elemento in **terminfo** per quella determinata stampante, e raramente devono essere cambiati; tuttavia, se necessario, potete cambiare le opzioni di default al momento dell'installazione della stampante con l'opzione **-o** nella linea di comando **lpadmin**. Per esempio, per disporre la velocità di comunicazione di una stampante a 1200 bps, dovete usare un comando tipo questo:

```
# lpadmin -p ATT470 -o stty='1200 cs8 ixon -ixany opost'
```

I valori specificati per **stty** diventano i normali default per la stampante indicata. Potete sopprimere la stampa della pagina di banner in modo simile:

```
# lpadmin -p PS -o nobanner
```

Notate che l'opzione **nobanner** è abitualmente richiesta per installare una stampante PostScript, o un'altra periferica che usa un linguaggio di descrizione pagina, perché molte versioni di sistemi **lp** stampano il banner in tipo di contenuto **simple**.

Potete cambiare molte altre opzioni di default, documentate nella man page di **lpadmin(1)**.

DESTINAZIONE DI DEFAULT

Se state installando la prima e unica stampante, conviene che questa diventi la stampante di default di sistema (*system default destination*); in questo modo potete utilizzare il comando **lp** senza indicare nessun nome di stampante come argomento. Se il sistema dispone di più stampanti, si assume generalmente come unica stampante di default quella di qualità inferiore, o quella di uso più comune. Potete anche comunicare al sistema **lp** quale stampante intendete assumere come destinazione di default. In questo caso,

dopo avere installato la stampante, utilizzate il comando **lpadmin** con l'opzione **-d** (default), specificando di seguito il nome della stampante, per esempio:

```
# lpadmin -d PS
```

In questo modo, PS diventa la stampante di default. Se non indicate una stampante di default, non viene definita la destinazione di default, e dovrete indicare esplicitamente il nome della stampante ogni volta che utilizzate il comando **lp**.

VERIFICA DELL'INSTALLAZIONE

Potete lanciare **lpstat** per verificare se avete installato correttamente la stampante nel sistema:

```
$ lpstat -t
scheduler is running
system default destination: PS
members of class Parallel:
        ATT470
device for ATT470: /dev/lp
device for PS: /dev/tty01s
ATT470 accepting requests since Thu Apr 19 18:58 MDT 1990
Parallel accepting requests since Thu Apr 19 18:58 MDT 1990
PS not accepting requests since Mon Apr 30 19:00 MDT 1990 -
        new destination
printer ATT470 is idle. enabled since Thu Apr 30 17:56 MDT 1990.
        available
printer PS disabled since Mon Apr 30 19:00 MDT 1990 available.
        new printer
$
```

Si può notare che la stampante è stata collegata correttamente, inoltre si hanno diverse altre informazioni; innanzitutto, che la stampante non sta accettando richieste ed è disabilitata. Se le informazioni ottenute non risultano simili a queste, dopo che avete aggiunto una stampante, il comando non ha funzionato come dovuto, per cui dovete ripetere l'installazione.

DISINSTALLAZIONE DI UNA STAMPANTE

Potete rimuovere, o disinstallare, una stampante dal sistema con l'opzione **-x** (exterminate):

```
# lpadmin -x PS
```

ACCETTAZIONE DI RICHIESTE DI STAMPA

Una nuova stampante aggiunta al sistema risulta installata ma non attivata. Altre due caratteristiche del sistema **lp** sono importanti in questo contesto. Potete configurare una stampante per accettare o respingere le richieste di stampa senza rimuoverla dal sistema; il risultato è che il comando **lp** rifiuta di accodare un lavoro quando il dispositivo non accetta le richieste di stampa. Utilizzate il comando **/usr/sbin/accept**, per consentire a una stampante di accettare le richieste di stampa.

```
# accept PS
destination "PS" now accepting requests
# lpstat -t
scheduler is running
system default destination: ATT470
members of class Parallel:
        ATT470
device for ATT470: /dev/lp
device for PS: /dev/tty01s
ATT470 accepting requests since Thu Apr 19 18:58 MDT 1990
Parallel accepting requests since Thu Apr 19 18:58 MDT 1990
PS accepting requests since Mon Apr 1 18:33 MDT 1990
printer ATT470 is idle. enabled since Thu Apr 30 17:56 MDT 1990.
        available
printer PS disabled since Mon Apr 30 19:00 MDT 1990. -
        new printer
#
```

PS accetta ora le richieste di stampa.

Utilizzate **/usr/sbin/reject** per impedire a una stampante di accettare le richieste di stampa. Il comando viene di solito usato quando la stampante deve essere messa fuori servizio per un tempo relativamente lungo, ma non deve essere eliminata permanentemente dal sistema, come con il comando **lpadmin -x**. Il comando **/usr/sbin/reject** accetta come argomento il nome della stampante, ed eventualmente l'opzione **-r** (reason) seguita da un messaggio che spiega il motivo per cui viene disabilitata:

```
# reject -r"Disattivata fino a giovedì per manutenzione" PS
destination "PS" is no longer accepting requests
#
```

Il comando **lpstat -t** visualizza il messaggio con il motivo che ha portato a disabilitare la stampante; questo stesso messaggio viene anche trasmesso all'utente che cerca di inviare dati alla stampante con il comando **lp**.

ABILITAZIONE DELLA STAMPANTE

Una stampante, una volta che inizia ad accettare le richieste di stampa, non è tuttavia ancora operativa; infatti accetta le richieste, ma i lavori sono solo introdotti nella coda di spool per la stampante. Il comando **lpstat −t** nei precedenti esempi, indica la stampante come disabilitata. Dovete abilitare la stampante perché possa effettivamente cominciare a stampare i lavori accodati. Inoltre, il software **lp**, diversamente dalla condizione di accettazione o rifiuto delle richieste di stampa (*accept/reject*), disabilita automaticamente una stampante se l'operazione di stampa non va a buon fine; se termina la carta o cade la tensione, la stampante viene disabilitata automaticamente. Per abilitare una stampante, dovete utilizzare il comando **/usr/bin/enable**.

```
# enable PS
# lpstat −t
scheduler is running
system default destination: ATT470
members of class Parallel:
        ATT470
device for ATT470: /dev/lp
device for PS: /dev/tty01s
ATT470 accepting requests since Thu Apr 19 18:58 MDT 1990
Parallel accepting requests since Thu Apr 19 18:58 MDT 1990
PS accepting requests since Mon Apr 30 21:02 MDT 1990
printer ATT470 is idle. enabled since Thu Apr 19 18:58 MDT 1990.
        available
printer PS is idle. enabled since Mon Apr 30 21:06 MDT 1990.
        available
#
```

Ora la stampante è pienamente operativa.

Il comando **lp** disabilita automaticamente una stampante quando rileva, o suppone, un malfunzionamento della stampante; una volta risolto il problema, dovete abilitarla manualmente. Spesso, può sembrare che la stampante non funzioni per un errore di configurazione, mentre in realtà la stampante è solo disabilitata. Verificate le informazioni fornite dal comando **lpstat −t** per accertare lo stato di una stampante non funzionante. Il sistema **lp** abitualmente invia al gestore messaggi di avviso via mail, quando una stampante viene disabilitata, e molte altre notizie per richiamare l'attenzione su condizioni anormali nel sottosistema di stampa. Esaminate regolarmente questa posta, o ritrasmettetela a un login di utente usato più spesso di **lp** e del login del gestore.

Potete disabilitare una stampante manualmente con il comando **/usr/bin/disable**.

```
# disable PS
```

Il comando **disable** accetta tre argomenti opzionali. L'opzione **-c** (cancel) elimina tutti i lavori che sono in stampa prima che il dispositivo venga disabilitato, al contrario, l'opzione **-W** (wait) disabilita la stampante solo dopo che tutti i lavori correnti sono stati completati. L'opzione **-r** (reason) permette di predisporre un messaggio che specifica il motivo per disabilitare il dispositivo:

```
# disable -c -r"Fine della carta" PS
```

Il comando **disable** permette di interrompere temporaneamente l'uscita a una stampante mentre aggiungete carta o togliete tensione per qualche motivo. Poiché **lp** accetta ancora le richieste, dovrete riportare abbastanza rapidamente la stampante in linea, altrimenti la coda dei lavori può raggiungere una dimensione notevole.

13.9 Trasferimento di lavori tra stampanti

Quando una stampante non funziona correttamente, o la coda dei lavori di una stampante ha raggiunto una dimensione eccessiva, potete decidere di distribuire i lavori tra più stampanti. Questa possibilità è data dal comando **/usr/sbin/lpmove**, che accetta come argomenti una lista di identificatori di richieste seguiti dall'identificatore della nuova destinazione. Con questo comando, le richieste vengono trasferite da una stampante a un'altra:

```
# lpmove ATT470-87 ATT470-88 daisy
```

Questo comando trasferisce le richieste ATT470-87 e ATT470-88 alla stampante daisy. In SVR4 dovete essere sicuri che **lpsched** sia in esecuzione prima di trasferire i lavori tra stampanti. Una seconda forma del comando **lpmove** trasferisce tutti i lavori accodati su una stampante a un'altra stampante:

```
# lpmove ATT470 daisy
```

Questo comando sposta tutti i lavori accodati per ATT470 sulla coda di daisy. Tenete presente che non potete trasferire normalmente lavori tra stampanti di tipi incompatibili; per esempio, non potete trasferire lavori accodati per una stampante a matrice di punti a una stampante PostScript. In questo caso, dovete cancellare i lavori ed effettuare nuovamente le richieste con **lp**.

13.10 Approfondimenti

Gli argomenti trattati finora forniscono informazioni sufficienti alla gestione di semplici stampanti in piccoli sistemi. Una macchina UNIX, tuttavia, può costituire anche un eccellente *print server* dedicato al controllo di molte stampanti di differenti tipi. Un sistema di questo tipo può essere collegato a una rete locale (LAN) per gestire efficientemente un'organizzazione di notevoli dimensioni; questo approccio può ridurre notevolmente i costi di un sistema di stampa, perché numerosi utenti possono condividere dispositivi di stampa di qualità elevata, e quindi di maggior costo. Le caratteristiche del sottosistema **lp** trattate nel seguito sono molto importanti in grandi installazioni con molte stampanti.

CLASSI DI STAMPANTI

Una stampante può essere assegnata a una classe di stampanti; una classe è un gruppo di stampanti che condividono una sola coda di lavori in spool. In altre parole, quando una stampante della classe diventa inattiva, stampa il primo lavoro della coda condivisa. Il software **lp** gestisce la coda assegnando i lavori alle stampanti che non sono impegnate.

In pratica, ciascuna classe include stampanti dello stesso tipo o di tipo analogo; per esempio, le stampanti laser possono appartenere a una classe, mentre le stampanti a matrice di punti possono appartenere a un'altra. Se non vi interessa distinguere le unità di stampa, potete definire un'unica classe che le contenga tutte.

Potete fare riferimento a un'intera classe di stampanti tramite un solo identificatore di destinazione; se la configurazione della stampante è la stessa dell'esempio precedente, questo comando

```
$ lp -d ATT470 /usr/spool/lp/model/serial
```

produce lo stesso risultato di

```
$ lp -d Parallel /usr/spool/lp/model/serial
```

Questo perché la stampante ATT470 appartiene alla classe **Parallel**. L'output del comando **lpstat -t** mostra le classi di stampanti definite nel sistema. Ogni volta che utilizzate una stampante nei comandi del sistema **lp**, potete utilizzare la classe della stampante.

Utilizzate l'opzione **-c** (class) con **lpadmin** per assegnare una stampante a una classe.

```
# lpadmin -p PS -c uno -m standard -v /dev/tty01s
# accept PS
```

```
destination "PS" now accepting requests
# enable PS
printer "PS" now accepting requests
#
```

Questo esempio assegna PS alla classe **uno**. Dovete utilizzare l'opzione **-c** quando inizialmente installate la stampante con **lpadmin**. Per cambiare la classe di una stampante esistente, rimuovetela completamente dal sistema con **lpadmin -x** e installatela nuovamente specificando una classe diversa. Il nome della classe che assegnate può essere o una classe di stampanti già esistente o una nuova classe; in quest'ultimo caso, **lp** crea una nuova classe che comprende solo quella stampante.

Dopo avere eseguito il comando dell'esempio precedente, **lpstat -t** visualizza dati diversi:

```
# lpstat -t
scheduler is running
system default destination: ATT470
members of class Parallel:
        ATT470
members of class uno:
        PS
device for ATT470: /dev/lp
device for PS: /dev/tty01s
ATT470 accepting requests since Thu Apr 19 18:58 MDT 1990
Parallel accepting requests Thu Apr 19 18:58 MDT 1990
PS accepting requests since Mon Apr 30 21:40 MDT 1990
uno not accepting requests since Mon Apr 30 21:40 MDT 1990 -
        new destination
printer ATT470 is idle. enabled since Thu Apr 19 18:58 MDT 1990.
        available
printer PS is idle. enabled since Mon Apr 30 21:06 MDT 1990
        available
#
```

In questo modo è stata definita una nuova classe a cui appartiene la stampante PS. Precedentemente abbiamo utilizzato il comando **accept** per consentire a PS di accettare le richieste di stampa, ma la nuova classe **uno** ancora le rifiuta. Con la configurazione descritta, potete accodare i lavori alla stampante PS, ma non direttamente sulla classe **uno**; per utilizzare la classe come destinazione dovete eseguire esplicitamente il comando **accept** anche per la classe **uno**.

```
# accept uno
destination "uno" now accepting requests
#
```

USO DI MODULI PRESTAMPATI

La versione SVR4 del sistema **lp** consente all'utente di richiedere che un lavoro di stampa venga stampato su un determinato modulo prestampato (*form*), o con un particolare insieme di caratteri (*printwheel*). Queste sono risorse specifiche di stampante, che devono essere "montate" manualmente prima di potere essere usate; cioè, dovete installare manualmente su una stampante un modulo (carta da lettere intestata o carta prestampata in un certo modo) prima che il lavoro venga stampato. Potete montare un determinato modulo su una stampante e informare il sistema **lp** che il modulo è disponibile su una determinata stampante. In seguito, quando gli utenti richiedono la stampa su quel modulo, il sistema **lp** stamperà il lavoro su quella stampante; nel caso il modulo non risulti montato, il lavoro verrà accodato, ma non verrà stampato fino a che il modulo non verrà montato. I lavori che richiedono moduli non montati originano l'invio di un messaggio di avviso al gestore, richiedendo il montaggio del modulo.

Una procedura del tutto simile viene supportata per *printwheel*, che sono dotazioni alternative di caratteri, sostituibili su certe stampanti, di solito stampanti *letter-quality* che consentono il cambio delle testine di stampa (margherite o sfere). Un utente può richiedere un determinato insieme di caratteri tramite un nome di riferimento; se quella risorsa non è montata, viene avvertito un gestore, mentre il lavoro viene salvato in una coda speciale; dopo che la risorsa è stata montata, il lavoro viene stampato e rimosso dalla coda.

Di norma, queste operazioni sono di interesse per i grandi centri di stampa con numerose stampanti, un gestore a tempo pieno, e molti utenti che richiedono differenti tipi di lavori di stampa. In sistemi più piccoli, con solo pochi utenti e un gestore non dedicato, è molto più semplice evitare complicazioni verificando di persona che la carta e la testina di stampa desiderate siano montate, prima di richiedere il lavoro che le utilizza.

A ogni modulo e testina di stampa definiti per i meccanismi di montaggio e avviso deve essere associato un nome e alcuni attributi. Per rendere disponibile un determinato modulo, il gestore deve includere il modulo nell'elenco dei moduli conosciuti al sistema, e predisporre gli attributi relativi; quindi deve informare il sistema **lp** dei moduli montati correntemente su ciascuna stampante del sistema. Le richieste di lavori di stampa da parte dgli utenti, con **lp**, devono specificare il modulo con l'opzione **–f** (form), e l'insieme dei caratteri con l'opzione **–S**, per esempio:

```
$ lp –f lettera –S elite mia.lettera
```

Il gestore deve montare le risorse e informare il sistema **lp** dell'operazione, perché il lavoro possa essere stampato.

CREAZIONE DI MODULI

Il comando **/usr/sbin/lpforms** genera le definizioni di moduli nuovi. Prima, dovete creare un file di controllo che elenca gli attributi dei moduli: lunghezza e ampiezza della pagina, numero di pagine del modulo, e così via. Ecco un esempio:

```
# cat lettera.form
Page length: 66
Page width: 80
Number of pages: 1
Character set choice: any
#
```

L'elenco completo di tutti i possibili attributi è incluso nella man page **lpforms(1)**; è ammesso includere nel file di controllo solo una parte degli attributi previsti; le voci mancanti assumono valori di default.

Dopo avere generato il file di controllo, dovete includere il nuovo modulo nella lista dei moduli disponibili, per esempio:

```
# lpforms -f lettera -F lettera.form -A mail
#
```

Il comando non fornisce segnalazioni se l'azione ha successo. Il nome del modulo segue l'opzione **-f** (form), il nome del file degli attributi segue l'opzione **-F** (File); ambedue le opzioni sono obbligatorie, tuttavia potete usare - (meno) in sostituzione della sola opzione **-F** se volete introdurre il file di attributi da standard input invece che da un file.

L'opzione **-A** (alert-type) consente di specificare il tipo di avviso da inviare quando un determinato modulo è richiesto ma non è montato; **mail** invia il messaggio per posta elettronica all'identificatore di login che ha eseguito il comando **lpforms**; **write** invia il messaggio direttamente alla console del gestore; ci sono anche diverse altre possibilità.

Dopo avere installato un gruppo di moduli, potete ottenere un elenco degli attributi di un determinato modulo, nell'esempio il modulo *modulo*, con l'opzione **-l** (list) di **lpforms**:

```
# lpforms -f modulo -l
page length: 66
page width: 80
number of pages: 1
line pitch: 6
character pitch: 10
character set choice: any
ribbon color: any
#
```

Potete listare i nomi e gli attributi di tutti i moduli con:

```
# lpforms -f all -l
```

Per cancellare il modulo *modulo*, usate l'opzione **-x**:

```
# lpforms -f modulo -x
```

ASSEGNAZIONE E MONTAGGIO DI MODULI

Dovete assegnare il modulo a ciascuna stampante che può usarlo. Per l'assegnazione usate il comando **lpadmin** come segue:

```
# lpadmin -p ATT470 -f allow:lettera
```

La speciale parola chiave **allow:** deve seguire l'opzione **-f** ed essere seguita dal nome del modulo. Questa assegnazione deve essere eseguita prima di montare il modulo.

Dopo che il modulo è stato stabilito e assegnato, e fisicamente installato su di una stampante, dovete informare il sistema **lp** che il modulo è montato, con **lpadmin**, come dall'esempio:

```
# lpadmin -p ATT470 -M -f modulo
```

Come d'uso, il nome della stampante segue l'opzione **-p**. L'opzione **-M** (mount) indica il montaggio; il nome del modulo segue l'opzione **-f**. L'opzione **-f** è obbligatoria in presenza dell'opzione **-M**. Il montaggio di un nuovo modulo su di una stampante provoca lo smontaggio del modulo precedente; potete forzare lo smontaggio di un modulo mediante il nome speciale *none*, come nell'esempio:

```
# lpadmin -p ATT470 -M -f none
```

USO DEGLI INSIEMI DI CARATTERI

La stessa procedura dei moduli è prevista per montare differenti insiemi di caratteri, o teste di stampa (*printweel*), che le richieste di stampa degli utenti rendono necessarie. In SVR4 sono previste due forme alternative di insiemi di caratteri. La prima forma, si applica con stampanti che accettano il caricamento via linea (*download*) di caratteri software; questa procedura richiede stampanti relativamente intelligenti, ma consente al sistema **lp** di gestire le richieste degli utenti senza la necessità di interventi manuali da parte del gestore, a patto che la procedura sia supportata. La seconda forma

si applica a stampanti che necessitano del montaggio fisico dell'insieme dei caratteri, generalmente una margherita o una sfera rimovibile.

Gli insiemi di caratteri software supportati da una stampante sono elencati nell'elemento di quella stampante nel file **terminfo**, e una richiesta di un utente, come:

```
$ lp -S elite file.da.stampare
```

(S = Set), verrà correttamente stampata con l'insieme di caratteri indicato, se quell'insieme è previsto. Il comando **lpstat -t** elenca gli insiemi di caratteri previsti per ciascuna stampante nel sistema. Il sistema **lp** eseguirà un montaggio "logico" dell'insieme di caratteri necessario, senza intervento del gestore nell'operazione.

I caratteri da montare fisicamente sono trattati in maniera molto simile ai moduli: il gestore deve dichiarare al sistema i nomi delle teste di stampa disponibili. Una richiesta di utente per un determinato insieme di caratteri, con il comando **lp -S**, provoca l'invio al gestore di un avviso, e il lavoro viene accodato; quando il gestore monta la testa di caratteri voluta e informa il sistema **lp** dell'esecuzione del montaggio, il lavoro viene stampato.

Il comando **lpadmin** viene usato per assegnare un nome alle teste di stampa disponibili, come nell'esempio:

```
# lpadmin -p nomestampante -S elite,math,graphics -A mail
#
```

Abbiate cura di non lasciare spazi nella lista di nomi, separati da virgole, che segue l'opzione **-S**. Il comando precedente assegna alla stampante *nomestampante* l'uso delle tre teste di stampa elencate dopo l'opzione.

Tenete presente che la stampante deve essere configurata per accettare le dichiarazioni di teste di stampa nel suo elemento in **terminfo**.

Per cancellare un insieme di nomi di *printweel*, usate

```
# lpadmin -p nomestampante -S none
```

La speciale parola chiave *none* rimuove l'elenco delle teste di stampa, e ripristina il comportamento di default.

Quando una richiesta di un utente rende necessario il cambio dell'insieme di caratteri, e dopo che il gestore ha risposto al messaggio di avviso montando sulla stampante la testa di stampa opportuna, il sistema **lp** deve essere informato dell'avvenuta sostituzione, col comando:

```
# lpadmin -p nomestampante -M -S elite
```

La testa montata rimane in servizio fino al montaggio di un'altra testa di stampa, o fino al comando di smontaggio, per esempio:

```
# lpadmin -p nomestampante -M -S none
#
```

Se la stampante non accetta la sostituzione della testa, il comando di montaggio viene rifiutato, e viene emesso un messaggio d'errore.

USO DI FILTRI

Potete anche configurare *filtri* di stampa, che accettano in input uno o più tipi di contenuto e riformattano la stampa richiesta per una determinata stampante. Per esempio, una stampante PostScript non è in grado di stampare correttamente molti tipi di file, perché l'input deve essere riformulato nel linguaggio di descrizione di pagina PostScript; molti altri tipi di file possono necessitare di un trattamento simile prima di essere stampati; questo compito viene svolto dai filtri di stampa. Di solito, un insieme di filtri standard è incluso nel sistema **lp** per trattare i più comuni tipi di contenuto e di stampanti. Un utente può specificare nella linea di comando **lp** che la richiesta di stampa ha un particolare tipo di contenuto con l'opzione **-T** (Type). Il sistema **lp** cerca di abbinare quel tipo di contenuto col tipo di contenuto accettato dalle stampanti disponibili; se viene trovato un abbinamento, il file viene inviato direttamente alla stampante; se, al contrario, non viene trovato alcun abbinamento, **lp** cerca di abbinare il tipo di contenuto del file col tipo di contenuto in input dei filtri disponibili, e il tipo di contenuto in output del filtro con il tipo di contenuto delle stampanti. Se ambedue questi abbinamenti vengono risolti, la richiesta di stampa viene passata al filtro prima di essere inviata alla stampante.

Il comando **/usr/sbin/lpfilter** gestisce l'elenco dei filtri disponibili. Potete esaminare i dati relativi a un filtro con:

```
$ lpfilter -f postprint -l
```

Il nome del filtro è specificato dopo l'ozione **-f** (filter); questa opzione è obbligatoria in tutte le linee di comando **lpfilter**. L'opzione **-l** (list) elenca i dati di controllo associati con quel filtro, come nell'esempio:

```
$ lpfilter -f postprint -l
Input types: simple
Output types: postscript
Printer types: any
Printers: any
Filter type: slow
Command: /usr/lib/lp/postscript/postprint
Options: PAGES * = -o*,COPIES * = -c*,LENGTH * = -l*,MODES
portrait = -pp,MODES landscape = -pl,MODES group= * = -n*,MODES
x= * = -x*,MODES y= * = -y*,MODES magnify= * = -m*,MODES
ptsize= * = -s*,CHARSET * = -f*
$
```

Il filtro **postprint** è il filtro standard usato per convertire semplici file di testo al linguaggio PostScript. Altri comuni filtri sono: **dpost**, che stampa l'output di **troff** con una stampante PostScript; **postio**, che stampa file già nel formato PostScript con stampanti PostScript. Se nella linea di comando **lp** non viene specificato alcun tipo di contenuto, il file è considerato del tipo di contenuto **simple**. Potete ottenere la lista completa di tutti i filtri disponibili con i loro attributi, con la linea di comando:

```
$ lpfilter -f all -l
```

Le informazioni ottenute con il comando **lpfilter -l** mostrano come ciascun filtro opera. L'elenco di tipi in input per il filtro viene comparato con il tipo di contenuto della richiesta dell'utente; poi, l'elenco dei tipi di stampante per quel filtro viene abbinato coi tipi di contenuto accettabili per ciascuna stampante (o per la stampante di destinazione selezionata dall'utente). Se ambedue le parti si abbinano, il filtro è appropriato per quel lavoro e la richiesta dell'utente viene passata attraverso quel filtro. Infine, il *comando* viene eseguito, tenendo conto delle *opzioni* elencate alla fine dell'output. Aggiungere nuovi filtri non è un'operazione facile, generalmente comporta numerosi tentativi. La creazione dei tipi in input, tipi in output e del comando (generalmente una procedura di shell) può non essere difficile, ma il sistema di filtro consente complesse opzioni che formano un linguaggio per elaborare gli argomenti delle linee di comandi degli utenti, all'interno del filtro. Per cominciare, create un file contenente i campi mostrati nell'output del precedente comando **lpfilter -l**, quindi create un comando di trattamento per implementare il filtro. Infine, usate **lpfilter** per includere quel filtro nel sistema, come nell'esempio:

```
# lpfilter -f nuovofiltro -F path
```

Il nome del filtro segue l'opzione **-f**, come al solito, e il pathname del file contenente il filtro segue l'opzione **-F**. Consultate la man page di **lpfilter(1)** per maggiori informazioni riguardo alle opzioni necessarie per far funzionare il filtro, o provate a copiare le opzioni da un filtro esistente.

IL SISTEMA UNIX COME SERVER

Un sistema UNIX che disponga del software **lp**, costituisce un eccellente print server. Poiché il supporto alla stampa utilizza una quantità relativamente ridotta di risorse di CPU, una sola macchina può facilmente supportare molte stampanti. I sistemi configurati principalmente come gestori di stampa possono generalmente supportare qualche altra attività contemporaneamente, come per esempio i collegamenti degli utenti e le comunicazioni di dati. È opportuno analizzare una configurazione particolare per deter-

minare quale sia il contributo di carico fornito dalle stampanti; generalmente, tre o quattro stampanti attive nel sistema utilizzano una quantità di risorse di CPU circa uguale a quella richiesta da un utente normale. Tuttavia, questa stima può variare considerevolmente se il server ha anche funzioni di elaborazioni che precedono il lavoro di stampa, come per esempio formattare i dati in uscita tramite **troff**.

Un problema che dovete considerare attentamente quando configurate un sistema gestore di stampa è l'impiego dello spazio su disco. I file che attendono di essere stampati occupano una certa quantità di spazio sul disco della macchina, per cui dovete valutare la velocità con cui i nuovi lavori si accodano, in rapporto con la velocità massima con cui i dispositivi collegati possono stampare. Il numero di stampanti da collegare al sistema dovrà essere adeguato alla quantità di richieste di stampa degli utenti, tuttavia, se una stampante non funziona, la coda dei lavori può crescere rapidamente. Esiste raramente un controllo sull'occupazione del disco da parte dei lavori accodati, per cui un disco si può riempire senza che il gestore del sistema ne venga a conoscenza.

Quando lavorate con print server dovete fare attenzione che le stampanti siano sempre abilitate e che la coda di lavori in attesa non superi le risorse di disco disponibili.

ACCESSO REMOTO ALLA STAMPANTE TRAMITE LAN

Il sistema **lp** vi consente di inviare richieste di stampa a una macchina print server centrale connessa alla vostra macchina via LAN. Questa caratteristica consente alle macchine connesse in rete di condividere le stampanti, contribuendo a ridurre il costo totale di una rete. Dal punto di vista dell'utente, le stampanti remote sono equivalenti a stampanti locali: ognuna ha un nome o un identificatore di destinazione. L'utente specifica questo nome dopo l'opzione **-d** nella richiesta **lp**, ma se la stampante remota è definita destinazione di default nella macchina locale, l'opzione **-d** non è necessaria. Il sistema **lp** ha cura di trasmettere il lavoro alla destinazione, dove viene trattato normalmente, in accordo col resto delle opzioni specificate nella linea di comando. Se è stata richiesta la spedizione di messaggi via posta, la posta viene tramessa indietro nella LAN e consegnata all'utente della richiesta. Tutte le segnalazioni di avviso, filtri e altre azioni, funzionano come per le stampanti locali.

La configurazione di stampanti accessibili in remoto avviene in due passi. Nel primo, il comando **/usr/sbin/lpsystem** viene usato per dichiarare le macchine remote accessibili dalla macchina locale, per esempio:

```
# lpsystem sysremoto
"sysremoto" has been added.
#
```

L'opzione **-t** (type) addizionale, nella linea di comando, identifica il sistema remoto come un sistema UNIX System V (con l'opzione **s5**), oppure come un sistema BSD (con l'opzione **bsd**); per esempio:

```
# lpsystem -t bsd sysremoto
```

Il default è **s5**. Se il print server è un sistema SVR4, è obbligatorio eseguire il comando **lpsystem** sia sul systema server che sul sistema cliente, altrimenti il server rifiuterebbe le richieste di stampa del cliente.

Altre opzioni consentono di specificare i parametri *timeout* e *retry*, che definiscono le modalità di ripetizione delle azioni di **lp** se la macchina remota non è disponibile; di norma i valori di default sono adeguati, quindi non è necessario specificare queste opzioni.

Nel secondo passo, una volta che la macchina remota è disponibile, potete aggiungere con **lpadmin** una stampante della macchina remota all'elenco di stampanti della macchina locale:

```
# lpadmin -p nuovonome -s nomemacchina!nomestampante
#
```

L'opzione **-s** (system) vuole come argomento un nome di macchina remota e un nome di stampante, separati da **!** (punto esclamativo), come in un indirizzo di **mail**; la stampante remota assume lo pseudonimo *nuovonome*. Se la sezione **!nomestampante** viene omessa, per servire le richieste viene usata la stampante di default della macchina remota; se viene omessa l'opzione **-p**, viene usato il nome remoto. Il comando **lpadmin** in questa forma deve essere usato con un *nuovo* nome di stampante; non potete usare una stampante che è stata già configurata nel sistema.

Dopo avere assegnato il nome, dovete configurare il nome locale come al solito per specificare il tipo della stampante, la classe, i moduli associati e altre informazioni. Potete anche definire la stampante remota come destinazione di default per il sistema locale.

Per eliminare un sistema remoto dall'elenco delle disponibilità in **lp**, usate l'opzione **-r**, come segue:

```
# lpsystem -r sysremoto
Removed "sysremoto"
#
```

Potete ottenere l'elenco degli attributi dell'accesso di **lp** al sistema remoto con l'opzione **-l** di **lpsystem**.

ACCESSO REMOTO ALLA STAMPANTE TRAMITE uucp

Oltre che con strumenti basati su reti LAN, potete anche trasferire lavori di stampa ad altre macchine mediante il sistema **uucp**. Potete scrivere una

semplice linea di comando usando **uux** per invocare **lp** su un sistema remoto, per esempio:

```
$ cat miofile ¦ uux - "sysremoto!lp -t$LOGNAME -s"
```

Questo comando trasferisce, con **cat**, il file al comando **lp** sulla macchina remota **sysremoto**, sopprime l'identificatore di richiesta tramite l'opzione **-s** e riporta il vostro identificatore di login sulla testata prodotta da **lp** (opzione **-t**). Dovete caratterizzare il sistema **uucp** sulla vostra macchina in modo che possa spedire file al server, e dovete anche caratterizzare il server in modo che consenta l'esecuzione delle attività di **uux** sulla macchina remota.

Il programma di stampa remota prevede generalmente controlli di errore addizionali sul file di ingresso; inoltre può prevedere la capacità di passare l'identificatore della stampante di destinazione dalla linea di comando locale al comando **uux** e al comando remoto **lp** e anche la possibilità di includere il nome della macchina locale **uname** sulla linea di intestazione. Per svolgere questi compiti potete creare una procedura di shell eseguibile, magari chiamata col nome **rlp**. Questa procedura locale potrebbe accettare nomi di file da stampare, oltre - o invece di - lo standard input, e ritrasmettere all'utente, mediante mail, un messaggio alla fine del lavoro.

Il sistema **lp** supporta direttamente la stampa su stampanti standalone connesse a un modem con risposta automatica (*autoanswer*). Quando un lavoro viene accodato per una tale destinazione, **lp** chiama la stampante ed esegue la stampa. Potete selezionare questa opzione quando configurate la stampante, se omettete l'opzione **-v**, usando invece l'opzione **-U** per selezionare la stampante. L'opzione **-U** accetta un nome di macchina, o numero di telefono, nella stessa forma accettata da **cu**; se potete usare **cu** con una stampante, potete stampare con quella stampante, per esempio:

```
# lpadmin -p daisy -c lettera -m standard -U cuname
```

Qui, *cuname* è configurato come nome di destinazione.

Notate bene che la forma **-U** non accoda un lavoro su di un sistema **lp** remoto, ma stampa un lavoro su una stampante che risponde alla chiamata telefonica, in rete pubblica commutata.

LA STRUTTURA DEL DIRECTORY lp

Gli strumenti di **lp** sono distribuiti nel file system. I semplici comandi utente **lp** e **lpstat** si trovano nel directory **/usr/bin**, mentre i restanti comandi di gestione sono memorizzati in **/usr/sbin**. La coda dello spool di lavori non stampati per ciascuna stampante si trova in **/var/spool/lp**; le tabelle dei filtri e altri dati di gestione in **/etc/lp**; vari strumenti, tabelle e filtri in

/usr/lib/lp/bin; alcuni symlink per comandi di **lp** possono essere in **/usr/lib**. Potete facilmente rintracciare i comandi disponibili perché iniziano di solito con **lp**, per esempio:

```
$ ls /usr/sbin/lp*
/usr/sbin/lpadmin
/usr/sbin/lpfilter
/usr/sbin/lpforms
/usr/sbin/lpmove
/usr/sbin/lpshut
/usr/sbin/lpsystem
/usr/sbin/lpusers
$
```

Inoltre, fanno parte del sistema **lp** anche i comandi **/usr/bin/enable**, **/usr/bin/disable**, **/usr/sbin/accept** e **/usr/sbin/reject**.

Il directory **/var/spool/lp** e i suoi subdirectory contengono le code e altre parti del sistema, per esempio:

```
# cd /var/spool/lp
# ls -F
SCHEDLOCK   bin@    requests/    temp@
admins/     fifos/  system/      tmp/
#
```

Il directory **fifos** contiene percorsi della comunicazione tra il comando utente **lp** e il demon **lpsched**. **SCHEDLOCK** è un *lock file* utilizzato da **lpsched** per evitare l'esecuzione simultanea di più demon **lpsched**. Quando **lpsched** non è attivo, **SCHEDLOCK** non deve essere presente; se **SCHEDLOCK** viene trovato presente al lancio di **lpsched**, il processo non può essere attivato; il comando **lpsched** non fornisce alcun avviso, tuttavia **lpstat** riporta che lo scheduler è ancora non attivo. In questa situazione, cancellate semplicemente il file **SCHEDLOCK** e fate ripartire **lpsched**; fate molta attenzione a non cancellare **SCHEDLOCK** quando **lpsched** è in esecuzione.

Il directory **requests** contiene un subdirectory per ognuna delle stampanti installate sulla macchina, e per le stampanti disponibili su sistemi remoti. Il comando **lp** trasferisce in uno di questi subdirectory il testo dei file accodati per la stampa; al termine della fase di stampa, il file viene cancellato da questa coda.

Il directory **/etc/lp** contiene informazioni relative alla specifica configurazione di stampanti della macchina, incluse tabelle delle stampanti esistenti, moduli, filtri e classi per esempio:

```
# cd /etc/lp
# ls -F
Systems     filter.table     interfaces/     printers/
```

```
classes/      filter.table.i  logs@       pwheels/
default       forms/          model@
#
```

Il file **default** contiene il nome della stampante di default usato quando nessuna delle opzioni **–d**, **–f**, **–T** è presente nella linea di comando **lp**. I moduli disponibili sono contenuti nel directory **forms**; le classi di stampanti configurate sono nel directory **classes**; il directory **printers** contiene strumenti usati per trattare ciascuna delle stampanti configurate; **Systems** contiene informazioni relative alla stampa con sistemi remoti; i file **filter.table** contengono informazioni necessarie per controllare i filtri configurati.

DRIVER DI STAMPANTE

Il device file associato con una stampante parallela è generalmente **/dev/lp**. Nelle release più vecchie di sistema UNIX il *device driver* che tratta i dati inviati a **/dev/lp** spesso aggiunge in uscita salti di pagina e qualche volta intestazioni, anche se dirigete l'uscita direttamente al device file. Oltre a **/dev/lp**, esiste spesso un secondo device file, denominato **/dev/rlp** (raw lp device), che non elabora l'uscita. Se volete modificare i dati in uscita prodotti da **/dev/lp**, potete modificare la procedura di interfaccia e assegnare l'uscita a **/dev/rlp** piuttosto che a **/dev/lp**. Può essere necessario procedere per esperimenti perché il comportamento di **/dev/lp** e **/dev/rlp** spesso differisce da sistema a sistema. Tenete anche presente che sia **/dev/lp** sia **/dev/rlp** indirizzano la stessa porta fisica di I/O sulla macchina, per cui non potete utilizzarli contemporaneamente senza stravolgere l'uscita. Le stampanti seriali che generalmente utilizzano una porta tty, come per esempio **/dev/tty01s**, non hanno un *raw device* associato, per cui tutto il trattamento dell'uscita deve essere eseguito nella procedura di interfaccia.

Quando collegate alla macchina un dispositivo di stampa a elevate prestazioni, questo può includere una piastra di interfaccia con connettore e cavo propri, per il collegamento alla stampante. Alcune stampanti laser o plotter o altri dispositivi grafici possono richiedere una piastra da includere nel sistema. Questi dispositivi non utilizzano né il modello parallelo né quello seriale, discussi in precedenza, e neppure i device file **/dev/lp** o **/dev/tty??**. La casa costruttrice fornisce un driver con questo hardware speciale, insieme alle istruzioni per installare l'hardware e il software. Se la stampante prevede l'installazione nel sottosistema **lp**, generalmente viene corredata anche di una procedura di interfaccia (modello) e viene creato un nuovo device file nel directory **/dev**. Utilizzando la speciale procedura di interfaccia e il nuovo device file, potete facilmente installare il dispositivo nel sistema **lp** usando il comando **lpadmin**. Può essere necessario creare o modificare la procedura di interfaccia, ma non esiste nessuna grande differenza concettuale rispetto alle normali stampanti. Potete collegare rapidamente a una macchina UNIX quasi tutti i tipi di unità di stampa mediante i programmi di gestione di **lp**.

Capitolo 14

Comunicazioni

Alcuni dei comandi più complessi del sistema UNIX interessano le *comunicazioni* tra utenti che condividono una macchina o che risiedono su macchine diverse. È opinione comune che UNIX disponga di alcuni tra gli strumenti più efficienti e affidabili presenti sul mercato nel campo delle comunicazioni di dati. Garantire il trasferimento dei dati sicuro, affidabile, senza errori, attraverso canali di comunicazione intrinsecamente inaffidabili e disturbati, come le linee telefoniche, è un potenziale problema e richiede soluzioni complesse. Il sistema UNIX fornisce strumenti che consentono di affrontare questo complesso argomento a vari livelli, dal semplice comando **mail** alla gestione sofisticata delle risorse e della sicurezza necessari nel sottosistema **uucp** e nelle reti locali (*local area network*) ad alte prestazioni. Gli strumenti di comunicazione in ambiente UNIX sono ritenuti complessi e di non facile interpretazione, e in un certo senso questa critica può essere accettata ma, in confronto al software di comunicazione disponibile in altri sistemi operativi, sono un esempio di chiarezza e di affidabilità. Infatti, è abbastanza frequente riscontrare una spiccata influenza di UNIX nei sistemi di comunicazione di altri ambienti di sistema, anche se i risultati finali sono sicuramente inferiori. Nel corso degli anni, questi strumenti del sistema UNIX sono stati costantemente migliorati nell'affidabilità, nelle funzioni supportate e nella sicurezza. Oggi, molti progettisti coinvolti nei problemi inerenti alle comunicazioni scelgono il sistema UNIX per risolverli, specialmente col crescente sviluppo delle reti locali, che rendono le comunicazioni sempre più importanti.

In questo capitolo tratteremo gli strumenti di comunicazione di base, che sono orientati principalmente verso i messaggi e i file ASCII; tratteremo anche di emulazione di terminali.

Gli argomenti di gestione relativi alle comunicazioni e il sottosistema di trasferimento file **uucp** sono trattati nel Capitolo 15; il Capitolo 24 tratta le reti locali.

14.1 Il comando news

Nei capitoli precedenti sono già stati introdotti i primi strumenti di base delle comunicazioni. Il primo tra questi è il comando **news**, che permette all'utente di leggere i messaggi "pubblicati" dal gestore di sistema. Quando lanciate il comando **news**, il sistema fornisce tutte le nuove informazioni create dal tempo della vostra ultima utilizzazione del comando, ma non quelle create prima di quel tempo. Un esempio del comando:

```
$ news

meeting (pat) Mon Jun 22 08:26:47 1990

    Lunedì alla 9.00 è fissata la riunione per la discussione del bilancio
    della società. Siate tutti puntuali. Grazie! – pat
$
```

Una volta che **news** ha visualizzato i messaggi, questi non sono più visualizzati lanciando nuovamente il comando, che termina in maniera silente:

```
$ news
$
```

I messaggi abitualmente hanno una lunghezza limitata; potete tuttavia interrompere l'uscita su video di un messaggio troppo lungo, premendo il tasto DEL. **news** interrompe il messaggio corrente e passa a visualizzare il messaggio successivo, se esiste. Se premete due volte il tasto DEL nell'arco di un secondo, interrompete l'esecuzione del comando **news** e restituite il controllo allo shell.

Il comando **news** dispone di diverse opzioni che ne variano il comportamento. L'opzione **–n** (name) induce **news** ad elencare solo il titolo dei messaggi non ancora visualizzati:

```
$ news –n
news: incontro a pranzo.
$
```

L'opzione **-s** (show) visualizza il numero di messaggi non ancora letti, senza indicare né titolo né contenuto:

```
$ news -s
3 news items
$
```

È conveniente utilizzare almeno una di queste opzioni nel vostro file **.profile**, in modo da essere informati se esistono nuovi messaggi, quando entrate nel sistema. L'opzione **-a** (all) induce **news** a visualizzare tutte le notizie diffuse, indipendentemente dal fatto che le abbiate già lette; può produrre in uscita una grande quantità di dati, soprattutto su macchine di grandi dimensioni. Tenete conto del fatto che l'opzione **-a** visualizza, oltre alle notizie non ancora lette, anche tutte le precedenti notizie diffuse nel sistema. Tutti gli altri argomenti nella linea di comando **news** vengono interpretati come i titoli, ovvero nomi, dei messaggi che vi interessa leggere.

```
$ news pranzo

pranzo (leo) Mon Jun 22 08:25:31 1990

    Non dimenticarti l'invito a pranzo. Cerca di esserci. -leo
$
```

Il comando **news** mantiene nel vostro home directory un file di lunghezza nulla di nome **.news_time**, con il tempo di modifica (visualizzato da **ls -l**) aggiornato all'ora dell'ultima esecuzione di **news**. Ogni volta che eseguite nuovamente questo comando, appaiono su video solo i file che hanno una data più recente rispetto a quella di **.news_time**, a meno che abbiate utilizzato l'opzione **-a**. Se il file **.news_time** non esiste, il sistema lo crea nel momento in cui lanciate il comando **news** per la prima volta.

Il directory **/usr/news** contiene i messaggi, ognuno dei quali costituisce un file distinto nel directory; il nome del file corrisponde al nome (o titolo) che identifica la notizia:

```
$ ls -l /usr/news
total 3
-rw-r--r--  1 leo     other      74 Jun 22 08:25 pranzo
-rw-r--r--  1 giorgio other     125 Jun 22 08:26 incontro
-rw-r--r--  1 pat     other      32 Jun 22 08:24 buongiorno
$
```

Il gestore di sistema è responsabile di solito della manutenzione di questo directory, cancellando le notizie obsolete. In questo caso, il directory **/usr/news** può essere accessibile in scrittura solo dal supervisore, ma deve essere accessibile in lettura ed esecuzione da parte di tutti gli utenti. Su al-

cuni sistemi il directory **/usr/news** è reso accessibile per tutti anche in scrittura, per cui ogni utente può creare un messaggio, scrivendo un file nel directory; in ogni caso i file creati nel directory devono essere necessariamente accessibili in lettura, altrimenti il comando **news** non può funzionare.

14.2 Il messaggio del giorno

UNIX fornisce un'altra funzione simile a **news**, il *messaggio del giorno*. Questo non è un comando, ma è una caratteristica disponibile sulla maggior parte dei sistemi. Il messaggio del giorno viene generalmente impiegato per messaggi che hanno una priorità superiore rispetto a quelli in **news**, come per esempio l'orario di chiusura del sistema o le ultime modifiche apportate al sistema. In pratica, il file **/etc/profile** di sistema include la linea di comando

```
cat /etc/motd
```

(dove **motd** indica "message of the day").

Diversamente da **news**, il contenuto del file **/etc/motd** appare automaticamente a video, perché il sistema esegue il comando **/etc/profile** ogni volta che un utente si collega alla macchina. I messaggi obsoleti contenuti nel file **motd** risultano particolarmente fastidiosi quando gli utenti si collegano ripetutamente. Il file **motd** deve essere accessibile in lettura da parte di tutti gli utenti, ma solo il supervisore ha la facoltà di aggiornarlo.

```
$ ls -l /etc/motd
-rw-r--r--  1 root     sys          1 Jun 6 1990 /etc/motd
$
```

14.3 Il comando write

Il comando **write** rende possibile la comunicazione diretta tra due utenti, inviando i messaggi direttamente ai loro terminali. La sua funzione è analoga a:

```
$ echo messaggio > /dev/altro-tty
```

dove *altro-tty* è il nome del terminale utilizzato dall'utente con cui intendete comunicare. Questi messaggi irrompono in qualunque visualizzazione sul terminale del ricevente. Il comando **write** è leggermente più sofisticato di quanto risulta dall'esempio precedente e fornisce diverse altre funzioni addizionali. Innanzitutto, accetta come argomento l'identificatore di login del ricevente, per cui potete scrivere a un altro utente senza la necessità di conoscere il terminale che utilizza.

```
$ write leo < messaggio
$
```

Il comando **write**, identifica il terminale ricevente, invia il messaggio e poi restituisce il controllo allo shell. Se su quel terminale non è collegato nessun utente, **write** visualizza sul terminale trasmittente un messaggio di errore:

```
$ write pat
pat is not logged in
$
```

Gli utenti possono disabilitare dal loro terminale questo tipo di comunicazione diretta scrivendo:

```
$ mesg n
$
```

La ricezione di messaggi può essere abilitata di nuovo con il seguente comando:

```
$ mesg y
$
```

Il comando **mesg** cambia i diritti di accesso sul device file che è associato al terminale. Se un utente disabilita la ricezione dei messaggi, **write** restituisce un particolare messaggio di errore:

```
$ write leo
Permission denied
$
```

L'accesso a **write** spesso viene disabilitato quando il tipo di uscita su terminale è di elevata qualità, come per esempio in alcune fasi di stampa.

UTILIZZO INTERATTIVO DI write

Poiché il comando **write** legge dallo standard input il messaggio da inviare, può essere usato in maniera più diretta. Se il comando è semplicemente

```
$ write pat
```

con nessuna ridirezione dell'input, **write** realizza il collegamento con l'utente ricevente e poi restituisce due segnali acustici per indicare l'apertura del colloquio. A questo punto, potete scrivere il messaggio direttamente da ta-

stiera e il comando **write** lo invia linea per linea sul terminale ricevente. Una volta stabilito, il collegamento permane fino a quando non battete un carattere di fine file (CTRL-D); fino a quel momento tutto quello che introducete al vostro terminale compare sul terminale ricevente. In ogni caso, il canale di comunicazione è a una sola via; il destinatario non può inviare nessun messaggio a voi, se non utilizzando a sua volta il comando **write**, specificando come argomento il vostro identificatore di login. In questo modo esistono due canali di comunicazione di **write**, da voi al destinatario, e viceversa. Ogni collegamento deve essere interrotto solo dall'utente che lo ha stabilito, per ripristinare completamente la situazione alla normalità.

Quando un utente utilizza **write** per chiamarvi, sul vostro terminale appare prima un'intestazione e di seguito il messaggio, linea per linea.

```
Message from pat on my_sys (console) [ Mon Jun 22 08:29:58 ] ...

ciao giorgio, ci sei oggi?
```

Dovete rispondere all'utente **pat** con **write**. Normalmente, si verifica un ritardo apprezzabile tra una linea e la successiva del messaggio in arrivo, ritardo dovuto in parte al tempo che occorre all'utente per scrivere il messaggio da tastiera e in parte alle operazioni che il sistema esegue su ogni linea del messaggio prima di spedirla. In una macchina caricata pesantemente, la comunicazione con **write** può risultare relativamente lenta; inoltre, la comunicazione può diventare confusa perché entrambi gli utenti possono inviare un messaggio contemporaneamente, oppure uno può rispondere prima che l'altro abbia completato il messaggio. Per questi motivi, tra gli utenti si è sviluppato un semplice protocollo di comunicazione che permette di utilizzare senza problemi il comando **write**, e che spieghiamo qui di seguito. Il mittente originale instaura un collegamento con il destinatario, inviando magari un messaggio di saluto. Il destinatario allora risponde, segnalando di essere pronto alla comunicazione. A questo punto, il mittente invia un messaggio multilinea che termina con la lettera **o** (over, in italiano, "passo"). Il destinatario attende di ricevere la **o** per essere sicuro che il messaggio è completo, quindi risponde con un messaggio di qualsiasi numero di linee e batte a sua volta la lettera **o** per segnalarne la fine. A questo punto il mittente riprende la parola e il colloquio può continuare quanto occorre. Una persona può decidere di terminare il colloquio con la sigla **o-o** oppure **oo** (over and out, cioè "passo e chiudo"); l'altra persona può continuare, fino a che decide a sua volta di inviare **o-o**. Infine, ambedue gli utenti devono battere CTRL-D per uscire da **write** e restituire il controllo allo shell. Ricordate: entrambi gli utenti devono chiudere **write** con CTRL-**d** per chiudere la rispettiva connessione. Questo protocollo, simile a quello in uso tra radioamatori e CB, non viene imposto dal programma **write**, ma è solo una convenzione adottata da molti utenti. Tuttavia, durante questa conversazione scritta bilaterale si può creare confusione, perciò nelle comunicazioni si preferisce maggiormente utilizzare il comando **mail** piuttosto che il comando **write**.

14.4 Il comando wall

Il comando **write** permette di comunicare con un solo utente, quello indicato come argomento sulla linea di comando. Potete avere necessità di spedire un messaggio a tutti gli utenti collegati alla macchina, per esempio per annunciare la chiusura del sistema, e consigliarli di chiudere la propria sessione prima che il sistema venga spento. Per non obbligarvi a usare un comando **write** per ciascun utente, il sistema UNIX prevede il comando UNIX **wall** (write all), collocato in **/usr/sbin/wall**. Tutti gli utenti possono lanciare questo comando, ma generalmente è il gestore di sistema che lo impiega con frequenza.

Come **write**, il comando **wall** legge dallo standard input il messaggio da inviare:

```
# wall < messaggio.file
```

Il messaggio viene inviato a tutti gli utenti che sono collegati alla macchina, mittente compreso; il messaggio è preceduto dalla stessa intestazione prodotta da **write**. Gli utenti possono rifiutare anche questo tipo di messaggi tramite il comando **mesg n**; non esiste una funzione che permetta ai destinatari di **wall** di rispondere al messaggio.

14.5 Il comando mail in dettaglio

Lo strumento usato più ampiamente per la comunicazione tra utenti è la funzione di posta elettronica (*electronic mail*), descritta brevemente nel Capitolo 2. La posta è usata da quasi tutti gli utenti, e sono disponibili parecchi programmi di posta elettronica, con caratteristiche e funzioni differenti; dal pacchetto semplice e facile, al sistema estremamente complesso e ricco di possibilità. Poiché il sistema UNIX è molto adatto alle comunicazioni, la posta elettronica è sempre stata una delle aree più fertili di sviluppi; di conseguenza, qualche volta le diverse versioni di UNIX dispongono di strumenti di comunicazione che differiscono in modo sensibile.

NOTA SULLE VERSIONI DEL SERVIZIO mail

Il programma standard di posta fornito con SVR4 è molto migliorato rispetto ai più vecchi equivalenti di System V. La versione SVR4, oltre ai file di testo in ASCII, può trasmettere file binari, come programmi eseguibili, o documenti prodotti da progrediti word processor; può creare messaggi multiparte, che contengono diversi componenti indipendenti; inoltre, sono state introdotte le convenzioni di indirizzamento della posta dei sistemi BSD UNIX e AT&T Mail. Queste nuove e migliorate caratteristiche possono

essere usate per trasmettere posta tra utenti di un singolo sistema SVR4, o tra utenti di differenti macchine SVR4 connesse tramite una rete locale. Tuttavia, la maggior parte degli altri programmi di posta può accettare solo messaggi di testo; cioè sono accettati solo normali file ASCII che possono essere accettati dagli editor standard. Questo significa che quando create messaggi per la posta, dovete avere cura di includere solo formati di dati accettati dagli strumenti di posta usati dal vostro corrispondente. Inoltre, e questo è molto più difficile, dovete essere sicuri che anche le macchine intermedie che possono avviare il messaggio nel cammino verso il destinatario, supportino tutte le caratteristiche di posta che voi usate. Per essere totalmente sicuri, dovete limitare la posta a destinatari che non conoscete bene, solo a semplici messaggi di tipo testo. Esistono altri strumenti che potete usare per trasferire file non-testo, a condizione che siate sicuri che il ricevente sia in grado di leggere il file da voi inviato.

In generale, non esiste una limitazione alla lunghezza dei messaggi, per cui potete trasferire per posta lunghi file di testo, documenti o archivi **shar**.

CONCETTI DI POSTA

Tutti i programmi di posta esaminano i messaggi in arrivo e li memorizzano in uno speciale contenitore o cassetta postale (*mailbox* o *inbox*) di proprietà del destinatario. La posta rimane nella cassetta fino a quando il destinatario legge il messaggio e decide di cancellarlo. Nella maggior parte dei sistemi, la posta in arrivo viene memorizzata nel directory **/var/mail**, dove ogni utente possiede un file, che costituisce la sua *mailbox*; è anche possibile creare mailbox addizionali per memorizzare e leggere la posta. Gli strumenti di posta concatenano i messaggi inviati a un utente in un singolo file, ma visualizzano i singoli messaggi separatamente quando l'utente legge la posta.

Il comando base di posta elettronica è **mail**: viene utilizzato sia per inviare i messaggi ad altri utenti sia per rintracciare i messaggi che gli altri utenti vi hanno inviato. I messaggi possono avere qualsiasi lunghezza; così, se necessario, possono essere trasferiti con sicurezza lunghi file di testo, documenti, o archivi **shar**.

INVIO DI POSTA

Per inviare posta, si usa il comando **mail** con un elenco di identificatori di utente destinatario, come argomenti nella linea di comando; il comando legge il messaggio da trasmettere dallo standard input. Quando più utenti sono presenti come argomenti della linea di comando, ognuno di loro riceve lo stesso messaggio, per esempio:

```
$ mail leo pat < messaggio.file
$
```

Inoltre, **mail** accetta lo standard input da terminale se non viene effettuata nessuna ridirezione; in questo caso, il carattere di fine file, CTRL-D, conclude il messaggio:

```
$ mail leo pat
questo è un breve messaggio di prova.
vediamoci a pranzo....giorgio
CTRL-D
$
```

Potete anche terminare il messaggio battendo il carattere . (punto) in una linea separata; questa convenzione impedisce di includere nei messaggi singole linee contenenti il punto.

INVIO DI FILE BINARI PER POSTA

Potete trasferire file non-testo tra sistemi SVR4, con il metodo di creare la posta con ridirezione dell'input; per esempio:

```
$ mail gino < binary.file
```

invia il file binario **binary.file** all'utente **gino**. Prima di usare questo metodo, dovete essere sicuri che il ricevente, e tutte le macchine che intervengono nella comunicazione, possono trattare file binari.

Alcune release di SVR4 richiedono l'abilitazione preventiva del modo posta binaria; se richiesto, questo viene trattato nel file **/etc/mail/mailsurr/**. Eseguite alcune prove di posta binaria, prima di fare affidamento su questo metodo.

LETTURA DELLA POSTA

Quando nuovi messaggi vengono consegnati nella vosta mailbox, lo shell vi avverte visualizzando il messaggio:

```
You have mail.
```

Questo messaggio appare dopo che il comando in corso ha terminato l'esecuzione, ma prima che lo shell visualizzi il prompt PS1. Potete ignorare il messaggio; lo shell non lo visualizza più fino a quando non arriva nuova posta. Quando eseguite il comando **mail** senza argomenti, il primo messaggio spedito in ordine di tempo viene estratto dalla mailbox e visualizzato:

```
$ mail
From leo Mon Jun 22 08:37 EDT 1990
```

```
non dimenticarti di restituirmi il libro... grazie. -leo
?
```

Se il messaggio in arrivo è binario (o dati non-testo qualsiasi), **mail** non lo visualizza, ma emette un messaggio di avviso, come nell'esempio:

```
$ mail
From root Mon Jun 22 08:37 EDT 1990
Content-Length: 31940

Message content is not printable: delete, write or save it to a file
?
```

Non è possibile visualizzare messaggi non-testo; dovete trasferirli in un file e usare l'applicazione appropriata per trattarli.

Una volta visualizzato il messaggio, **mail** attende ordini, come si può notare dalla presenza del prompt **?**. In particolare, dovete informare il comando se intendete salvare il messaggio, cancellarlo, passare al successivo, o intraprendere qualche altra azione. Per richiedere l'elenco completo delle azioni possibili battete il carattere **?** (help) in corrispondenza del prompt. (La Figura 2.1 elenca l'insieme delle azioni possibili.) Rispondete con **d** (delete) per cancellare il messaggio, + (next) per passare al messaggio successivo senza rimuovere quello corrente, **q** (quit) per uscire dal programma **mail**, **s** (save) per rimuovere il messaggio dalla mailbox e salvarlo in un file; esistono molte altre opzioni. Se passate a un nuovo messaggio, il comando **mail** lo visualizza e poi si mette in attesa di una ulteriore decisione visualizzando il prompt **?**; se non esistono più messaggi, **mail** termina l'esecuzione e restituisce il controllo allo shell.

L'opzione **s** accetta come argomento il nome del file che deve salvare il messaggio; se battete **s** senza argomenti, il comando utilizza per default il file **mbox** nel vostro home directory. Quando salvate un messaggio in un file esistente, **mail** lo inserisce in coda a quelli preesistenti, per cui un file può contenere più messaggi.

Per rileggere la posta salvata in altri file, il comando **mail** accetta come argomento l'opzione **-f** (file), seguita dal nome di un file, che specifica una mailbox diversa da quella di sistema. Questo comando

```
$ mail -f mbox
```

apre il file **mbox** invece che la mailbox di sistema. Se il file corrisponde a una mailbox creata con l'opzione **s** durante la lettura della posta, allora tutte le funzioni del programma **mail** operano sul file come sulla normale mailbox.

STRUTTURA DEI MESSAGGI DI POSTA

Le prime linee di ogni messaggio contengono un'intestazione con informazioni che ne identificano il tragitto, come un timbro postale (*postmark*). Queste linee si riducono a una per i messaggi inoltrati da utenti collegati alla stessa macchina del destinatario; se il messaggio arriva da un'altra macchina, questo "timbro" contiene una linea per ogni passo che il messaggio ha compiuto per giungere alla mailbox del destinatario. Questa intestazione non viene creata dal mittente, ma viene aggiunta dal programma **mail** e può risultare di notevole utilità, come vedremo tra breve.

Dopo l'intestazione "timbro", e prima del corpo del messaggio, compaiono anche altre intestazioni con informazioni che generalmente hanno una struttura del tipo:

nome: valore

dove *nome* indica il nome della specifica intestazione e *valore* il suo contenuto. I programmi **mail** possono inserire numerose intestazioni, con le voci **To:** (destinatario), **From:** (mittente), **Date:** (data) e altre informazioni, per un totale anche di 20 o 30 linee. Anche queste intestazioni non vengono di solito create dal mittente, ma inserite da **mail** prima di inoltrare il messaggio.

CREAZIONE DI MESSAGGI CON INTESTAZIONE

Il programma **mail** della versione standard SVR4 genera sempre l'intestazione **Content-Length:**, che fornisce la dimensione (in byte) del messaggio che segue; inoltre può produrre due intestazioni opzionali, non inserite per default, la linea **To:** e la linea **Message-Type:**. L'opzione **–t** nella linea di comando quando create il messaggio aggiunge l'intestazione **To:**, per esempio:

```
$ mail –t giorgio leo
```

In questo caso il messaggio, oltre alle usuali informazioni di riconoscimento, include una linea **To:** che elenca i destinatari. Questa indicazione può rivelarsi utile quando il messaggio viene inoltrato a più utenti, e vi interessa far conoscere a ogni destinatario i nomi degli altri destinatari. Nella linea di comando può essere usata anche l'opzione **–m** seguita da un tipo di messaggio; il tipo di messaggio è definito dall'utente, non coincide col tipo di contenuto.

Potete anche creare una intestazione **Subject:** introducendola direttamente come prima linea del testo del messaggio, per esempio:

```
Subject: segue qui il testo del soggetto
```

Il programma **mail** sposta la linea **Subject:** dal corpo del messaggio nelle intestazioni.

VISUALIZZAZIONE SOMMARIA DEL CONTENUTO DI MAILBOX

La struttura di un messaggio in arrivo comprende, oltre al corpo del messaggio, alcune informazioni di instradamento e intestazioni composte di più voci. Potete chiedere al programma **mail**, specificando l'opzione **-h** (headers), di visualizzare solo il sommario delle voci di intestazione, per poter conoscere il contenuto della mailbox senza dover esaminare completamente ciascun messaggio.

```
$ mail -h
```

In alternativa, potete specificare l'opzione **h** anche in risposta al prompt **?** di **mail**, per esempio:

```
? h
2 letters in /var/mail/giorgio, 0 scheduled for deletion, 2 newly arrived
> 2     250   pat             Tue Jun 23 14:37 EDT 1990
  1     556   leo             Mon Jun 22 08:37 EDT 1990
?
```

I messaggi vengono elencati in ordine inverso, per cui il più recente diventa il primo messaggio della lista. Il messaggio corrente è contrassegnato, all'estrema sinistra, dal carattere > e molti dei comandi disponibili, come **s** e **d**, operano in modo implicito sul messaggio corrente. Ogni messaggio viene identificato nella lista con un numero nella colonna dopo >; potete richiedere la visualizzazione di un messaggio rispondendo al prompt **?** di **mail** con il corrispondente numero di messaggio nella lista. Per esempio, il seguente comando

```
? 2
```

permette di visualizzare il messaggio numero 2. Inoltre, il numero di messaggio può anche essere indicato di seguito ad altri comandi. Per esempio, per cancellare il messaggio numero 1, scrivete

```
? d 1
```

indipendentemente da quale sia il messaggio corrente.

Gli altri dati nell'elenco indicano rispettivamente il nome del mittente, la data e l'ora di creazione del messaggio; informazioni che si rivelano utili soprattutto se la posta in arrivo è numerosa.

I messaggi cancellati non vengono eliminati fisicamente dalla mailbox

finché il programma **mail** rimane in esecuzione; vengono solo marcati per la cancellazione e non appaiono più nella lista di intestazioni. Potete ancora leggere i messaggi cancellati fino a quando non uscite dal programma **mail**; solo a seguito di questa operazione vengono fisicamente rimossi dalla mailbox e perduti. Il comando **h a** (headers of all messages) visualizza tutti i messaggi, anche quelli cancellati (marcati).

```
? h a
2 letters found in giorgio, 1 scheduled for deletion, 0 newly arrived
    2    250    pat          Tue Jun 23 14:37 EDT 1990
>   1  d 556    leo          MonJun 22 08:37 EDT 1990
?
```

Il messaggio numero 1 è stato marcato per la cancellazione, come indica la presenza della lettera **d**; potete recuperarlo prima di uscire dal programma **mail**, con

```
? u 1
```

(undelete). Viene recuperato il messaggio indicato; se non viene indicato alcun numero di messaggio, viene recuperato il messaggio corrente.

RISPOSTA A UN MESSAGGIO RICEVUTO

Se, quando ricevete un messaggio, intendete rispondere, selezionate l'opzione **r** (reply), al prompt **?** del comando **mail**. Il comando **r** determina dal messaggio ricevuto il nome e l'indirizzo di mail dell'utente che ha spedito il messaggio e genera un nuovo processo **mail** per inoltrare la vostra risposta. A conclusione del testo della risposta, il programma **mail** che inoltra il messaggio termina l'esecuzione e restituisce il controllo al programma **mail** di lettura della posta. In ogni caso, l'operazione di risposta può risultare complicata, per cui dovete usarla con molta attenzione, specialmente se il messaggio è stato spedito da una macchina remota. Il processo di interpretazione delle informazioni di instradamento del messaggio si dimostra qualche volta molto difficile, e l'opzione **r** non sempre agisce come dovrebbe, per cui può accadere che siate costretti a inoltrare manualmente il messaggio, come spiegato nella sezione successiva.

INDIRIZZI DI POSTA DI ALTRI UTENTI

Nei precedenti esempi, come destinatario del messaggio abbiamo un utente collegato alla stessa macchina del mittente.

```
$ mail leo pat
```

Anche in questo caso, **leo** e **pat** devono essere identificatori di login di due utenti collegati alla vostra macchina. La funzione **mail** permette di inoltrare messaggi a utenti collegati anche su altre macchine UNIX; allora si usa il termine posta remota (*remote mail*), perché il destinatario è in collegamento remoto con la vostra macchina. Potete anche inoltrare i messaggi a un utente che lavora su una macchina diversa dalla vostra, attraverso una o più macchine intermedie; allora si usa il termine *multi-hop mail*. Questo meccanismo trova applicazione specialmente nelle reti locali; inoltre, diverse reti internazionali di utenti di sistemi UNIX comunicano in tutto il mondo mediante un servizio di questo tipo. Esistono vari formati di indirizzamento per supportare i differenti tipi di reti.

La forma più semplice di indirizzamento consiste nello specificare il percorso di instradamento completo, dalla vostra macchina a quella ricevente. Con questa forma, l'indirizzo di posta di utenti remoti è espresso nella linea di comando **mail** specificando l'identificatore univoco della macchina, seguito dall'indicatore univoco di utente destinatario, in quella macchina. Il carattere ! separa le due voci e non deve essere inserito alcuno spazio:

```
$ mail altro!utente
```

Potete specificare più destinatari, con lo stesso formato:

```
$ mail altro!utente amico!remoto
```

dove **altro** e **amico** sono gli identificatori delle macchine e **utente** e **remoto** sono gli identificatori degli utenti a cui intendete inoltrare il messaggio. Il cammino di instradamento della posta può coinvolgere più macchine, prima di raggiungere il destinatario:

```
$ mail altro!utente!remoto
```

In questo caso il messaggio viene prima inviato alla macchina **altro**, dove il programma **mail** lo riceve, e lo inoltra alla macchina di nome **utente**, dove viene consegnato all'utente **remoto**. Questa modalità di indirizzamento permette di specificare fino a 20 riferimenti intermedi per raggiungere il destinatario.

DOMINIO DI INDIRIZZI

Può essere usato un secondo formato di indirizzamento, con sistemi di mail che lo supportano. Questo stile diverso viene detto indirizzamento per dominio; consente di usare un indirizzo *logico*, anziché l'instradamento completo fino alla macchina di destinazione. Di solito, con questo metodo la posta viene inviata a un processo *mail server* su una macchina *gateway*; il pro-

cesso esamina una tabella di indirizzi noti, e invia il messaggio a una macchina indicata come obiettivo di quell'indirizzo. La macchina obiettivo ripete il procedimento fino a che il messaggio non raggiunge la destinazione finale. Il programma **mail** standard di SVR4 può trattare l'indirizzamento per dominio, ma alcuni sistemi più vecchi non lo possono.

Con l'indirizzamento per dominio, il messaggio viene inviato a un utente a un dominio destinazione, usualmente nella forma:

```
utente@gateway.dominio
```

dove l'identificatore di utente è seguito dal carattere @ (chiocciola), poi dalla macchina gateway del dominio, poi dal carattere . (punto), e infine dal nome del dominio, per esempio:

```
$ mail edigeo@milano.com
```

Talvolta occorre aggiungere nomi di altre macchine dopo @, per specificare una macchina nel gateway, per esempio:

```
$ mail edigeo@milano.centro.com
```

In questo caso, **centro** può essere una *sottorete*, mentre **milano** può essere un gateway, o la macchina obiettivo. L'indirizzamento per dominio può risultare molto complesso; dovete curare di ottenere dai vostri corrispondenti gli indirizzi esatti. Alcuni indirizzi possono fare uso di % invece di uno dei punti, oppure del punto invece di @.

Sono stati definiti solo pochi domini, mentre in un dominio possono esistere molti gateway, e un gateway può supportare molte singole macchine. I domini più comuni sono: **com** (commercial), **edu** (education), **gov** (government).

Con l'indirizzamento per dominio, il sistema di mail (o gateway server) esamina un database locale di nomi di dominio per *risolvere* l'indirizzo logico e trovare la via più efficiente per passare il messaggio alla macchina successiva. In SVR4 questo database si trova nel file **/etc/mail/mailsurr** (mail surrogate); modificarlo correttamente può risultare molto complicato. Consultate il gestore della vostra rete prima di modificare **mailsurr** o altri file relativi ai domini.

L'indirizzamento nello stile con punti esclamativi si basa sul sottosistema di comunicazione dati **uucp**, trattato nel Capitolo 15. Tenete ben presente che non potete spedire posta a una macchina remota, se questa non è nota al vostro sistema **uucp** o al vostro database di dominio, anche se può essere possibile ricevere posta da quella stessa macchina. Quindi, se non riuscite a spedire posta col vostro sistema in remote-mail o multi-hop, la causa si trova nell'elenco di macchine remote note a **uucp**, o nel database di dominio. Con la posta in multi-hop, ciascuna macchina che interviene nell'instra-

damento deve conoscere la successiva macchina nel percorso di invio; dovete essere molto sicuri che il percorso sia valido prima di spedire un messaggio, altrimenti potrebbe non essere consegnato a destinazione. Infatti, una macchina intermedia può scartare un messaggio, senza restituire una comunicazione di errore al mittente originale, se la successiva macchina nell'instradamento risulta sconosciuta.

INOLTRO DI POSTA

Potete richiedere che la posta diretta a voi sia inoltrata a un altro indirizzo. Il sistema **mail** supporta il concetto dell'indirizzo di rispedizione, o inoltro, ovvero la possibilità per la posta in arrivo di non essere necessariamente consegnata al destinatario previsto, ma rispedita a un altro indirizzo. Nei sistemi SVR4, il comando **mail** fornisce l'opzione **-F** (forward) per specificare l'indirizzo di inoltro. Potete usare questa opzione, solo se la vostra mailbox è vuota; dovete aver cancellato o salvato altrove la posta contenuta nella vostra mailbox. Il comando

```
$ mail -F nuovo!indirizzo
```

stabilisce l'inoltro della posta a **nuovo!indirizzo**. Tutte le modalità di indirizzamento di messaggi, già trattate, si applicano anche in questo caso. Per esempio, potete specificare sia un destinatario locale

```
$ mail -F leo
```

che un indirizzo remoto

```
$ mail -F altro!giorgio
```

La prima forma del comando **mail** viene impiegata quando utilizzate più identificatori di login sulla stessa macchina o quando qualche altro utente evade la vostra posta, mentre la seconda forma trova impiego quando disponete di identificatori di login su più macchine, e uno è preferito. La maggior parte degli utenti inoltra i messaggi dall'identificatore di login meno utilizzato alla macchina o all'identificatore di login utilizzato più di frequente.

Una volta stabilito, l'inoltro della corrispondenza permane indefinitamente e tutta la posta in ingresso viene rinviata all'indirizzo specificato. Per ritornare alla situazione originale, specificate un argomento nullo dopo l'opzione **-F**:

```
$ mail -F ""
```

La coppia di virgolette, non separate da alcuno spazio, induce lo shell a riservare un argomento vuoto per l'opzione **-F**; questo disattiva l'operazione di inoltro, per cui la corrispondenza in arrivo viene di nuovo depositata nella vostra mailbox originale.

Quando la corrispondenza viene inoltrata non può essere letta, ma è possibile accertare se l'operazione procede:

```
$ mail -F leo
Forwarding to leo
$ mail
Your mail is being forwarded to leo
$
```

Nelle versioni precedenti a SVR3, il comando **mail -F** non esiste, per cui dovete utilizzare un meccanismo diverso per inoltrare la corrispondenza. Innanzitutto, cancellate o salvate altrove tutta la vostra posta, inviatevi poi un messaggio fittizio, infine entrate in editor sul file **/usr/mail/login**, dove **login** corrisponde al vostro identificatore di utente. Una volta nell'editor, cancellate l'intero contenuto del file e poi aggiungete all'inizio del file la linea

```
Forward to indirizzo
```

La lettera F maiuscola a inizio della linea è obbligatoria, **indirizzo** indica l'indirizzo di inoltro, come già visto. A questo punto scrivete il file e poi uscite dall'editor. L'operazione di inoltro rimane abilitata fino a quando riscrivete il file, cancellando la linea "Forward to...". Non create manualmente il file, ma usate **mail** per spedire un messaggio a voi stessi. Questo assicura che il mailbox venga creato con i diritti di proprietà e di accesso corretti, in modo che **mail** possa trattarlo correttamente in seguito.

RISPOSTA AUTOMATICA A POSTA IN ARRIVO

Se non siete presenti al sistema per qualche motivo, come una vacanza, potete configurare **mail** di SVR4 per rispondere automaticamente ai messaggi in arrivo con una risposta di riserva. La vostra posta viene conservata, e i vostri corrispondenti ricevono una risposta informativa immediata; potete definire il testo della risposta, o usare una risposta di default prevista dal sistema. Quando rientrate, potete ripristinare il normale servizio di mail. Usate il comando **vacation** per stabilire il servizio di risposta automatica, come segue:

```
$ vacation
Forwarding to |/usr/lib/mail/vacation2 -o %R
$
```

Dovete avere prima cancellato o salvato tutta la vostra posta, altrimenti **vacation** segnala la mancanza, per esempio:

```
$ vacation
mail: Cannot install/remove forwarding without empty mailfile
$
```

Se non gradite il messaggio predisposto, indicato in man page **vacation(1)**, potete creare un file col vostro messaggio alternativo e specificarlo con l'opzione **-M** (Message), per esempio:

```
$ vacation -M mia.risposta
```

Per default, **vacation** salva i vostri messaggi in arrivo nel file **$HOME/.mailfile**; potete cambiare questo file specificando il nome di un altro file con l'opzione **-m** (mailfile) di **vacation**. Al vostro ritorno, potete leggere il file riservato con la linea di comando:

```
$ mail -f $HOME/.mailfile
```

Ricordate anche di ristabilire il normale servizio di **mail**, con:

```
$ mail -F ""
```

Come avrete capito, **vacation** è una forma intelligente di rinvio della posta.

IL COMANDO rmail

Esiste un altro programma eseguibile associato con la funzione di **mail**, che però non dovete mai lanciare direttamente. Si tratta del comando **rmail** (remote mail) che viene utilizzato internamente nelle operazioni per spedire la corrispondenza; deve essere presente nel directory **/usr/sbin** perché il sistema possa funzionare correttamente. Il comando **rmail** tratta la risoluzione degli indirizzi di dominio, e l'accodamento della posta per **uucp** o sistemi mail basati su LAN.

14.6 Emulazione di terminale con il comando cu

Il sistema UNIX dispone di un programma di *emulazione di terminale* denominato **cu** (call up). Questo comando viene spesso chiamato "call UNIX" dagli utenti, ma il suo uso non è limitato al collegamento con un altro sistema UNIX; **cu** consente collegamenti con quasi ogni altro sistema che permetta comunicazioni asincrone ASCII, includendo macchine MS-DOS, e converti-

tori di protocollo che convertono da terminali tipo IBM 3270 al formato asincrono ASCII. Naturalmente, il programma **cu** consente funzioni addizionali se anche la macchina remota lavora in ambiente UNIX.

Il comando **cu** fa parte del sistema di trasferimento di dati **uucp** e utilizza i file di controllo **uucp**, e qualsiasi modem esterno o integrato collegato alla macchina. Di solito per utilizzare le funzioni di emulazione di terminale, è necessario avere un modem a chiamata automatica; sui sistemi più semplici è possibile comporre manualmente il numero, a patto che venga prima eseguito il comando **cu**, e quindi composto il numero.

L'esecuzione di **cu** richiede il nome di una macchina come argomento:

```
$ cu remota
```

Il comando **cu** esamina i file di controllo **uucp** per determinare come stabilire un collegamento con la macchina specificata e poi chiama quella macchina. La chiamata può essere effettuata su linee telefoniche commutate, o su alcuni tipi di rete locale. I parametri di comunicazione (velocità di linea, ecc.) e le procedure di shell di chiamata si devono trovare nel database di **uucp**, altrimenti **cu** non funziona.

Quando il collegamento viene stabilito, **cu** restituisce un messaggio:

```
$ cu remota
Connected
```

Se la connessione non avviene, **cu** restituisce un messaggio di errore tra quelli stessi previsti dal sistema **uucp**. Una volta effettuato il collegamento, **cu** attende i dati in ingresso dalla vostra tastiera e li trasferisce immutati alla macchina remota, dove potete iniziare un login od ogni altra adeguata operazione. Tutti i caratteri inviati dalla macchina remota alla macchina locale arrivano, tramite **cu**, al vostro terminale; se sulla macchina remota utilizzate la stessa variabile TERM che usate sulla macchina locale, le funzioni di gestione del cursore sulla macchina remota funzionano come previsto.

Inoltre, **cu** permette di effettuare la connessione remota indicando il numero di telefono del sistema remoto; ovviamente questo formato funziona solo se il sistema locale dispone di un modem a chiamata automatica. Questo comando

```
$ cu 5559876
```

esegue la chiamata telefonica del numero indicato. Il numero di telefono può avere una lunghezza qualsiasi, ma deve essere composto solo da cifre e da alcuni caratteri che hanno uno speciale significato. Il carattere – (meno) segnala un ritardo di circa 4 secondi, per cui il comando **cu** può chiamare attraverso reti telefoniche che hanno ritardi di connessione o che usano commutatori con alta latenza. Il carattere = (uguale) fa attendere un secon-

do tono di linea nel punto in cui compare il carattere. Per esempio, in uscita, un centralino può richiedere di comporre un numero (in questo caso il 9), per ottenere la linea esterna; dopo avere atteso il tono di linea, può essere composto il numero di selezione intercomunale o internazionale, quindi il numero di abbonato; dall'altro lato, un PBX remoto può rispondere con un altro tono di linea, e voi potete comporre un numero interno. Tutte queste operazioni possono essere ottenute, con il seguente comando **cu**:

```
$ cu 9=01-9336=8453
```

Numeri di telefono complessi devono essere verificati per essere sicuri che funzionino correttamente, in quanto le sequenze di toni e di ritardi di solito devono essere stabilite caso per caso.

USCITA DA UNA SESSIONE cu

Dopo che avete completato la vostra sessione di emulazione di terminale, uscite normalmente (logout) dalla macchina remota. Dopo di questo dovete segnalare a **cu** di chiudere la linea, e ritornare a shell, usando il comando ~. (tilde punto). Il comando deve essere introdotto su una linea separata, subito dopo un ritorno a capo, e deve essere seguito a sua volta da un altro ritorno a capo.

```
$ cu remote
Connected

~[uname].
Disconnected
$
```

Dopo che avete battuto il comando, **cu** visualizza il nome della macchina locale tra parentesi quadre; questo è un esempio di *comando interno*, che descriveremo tra breve. Dovete uscire correttamente dalla macchina remota prima di battere tilde-punto perché alcune macchine remote non rispondono correttamente quando la linea di comunicazione si interrompe bruscamente. Non esiste altro modo per terminare una sessione **cu**, se non staccando il cavo dalla porta di comunicazione, perché tutti i caratteri vengono passati alla macchina remota, incluso il carattere di *job control* CTRL-Z.

OPZIONI DELLA LINEA di COMANDO cu

Per default, il comando **cu** presuppone che il collegamento sia *full-duplex* e che il sistema remoto fornisca l'eco dei caratteri. La velocità in baud del collegamento viene determinata dai file di dati di **uucp**, mentre viene assun-

ta la velocità di 1200 baud come valore predefinito quando è specificato un numero telefonico; potete cambiare questi valori selezionando le relative opzioni nella linea di comando **cu**. L'opzione **–s** (speed) accetta come argomento l'indicazione della velocità, che può essere 300, 1200, 2400, 4800 o 9600 baud. Questo comando

```
$ cu –s 2400 2345678
```

effettua una chiamata telefonica e imposta la velocità di comunicazione a 2400 baud. L'opzione **–s** sostituisce l'eventuale valore presente nel file di dati di **uucp**. Il seguente comando

```
$ cu –s 9600 sistema
```

esegue il collegamento con la macchina **sistema** a 9600 baud. Ovviamente questi valori devono corrispondere alle caratteristiche hardware del vostro sistema e di quello remoto.

Se il sistema remoto non genera l'eco dei caratteri inviati, potete utilizzare l'opzione **–h** (half-duplex), che fa visualizzare immediatamente tutti i caratteri battuti da terminale. Non si tratta di un vero canale di comunicazione "half-duplex" perché i caratteri battuti da tastiera vengono inviati immediatamente alla macchina remota anche se è in corso un trasferimento di caratteri alla vostra macchina. Altre opzioni disponibili in **cu** sono **–o** (odd) o **–e** (even) che attivano il controllo di parità dispari o pari durante il trasferimento dei caratteri alla macchina remota. Per default, il sistema non utilizza alcun controllo di parità.

Utilizzate l'opzione **–l** (line) per specificare una particolare porta **tty** da utilizzare nelle comunicazioni. Questa opzione è utile per sostituire i dati predefiniti di **uucp** o quando un dispositivo di comunicazione è collegato direttamente a una porta seriale. Per esempio, se un modem intelligente con un database di numeri telefonici è collegato alla porta COM2, il seguente comando permette l'accesso diretto al modem tramite **cu**:

```
$ cu –l /dev/tty01s
```

Quando il modem risponde alzando il segnale *carrier detect*, **cu** restituisce il messaggio **Connected** e i caratteri introdotti da tastiera vengono letti dal modem. Potete riprogrammare il modem o utilizzare i suoi comandi predefiniti per effettuare una chiamata esterna. Potete utilizzare l'opzione **–l** anche quando una macchina è collegata a un'altra con linea diretta su una porta seriale.

L'opzione **–d** (debug) induce **cu** a visualizzare una traccia delle operazioni che esegue quando realizza un collegamento; aiuta a diagnosticare i problemi connessi con i collegamenti che **cu** cerca di realizzare, e permette anche di acquisire familiarità con il comando **cu**. Tratteremo l'argomento *de-*

bug nel Capitolo 15, perché in realtà interessa il sottosistema **uucp**. Inoltre, **cu** prevede molte altre opzioni, ma la loro utilità è sicuramente inferiore a quella delle opzioni descritte.

14.7 Approfondimenti

Il comando **cu** ha ulteriori caratteristiche, oltre a quelle esaminate finora. La sua notevole potenza si basa sui suoi comandi interni e sulla capacità di leggere e scrivere l'output di comandi al sistema remoto, attraverso la rete di comunicazione. Tratteremo queste caratteristiche di **cu** e successivamente altre funzioni addizionali di **mail**.

I COMANDI INTERNI DI cu

Il comando **cu** dispone di alcuni *comandi interni* che ne accrescono le potenzialità. Per lanciare questi comandi, dovete far precedere il loro nome dal carattere ~ (tilde) (all'inizio della linea, dopo un ritorno a capo) per segnalare a **cu** che il comando che segue è un comando interno. Il programma **cu** interpreta in tal senso il resto della linea e non la trasferisce alla macchina remota. Per esempio, perché **cu** restituisca il controllo allo shell, scrivete il comando ~. (tilde punto), già trattato in precedenza. Quando **cu** riconosce il carattere tilde, lo visualizza immediatamente sul terminale, quando incontra il successivo carattere punto, visualizza il nome del sistema tra parentesi quadre, seguito dal carattere punto. Il nome del sistema viene sempre visualizzato per tutti i comandi interni, però solo dopo che il programma **cu** ha riconosciuto il carattere che segue il carattere tilde.

Esistono molti altri comandi interni. Per esempio, potete temporaneamente passare il controllo a un subshell locale con il comando ~! (tilde punto esclamativo) e, in questo ambiente, eseguire qualunque comando, senza interrompere il collegamento con la macchina remota. Quando avete terminato il lavoro con lo shell locale, lo disattivate normalmente con CTRL-D oppure con il comando **exit**; a questo punto, **cu** visualizza il carattere ! per segnalare di avere il controllo, e riprende a spedire alla macchina remota i caratteri introdotti da tastiera.

```
~[my_sys]!
$ echo $HOME
/home/giorgio
$ exit
!
```

Quando il programma **cu** riprende il controllo, non ripristina lo schermo video, perché inizia immediatamente a trasferire i caratteri alla macchina re-

mota; operazioni di "rinfresco" del video locale devono essere trattate nella macchina remota.

Potete lanciare un comando sulla macchina locale specificando **~!cmd**, dove *cmd* è una linea di comando o un pipeline qualsiasi. Il programma **cu** sospende temporaneamente la comunicazione al sistema remoto ed esegue la linea di comando in un subshell locale; quando il comando termina, **cu** visualizza il carattere ! e ritorna in azione.

TRASFERIMENTO DI FILE ASCII CON cu

Se la comunicazione avviene tra due sistemi UNIX, il programma **cu** dispone di alcuni comandi interni che permettono di trasferire dati in entrambe le direzioni. Si tratta dei comandi **~%put** e **~%take** (tilde percento, seguiti da put o take), che trasferiscono rispettivamente un file dalla macchina locale alla macchina remota e viceversa. Questi comandi accettano come argomento il nome di un file:

```
~[my_sys]%put my.file
```

La macchina remota deve essere in attesa al prompt PS1 perché il trasferimento di dati abbia successo. L'esempio precedente crea un file di nome **my.file** sulla macchina remota in cui copia un file locale di nome **my.file**. Durante la fase di trasferimento, **cu** visualizza una cifra progressiva sulla macchina locale a ogni trasferimento di un blocco di 1000 caratteri. Quando il procedimento di copiatura è completato, **cu** visualizza il numero complessivo di byte trasferiti e un messaggio finale.

```
~[my_sys]%put my.file
stty -echo;(cat -> my.file)¦¦cat -> /dev/null; stty echo
1234+
247 lines/4257 characters
$
```

La prima linea dell'esempio indica la struttura esatta del comando **put** (o **take**), che **cu** invia al sistema remoto UNIX. L'opzione **echo** è disattivata e lo standard input è ridiretto al file specificato (**my.file** in questo caso); al termine dell'esecuzione del comando **cat** viene riattivato **echo**. Il programma **cu** sulla macchina locale trasferisce alla macchina remota il file **my.file**, visualizza un numero progressivo (1, 2, 3 e poi 4) a ogni trasferimento di un blocco di dati, visualizza il conteggio completo di byte e linee trasferite, infine visualizza il prompt PS1 sulla macchina remota. Analoghe azioni, all'inverso, vengono intraprese se viene lanciato il comando **take**.

Potete cambiare il nome del file trasferito tra la macchina locale e la mac-

china remota, indicando un secondo nome di file come argomento del comando **put** o **take**:

```
~[my_sys]%take from.file to.file
```

In questo caso, **from.file** viene copiato dalla macchina remota alla macchina locale in **to.file**. La linea di comando **put** ha una struttura analoga:

```
~[my_sys]%put from.file to.file
```

dove **from.file** è il nome del file locale, mentre **to.file** è il nuovo file creato sulla macchina remota.

Tenete presente che il programma **cu** non esegue alcun controllo sugli errori che si possono verificare durante le operazioni di trasferimento dei file, per cui dovete sempre controllare che il file creato abbia la stessa dimensione del file sorgente. Il comando **wc** consente un controllo di questo tipo; tuttavia, **cu** qualche volta sostituisce nel file copiato i caratteri di tabulazione presenti nel file sorgente con un adeguato numero di caratteri spazio, per cui generalmente le sole informazioni attendibili di **wc** sono il conteggio delle linee e delle parole, mentre il conteggio dei caratteri può differire nei due file.

TRASFERIMENTO DI FILE BINARI CON cu

Il programma **cu** è destinato solo all'emulazione di terminale ASCII. Può trasferire file ASCII da una macchina a un'altra, ma non può trasferire direttamente file binari o altri file non di testo. Il programma **cu** non dispone di nessun protocollo di trasferimento di file con verifica di errore, come per esempio KERMIT o XMODEM; per trasferire file binari, si ricorre generalmente al sistema **uucp**. Il comando **cu** consente tuttavia l'esecuzione di un programma su entrambi i lati del collegamento di dati, ridirigendo l'ingresso e l'uscita attraverso il canale di comunicazione ASCII realizzato. In questo modo, potete aggiungere a entrambi i lati della connessione un'applicazione che permetta di convertire i dati binari nella forma ASCII, di trasferirli all'altro lato e poi di riconvertirli in forma binaria.

Un comando interno di **cu** supporta questa funzione, si tratta di **~$cmd** (tilde dollaro), dove **cmd** indica una linea di comando qualsiasi. La linea di comando *cmd* viene eseguita sulla macchina locale, ma l'output di *cmd* viene trasferito al sistema remoto lungo la linea di comunicazione, invece di essere visualizzato sullo schermo locale. Questo consente di definire una procedura (per esempio una procedura di login) sulla macchina locale che permette di guidare la macchina remota. Oppure, potete sviluppare un'applicazione di trasferimento di file, in due parti, eseguendone una parte sul sistema remoto, e utilizzando il comando **~$** per lanciare sulla macchina locale la parte corrispondente.

Esaminiamo lo scenario nei dettagli. L'applicazione remota legge lo standard input, in modo da ricevere i dati attraverso il canale di comunicazione; quindi il comando ~$ (tilde dollaro) trasmette i risultati dell'esecuzione del comando locale *cmd*, che vengono letti dall'applicazione remota; l'applicazione remota risponde con la stringa ~>:***filename*** in cui *filename* indica il nome di un file della macchina locale. Dopo questo comando, tutti i dati inviati dalla macchina remota vengono ridiretti al file. Quando il sistema remoto spedisce una linea che consiste solo di ~>, il file *filename* viene chiuso e **cu** inizia a visualizzare su terminale i dati ricevuti. Si tratta quindi di un meccanismo di uso generale per il trasferimento di dati tra due macchine. I programmi applicativi supportati dalle due macchine eseguono la conversione dei dati in formato ASCII e ricostruiscono il formato binario originale dopo la trasmissione. Potete provare l'uso di **uuencode/uudecode** oppure **btoa/atob** per la conversione tra dati ASCII e binari, e viceversa; di solito nelle macchine SVR4 è presente una di queste coppie.

ALTRI COMANDI INTERNI DI cu

Il programma **cu** prevede molti altri comandi interni. Per esempio, il comando ~%**cd** cambia il directory corrente della macchina locale nel nuovo directory indicato come argomento del comando **cd**:

```
~[my__sys]%cd /usr/src
```

Questo comando cambia il directory corrente visto da **cu**, invece il seguente comando

```
~[my__sys]!cd /usr/src
```

non ha lo stesso, perché viene eseguito da un subshell e non direttamente dal programma **cu**. Quando il subshell termina, **cu** ha lo stesso directory corrente che aveva prima dell'esecuzione del comando ~!.

Infine, il comando ~%**b** invia un segnale di break alla macchina remota. In molti sistemi questo comando non è necessario, in quanto lo stesso programma **cu** riconosce il tasto BREAK e può così inviare correttamente il segnale break alla macchina remota. Il comando ~%**d** attiva o disattiva, alternativamente, l'output diagnostico; i comandi ~%**ifc** (input flow control) e ~%**ofc** (output flow control), attivano o disattivano, alternativamente, il controllo di input e output mediante i tasti CTRL-S – CTRL-Q. Gli altri comandi interni di **cu** non vengono utilizzati di frequente in quanto sono riservati all'attività di messa a punto, o debugging.

IL COMANDO mailx

Il sistema standard SVR4 mette a disposizione un altro programma mail: il comando **mailx** (experimental mail). Si tratta di un *front-end processor* di **mail** che fornisce caratteristiche avanzate di interfaccia utente, molto simili a quelle di **sendmail** di BSD. I messaggi di posta creati da **mailx** vengono inviati realmente dal programma **rmail**, mentre **mailx** legge la corrispondenza in ingresso dalla stessa mailbox **/var/mail/login** utilizzata da **mail**. Nella pratica, gli utenti scelgono di utilizzare o **mail** o **mailx** (o qualche pacchetto più evoluto), ma raramente li usano entrambi.

Le modalità di esecuzione del comando **mailx** sono identiche a quelle di **mail**, ma esistono molte differenze nelle opzioni possibili nella linea di comando. Questo comando

```
$ mailx
```

viene utilizzato per leggere posta, mentre

```
$ mailx lista-destinatari
```

viene utilizzato per inviare posta. L'opzione **-H** (headers) induce **mailx** a stampare solo le intestazioni dei messaggi e poi a terminare l'esecuzione. Il programma **mailx** fornisce le operazioni fondamentali di **mail**, con notevoli miglioramenti e facilitazioni nell'uso. Oltre a sofisticati strumenti con cui elencare le intestazioni dei messaggi, **mailx** consente di predisporre un profilo di esecuzione. Quando **mailx** viene lanciato, esamina due file: **/etc/mail/mailx.rc** – un profilo di sistema comune a tutti gli utenti – e **$HOME/.mailrc** (mail run control) – un file di controllo di proprietà dell'utente. Questi due file possono contenere alcuni comandi in un formato speciale, definito per **mailx**, che rendono effettivamente **mailx** un linguaggio di programmazione *mail-oriented*, permettendo una precisa personalizzazione delle sue operazioni. Non descriveremo dettagliatamente le caratteristiche del linguaggio di controllo, ma sottolineiamo che può modificare sensibilmente l'aspetto dell'intestazione per i messaggi in arrivo. Inoltre, il linguaggio di controllo permette di definire un insieme semipermanente di intestazioni di messaggi, che potete aggiungere a ogni nuovo messaggio spedito, senza doverle ogni volta introdurre da tastiera. Una funzione di espansione di metacaratteri, con sostituzioni controllate da variabili, consente di aggiungere al messaggio informazioni come la data corrente o l'elenco dei destinatari del messaggio. Inoltre, *signature* permette di predefinire un insieme di linee che viene aggiunto alla fine di ogni messaggio creato. Il file di controllo di **mailx** fornisce molte altre funzioni; il programma **mailx** è uno strumento di posta elettronica estremamente potente, anche se può risultare piuttosto lento in esecuzione, specialmente su macchine caricate pesantemente.

IL COMANDO ct

Potete predisporre la vostra macchina a compiere una chiamata telefonica in uscita a un terminale remoto, in luogo di un'altra macchina UNIX. Questo consente di stabilire da *host* la connessione in login di un terminale, ottenendo l'attribuzione delle spese telefoniche al sistema centrale. Il comando **ct** (call terminal), è predisposto a questo scopo, tuttavia non funziona correttamente in tutte le release; dovrete verificarne il funzionamento nel vostro sistema.

Il comando **ct** accetta un numero telefonico o un nome di una macchina come argomento nella linea di comando, assieme ad altri argomenti nello stesso formato accettato da **cu**, per esempio:

```
$ ct -s2400 5556789
```

Il comando **ct** compone il numero e, se riceve in risposta un tono di modem, esegue un **login:** al terminale, in modo che sia possibile entrare nel sistema come se la chiamata fosse stata fatta dal terminale; quando la sessione termina, **ct** rileva la disconnessione e termina esso stesso.

Potete includere **ct** in un lavoro programmato con **at** per chiamare un numero in un momento prefissato del giorno e ora.

Capitolo 15

Il sottosistema di comunicazione dati uccp

Oltre agli strumenti di comunicazione già visti, il sistema UNIX dispone di un programma di utility per la trasmissione di dati in background molto sofisticato e potente che rende possibile il trasferimento di file tra elaboratori, senza richiedere un intervento particolare da parte degli utenti. Si tratta del sottosistema **uucp** (UNIX to UNIX copy), un pacchetto che permette di trasferire file binari o ASCII tra macchine diverse, di controllare l'esecuzione di comandi su una macchina remota, di accodare lavori di trasferimento ed eseguirli in un secondo tempo, di ripetere automaticamente le operazioni quando il trasferimento non ha avuto successo. Il sottosistema **uucp** comprende molti comandi e funzioni utente, assieme a completi strumenti di caratterizzazione per le diverse reti di comunicazione; possiede eccellenti meccanismi di protezione, strumenti di registrazione di dati storici, mezzi per diagnostica di errori, oltre a diversi protocolli per il trasferimento di dati, con verifiche di errore adeguate a ciascun tipo di rete.

Il sottosistema **uucp** fornisce a **mail** il supporto per il trasferimento della posta elettronica tra macchine diverse, e gestisce le necessarie interazioni, senza che l'utente abbia necessità di conoscere in dettaglio il sottosistema **uucp** stesso. In SVR4 **mail** può trasferire file sia ASCII che binari, quindi, se il vostro corrispondente risiede su un'altra macchina SVR4, non avete necessità di conoscere granché del sistema **uucp**; al contrario, se dovete scambiare file con macchine che non sono SVR4, **uucp** diventa il vostro principale strumento di comunicazione. Per le sue caratteristiche di poten-

za e di efficienza, **uucp** può risultare piuttosto difficile da gestire e da caratterizzare correttamente. In questo capitolo tratteremo le modalità di gestione di **uucp**, ma nei primi paragrafi presupporremo che il sottosistema **uucp** sia già attivo sulla vostra macchina.

15.1 Il comando uuto

Il mezzo più semplice per trasferire file tra due elaboratori è dato dal comando **uuto** (UNIX to UNIX to) per spedire file, e dal comando **uupick** (UNIX to UNIX pickup) per ricevere file inviati da un'altra macchina. Il comando **uucp** accetta come argomenti un elenco di nomi di file e l'indirizzo di una macchina remota:

```
$ uuto dati1 dati2 macchina!remota
```

Dopo l'elenco dei nomi di file, l'ultimo argomento è l'indirizzo della macchina remota, nello stesso formato utilizzato nel comando **mail**: il nome della macchina remota e l'identificatore di login, separati dal carattere **!**. A differenza di **mail**, il comando **uuto** accetta solo un indirizzo remoto, per cui non è possibile trasferire contemporaneamente a più destinatari i file indicati sulla linea di comando. Il numero di nomi di file nella linea di comando non è soggetto a limitazione; l'indirizzo deve essere l'ultimo argomento.

Il comando **uuto** non esegue immediatamente il trasferimento dei file, ma li accoda in attesa che il sistema **uucp** li trasferisca al momento opportuno; il trasferimento può essere ritardato dal sistema per lungo tempo, per cui possono trascorrere ore o anche giorni prima che i file arrivino a destinazione. Se le operazioni di trasferimento vengono accodate correttamente, **uuto** restituisce il controllo allo shell dopo 1 o 2 secondi, e potete continuare la vostra sessione.

Il comando **uuto** accetta due opzioni: l'opzione **-m** (mail) causa la spedizione di un messaggio sia al mittente, che al destinatario, una volta completato il trasferimento dei dati; l'opzione **-p** provoca l'esecuzione di una copia del file nell'operazione di accodamento. Senza l'opzione **-p** non potete cambiare il nome del file o cancellare il file se il trasferimento non si è concluso; se modificate il contenuto del file dopo averlo accodato, ma prima di averlo spedito, il sistema **uucp** trasferisce la versione aggiornata del file. L'opzione **-p**, invece, trasferisce la versione del file che **uuto** ha accodato e permette di spostare, cambiare o cancellare la versione originale. Poiché l'opzione **-p** duplica il file su disco, comporta una diminuzione di spazio disponibile e di conseguenza può creare qualche problema se i file da trasferire hanno una dimensione elevata.

A differenza di **mail**, nel comando **uuto** non è permesso l'indirizzamento *multi-hop*. Se non esiste un collegamento diretto tra sistema sorgente e si-

stema destinatario, dovrete utilizzare **uuto** per trasferire i file a una terza macchina, connessa con ambedue i sistemi, su cui lavori un utente che sia disposto a inoltrare il file a destinazione.

15.2 Il comando uupick

Quando i file sono stati trasferiti, il sistema **uucp** attivo sulla macchina ricevente avverte l'utente destinatario dell'arrivo di file dall'altra macchina, con un messaggio tramite posta elettronica. Ricevuta questa posta, il destinatario può utilizzare il comando **uupick** per copiare nel directory corrente i file che **uucp** ha memorizzato in un directory temporaneo. Il comando **uupick** viene generalmente lanciato senza argomenti, individua i file che sono stati trasferiti tramite **uuto** e per ciascuno chiede quale azione intraprendere:

```
$ uupick
from system unix1: file dati1 ?
```

Come potete notare, **uupick** visualizza l'identificatore della macchina remota (*unix1*) da cui proviene il file e il nome del file trasferito (*dati1*); poi attende un vostro comando.

Se battete il comando *, **uupick** visualizza l'elenco dei comandi disponibili. Il comando più utile è **m** (move), che permette di spostare il file (*dati1* nel nostro caso) in un directory qualunque; a questo scopo, il comando **m** accetta un argomento con l'indicazione del directory in cui spostare il file, in luogo del directory corrente, per esempio:

```
$ uupick
from system unix1: file dati1 ? m /tmp
642 blocks
from system unix1: file dati2 ?
```

Il comando **uupick** visualizza la dimensione del file e passa all'esame dei successivi file; se non ne esistono altri, restituisce il controllo allo shell. Per spostare un file nel directory corrente utilizzate il comando **m .** (m punto).

Il comando **a** (all), ha una funzione analoga a quella di **m** ma, spostando tutti i file che la macchina remota ha trasferito, se il loro numero è elevato risulta più veloce di tanti comandi **m**. Il comando **d** (delete) cancella il file, **p** (print) ne visualizza il contenuto (fate attenzione a utilizzare questo comando perché **uuto** può trasferire anche i file binari) e il comando **q** (quit) permette di uscire da **uupick**. I file che non avete spostato in un directory o cancellato rimangono nel directory di spool fino alla successiva esecuzione di **uupick**. Infine, il comando **!cmd** attiva un subshell che esegue un qualsiasi comando (**cmd**) e poi restituisce il controllo a **uupick**. In questo modo,

è possibile creare nuovi directory o cambiare i diritti di accesso prima di trasferire i file.

Il comando **uupick** accetta l'opzione **–s** (system), seguita dal nome di un sistema, per limitare la ricerca solo ai file trasferiti dalla macchina indicata. I comandi **uuto** e **uupick** si trovano nel directory **/usr/bin**; sono entrambi file di comandi che potete esaminare per avere maggiori dettagli sulle loro caratteristiche.

15.3 Considerazioni sulla sicurezza di uucp

La sicurezza è un fattore importante per il sottosistema **uucp**, perché il trasferimento di file da una macchina a un'altra può risultare estremamente pericoloso per le macchine e per l'integrità dei loro file. I comandi **mail** e **uuto** sono i preferiti in molti trasferimenti di dati tra macchine, perché permettono solo di trasferire i dati dalla macchina locale alla macchina remota e tutti i file vengono trasferiti in directory di dominio pubblico, di proprietà di un determinato sottosistema. Tuttavia, il sottosistema **uucp** nel suo insieme ha molte altre potenzialità, derivanti in parte dalla possibilità di esecuzione remota di comandi qualsiasi. Di conseguenza, il sottosistema **uucp** prevede notevoli misure di sicurezza e quanto diremo sui comandi e sulle loro azioni può essere permesso per alcune macchine, ma non per altre. Nel Capitolo 22 tratteremo in dettaglio la gestione della sicurezza di **uucp**, ma per ora è importante teniate conto che molte caratteristiche significative di **uucp** possono essere disabilitate su alcuni sistemi o reti, per ragioni di sicurezza. Nel corso degli anni, le misure protettive di **uucp** sono diventate sempre più restrittive, come risultato delle amare esperienze incontrate nelle comunità di sistemi UNIX. Solo in ambienti molto sicuri, in cui non è previsto nessun accesso alle linee telefoniche pubbliche o modem a chiamata automatica, possono essere usate con fiducia le caratteristiche più avanzate di **uucp**.

15.4 Il comando uucp

Nel caso di trasferimenti di file, il comando **uuto** rappresenta il metodo più semplice per accedere al sottosistema **uucp** tuttavia, il comando **uucp** può fornire, in situazioni particolari, un maggiore controllo sul trasferimento di dati. A differenza del comando **uuto**, mediante **uucp** potete richiedere di copiare un file da una macchina remota alla macchina locale.

Il seguente comando

```
$ uucp file.sorgenti file.dest
```

copia l'elenco di file **file.sorgenti** nel file **file.dest**. Questi argomenti specificano in realtà indirizzi nel consueto formato *macchina*!*obiettivo*, eccetto che, in questo caso, *obiettivo* non rappresenta un identificatore di login di utente, ma un pathname del file o directory da copiare. Nella sintassi di **uucp**, la parte *macchina*! può essere omessa se si riferisce alla macchina locale. Per esempio, questo comando

```
$ uucp /etc/profile uname!/var/spool/uucppublic/profile
```

copia il file **/etc/profile** della macchina locale nel file **/var/spool/uucppublic/profile** della macchina di nome **uname**.

Potete trasferire più file se *obiettivo* è un directory. Questo comando

```
$ uucp $HOME/* uname!/var/spool/uucppublic/xfer.dir
```

trasferisce tutti i file che appartengono al directory $HOME nel directory **/var/spool/uucppublic/xfer.dir** della macchina remota.

Inoltre, **uucp** può copiare i file dalla macchina remota alla macchina locale se il primo argomento include la parte *macchina*!. Questo comando

```
$ uucp "uname!/tmp/salve/*" /var/spool/uucppublic/xfer
```

trasferisce tutti i file contenuti nel directory **/tmp/salve** del sistema remoto nel directory **/var/spool/uucppublic/xfer** del sistema locale. Abbiamo delimitato con virgolette l'elenco dei file sorgente per evitare che lo shell locale espanda il carattere *, che dovrà invece essere espanso sul sistema remoto. Potete anche inserire la parte *macchina*! specifica in entrambi gli argomenti e, in questo caso, il comando **uucp** esegue un trasferimento per conto terzi da una macchina a un'altra:

```
$ uucp macchina1!file macchina2!file
```

Nessuna delle due macchine deve essere necessariamente la macchina locale, a condizione che le misure di protezione di **uucp** accettino una linea di comando come quella precedente.

PATHNAME LOGICI

In ambiente **uucp**, di solito, le misure di sicurezza sono predisposte in maniera da permettere di copiare i file solo da alcuni directory di dominio pubblico, tra questi il directory **/var/spool/uucppublic**. Di conseguenza, il meccanismo di definizione dei pathname spesso non funziona esattamente come descritto finora; per sormontare queste limitazioni di sicurezza, il comando **uucp** accetta alcune forme di definizione di pathname logici. Se la

parte *obiettivo* nel modello *macchina*!*obiettivo* è ~ *user* (tilde user), dove *user* è un identificatore di login sulla macchina destinazione, **uucp** crea il file nell'home directory di quell'utente. Per esempio, molti sistemi permettono agli utenti di creare un directory di dominio pubblico denominato **rje** (remote job entry) all'interno del loro home directory. Questo comando

```
$ uucp file.locale my_sys!~giorgio/rje
```

scrive il file **file.locale** nel directory **rje** dell'utente **giorgio** sulla macchina **my_sys**. Se la parte *obiettivo* inizia con i caratteri ~/ (tilde barra), come per esempio */destinazione*, il comando **uucp** scrive il file, specificato come primo argomento, nel directory *destinazione* che fa parte del directory standard di dominio pubblico definito sulla macchina remota per **uucp**. Questo directory pubblico di **uucp** è generalmente **/var/spool/uucppublic**, per cui il comando

```
$ uucp file.locale my_sys!~/giorgio
```

copia **file.locale** nel directory **/var/spool/uucppublic/giorgio** sulla macchina **my_sys**. Questo per il trasferimento di file singoli, ma se intendete trasferire più file dovete aggiungere a ~/*destinazione* una barra (/) finale, che avverte **uucp** di creare un directory con quel nome in cui copiare tutti i file sorgente, conservandone i nomi originali.

```
$ uucp $HOME/* sistema!~/giorgio/
```

Questo comando assomiglia molto a **uuto**, ma quest'ultimo utilizza una notazione più complessa per indicare il directory obiettivo, che viene derivato dal nome della macchina locale e dall'identificatore di login sulla macchina ricevente dell'utente destinatario dei file trasferiti.

LE OPZIONI DELLA LINEA DI COMANDO uucp

Il comando **uucp** dispone di molte opzioni che ne modificano le funzioni. Molte di queste opzioni sono utilizzate soprattutto dagli utenti molto esperti di **uucp** per ottimizzare l'operazione di trasferimento di dati, mentre alcune di esse hanno utilità generale. L'opzione **-m** (mail) ottiene di informare via mail l'utente che ha lanciato il comando, che il trasferimento di dati è terminato; l'opzione **-n** (notify) accetta come argomento un identificatore di login di un utente che lavora sulla macchina remota affinché il comando **uucp** lo informi della conclusione del trasferimento del file; l'opzione **-C** (copy) permette di copiare, prima che venga trasferito, il file nel directory di dominio pubblico di **uucp**, per cui potete cancellare o modificare la versione originale del file. Se non utilizzate quest'ultima opzione, potete cancellare o rinominare il file solo dopo che il trasferimento è terminato.

Per default **uucp** non restituisce nessun messaggio se ha accodato correttamente il file da trasferire, ma, specificando l'opzione **-j** (job), il comando visualizza il numero di identificazione del lavoro accodato; questo numero è utile per seguire l'esito del trasferimento, come vedremo in seguito. L'opzione **-x** induce il comando **uucp** a visualizzare in fase di esecuzione alcune informazioni a uso diagnostico e il grado di dettaglio di queste informazioni dipende da un numero (compreso tra 1 e 9) che segue l'opzione.

Il comando **uucp** inizia a trasferire immediatamente il file appena immesso correttamente nella coda, tuttavia potete inibire questo comportamento. Se non volete iniziare subito il collegamento con la macchina remota, utilizzate l'opzione **-r**; il lavoro viene accodato, ma il trasferimento non viene iniziato. Il comando **uucp** inizia il trasferimento del file durante il suo successivo regolare ciclo di temporizzazione, che ha luogo usualmente dopo circa un'ora.

Si ricorre all'opzione **-r** in particolare quando non è in esecuzione la macchina a cui si intende trasferire il file oppure quando non è ancora attivo il collegamento.

15.5 Il comando uux

Il comando più potente disponibile in ambiente **uucp** è **uux** (UNIX to UNIX execute) con cui potete generare linee di comando da eseguire su una macchina remota. Il comando **uux** può collezionare da diverse macchine i file specificati nella sua linea di comando, assemblare il comando nella macchina obiettivo, quindi eseguire il comando su quella macchina. Per queste eccezionali caratteristiche, **uux** rappresenta un pericoloso rischio per la sicurezza, per cui non può essere liberamente utilizzato nella maggior parte dei sistemi. Quando invece è disponibile, si dimostra particolarmente utile negli ambienti di rete in cui il trasferimento di file tra le macchine è molto rapido ed efficiente.

Il comando **uux** accetta come argomento una normale linea di comando, con la differenza che i nomi del comando e dei file che costituiscono gli argomenti della linea possono essere preceduti dall'indicazione (*macchina*!) della macchina in cui sono presenti. Questa notazione particolare indica a **uux** su quale macchina eseguire il comando e da quali macchine derivano i file che occorrono per eseguire quel comando. Per esempio, questo comando

```
$ uux "sist1!cat sist2!/etc/profile sist3!/etc/rc2 > !/tmp/output"
```

richiede il file **/etc/profile** dalla macchina **sist2** e il file **/etc/rc2** dalla macchina **sist3** e li trasferisce alla macchina **sist1**, in cui avviene l'esecuzione del comando **cat**. Il risultato in output viene ridiretto sulla macchina locale al

file **/tmp/output**. Se, negli argomenti della linea di comando, la parte *macchina*! del modello *macchina*!*nome* contiene solo il carattere ! e non specifica nessuna macchina, il comando **uux** sottintende che si tratti del sistema locale. L'intera linea di comando è delimitata da virgolette per evitare che lo shell locale interpreti il carattere > prima di **uux**.

Nei pipeline multicomando nessun comando successivo al primo può contenere l'indicazione *macchina*! perché **uux** richiede che tutte le parti del pipeline siano eseguite sulla stessa macchina, per esempio:

```
$ uux "sist1!cat !/etc/profile | grep HOME > !/tmp/out"
```

Il comando **uux** richiede il file **/etc/profile** dal sistema locale e lo trasferisce alla macchina **sist1**, dove viene eseguito il pipeline **cat**|**grep**. Il risultato in uscita viene ridiretto sulla macchina locale al file **/tmp/out**.

Il carattere – (meno) può comparire nella linea di comando **uux** come parte *nome* di file per specificare lo standard input. Una forma alternativa del comando precedente è questa:

```
$ cat /etc/profile | uux "sist1!cat – | grep HOME > !/tmp/out"
```

È preferibile indicare nel comando **uux** i pathname completi, ma gli operatori speciali ~*user* e ~!*path* agiscono sulla parte *nome* del modello *macchina*!*nome* con le stesse modalità adottate per il comando **uucp**. Il comando **uux** non funziona correttamente se i file specificati non soddisfano le attese o se le misure di protezione di **uucp** non permettono di utilizzare i comandi sulla macchina obiettivo; in questi casi, **uux** spedisce un messaggio all'utente che ha richiesto l'azione, sulla macchina originaria dove è stato lanciato **uux**. Il comando **uux** accetta diverse opzioni. L'opzione **–C** (copy) copia i file locali nel directory di spool per consentire di spostarli o cancellarli prima che termini il trasferimento dei dati. L'opzione **–n** (notify) impedisce a **uux** di fornire informazioni se il comando non termina correttamente l'esecuzione; si dimostra quindi particolarmente utile per applicazioni **uux** in background o automatiche, come per esempio applicazioni che regolarmente effettuano raccolta dati (*data collection*) da macchine collegate in rete locale. L'opzione **–j** (job) permette di visualizzare l'identificatore dell'applicazione che viene accodata, e **–r** induce **uux** ad accodare l'applicazione, senza lanciarne l'esecuzione. Se viene specificata l'opzione **–z**, **uux** avverte l'utente del buon esito della sua applicazione, mentre normalmente una conclusione positiva non viene segnalata. L'opzione **–x** accetta come argomento una cifra compresa tra 1 e 9 che stabilisce il livello di dettaglio in uscita delle informazioni di diagnostica per la messa a punto, in fase di esecuzione.

15.6 Il comando uustat

Poiché il sottosistema **uucp** esegue in background l'attività di trasferimento di file, non esiste modo di avere una conferma immediata dell'esito positivo delle operazioni. Per questo motivo, esiste in ambiente UNIX il comando **uustat** (UNIX to UNIX status) che riferisce sullo stato della coda di trasferimento di file **uucp**; fornisce un breve elenco riassuntivo delle applicazioni accodate ma non inviate, lo stato della comunicazione delle macchine in collegamento con la vostra, e altra possibilità. Senza opzioni, **uustat** riporta solo i lavori creati dall'utente che lo lancia:

```
$ uucp /etc/profile my__sys!~/root/
$ uustat
my__sysN74d6        06/27-18:58  S  my__sys  root 4290 /etc/profile
$
```

In questo caso, esiste una sola applicazione accodata alle ore 18:58 del 27 giugno e **my__sysN74d6** è il suo identificatore di lavoro; questa applicazione è diretta alla macchina **my__sys** ed è stata creata dall'utente **root**. La dimensione del file è di 4290 byte e il suo nome è **/etc/profile**. La lettera **S** (send) significa che il file deve essere trasferito, mentre una lettera **R** (received) indicherebbe che il file deve essere ricevuto.

Una singola applicazione in coda può comprendere sotto lo stesso identificatore di lavoro più file:

```
$ uustat
my__sysN74d7        06/29-17:47  S  my__sys  root 62 D.tune2317f15
                    06/29-17:47  S  my__sys  root rmail leo
$
```

Questa richiesta di trasferimento di file a una macchina remota è stata formulata da un comando **mail** che spedisce il messaggio come file di dati. In questo caso, il messaggio corrisponde al primo file tra quelli visualizzati in uscita dal comando **uustat**. Il secondo file corrisponde invece a un comando **uux** che rende possibile l'esecuzione, sulla macchina **my__sys**, del comando **rmail leo** che si occupa di consegnare il messaggio. Se i comandi **uuto** o **uucp** trasferiscono più file, questi vengono catalogati sotto un unico identificatore di lavoro ma, se sono accodati più comandi **uucp**, **uustat** visualizza più identificatori di lavoro.

Una volta concluso il trasferimento dei dati, il corrispondente identificatore di lavoro non compare più nell'elenco di **uustat**; se non esistono altre applicazioni da trasferire, **uustat** termina l'esecuzione senza restituire alcun messaggio. In altre parole, quando il comando **uustat** non visualizza nessuna informazione, vuol dire che tutte le applicazioni in coda sono state trasmesse correttamente.

Se lanciate **uustat** senza opzioni, il comando riferisce solo sui vostri lavori accodati. L'opzione **–a** (all) permette invece di visualizzare tutte le applicazioni accodate sulla macchina, indipendentemente dagli utenti a cui appartengono; l'opzione **–u** (user) accetta un identificatore di utente come argomento e produce l'elenco delle applicazioni accodate di proprietà solo di quell'utente.

```
$ uustat –u giorgio
```

Si ottengono gli stessi risultati che si hanno se il comando **uustat** viene eseguito dall'utente **giorgio**.

STATO DI ALTRE MACCHINE

Il comando **uustat** può visualizzare l'elenco delle applicazioni accodate su una particolare macchina, se utilizzate l'opzione **–s** (system) seguita dall'identificatore della macchina. Questo comando

```
$ uustat –s my_sys
```

genera in uscita l'elenco delle applicazioni che sono in coda nella macchina **my_sys**. Il comando **uustat** può anche riferire sullo stato di comunicazione dei trasferimenti che hanno luogo dalla macchina locale alle macchine remote recentemente contattate. Utilizzate l'opzione **–m** (machine):

```
$ uustat –m
my_sys          1C          06/29 – 17:47 SUCCESSFUL
sys2            1C          06/29 – 18:20 TALKING
$
```

In questo esempio, la macchina locale ha contattato recentemente due altre macchine. La macchina **my_sys** è stata contattata con successo alle 17:47 del 29/6, mentre il collegamento della macchina locale con la macchina **sys2** è ancora attivo e alcuni dati sono in corso di trasferimento. Altre informazioni sullo stato del collegamento possono apparire alla fine della linea. La procedura di gestione settimanale di **uucp** cancella il giornale storico dello stato dei collegamenti, per cui raramente il comando **uustat –m** produce delle informazioni in uscita dopo l'esecuzione della procedura settimanale, a meno che non sia stata contattata qualche altra macchina.

ELIMINAZIONE DI UN LAVORO DALLA CODA

Potete utilizzare il comando **uustat** per cancellare dalla coda di **uucp** un'applicazione che non intendete più spedire o un'applicazione che non può es-

sere spedita perché una connessione non si completa correttamente. Il comando **uustat** visualizza tutti i trasferimenti **uucp** che sono attivi, tra cui quelli richiesti da **mail** o da **uux**. Per cancellare un'applicazione, dovete conoscerne l'identificatore di lavoro, che viene visualizzato quando l'applicazione è originariamente accodata, o che può essere reperito nell'elenco fornito da **uustat**.

Per eliminare un'applicazione, lanciate il comando **uustat** specificando l'opzione **-k** (kill) e, come argomento, l'identificatore di lavoro:

```
$ uustat -k my_sysN74d7
Job: my_sysN74d7 successfully killed
$
```

Quando lanciate il comando **uustat -k**, potete specificare un solo identificatore di lavoro. Se il trasferimento è già stato completato, o se è ancora in corso, l'applicazione non può essere cancellata e **uustat** restituisce un messaggio di errore:

```
$ uustat -k my_sysN74d7
Can't find Job my_sysN74d7; Not killed
$
```

15.7 Gestione del sottosistema uucp

Il sottosistema **uucp** è uno strumento di trasmissione dati estremamente flessibile e potente, ma questa flessibilità si paga in termini di gestione. La gestione di **uucp** infatti risulta, in alcuni casi, estremamente complessa e necessita di grande capacità da parte del gestore, soprattutto quando la macchina deve essere adattata a una nuova rete di comunicazione o quando non funziona correttamente. Tuttavia, gli utenti ordinari possono curare senza eccessive difficoltà la gestione del sottosistema **uucp** per i sistemi personali, e lo stesso sottosistema potrà funzionare senza problemi a meno di qualche guasto di natura patologica.

UNA NOTA SULLE VERSIONI DI uucp

Il sistema SVR4 dispone di una versione di **uucp**, che è conosciuta formalmente come BNU (Basic Networking Utilities) e informalmente come HDB (HoneyDanBer), nome derivato dal login dei tre implementatori. Questa versione differisce molto sia dalle versioni meno recenti che dalle versioni BSD. La nostra attenzione si focalizza sulla versione HDB, che è sicuramente la migliore per via dell'assenza di difetti noti (bug), dell'elevata flessibilità e dell'affidabilità.

LA STRUTTURA DEL DIRECTORY uucp

Esistono principalmente quattro strutture di directory di proprietà del sottosistema **uucp**: **/etc/uucp** contiene i file di controllo utilizzati da **uucp** per realizzare i collegamenti con altre macchine; **/usr/lib/uucp** contiene i file eseguibili di alcuni comandi e strumenti usati dal sistema; **/var/uucp** contiene i file temporanei per i trasferimenti non ancora completati, file di giornale storico e di stati, utilizzati da **uustat** e da altri comandi; infine **/var/spool/uucppublic** è il directory interessato dai trasferimenti pubblici di file tra le macchine. I comandi **uuto** e **uupick** inseriscono ed estraggono i file da questo directory. Gli ultimi due directory contengono molte strutture di subdirectory gestite dagli strumenti di gestione di **uucp**; non devono mai essere cambiati o cancellati manualmente. Il directory **/etc/uucp** contiene i file di controllo che gli utenti modificano con edit quando aggiungono nuovi sistemi remoti o cambiano il modem o i parametri di rete della loro macchina.

Il directory **/var/spool/uucppublic** contiene lo spool di dominio pubblico, utilizzato nei normali trasferimenti di **uucp**; può avere molti subdirectory, che generalmente vengono creati con il seguente comando:

```
$ uucp file dest!~/dir/
```

per cui **dir** diventa un subdirectory di **uucppublic** sul sistema **dest**. Il comando **uuto** utilizza **/var/spool/uucppublic/receive**, e ogni utente dispone di un subdirectory personale all'interno di **receive**. All'interno di questo directory di proprietà dell'utente esiste un subdirectory per ogni macchina sorgente e i file effettivamente trasferiti appaiono in quel directory. In altre parole, se la macchina **uname** spedisce questo comando

```
$ uuto file1 my_sys!giorgio
```

file1 appare nel directory **/var/spool/uucppublic/receive/giorgio/uname** che si trova sul sistema **my_sys**. Una struttura di directory così articolata evita che si verifichi una situazione in cui più file, con lo stesso nome ma spediti da macchine diverse, si sovrappongano.

Il directory **/var/uucp** contiene i file di giornale storico (*log file*) e i file accodati in spool di **uucp**:

```
$ ls -aF /var/uucp
./    .Admin/    .Log/     .Sequence/    .Workspace/    my_sys/
../   .Corrupt/  .Old/     .Status/      .Xqtdir/
$
```

Tutti questi file sono contenuti in subdirectory, la maggior parte dei quali inizia con il carattere . (punto), per cui, per avere un listato completo, utiliz-

zate il comando **ls -a**. I directory che invece non iniziano con il carattere . contengono delle informazioni temporanee sulle applicazioni che sono accodate nelle macchine indicate dal nome dei directory. Il sistema **uucp** crea questi directory di spool quando occorrono, mentre la procedura di pulizia settimanale li cancella. Le informazioni di gestione di **uucp** appartengono ad altri directory, in particolare a **.Log**, e potete esaminarle senza problemi. Il file **/var/uucp/.Admin/xferstats** contiene alcune statistiche che riguardano l'efficienza di comunicazione di tutti i trasferimenti di dati, in termini di byte trasferiti ogni secondo di tempo di collegamento. Un collegamento telefonico a 1200 baud ha un valore medio di circa 105 caratteri al secondo e risulta molto efficiente in confronto ad altri algoritmi di trasferimento di dati. Se il collegamento permette un trasferimento di dati più rapido, le prestazioni che ne derivano sono sicuramente migliori.

Il directory **/usr/lib/uucp** contiene i programmi eseguibili di proprietà esclusiva del sottosistema **uucp**, i processi demon di gestione, che eseguono lavori di pulizia e altri lavori di routine.

```
$ ls -F /usr/lib/uucp
Config@          Permissions@     bnuconvert*      uudemon.hour*
Devconfig@       Poll@            remote.unknown   uudemon.poll*
Devices@         SetUp*           uucheck*         uugetty@
Dialcodes@       Sysfiles@        uucico*          uusched*
Dialers@         Systems@         uucleanup*       uuxqt*
Grades@          Teardown*        uudemon.admin*
Limits@          Uutry*           uudemon.cleanup*
$
```

In SVR4 i file di controllo che permettono ai programmi **uucp** di decidere come effettuare il collegamento con le altre macchine sono stati spostati in **/etc/uucp**, ma i symbolic link sono rimasti in **/usr/lib/uucp**. Ecco l'elenco dei file contenuti in **/etc/uucp**:

```
$ ls -F /etc/uucp
Config           Dialcodes        Limits           Sysfiles
Devconfig        Dialers          Permissions      Systems
Devices          Grades           Poll
$
```

I processi di gestione sono tutti quelli che iniziano con la parola **uudemon** e, poiché sono file di comandi, potete esaminarli per studiarne le funzioni. Il programma **uucico** costituisce la parte fondamentale del sottosistema **uucp**, in quanto realizza la connessione e trasferisce i dati. Il programma **uugetty** è una versione del programma **getty** e lo può sostituire in **/etc/inittab**. In SVR4 **uugetty** è obsoleto; è stato sostituito con *Service Access Facility*, che supporta le comunicazioni bidirezionali. Anche se **uugetty** è stato conservato nel sistema per ragioni di compatibilità, il suo uso è sconsigliato.

Quasi tutti gli altri file, quelli realmente in **/etc/uucp**, sono file di controllo di **uucp** che specificano le modalità con cui si realizza il collegamento con una macchina remota. Li tratteremo più avanti nel capitolo.

I contenuti dei directory **/usr/lib/uucp**, **/etc/uucp** e **/var/uucp** sono riservati al gestore di **uucp**; sono di proprietà dell'identificatore di login **uucp** e del gruppo **uucp**. Questi diritti di proprietà non devono assolutamente essere cambiati, né devono essere modificati i diritti di accesso ai file in essi contenuti rispetto ai valori assegnati in fase di creazione. Qualunque cambiamento apportato ai diritti di accesso o alla proprietà di questi file e directory genera un errato funzionamento del sottosistema **uucp** e, in questo caso, è conveniente ricaricare il software di sistema dal supporto originale, ripristinando le iniziali condizioni di funzionamento. Dovete salvare i file di controllo prima dell'operazione di caricamento, ma fate attenzione a non riscrivere i vecchi file su quelli ripristinati, perché altrimenti i diritti di accesso rimangono errati. Solo il supervisore è abilitato ad apportare cambiamenti al sottosistema **uucp**, ovvero edit ai file, ed è responsabile del mantenimento corretto dei diritti di accesso e della proprietà dei singoli file e directory.

Il sottosistema **uucp** utilizza un altro directory, **/var/spool/locks**, che contiene i file di assegnamento (*lock file*) di risorse del sistema. Questi file vengono creati quando i programmi **uucico** e **cu** richiedono di accedere a un dispositivo o una porta. Anche il programma **uugetty** crea un file di lock quando un utente chiama la macchina. La presenza di questi file segnala agli altri programmi, in esecuzione nel sottosistema **uucp**, che una porta è impegnata, ed evita la contesa della stessa risorsa da parte di più programmi contemporaneamente. I programmi rimuovono il file di assegnamento quando finiscono di utilizzare il dispositivo e lo rendono disponibile ad altri impieghi. I file di assegnamento vengono cancellati durante l'operazione di inzializzazione del sistema, per cui, se un programma **uucico** non rilascia un file di assegnamento per un errore in esecuzione, la situazione di blocco che si crea è solo temporanea.

L'ARCHITETTURA DEL SOTTOSISTEMA uucp

Quando un'applicazione viene messa in coda per essere trasferita con **mail** o con uno dei programmi **uucp**, le funzioni di utility di **uucp** esaminano la lista di macchine remote conosciute; se il sistema remoto voluto viene individuato, i programmi creano un nuovo file nel directory **/var/uucp**, in cui memorizzano la richiesta di trasferimento di dati. Se non è in corso nessun trasferimento di dati verso quella macchina, le funzioni di utility di **uucp** eseguono il programma **/usr/lib/uucp/uucico** (uu copy in copy out); **uucico** utilizza altri file di controllo per determinare il miglior modo di realizzare il collegamento con il sistema remoto e, se il collegamento ha successo, esegue il trasferimento. Su entrambe le macchine interconnesse è in esecuzione di fatto un programma **uucico**, come può confermare il comando **ps -fe**.

Al termine del trasferimento, i file di controllo contenuti nel directory **/var/uucp** vengono cancellati.

Se il collegamento non riesce, **uucico** può tentare altre connessioni prima di abbandonare, ma questo dipende dalla gestione dei file di controllo.

Se il trasferimento non riesce per qualsiasi motivo, il lavoro rimane in coda. Nella maggior parte dei sistemi, la caratteristica di temporizzazione **cron** seleziona, una volta ogni ora, un processo **uucp** regolarmente scadenziato; questo ricerca in **/var/uucp** e nei suoi subdirectory le applicazioni che sono ancora in coda e lancia il programma **uucico** per trasferirle. L'algoritmo di scheduling ha un comportamento adattabile, per cui seleziona più spesso le applicazioni che si sono accodate di recente rispetto a quelle che non è stato possibile trasferire in più occasioni. Infine, se il trasferimento fallisce ripetutamente per una settimana, il processo di pulizia settimanale di **uucp** invia al mittente un messaggio di avvertimento tramite **mail**. Generalmente, questa situazione non si verifica, a meno che la macchina che deve ricevere i dati non sia operativa o il canale di comunicazione non sia attivo.

SCELTA DEL METODO DI COLLEGAMENTO CON UN SISTEMA REMOTO

Ogni macchina remota, a cui potete accedere tramite **uucp** o **mail**, deve essere specificata in un file di controllo di **uucp**. Quando le funzioni di utility di **uucp** accodano un'applicazione in attesa di trasferirla, esaminano il contenuto del file di controllo per verificare se quella macchina remota vi è compresa. Se la ricerca ha esito negativo, i comandi **uucp** non sono eseguiti e restituiscono un messaggio di errore:

```
$ uucp file1 nosys!~/user/
bad system: nosys
uucp failed completely (11)
$
```

Il comando **uname** consente di determinare l'insieme completo dei sistemi remoti collegati; produce un elenco di nomi di macchine, uno per linea. Lanciate il comando **grep** su questo elenco, in modo da selezionare una macchina particolare, invece di richiedere la lista completa che può risultare voluminosa; usate questa linea di comando:

```
$ uuname | grep nosys
$
```

L'elenco di macchine riconosciute differisce tra **uucp** e **cu**; utilizzate l'opzione **-c** (**cu**) di **uname** per avere l'elenco di macchine conosciute a **cu** invece di quelle conosciute da **uucp**.

L'elenco delle macchine che il sistema riconosce costituisce una struttura complessa di file interconnessi all'interno del directory **/etc/uucp**, in modo che i gestori possono amministrare in maniera flessibile l'elenco dei sistemi disponibili a seconda delle circostanze. Il file **/etc/uucp/Sysfiles** contiene una tabella dei sistemi disponibili, insieme ad altre informazioni:

```
$ tail -7 /etc/uucp/Sysfiles
service = cu        systems = Systems \
                    devices = Devices \
                    dialers = Dialers

service = uucico    systems = Systems.cico:Systems \
                    devices = Devices.cico:Devices \
                    dialers = Dialers.cico:Dialers
$
```

La prima parte di **Sysfiles** contiene generalmente una lunga nota di commenti che descrivono come utilizzare il file. Le informazioni fornite da **Sysfiles** si dividono in varie sezioni a seconda dei servizi disponibili sulla macchina, che generalmente si riducono a **cu** e **uucico**, come già indicato; possono essere presenti servizi addizionali se la vostra macchina fa parte di una rete LAN. La parola chiave **service=** delimita l'inizio della descrizione di un servizio; ogni descrizione inizia su una linea distinta, ma può contenere al suo interno più linee separate dal carattere \ (barra inversa), che annulla il carattere di ritorno a capo.

Ogni servizio dispone di tre diversi tipi di informazioni (sul sistema, sui dispositivi e sulle modalità di chiamata) che occorrono al software per realizzare il collegamento: *systems information*, *devices information* e *dialers information*. Ogni tipo di informazione dispone di una parola chiave in **Sysfiles** che specifica uno o più file nel directory **/etc/uucp** dove si trova quella informazione, per cui più file possono contenere l'informazione sul sistema, sui dispositivi, ecc. I nomi di questi file seguono la corrispondente parola chiave che segue a sua volta la parola chiave **service**. I file di controllo sono separati dal carattere **:** (due punti) e le corrispondenti parole chiave sono separate tra loro da un carattere spazio. Ogni servizio dispone di una personale lista di file di controllo.

Nel precedente esempio le sezioni **systems=** di **Sysfiles** forniscono i nomi dei file di controllo che contengono gli elenchi delle macchine remote a cui **uucico** (o **cu**) può accedere. I file **Systems.cico** e **Systems**, memorizzati nel directory **/etc/uucp**, contengono le tabelle che il servizio **uucico** consulta in esecuzione; se il file **Systems.cico** non contiene la macchina richiesta, allora il programma **uucico** consulta il file **Systems**.

I FILE SYSTEMS

Il contenuto dei file **Systems** stabilisce le modalità con cui la funzione di utility cerca di realizzare il collegamento con la macchina remota. Se esistono reti diverse, diversi tipi di modem, collegamenti diretti e differenti sequenze di login e password per macchine diverse, il file **Systems** contiene una grande quantità di informazioni, tra cui alcune note di commento che spiegano il significato dei diversi campi di cui si compone il file. Per esempio:

```
$ tail -3 /etc/uucp/Systems
my_sys Any ACU 1200 5300548 "" \d ogin:--ogin:--ogin: nuucp
packet Any DIR 9600 - "" \d ogin:-ogin:--ogin: nuucp
another Any D2 4800 - "" \d ogin:-ogin:--ogin: nuucp
$
```

Per ogni macchina remota, il file **Systems** riserva più linee. Procedendo da sinistra a destra, il primo campo della linea indica il nome della macchina; il secondo campo indica i periodi di tempo in cui è permessa la chiamata (**Any** significa che la chiamata può avvenire in ogni momento); il terzo campo specifica il dispositivo (*device*) da utilizzare per il collegamento, cioè, più precisamente, il puntatore a un descrittore del dispositivo contenuto nel file **Devices**. Il quarto campo in **Systems** indica la velocità richiesta di trasferimento dei dati, che coincide con le capacità del dispositivo. Spesso si utilizza **Any** come indicatore di velocità, per rimettere la decisione sul valore effettivo della velocità di trasferimento dei dati alla logica di **Devices**. Il quinto campo contiene il numero di telefono, se viene utilizzato un modem a chiamata automatica, oppure il nome di un file (**Dialer**) per i collegamenti in rete. Il segno - (meno) indica invece che il campo non viene utilizzato. Il resto della linea costituisce una sequenza di dialogo, *chat*, costituita da una serie di stringhe per realizzare la procedura detta *handshaking*.

La sequenza di *chat* ha il seguente funzionamento. In fase di collegamento, la macchina locale attende di sentire la prima stringa (una stringa nulla "", significa attendi nulla). Quando la stringa viene ricevuta dalla linea di trasmissione dati, la macchina locale invia la stringa successiva nella sequenza, poi attende quella successiva e così via. Questo permette al servizio di realizzare il collegamento con le macchine remote, svolgendo complesse sequenze di login. Di solito, l'ultima parte della sequenza attende il prompt **login:** dalla macchina remota e poi invia l'identificatore di login per il programma **uucico**, cioè **nuucp**. Se la macchina remota lo richiede, la procedura *chat* attende il prompt **passwd:** e poi invia la password per quella macchina. Molte macchine non richiedono una password per il login **nuucp** perché il programma **uucico** ottiene il controllo in luogo di uno shell.

Oltre a queste normali stringhe di caratteri, **chat** può includere alcune speciali parole chiave con cui la macchina genera un ritardo, invia un se-

gnale di interruzione, ecc.; queste parole chiave sono documentate nei commenti del file **System**.

Una macchina remota può avere più di un elemento nel file **Systems** se esistono più modi per raggiungerla. Per esempio, una macchina può essere accessibile su una rete locale o tramite un collegamento telefonico commutato. In fase di collegamento con la macchina, **uucico** tenta il primo elemendo di **Systems** e, se il tentativo fallisce, tenta il secondo e così via. In altre parole, se un collegamento di dati non è disponibile, il trasferimento di dati può seguire un altro cammino.

IL FILE DEVICES

In fase di collegamento, al servizio (**cu** o **uucico**) può interessare conoscere alcuni dettagli sulla configurazione dei dispositivi locali di comunicazione. Per dispositivo si intende generalmente un modem o un collegamento in rete che prevede una porta sulla macchina locale e che può richiedere una sequenza **chat** per attivarla. Per esempio, un modem a chiamata automatica mediante comandi **AT** può essere connesso a una porta **tty** della macchina con un collegamento RS-232. Per rendere possibile la chiamata automatica, dovete inviare il corretto comando **AT** e inviare il numero di telefono da chiamare. Dopo ogni comando, il modem risponde con una stringa ASCII; questa informazione è logicamente associata al modem e non al sistema a cui intendete accedere tramite quel modem. Per questo motivo, le stringhe **chat** e le altre informazioni relative al dispositivo sono considerate dati del dispositivo, mentre le informazioni che riguardano il sistema, come per esempio la sequenza di login, fanno parte dei dati di sistema. In altre parole, come già detto le informazioni che interessano il sistema sono contenute nel file **Systems**; mentre uno dei file **Devices**, come specificato nel file **Sysfiles**, memorizza le informazioni che riguardano il dispositivo e molte note di commento con istruzioni d'uso del file.

Il terzo campo del file **Systems** indica un dispositivo da impiegare per realizzare il collegamento. Il servizio esamina il file **Sysfiles** alla ricerca della linea **devices=** del servizio, quindi esamina i file qui indicati. Se trova una linea il cui il campo più a sinistra corrisponde al terzo campo di **Systems**, esegue l'operazione specificata nella linea.

Gli altri campi interni al file **Devices** specificano altre informazioni che interessano i dispositivi. Per esempio:

```
$ tail -3 /etc/uucp/Devices
ACU ttyO1s - 1200 hayes
DIR  ttyO0s - 9600 direct
STARLAN,eg starlan - - TLIS \D
$
```

Questo esempio illustra tre tipi molto comuni di elementi di **Devices**. I tre elementi di **Devices** riguardano, rispettivamente, un modem con comandi **AT**, una macchina direttamente connessa tramite collegamento RS-232 su **/dev/tty00s** e un accesso alla rete locale StarLan.

Se esistono più dispositivi per una singola strategia di collegamento, allora il file **Devices** può contenere più linee con lo stesso nome. Per esempio, due modem possono essere collegati alla macchina su porte differenti. Generalmente non interessa conoscere quale modem effettua la chiamata, per cui ci possono essere due elementi **ACU** nel file **Devices**, che indicano separatamente la porta utilizzata da ogni modem.

Come abbiamo già detto, il primo campo di ogni linea del file **Devices** contiene un token (contrassegno) che identifica un dispositivo. Questi token sono presenti anche nel terzo campo di ogni linea del file **Systems**. Per esempio, il token **ACU** (automatic calling unit) identifica generalmente i modem a chiamata automatica, mentre **DIR** (direct) indica le macchine che sono collegate direttamente. Il secondo campo nel file **Devices** specifica il file **/dev**, o porta utilizzata per il collegamento, privato della parte **/dev/**. Nell'elemento ACU nell'esempio precedente, il modem è collegato alla porta COM2, che è conosciuta come **tty01s** nella terminologia UNIX. Il terzo campo può indicare un dispositivo di chiamata che differisce dal dispositivo di collegamento; raramente viene utilizzato nell'hardware moderno, per cui spesso appare il segno – (meno).

Il quarto campo specifica la velocità di trasferimento dei dati, che viene effettivamente definita dall'elemento del file **Devices** se il campo velocità nel file **Systems** è pari ad **Any**. Poiché la velocità generalmente dipende dal dispositivo, il file **Systems** non contiene un riferimento numerico preciso di velocità, ma si rimette alla decisione del file **Devices**, indicando che qualunque velocità (**Any**) è di suo gradimento. In caso contrario, la velocità di **Systems** deve corrispondere con la velocità di **Devices**, altrimenti il collegamento non ha luogo con quella linea. Il quinto campo indica il nome di una linea memorizzata in un file *Dialers*, in cui è contenuta un'altra sequenza di **chat** per quel dispositivo, che specifica la modalità di chiamata. Nel caso di **ACU**, il riferimento in **Dialers** è **hayes**. Talvolta il nome **dialer** specifica un *dialer* intrinseco predefinito associato con qualche hardware speciale, come nel caso di STARLAN, in cui **TLIS** specifica una modalità di chiamata basata sul meccanismo di I/O *stream*. Questi modi di chiamata interni predefiniti non appaiono comunque nel file **Dialers**. Consultate il gestore della vostra rete per maggiori dettagli sulla configurazione di LAN, specialmente usando Ethernet e il protocollo TCP/IP.

Infine, una sequenza di **chat** specifica di un dispositivo può seguire questi cinque campi, come per l'operatore **\D** nella linea STARLAN del file **Systems**; questo *chat* viene descritto nelle note di commento interne al file **Devices**. In ogni caso, è molto più frequente trovare il **chat** del dispositivo nel file **Dialers** piuttosto che nel file **Devices**.

IL FILE DIALERS

Il file **Dialers** viene anche specificato nella lista nel file **Sysfiles**, per cui i file di chiamata possono differire a seconda del servizio in esecuzione. Un file di chiamata consiste nella sequenza di **chat** che permette al dispositivo di effettuare la chiamata di collegamento. In realtà il termine *chiamata* (dial) può trarre in inganno perché non solo i modem a chiamata automatica utilizzano il file **Dialers**. Molte reti locali utilizzano complesse sequenze di login per eludere la sicurezza di rete e indirizzare una macchina remota nella rete e anche il controllo di questi collegamenti è incluso nel file **Dialers**. Il file **/etc/uucp/Dialers** contiene note di commento che descrivono il formato e gli operatori disponibili con i file di chiamata.

Il quinto campo di ogni linea del file **Devices** contiene un identificatore che corrisponde al primo campo della linea del file **Dialers**:

```
$ grep hayes /etc/uucp/Dialers
hayes =,-, "" \dAT\r\c OK\r \EATDT\T\r\c CONNECT
$
```

Il secondo campo di questa linea elenca i token che appaiono nel numero telefonico presente nell'elemento nel file **Systems** e le relative traduzioni in token nel file di chiamata. Per esempio, l'entry **hayes** specifica al programma **uucico** di tradurre il carattere = nel carattere , e il carattere - nel carattere ,. Il carattere =(segno di uguaglianza) indica l'attesa di un secondo tono di chiamata, mentre il carattere - (segno meno) indica un ritardo di due secondi. In questo esempio, entrambi i token vengono convertiti nel carattere , (virgola), che corrisponde all'operatore di ritardo in stile **AT**. Il resto della linea contiene la sequenza di **chat** da usare per comunicare col modem.

Riassumendo, i servizi di comunicazione consultano il file **Sysfiles** per determinare quali file utilizzare per collegarsi con altre macchine; **Sysfiles** contiene l'indicazione del file **Systems** da consultare per individuare il tipo di collegamento che permette di effettuare la chiamata, determinare quando sono consentite le chiamate, e così via. Successivamente, i servizi cercano il nome del dispositivo (terzo campo di **Systems**) nel file **Devices** e ne deducono quale hardware realizza il tipo di collegamento richiesto. Poi ricercano il quinto campo del file **Devices** nel file **Dialers** (o utilizzano un modo di chiamata intrinseco) per stabilire quali siano le modalità di colloquio con quel dispositivo. Quando finalmente la chiamata ha luogo, utilizzando le informazioni dei file **Devices** e **Dialers**, e le sequenze **chat** hanno concluso il collegamento, il programma **cu** restituisce il controllo alla tastiera, mentre il programma **uucico** ritorna nell'elemento del file **Systems** e si collega alla macchina utilizzando la sequenza di **chat** contenuta nell'elemento. Al termine di tutte queste fasi preliminari, il collegamento esiste e inizia il trasferimento di dati.

15.8 Approfondimenti

Il sottosistema **uucp** è un sistema di comunicazione molto potente e flessibile che può gestire diversi tipi di reti e di dispositivi; di conseguenza presenta molti problemi potenziali quando si cerca di realizzare un nuovo tipo di collegamento.

In questo paragrafo accenneremo ad alcuni di questi problemi e tratteremo alcuni strumenti addizionali disponibili in SVR4.

DIAGNOSTICA DI ERRORI NEI COLLEGAMENTI DI uucp

Il sottosistema **uucp** dispone di eccellenti strumenti diagnostici che possono seguire l'evolversi delle diverse fasi del collegamento e diagnosticare gli errori nelle sequenze di **chat**. Il comando **cu** include l'opzione **-d** (debug) che visualizza una traccia di come evolve il collegamento:

```
$ cu -d giorgio
conn (giorgio)
Trying entry from '/etc/uucp/Systems' - device type HAYES.
Device Type HAYES wanted
Trying device entry from '/etc/uucp/Devices'.
processdev: calling setdevcfg(cu, HAYES)
fd__mklock: ok
fixline(6,1200)
gdial(hayes) called
Trying caller script 'hayes' from '/etc/uucp/Dialers'.
expect: ("")
got it
sendthem (DELAY
AT^M<NO CR>)
expect: (O^M)
O^Mgot it
sendthem (ECHO CHECK OFF
ATDT5551239^M<NO CR>)
expect: (1)
1got it
getto ret 6
device status for fd = 6
F__GETFL = 2,iflag = '12045',oflag = '0',cflag = '2651',lflag = '0',line = '0'
cc[0] = '177',[1] = '34',[2] = '43',[3] = '100',[4] = '1',[5] = '0',[6] = '0', [7] = '0',
call __mode(1)
Connected
__receive started
transmit started

login:
```

A collegamento effettuato, non appaiono più le informazioni diagnostiche e potete utilizzare **cu** normalmente. In ogni caso, quando vi scollegate, il sistema visualizza altre informazioni dignostiche:

```
~[my_sys].
call tilda(.)
call _quit(0)
call _bye(0)

Disconnected
call cleanup(0)
call _mode(0)
$
```

In maniera simile, anche il sottosistema **uucp** mette a disposizione uno strumento diagnostico, nel directory **/usr/lib/uucp/Uutry**, che genera una traccia dell'evoluzione del collegamento, come mostra la Figura 15.1. Il programma **Uutry** accetta come argomento il nome della macchina. La Figura 15.1 indica le diverse fasi di un collegamento con una rete locale StarLan, con trasferimento di un file.

Uutry è un eccellente strumento che dovreste usare spesso sulla vostra macchina per avviare i vostri trasferimenti di dati con **uucp**. Quando lanciate il programma **Uutry**, sembra che la macchina sia bloccata, perché non riappare il prompt PS1; si tratta di una situazione normale perché **Uutry** utilizza il comando **tail** **-f** per visualizzare il suo log file. Per ritornare allo shell in qualunque momento, durante o dopo l'esecuzione di **Uutry**, dovete premere il tasto DEL. Il controllo del vostro terminale ritorna quindi immediatamente allo shell, terminano le informazioni prodotte in uscita da **Uutry**, ma il trasferimento di dati continua fino alla fine. In questo modo, potete osservare la traccia prodotta da **Uutry** durante la fase di collegamento e poi abbandonare il video mentre i file (che possono anche essere molti) vengono trasferiti.

In entrambi gli esempi precedenti la maggior parte dell'output diagnostico descrive la sequenza **chat** che è stata eseguita per completare il collegamento. Queste sequenze **chat** differiscono significativamente nei due esempi perché i dispositivi interessati al collegamento sono diversi. Nell'esempio di **cu** è usata una connessione commutata con modem con comandi **AT**, con l'elemento **Systems**:

```
yoursys Any HAYES 1200 5551239 in:--in: nuucp word: any4pw
```

l'elemento **Devices**:

```
HAYES tty01s - 1200 hayes
```

e questo elemento **Dialers**:

```
hayes   =,-,        "" \dAT\r\c 0\r \eATDT\T\r\c 1
```

```
$ /usr/lib/uucp/Uutry my__sys
/usr/lib/uucp/uucico -r1 -smy__sys x5 >/tmp/my__sys 2>&1&
tmp=/tmp/my__sys
conn(my__sys)
Device Type STARLAN wanted
expect: ("")
got it
sendthem (<NO CR>NLPS:000:001:002^RSTR)
getto ret 6
expect: (in:)
login:got it
sendthem (@nuucp^M)
expect: (word:)
login: @^M^Jnuucp^M^JPassword:got it
sendthem (^M)
Login Successful: System=my__sys
wmesg 'U'eg
Proto started e
*** TOP *** - role=1, setline - X
Request: giorgio!/home/leo/s - -> my__sys!/home/giorgio/root (giorgio)
setline - S
wrktype - S
 wmesg 'S' /home/giorgio/s /home/giorgio/root giorgio -dc D.0 644 giorgio
rmesg - 'S' got SN2
 PROCESS: msg - SN2
SNDFILE:
mailopt 0, statfopt 0
*** TOP *** - role=1, setline - X
Finished Processing file: /usr/spool/uucp/my__sys/C.my__sysN3f5a wmesg 'H'
rmesg - 'H' got HY
 PROCESS: msg - HY
HUP:
wmesg 'H'Y
cntrl - 0
send OO 0, exit code 0
Conversation Complete: Status SUCCEEDED
$
```

Figura 15.1 Traccia di esecuzione di **Uutry**.

Le informazioni in uscita dell'esempio descrivono la selezione del dispositivo *HAYES* e mostrano il successo del suo collegamento attraverso la linea **getto ret 6**; reti più sofisticate mostrerebbero sequenze **chat** differenti. Quindi, il programma **cu** visualizza in un formato conciso l'inizializzazione del terminale per il collegamento e, infine, annuncia l'avvio dei processi di trasmissione e di ricezione.

Il programma **Uutry** visualizza in uscita (Figura 15.1) informazioni più dettagliate sulla sequenza **chat**, e quindi gli utenti preferiscono **Uutry** a **cu**

-d, per diagnosticare le operazioni di messa a punto di un collegamento. Ricordatevi comunque che il file **Sysfiles** definisce sequenze **chat** diverse per **cu** e **Uutry**, per cui alcuni collegamenti possono funzionare con un comando, ma non con l'altro. **Uutry** realizza il collegamento anche se non esiste alcun file da trasferire, allo scopo di facilitare la messa a punto delle sequenze di **chat**. Tuttavia, il trasferimento corretto di file ha un significato più importante della realizzazione di una semplice connessione, per cui controllate sempre l'efficacia del collegamento trasferendo qualche file in entrambe le direzioni.

La prima parte dei dati prodotti da **Uutry** indica il comando **uucico** che è in esecuzione e poi il dispositivo selezionato (ricavato dai file **Systems** e **Devices**). Segue una dettagliata descrizione della sequenza **chat**, in cui le linee che iniziano con la parola **sendthem** visualizzano l'uscita da una macchina locale, mentre le linee che iniziano con la parola **expect** visualizzano quello che la macchina locale attende dalla macchina remota. Questa alternanza di messaggi continua fino a quando esiste il dialogo tra la macchina locale e la macchina remota, dopodiché appare il messaggio "Login Successful". A questo punto inizia l'operazione di trasferimento di file (dopo l'indicazione **TOP**). In ogni file appaiono le sezioni **Request** e **SNDFILE**. Quando tutti i trasferimenti hanno avuto luogo, le due parti del collegamento segnalano la conclusione dell'operazione visualizzando la lettera **H** e il collegamento termina.

Se il collegamento non riesce, le informazioni visualizzate in uscita differiscono da quelle che abbiamo trattato. L'errore dipende generalmente da un'errata sequenza di **chat**, in questo caso, viene a mancare la sezione **expect** e appare il messaggio "Timeout", seguito da "Status FAILED". La presenza di questo messaggio indica il punto in cui la sequenza **chat** ha generato l'errore. Il programma **Uutry** visualizza molte altre codifiche di errore che indicano altri tipi di problemi. Due tra le codifiche più significative sono **No Device** e **Can't Access Device** e indicano rispettivamente che l'ingresso al file **Devices** non è corretto o che il dispositivo fisico non funziona correttamente. Qualche volta, in caso di trasferimento di file, appare in uscita il messaggio "Permission Denied", a indicare che si cerca di referenziare directory che non sono accessibili sulla parte ricevente o mittente. In altre parole, si attiva il sistema di protezione di **uucp** sui file, mentre il directory di dominio pubblico **/var/spool/uucppublic** deve risultare sempre accessibile. La maggior parte delle altre codifiche di errore ha un significato chiaro, tuttavia può essere necessaria la presenza di un esperto del sistema **uucp** per la messa a punto del buon funzionamento di complessi collegamenti di rete.

IL COMANDO uulog

Anche quando non richiedete esplicitamente col programma **Uutry** di visualizzare le diverse fasi del collegamento tra le macchine, il sottosistema **uucp**

conserva una traccia di alcune informazioni storiche che riguardano tutti i sistemi che sono stati contattati dal momento dell'ultima cancellazione settimanale dei dati di **uucp**. Il comando **uulog** visualizza queste informazioni storiche, indicando tutti gli accessi che sono stati effettuati su una macchina e, poiché si tratta di informazioni specifiche di quella macchina, il comando **uulog** richiede come argomento il nome del sistema che interessa, come indicato in Figura 15.2.

Questo esempio indica il colloquio che la macchina remota **my__sys** instaura con la macchina locale **giorgio**. Si tratta di quattro chiamate, di cui le prime due sono originate dalla macchina **my__sys** e sono memorizzate in due file che vengono trasferiti. La chiamata inizia con la parola **startup** a indicare che la sequenza **chat** ha avuto successo e il trasferimento dei file è iniziato, mentre il messaggio "conversation complete" specifica la conclusione della chiamata. L'esempio di Figura 15.2 contiene anche due chiamate errate nelle due ultime linee del colloquio. La prima di queste (LOGIN FAILED) riferisce che la sequenza **chat** non era corretta per la macchina selezionata e non è stata accettata. La seconda chiamata errata (CAUGHT) deriva da un *timeout*, ovvero dal fatto che il collegamento non è stato effettuato entro il periodo di tempo definito dal programma **uucico**. Questa situazione si verifica quando la linea di trasferimento dati non è inizializzata correttamente e lo scambio dei segnali tra la macchina locale e la macchina remota non funziona correttamente.

Le informazioni storiche possono essere molto numerose se le due macchine conversano attivamente, ma vengono cancellate durante la pulizia settimanale di **uucp**. Una lunga traccia del colloquio con una macchina dal comportamento sospetto può fornire una grande quantità di informazioni per capire i motivi per cui i file non arrivano a destinazione.

```
$ uulog my__sys
uucp giorgio (7/2-9:48:26,12288,0)  OK (startup)
uucp giorgio (7/2-9:48:26,12288,0)  REMOTE REQUESTED(giorgio!D.giorgio791d37a
- - >my__sys!D.giorgio791d37a (root))
uucp giorgio (7/2-9:48:28,12288,1)  REMOTE REQUESTED(giorgio!D.my__sys2ac61b4
- - > my__sys!X.my__sysN2ac6 (root))
uucp giorgio (7/2-9:48:29,12288,2)  OK (conversation complete tty00 5)
uucp giorgio (7/2-14:24:29,21372,0) OK (startup)
uucp giorgio (7/2-14:24:29,21372,0) REMO TEREQUESTED(giorgio!D.giorgio791f84b
- - > my__sys!D.giorgio791f84b (root))
uucp giorgio (7/2-14:24:30,21372,1) REMOTEREQUESTED(giorgio!D.my__sys2ac7685
- - > my__sys!X.my__sysN2ac7 (root))
uucp giorgio (7/2-14:24:31,21372,2) OK (conversation complete tty00 4)
uucp giorgio (7/2-18:10:07,156,0)   CONN FAILED (LOGIN FAILED)
uucp giorgio (7/2-18:34:59,206,0)   CAUGHT (SIGNAL 15)
$
```

Figura 15.2 Dati visualizzati dal comando **uulog**.

I PROCESSI DI GESTIONE DI uucp

Gran parte delle funzioni del sottosistema **uucp** è programmata, ovvero scadenziata nel sistema (*scheduled*), e non viene lanciata direttamente dall'utente. Una volta che un'applicazione è in coda, **uucp** tenta periodicamente di trasferirla fino a quando non riesce nell'intento o l'applicazione viene eliminata o rimossa dalla coda. Non è richiesta nessuna azione da parte dell'utente quando la macchina con cui intende dialogare è spenta o la macchina locale viene reinizializzata, perché la coda rimane sempre attiva. In particolare, viene utilizzato il comando **cron** per attivare periodicamente il sottosistema **uucp**. Gli strumenti di **uucp** esaminano la coda delle applicazioni e cercano di trasferire quelle che trovano, inoltre, periodicamente cancellano le registrazioni storiche; il sottosistema **uucp** può essere caratterizzato per interrogare (*poll*) regolarmente una macchina remota e verificare quindi se qualche applicazione deve essere trasferita alla macchina locale.

Tutte queste attività sono gestite dalle procedure **uudemon** contenute nel directory **/usr/lib/uucp**, che vengono attivate dalla funzione **crontab** con una frequenza che dipende dalla loro funzione. I programmi **uudemon** sono generalmente file di comandi che potete listare per avere maggiori informazioni sul loro comportamento.

Il processo **uudemon.hour** viene lanciato una volta ogni ora mentre la macchina è attiva; esegue il programma **uuxqt**, che ricerca le applicazioni in coda e, utilizzando un preciso algoritmo, determina se l'applicazione deve essere trasferita; se nel trasferimento si incontrano ripetutamente errori, **uuxqt** ne ritarda i successivi tentativi di trasferimento. Uno tra gli algoritmi più comuni effettua un tentativo il primo giorno una volta ogni ora, il giorno seguente una volta ogni due ore, il terzo giorno una volta ogni dodici ore, poi una volta al giorno per il resto della prima settimana. Se questi ripetuti tentativi non si risolvono positivamente, il programma **uuxqt** invia un messaggio di avvertimento all'utente che ha creato l'applicazione. Generalmente, se l'applicazione rimane in coda per un'intera settimana, il sistema cui è destinata non esiste oppure la sequenza **chat** non è corretta per realizzare il collegamento con quella macchina.

Il processo **uudemon.admin** viene eseguito una volta al giorno, di solito nelle ore notturne, e produce un rapporto gestionale dell'attività giornaliera di **uucp**; riporta i trasferimenti delle applicazioni, le macchine contattate, sospette violazioni delle misure di protezione e lo spazio su disco utilizzato dalle code di **uucp**; invia il suo rapporto tramite **mail** all'identificatore di login **uucp**, che spesso corrisponde al gestore di sistema.

Il processo **uudemon.cleanup** viene lanciato una volta alla settimana; cancella vecchi file, informa gli utenti sui file non trasferiti ed esegue altri compiti di pulizia.

A causa della natura gestionale, questi processi devono essere scadenziati per andare in esecuzione quando la macchina è attiva. Se la macchina viene disattivata nelle ore notturne e durante i fine settimana, dovete prevedere

l'opportuna temporizzazione in **crontab** opportuna per lanciare questi processi quando la macchina è in funzione.

POLLING DI ALTRE MACCHINE

Il sottosistema **uucp** può interrogare periodicamente un'altra macchina e chiedere se ha qualcosa da trasmettere; questa operazione è detta *polling*. Questa tecnica può essere usata quando la macchina locale è in grado di contattare la macchina remota, ma la macchina remota, per qualche motivo, non può chiamare la macchina locale. In questo caso, se la macchina locale non chiamasse la macchina remota, qualunque applicazione accodata sulla macchina remota e destinata alla macchina locale non verrebbe mai trasmessa. Potete inserire in **crontab** il processo **uudemon.poll**, per effettuare una regolare operazione di polling, in modo che venga lanciato una volta ogni ora, pochi minuti prima che vada in esecuzione il processo **uudemon.hour**.

Il processo **uudemon.poll** inizialmente consulta il file **/etc/uucp/Poll**, che contiene l'elenco dei sistemi da interrogare e l'ora e giorno per interrogarli, e poi inserisce nella usuale coda di **uucp** un'applicazione fittizia. Quando **uudemon.hour** viene lanciato, trova l'applicazione fittizia e chiama la macchina. Se le misure di protezione di **uucp** sono gestite correttamente, la macchina remota trasferisce, durante questa chiamata, il primo lavoro in coda destinato alla macchina locale. Il file **Poll** riserva una linea a ogni macchina da interrogare, contenente un elenco di ore per l'interrogazione. Potete inizializzare questo file in modo da interrogare una macchina remota con la frequenza che ritenete opportuna, da una volta al giorno a una volta ogni ora. In ogni caso, è preferibile chiamare una macchina il minimo indispensabile, per ridurre il traffico sulla rete di collegamento e sulle due macchine.

MODIFICA DEL PROTOCOLLO DI TRASFERIMENTO DI DATI

Il sottosistema **uucp** supporta diversi protocolli di comunicazione per trasferire i dati attraverso differenti tipi di reti. UNIX fornisce di default il protocollo **g** e tutte le versioni di **uucp** lo supportano. Si tratta di un protocollo a correzione di errore che ha dimostrato la sua validità nel corso di molti anni, ma le reti più veloci, che dispongono di proprie procedure di correzione di errori, possono raggiungere migliori prestazioni se si utilizza un protocollo diverso, per esempio il protocollo **e** (error-free) che si adatta bene alle reti locali. Quando sono disponibili più protocolli, il programma **uucico** "negozia" automaticamente con l'altra macchina in collegamento il protocollo di colloquio adatto a entrambe le parti.

Potete specificare nei file **Devices** il protocollo da utilizzare per un particolare tipo di rete. Al primo campo del file **Devices** usato dal programma

uucico (come specificato nel file **Sysfiles**), aggiungete il carattere , (virgola) e i nomi dei protocolli che intendete utilizzare su quel dispositivo, senza nessun carattere spazio né prima né dopo la virgola. Per esempio, questo comando

```
STARLAN,eg starlan - - TLIS \D
```

utilizza il protocollo **e**, quando la macchina remota è in grado di supportarlo, altrimenti utilizza il protocollo **g**. Non è richiesto nessun cambiamento al file **Systems** o ad altri file di controllo. Poiché i protocolli vengono definiti in fase di configurazione del programma **uucico**, non tutte le versioni di **uucp** supportano protocolli alternativi. Notate bene, dovete controllare che la vostra rete sia realmente in grado di garantire il trasferimento dati error-free prima di fare assegnamento sul protocollo **e**.

GRADI DI SERVIZIO

Il sistema **uucp** di SVR4 include la capacità di selezionare un determinato "grado di servizio" per il trasferimento dati. Con questa possibilità potete specificare la priorità di certi lavori rispetto ad altri. È possibile definire le priorità di singoli lavori, utenti e sistemi, determinando un complesso ordine di estrazione dalla coda delle attività di **uucp**. Questa tecnica può risultare utile per eseguire certi trasferimenti prioritari, in macchine con un notevole carico di lavoro, tuttavia risulta necessaria solo in condizioni non usuali. I gradi di servizio sono controllati tramite il file **/etc/uucp/Grades**, che contiene anche commenti che spiegano l'utilizzazione della tecnica dei gradi; il comando **uuglist** elenca i gradi di servizio correntemente configurati su una macchina. Potete usare i comandi **uucp** con l'opzione **-g** nella linea di comando, per selezionare un grado di servizio dei trasferimenti; dovete usare valori di grado compresi tra quelli a voi permessi, specificati nel file **Grades**.

IL FILE DEVCONFIG

Esistono molte altre caratteristiche del sottosistema **uucp**, troppo complicate per esaminarle ora in dettaglio. Una di queste è il file **Devconfig** (device configuration), una recente estensione a **uucp**, impiegato con *dialer* di trasmissione dati basati sullo schema di driver **stream device**. Quando il sistema **uucp** decide di utilizzare un dispositivo "stream-based", consulta il file **Devconfig** per determinare quali moduli *stream* collegare nel percorso dei dati verso il dispositivo. Le reti locali StarLan ed Ethernet, e altre, utilizzano il meccanismo di *stream* nel loro software.

L'uso di **Devconfig** può presentare difficoltà e incertezze; consultate il gestore della vostra rete per ottenere dettagli sulle specifiche caratteristiche.

USO DI uucp SU RETI TCP/IP

Potete usare il sistema **uucp** attraverso un collegamento Ethernet o StarLan, come su un collegamento commutato, anche se questo impiego non è molto comune. Gli utenti Ethernet preferiscono in genere l'uso del comando **rcp**, invece di **uucp**, per il trasferimento di file. Per impiegare il sistema **uucp** in questi tipi di collegamento dovete usare la versione HDB di **uucp** su entrambi i lati della connessione; di solito occorre un discreto numero di tentativi prima di mettere a punto un sistema che funziona correttamente. La predisposizione delle macchine differisce anche tra SVR3 e SVR4. Questa sezione deve essere considerata solo come esempio; consultatevi col gestore della vostra rete per conoscere le caratteristiche di questi tipi di collegamento.

Per ciascuna macchina che volete chiamare via TCP/IP deve esistere nel file **systems** un elemento che contiene l'indirizzo Internet della macchina, come segue:

```
anysys Any TCPDEV - \x00020401ff003f00
```

Dovete definire un elemento speciale di device come TCPDEV in questo esempio. La lunga stringa di cifre che segue \x è composta dalla parte fissa 00020401, seguita dall'indirizzo Internet della macchina (255.0.127.0 in questo esempio), in formato esadecimale. Notate che in questo caso non esiste la sequenza *chat*, per cui si presuppone che *anysys* fornisca il servizio **uucico** senza passare attraverso l'operazione di **login**.

L'elemento **Devices** per TCPDEV sarà come questo:

```
TCPDEV,eg ip - - TLI \D nls
```

La sigla *ip* è il nome del device file per la connessione TCP; *nls* punta al seguente elemento nel file **Dialers**:

```
nls "" "" NLPS:000:001:1\N\c
```

Quando questo elemento **Dialers** viene eseguito richiede il codice di servizio 1 (uno) alla macchina di destinazione; dovete configurare il processo in ascolto alla destinazione (*listener*) per rispondere con questo codice di servizio, mediante comandi tipo:

```
# nlsadmin -l \x00020401ff003f00
# nlsadmin -a 1
# nlsadmin -s
```

Nell'esempio viene usato l'equivalente formato esadecimale dell'indirizzo Internet della macchina ricevente, dopo la parte fissa \x00020401, per ascoltare il traffico in arrivo per questa macchina.

Può anche essere necessario aggiungere un elemento in **Devconfig** dal lato ricevente, per fare intervenire il modulo tirdwr nel trattamento dello stream, per esempio:

```
service = uucico device = ip push = tirdwr
```

Infine, non trascurate le modifiche necessarie nei file di permessi e di sistema su ciascuna macchina.

rje ED EMULAZIONE DI IBM 3270

Storicamente, il sistema UNIX ha dato prova di buone capacità di collegamento con la rete IBM SNA, rendendo possibile l'accesso in **rje** (remote job entry) tra elaboratori centrali e la macchina UNIX. Attualmente, i pacchetti software disponibili per il sistema UNIX permettono anche l'emulazione di IBM 3270. Tuttavia, entrambe queste caratteristiche richiedono un hardware addizionale e particolari driver per i dispositivi esterni. I pacchetti relativi non fanno parte del sistema UNIX standard, ma sono disponibili da diverse fonti.

Capitolo 16

Korn shell, C shell

Lo shell è un processo a livello utente come ogni altro comando. Non ha speciali relazioni col kernel, né alcun privilegio speciale che non sia concesso ad altri comandi; può quindi essere cambiato a discrezione dell'utente, e infatti sono disponibili molti shell alternativi con caratteristiche diverse, oppure ottimizzati per diverse funzioni.

Lo shell "standard", quello trattato finora, è noto come *Bourne shell* dal nome del suo progettista, Steve Bourne degli AT&T Bell Laboratories. Questo shell, progettato per un impiego generale e introdotto circa nel 1978, è di dimensioni ridotte e relativamente efficiente. Nel corso degli anni sono stati apportati miglioramenti e aggiunte nuove caratteristiche, per esempio le funzioni di shell, per aggiornarlo con il resto del sistema UNIX, ma presenta tuttora alcuni svantaggi per gli utenti esperti. In primo luogo, non esiste una *memoria storica* dei comandi; non esiste nessun modo per ripetere una linea di comando senza riscriverla interamente. Inoltre, lo shell standard non prevede il concetto di *aliasing*, cioè la possibilità per gli utenti di utilizzare pseudonimi per i comandi utilizzati più frequentemente. Potete risolvere entrambi questi problemi con il Bourne shell definendo nuovi comandi come programmi di shell, salvandoli in un file, e associando al file il nome che preferite. Purtroppo, questa soluzione è inefficiente perché è richiesto un subshell per leggere ed eseguire il programma di shell, inoltre non fornisce un modo rapido e semplice per modificare i comandi perché il programma di shell risiede permanentemente nel file.

Queste e altre considerazioni hanno motivato lo sviluppo di due shell migliorati, molto popolari, il C shell e il Korn shell, inclusi nel sistema SVR4. Entrambi questi shell possono sostituire il Bourne shell per sessioni interattive a livello utente e per l'esecuzione di file di procedura shell. Il C shell fu sviluppato originariamente come parte della versione BSD, mentre il Korn shell è stato sviluppato da David Korn degli AT&T Bell Laboratories in risposta al C shell. Il Korn shell è più recente, e quindi in molti casi presenta caratteristiche e metodi migliorati rispetto al C shell. Il C shell è anche relativamente inefficiente in confronto sia al Bourne shell che al Korn shell; tuttavia è apprezzato da molti utenti specialmente fra gli entusiasti dei sistemi BSD. Nonostante che il Korn shell abbia dimensioni maggiori del Bourne, è notevolmente più efficiente grazie alle numerose funzioni interne, che vengono svolte direttamente dallo shell e non richiedono un subshell separato.

Né il C shell né il Korn shell possono interamente sostituire il Bourne shell; nessuno dei due può essere rinominato **/sbin/sh**. Il Korn shell venne in origine progettato per essere totalmente compatibile con il Bourne shell per poterlo rimpiazzare, tuttavia esistono ancora piccole differenze che sembrano affiorare nei momenti meno opportuni; al contrario il C shell non intese mai rimpiazzare il Bourne shell, dal quale differisce totalmente.

Entrambi questi shell di più recente progettazione presentano notevoli miglioramenti rispetto al Bourne shell, alcuni fra i quali sono: un numero maggiore di operatori per la programmazione di shell, operatori aritmetici interni predefiniti che sostituiscono il comando **expr** e funzioni per ottimizzare la manipolazione di stringhe. Comunque i più importanti vantaggi offerti dal Korn shell e dal C shell riguardano la possibilità di edit dei comandi, la memorizzazione storica dei comandi e il concetto di alias, o pseudoni-

mi. La possibilità di edit e la memorizzazione storica dei comandi agiscono assieme, consentendo di recuperare un comando eseguito in precedenza, modificarlo in parte per una nuova utilizzazione (usando le caratteristiche degli editor) e infine di eseguire il comando così modificato come se fosse stato introdotto da tastiera. La possibilità di creare degli alias fornisce un efficiente modo di creare versioni personalizzate delle linee di comando di uso più comune.

16.1 Scelta di uno shell progredito

Molti utenti, superato lo stato di neofiti, si rendono conto che lo shell standard li rallenta nell'uso del sistema, diminuendone la produttività; a questo punto, è opportuno passare all'uso di Korn shell o C shell. La maggior parte degli utenti sceglie di utilizzare esclusivamente uno oppure l'altro dei due, in quanto poche persone trovano comodo alternarne l'uso in un ambiente misto.

In generale, il Korn shell offre un più ampio insieme di caratteristiche avanzate rispetto a C shell; in Korn shell l'edit dei comandi e l'uso della memoria storica dei comandi sono più facili da apprendere e usare; inoltre, Korn shell è esattamente uguale a Bourne shell nelle caratteristiche fondamentali; infine, risulta notevolmente più veloce ed efficiente del C shell. Riguardo alla programmazione shell il Korn shell è un'estensione del Bourne shell, mentre il C shell è notevolmente differente; d'altra parte il C shell è più ampiamente diffuso in differenti versioni di UNIX, specialmente nei vecchi sistemi BSD. Per questo motivo, in molte installazioni UNIX si possono trovare vasti gruppi di utenti con esperienza di C shell, che possono aiutare i nuovi utenti. In conclusione, adottate il C shell se gli altri utenti che conoscete sono utenti di C shell, oppure se il vostro sistema dispone solo del C shell, altrimenti date la preferenza al Korn shell.

16.2 Korn shell

Il Korn shell è progettato come un'estensione delle caratteristiche del Bourne shell. Questo significa che ogni linea di comando o procedura accettata dal Bourne shell *dovrebbe* funzionare con il Korn shell, incluse le funzioni shell e le altre caratteristiche di programmazione. Non è invece previsto il contrario; procedure di Korn shell possono non funzionare correttamente sotto il Bourne shell.

LANCIO DI KORN SHELL

Potete lanciare il Korn shell dalla vostra normale sessione di login, come un qualsiasi altro comando, per esempio:

```
$ ksh
$
```

Il Korn shell usa le variabili di ambiente **PS1** e **PS2** per i suoi prompt, in questo modo sembra che nulla sia cambiato dopo aver lanciato il Korn shell. Molti utenti preferiscono predisporre un prompt per distinguere quando stanno usando il Korn shell; per ottenere ciò, naturalmente, vi sono diverse possibilità. Lo stesso Korn shell fornisce l'operatore ! in **PS1**, che viene interpretato come il numero di comandi eseguiti dopo l'ultima pulizia del file storico di **ksh** (di solito quest'operazione viene fatta al login). Per esempio:

```
$ PS1 = "[!]$ "
[!]$ ksh
[1]$ echo $PS1
[!]$
[2]$
```

Notate che il prompt continua a cambiare sotto **ksh**, mentre il Bourne shell non esegue su ! nessun trattamento particolare.

Quando lo invocate in questo modo, **ksh** funziona sotto lo shell usuale come un normale comando, per cui quando uscite da **ksh**, rientrate allo shell regolare; per esempio:

```
[!]$ ksh
[1]$ exit
[!]$
```

Potete anche uscire con CTRL-D al prompt, invece di usare il comando **exit**. Un altro modo di entrare in **ksh** è dato dal comando **exec** (execute), che rimpiazza lo shell corrente con il nuovo.

```
[!]$ exec ksh
[3]$
```

In questo caso, lo shell originale viene perduto, e quando uscite da **ksh** venite scollegato dal sistema.

```
[3]$ exit
my_sys login:
```

Questa forma di **exec** può essere introdotta alla fine del vostro **.profile**, in modo che al momento del login nella macchina ottenete **ksh** come normale shell.

Quando **ksh** viene lanciato, ricerca un file simile a **.profile** per leggere degli specifici comandi di caratterizzazione dell'ambiente, che consentono di

configurare **ksh** alle vostre particolari necessità. Poiché **ksh** viene lanciato dopo il login, il file **.profile** non può contenere comandi di **ksh** non conosciuti al Bourne shell. Il file di ambiente per **ksh** consente di tenere distinti comandi particolari che vengono eseguiti dagli shell **ksh**, ma non dagli altri shell.

Questo speciale file di ambiente è una normale procedura **ksh**, ovvero una serie di comandi ammessi e interpretati da **ksh**; per usare il file dovete assegnare il riferimento del file alla variabile **ENV** e quindi esportare la variabile; per esempio:

```
[!]$ ENV = $HOME/.kshrc
[!]$ export ENV
```

Quando **ksh** entra in esecuzione ricerca la variabile **ENV**; se esiste, esegue il contenuto del file in maniera simile all'esecuzione di **.profile** da parte del Bourne shell.

Potete eseguire **ksh** per interpretare una procedura di shell, per esempio:

```
[5]$ ksh procedura
```

oppure:

```
[6]$ ksh < procedura
```

Il file procedura viene eseguito come procedura di shell da **ksh**. Potete utilizzare questa forma anche mentre state usando **ksh** per interpretare le vostre linee di comando.

16.3 Memoria storica dei comandi in ksh

Quando disponete di **ksh** come shell della vostra sessione, potete trarre vantaggio dalle sue caratteristiche di memorizzazione storica dei comandi e di edit dei comandi. Ogni comando da voi introdotto per l'esecuzione, viene da **ksh** memorizzato in un file storico di comandi, numerati sequenzialmente; il numero di sequenza di ciascun comando viene visualizzato dall'operatore ! nel prompt PS1. Vengono memorizzati nel file storico, per default, gli ultimi 128 comandi; se, introducendo ulteriori comandi, il valore di ! supera 128, **ksh** scarta dal file i comandi più vecchi, e memorizza i più recenti; i comandi scartati vengono perduti. È possibile cambiare le dimensioni del file storico, assegnando ed esportando la variabile di ambiente **HISTSIZE**; per esempio:

```
[8]$ HISTSIZE = 400
[9]$ export HISTSIZE
```

Questo istruisce **ksh** di conservare nel file storico gli ultimi 400 comandi; per pulire il file storico e azzerarne il contatore potete cancellare o ridurre il file storico prima del lancio di **ksh**.

La memoria storica viene tenuta, per default, nel file **$HOME/.sh_history**; potete cambiare il nome del file assegnando ed esportando la variabile d'ambiente **HISTFILE**; per esempio:

```
[!]$ HISTFILE = $HOME/.ksh.storia
[!]$ export HISTFILE
[!]$
```

Dopo avere fatto questo, subito dopo il login, e prima di lanciare **ksh**, potete introdurre

```
[!]$ rm $HISTFILE
[!]$
```

Questo inizia un nuovo conteggio dall'inizio della sessione di login. Potete includere questi comandi relativi alla gestione del file storico in **.profile**, per iniziare un nuovo conteggio a ogni sessione di login; altrimenti il file storico viene conservato da un login all'altro.

16.4 Editing con vi dei comandi ksh

Potete accedere al file storico, e recuperare linee di comando eseguite in precedenza, usando le funzioni di editing del comando. L'editing del comando vi consente sia di modificare le linee di comando che state introducendo, che di richiamare dal file storico comandi eseguiti in precedenza e modificarli.

Il Korn shell fornisce due metodi per l'editing di linee di comando, secondo le convenzioni degli editor di testo **vi** oppure **emacs**; gli utenti possono scegliere l'editor a loro più familiare. La scelta dell'editor viene fatta assegnando ed esportando la variabile di ambiente **VISUAL**; il valore è il nome dell'editor scelto. Per esempio, selezionate i comandi dell'editor **vi** in questo modo:

```
[9]$ VISUAL = vi
[10]$ export VISUAL
[11]$
```

Per selezionare i comandi in modo **emacs**, invece, usate:

```
[12]$ VISUAL = emacs
[13]$ export VISUAL
[14]$
```

Di solito questi comandi vengono inclusi in **.profile** per evitare di introdurli manualmente.

Dopo che avete scelto un editor, potete usare i comandi relativi per modificare le linee di comando. Lo shell **ksh** tratta il file storico come un file in modifica da parte dell'editor; normalmente vi trovate in modalità introduzione (per usare i termini **vi**) in quel file; ogni linea da voi introdotta viene aggiunta alla fine del file quando battete il RETURN, e il comando viene eseguito.

Per passare alla modalità comando, quando **VISUAL** ha il valore **vi**, usate il tasto operatore ESC; dopo questo, **ksh** vi consente di modificare il comando con gli operatori **vi**, anziché aggiungere al comando i caratteri introdotti. Per esempio, per spostare il cursore due parole indietro nella linea corrente, potete usare il comando **b2w**. Vengono accettati tutti i normali comandi **vi**; per esempio, potete ritornare alla modalità introduzione con **i** o **a**, aggiungere del testo, tornare ancora in modalità comando con ESC per cambiare tre parole con **c3w**. Potete eseguire sequenze di operazioni accettate in **vi** tutte le volte che occorre. Potete usare i normali operatori **vi** di ricerca e sostituzione; anche l'operatore **u** funziona come in **vi**.

Quando introducete un newline con RETURN, **ksh** presuppone che abbiate finito la modifica della linea, esegue la linea di comando e la accoda nel file storico.

Potete usare gli operatori **vi** "muovi" e "cerca" per recuperare comandi dal file storico; per esempio, se volete recuperare un comando **grep** introdotto in precedenza, per rieseguirlo o modificarlo, potete premere ESC, quindi introdurre **/grep** per ricercare la prima linea nel file storico che contiene la stringa *grep*. Il **ksh** visualizza la linea di comando trovata e vi lascia in modalità comando per permettervi di continuare modificando la linea di comando; quando introducete RETURN, il comando (modificato) viene eseguito. Potete ricercare in questo modo qualsiasi altra stringa, o linea di comando. Gli operatori di ricerca in file storico non accettano espressioni regolari complesse; sono ammesse solo stringhe semplici.

La modalità ultima linea non è supportata nei comandi di edit in **ksh**, pertanto gli operatori **vi** che iniziano con **:** (due punti) non sono ammessi; questo implica che non risulta agevole la ricerca nel file storico mediante i numeri di comando visualizzati dal **!** nel vostro PS1. Questi numeri sono invece utilizzati dal comando **fc**, spiegato più oltre in questo capitolo. L'elenco completo degli operatori di edit in **ksh**, sia per il modo **vi** che **emacs**, si trova nella man page **ksh(1)**.

Normalmente vi aspettereste dal comando **vi** **/** una ricerca in avanti e dal comando **?** una ricerca indietro; tuttavia, **ksh** tratta questi operatori all'inverso. In altre parole, l'operatore **/** ricerca indietro nel file storico, a partire dal comando corrente, la più recente linea di comando che contiene la stringa specificata; l'operatore **?** agisce all'inverso. Questo non ha mai costituito un problema, in quanto gli utenti hanno sempre trovato intuitivo questo comportamento.

16.5 Editing con emacs dei comandi ksh

Dopo avere selezionato il modo di editor **emacs** con

```
[17]$ VISUAL = emacs ; export VISUAL
```

potete usare i normali comandi di editor **emacs** per modificare i vostri comandi. Poiché **emacs** non prevede vere modalità introduzione e comando, dovete usare meta comandi e comandi CTRL di **emacs** per manipolare le linee di comando. Ricordiamo che i comandi CTRL vengono composti tenendo abbassato il tasto CTRL mentre viene introdotto un carattere comando; usiamo la forma **^k** per significare la sequenza CTRL-K. I meta comandi vengono generati premendo il tasto ESC e un altro tasto; nell'uso **emacs**, ESC seguito dal tasto *k* viene simbolizzato con **M-k**. Normalmente, tutti i caratteri introdotti vengono interpretati come testo e aggiunti alla linea di comando nella posizione del cursore.

Purtoppo, i comandi **emacs** usati da **ksh** non sono identici ai comandi usati dal molto noto Gnu emacs, citato nel Capitolo 5; tuttavia, gli utenti di **emacs** non dovrebbero avere difficoltà con la sintassi di **ksh**. Per spostare il cursore di un carattere indietro, usate **^b** (CTRL-B); di un carattere avanti, **^f**; di una parola indietro, **M-b**; di una parola avanti, **M-f**. Per spostare il cursore in avanti di *n* parole, usate **M-*n*M-f**, dove *n* è una cifra che stabilisce il numero di ripetizioni dell'operazione. In altre parole, definite un argomento numerico con **M-*n***, quindi introducete il comando che volete sia ripetuto *n* volte. A differenza di molti editor **emacs**, il modo di edit **emacs** non supporta l'uso diretto di argomenti in un comando; tuttavia, molti dei normali comandi, inclusi cancellazione di testo, operazioni di *yank* e copia, apposizione di mark e operazioni di ricerca, funzionano alla stessa maniera.

Per recuperare il comando precedente dal file storico, usate **^p**; usate **^n** per recuperare il comando successivo, se non siete alla fine del file storico; in altre parole, non ha senso **^n** se non avete prima eseguito almeno un **^p**.

Potete richiamare un comando più indietro nel file storico con **M-*n*^p**, dove *n* è un contatore di quanti comandi occorre risalire indietro.

Potete ricercare una stringa nel file storico con **^r*stringa***; se la stringa esiste il comando che la contiene viene visualizzato sulla linea di comando e può essere eseguito o modificato. Se viene specificato un argomento o contatore zero, la ricerca avviene nel senso avanti; altrimenti avviene nel senso indietro. In altre parole, normalmente non dovete indicare nessun argomento per trovare la più recente occorrenza della stringa nel file; potete usare la forma **M-0^r*stringa*** per ricercare in avanti, quando non siete alla fine del file storico.

Ancora, **ksh** accetta molti comandi di **vi** ed **emacs**, eccetto che un ritorno linea (newline) viene inteso come segnale di eseguire il comando corrente. Non è necessario che il cursore sia alla fine della linea quando date il RE-

TURN, perché **ksh** esegue l'intera linea di comando come risulta dopo le modifiche. Per annullare la linea di comando corrente senza eseguirla, premete il tasto DEL; **ksh** scarta la linea di comando corrente ed emette un nuovo prompt. Infine, se aggiungete un carattere # (pound, operatore shell di commento) come primo carattere della linea corrente, e premete RETURN, **ksh** aggiunge la linea al file storico, ma non la esegue. Questo consente di includere commenti o linee di comando non eseguite nel file storico.

16.6 Alias in ksh

Un'altra importante caratteristica di **ksh** è la possibilità di definire *alias*, o pseudonimi. Questa caratteristica consente di cambiare l'ambiente di **ksh** in maniera tale che nomi di comandi nella linea di comando vengono cambiati in altri comandi prima di essere eseguiti; in questo modo potete modificare permanentemente il vostro "mondo di comandi", usando comandi personalizzati, pur mantenendo in uso i normali comandi. Questa sostituzione dei nomi di comandi con pseudonimi è detta *espansione di alias*. Vediamo un esempio. Il comando **rm** cancella i file specificati, e non emette alcun messaggio; il comando **rm −i** invece richiede interattivamente una conferma dell'utente prima di cancellare un file. Molti utenti non apprezzano il comportamento di **rm**, ma non gradiscono neanche introdurre **rm −i** ogni volta che usano il comando. Potete definire **rm** alias di **rm −i**, in modo che quando introducete

```
[20]$ rm file
```

ksh in realtà esegue il comando

```
rm −i file
```

Sono possibili anche molti altri usi di alias.

Per stabilire un alias, usate il comando **alias**, per esempio:

```
[21]$ alias rm="rm −i"
[22]$
```

Lo shell **ksh** sostituirà il comando **rm** col suo alias tutte le volte che compare in una linea di comando.

L'espansione di alias segue alcune regole. Un alias può essere composto di un numero qualsiasi di parole, a condizione che siano racchiuse tra virgolette (") nel comando **alias**. La sostituzione di alias avviene solo nella prima parola nella linea di comando e nella prima parola di ciascun elemento di un pipeline. Per esempio:

```
[23]$ alias grep="fgrep -i"
[24]$ grep hello file | grep grep
```

causa l'esecuzione della linea di comando:

```
fgrep -i hello file | fgrep -i grep
```

Inoltre, quando un comando inizia con / (barra) per specificare un pathname completo, non viene applicato l'alias; di conseguenza, per utilizzare la forma normale, non alias, di un comando, ne dovete usare il pathname completo.

Potete rimuovere un alias stabilito in precedenza, col comando **unalias**; per esempio:

```
[25]$ unalias rm
```

ristabilisce il significato originale del comando **rm**. Il comando **rm** senza argomenti fornisce un elenco degli alias correnti, come per esempio:

```
[26]$ alias
autoload=typeset -fu
cat=/bin/cat
cd=__cd
cls=clear
del=/bin/rm -rf
dir=/bin/ls -F
false=let 0
functions=typeset -f
hash=alias -t -
history=fc -l
integer=typeset -i
ls=/bin/ls -F
mail=mail -v
nohup=nohup
p=enscript -2 -G -r -Plw
r=fc -e -
rm=/bin/rm -i
space=df -a
stop=kill -STOP
suspend=kill -STOP $$
true=:
type=whence -v
vi=/usr/ucb/vi
[27]$
```

Gli alias contenenti spazi bianchi devono essere protetti, quando vengono definiti; tuttavia, nella visualizzazione il comando rimuove i caratteri virgolette di protezione.

Molti di questi alias vengono stabiliti automaticamente da **ksh**, che li usa per incrementare la velocità di esecuzione; alcuni alias di questo tipo sono: **suspend**, **stop**, **type**. Altri sono noti come *tracked alias*, perché quando **ksh** esegue per la prima volta uno di questi comandi, automaticamente inserisce nella definizione il pathname completo, per ridurre le volte successive il tempo necessario per la ricerca del comando in **PATH**; gli alias **vi** e **cat** sono di questo tipo. Altri alias sono definiti dagli utenti per le proprie sessioni, come **rm**, **space**, **ls**. Molti utenti pratici di MS-DOS configurano degli alias in modo che comuni comandi MS-DOS introdotti per errore vengano tradotti nei loro corrispondenti UNIX; nell'esempio precedente sono di questo tipo **cls** e **del**.

Non esistono limitazioni al numero di alias definibili, ma poiché sono gestiti internamente a **ksh**, non permangono da una sessione all'altra come le procedure di shell. Potete, come molti utenti fanno, aggiungere i comandi **alias** nel file ENV di modo che questi alias vengano stabiliti ogniqualvolta **ksh** viene eseguito.

16.7 Il comando whence

Il **ksh** include anche diverse caratteristiche di comodità. Potete usare il comando **whence**, seguito dal nome di un comando, per conoscere il formato esatto del comando usato da **ksh** quando invocate il comando. Per i normali comandi **whence** indica il pathname completo, ricercando il comando nella vostra **PATH**; per esempio:

```
[25]$ whence ksh
/usr/bin/ksh
[26]$
```

Se il comando è una funzione di shell o un comando interno, **ksh** indica il nome senza il percorso; per esempio:

```
[27]$ whence echo
echo
[28]$
```

Infine, **whence** esegue l'espansione degli alias, di modo che potete verificare quale comando viene realmente eseguito; per esempio:

```
[29]$ whence ls
/bin/ls -F
[30]$
```

Il comportamento del comando **whence** può essere condizionato con numerosi argomenti.

16.8 Il comando fc

Il comando **fc** (fixed command) consente uno dei pochi usi possibili della numerazione dei comandi fornita da ! nel prompt. Il comando **fc** consente di selezionare una serie di comandi dal file storico, modificare questa serie con l'editor prescelto, quindi eseguire il gruppo di comandi come se aveste creato con questi un file di procedura di shell. Per esempio:

```
[31]$ fc 20 35
```

lista, uno per linea, ed esegue un editor sui comandi dal 20 al 35; potete modificare i comandi e, quando uscite dall'editor, la serie di comandi viene eseguita. Per default **fc** usa un proprio file temporaneo per la sessione di editor, ma se volete salvare i comandi, potete scriverli su un vostro file mentre siete in editor.

Il comando **fc** usa l'editor specificato nella variabile di ambiente **FCEDIT**; volendo usare il vostro abituale editor, dovete usare nel vostro **.profile** un comando come

```
FCEDIT=$EDITOR ; export FCEDIT
```

per stabilire **FCEDIT** al valore corretto.

Il comando **fc** prevede diverse opzioni. Se specificate

```
[32]$ fc -l
```

(list), **fc** visualizza il contenuto del file storico. Per listare una serie di comandi, usate

```
[33]$ fc -l 20 35
```

Potete usare **fc -e**, seguito dal nome di un editor, per sostituire l'editor della variabile **FCEDIT**. Se usate l'opzione **-e**, e specificate - (meno) come nome di editor, **fc** presuppone che volete semplicemente eseguire la serie di comandi senza modificarli. Per esempio, se volete rieseguire il comando 14 del file storico, potete usare

```
[34]$ fc -e - 14 14
```

Dovete indicare i due estremi della serie, altrimenti **fc** eseguirà tutti i comandi dal numero dato fino alla fine del file storico. Invece di specificare un numero come argomento di **fc**, potete indicare il primo o i primi caratteri del comando. Se avete nel file storico il comando

```
grep -v "^[a-z]hello[1-9]*" file | pr | lp
```

lo potete rieseguire con

```
[35]$ fc -e gr
```

In questo caso, non occorre indicare gli estremi della serie.

Infine, **ksh** usa queste caratteristiche per predefinire l'alias **r** in maniera tale che eseguendo **r** da solo rieseguirà il comando immediatamente precedente; mentre

```
[36]$ r gr
```

avrà lo stesso effetto dell'esempio precedente.

16.9 Sostituzione di tilde

Un'altra utile caratteristica è la sostituzione di tilde. Quando **ksh** incontra il carattere ~ (tilde) all'inizio di un pathname, interpreta la stringa successiva, fino al carattere / (barra) come identificatore login di un utente; se l'utente esiste, rimpiazza la sequenza tilde-stringa con l'home directory dell'utente. Per esempio:

```
[37]$ cd ~paolo
```

cambierà il directory corrente all'home directory dell'utente **paolo**, e

```
[38]$ cat ~matteo/lib/script
```

visualizzerà il file **lib/script** situato nell'home directory dell'utente **matteo**.

16.10 Cambio di directory sotto ksh

Il **ksh** prevede anche comandi estesi per operazioni di cambio di directory. La forma **cd** – cambia il vostro directory nel precedente directory in uso, come qui illustrato:

```
[39]$ pwd
/home/paolo
[40]$ cd /tmp
[41]$ pwd
/tmp
[42]$ cd -
[43]$ pwd
/home/paolo
[44]$
```

Il **ksh** supporta la variabile **CDPATH** esattamente come Bourne shell.

16.11 Il comando set

Il **ksh** include molte opzioni che ne modificano il comportamento e consentono di personalizzarlo secondo le vostre preferenze. Le opzioni sono esaminate e modificate con il comando **set**; esiste un grande numero di opzioni, documentate completamente nella man page **ksh(1)**; questa sezione ne illustra solo alcune maggiormente utili.

Il comando **set** può modificare due differenti tipi di comportamento di **ksh**. Nel primo tipo, un gruppo di opzioni cambia le modalità di trattamento delle procedure di shell; per esempio:

```
[44]$ set -n
```

istruisce **ksh** di leggere i comandi delle procedure di shell, senza eseguire i comandi; l'istruzione è ignorata negli shell interattivi, come la sessione di login. Per ripristinare il comportamento originale dello shell, usate:

```
[45]$ set +n
```

Questo esempio illustra una proprietà generale del comando **set**: l'opzione viene selezionata se l'argomento inizia con – (meno), viene deselezionata se l'argomento inizia con + (più).

Il comando:

```
[46]$ set -a
```

causa l'esportazione di tutte le variabili di ambiente definite successivamente; in questo modo potete evitare l'uso di **export** dopo ogni variabile definita.

Il comando "verbose"

```
[47]$ set -v
```

richiede a **ksh** la visualizzazione di ogni comando letto da una procedura di shell.

Infine, il comando:

```
[48]$ set -m
```

richiede a **ksh** di informarvi quando un qualunque vostro lavoro in background si conclude; per esempio:

```
[49]$ sleep 5 &
[1] 18165
[50]$
```

Qui, **ksh** riporta l'identificatore di job del lavoro in background, quindi restituisce un altro prompt; quando il lavoro si conclude (5 secondi più tardi in questo semplice lavoro) **ksh** visualizza:

```
[1] + Done          sleep 5 &
```

prima di emettere un prompt. Questa notificazione è quasi sempre molto utile e viene sempre emessa per default, ma volendo la potete escludere con:

```
[50]$ set +m
```

Il secondo tipo della funzione **set** seleziona opzioni che controllano le vostre sessioni interattive; questi comandi hanno la forma:

```
set -o opzione
```

dove *opzione* è uno dei tipi supportati. Anche in questo caso si usa la forma:

```
set +o opzione
```

per deselezionare un'opzione.

Il comando

```
set -o ignoreeof
```

istruisce il vostro shell corrente di ignorare il comando CTRL-D, evitando la disconnessione se introducete per errore CTRL-D; dopo avere selezionato **ignoreeof** dovete usare il comando **exit** per concludere la vostra sessione.

Il comando:

```
set -o noclobber
```

evita che l'operatore di ridirezione > distrugga un file esistente; in questo modo si possono evitare errori che porterebbero alla perdita di file. Potete usare:

```
set -o emacs
```

oppure:

```
set -o vi
```

per sostituire l'editor della variabile di ambiente **VISUAL** e cambiare il modo di edit della linea di comando. Infine, il comando:

```
set -o bgnice
```

provoca l'esecuzione dei vostri lavori in background (& alla fine della linea di comando) con una priorità inferiore a quella solita; l'opzione viene selezionata per default, di conseguenza dovete usare il comando:

```
set +o bgnice
```

per eseguire i comandi in background alla priorità normale. Esistono molte altre opzioni controllate con **set -o**, come abbiamo già detto.

16.12 Miglioramenti di ksh per la programmazione di shell

Oltre alle caratteristiche di comodità, **ksh** include miglioramenti per un'avanzata programmazione shell. Sono disponibili molti nuovi operatori che consentono una logica migliore nelle procedure, e diversi operatori che incrementano l'efficienza in velocità delle procedure; tuttavia, l'uso di questi nuovi operatori in una procedura ne esclude la compatibilità, e quindi l'eseguibilità, con Bourne shell. Al contrario, in generale **ksh** accetta tutti i normali operatori di programmazione in Bourne shell.

OPERATORI ARITMETICI

Uno dei più importanti miglioramenti in **ksh** riguarda le operazioni aritmetiche. Nel Bourne shell, dovete usare il comando **expr** per risolvere espressioni aritmetiche; questo risulta lento e inefficiente, perché **expr** è un comando standalone eseguito come processo separato; al contrario, **ksh** tratta direttamente molte operazioni aritmetiche, con una maggiore velocità.

Usate il comando **let** per assegnare o cambiare il valore di una variabile di ambiente, per esempio:

```
[38]$ let x=42
[39]$ echo $x
42
[40]$
```

In questo stesso modo potete usare normali operatori numerici per aggiornare o modificare il valore, come per esempio:

```
[40]$ let x=42
[41]$ let x=$X*3-11
[42]$ echo $x
115
[43]$
```

Notate che viene rispettata la normale precedenza aritmetica; per cambiare l'ordine di valutazione, dovete usare assegnazioni intermedie, per esempio:

```
[43]$ let x=42
[44]$ let y=3-11
[45]$ let x=$x*$y
[46]$ echo $x
-336
[47]$
```

Nei comandi **let** potete usare parametri posizionali di shell, e variabili di ambiente in qualsiasi numero; potete anche omettere l'operatore **let**, perché il segno = (uguale) è sufficiente a **ksh** per riconoscere un'operazione aritmetica; non dovete invece includere spazi all'interno delle espressioni.

ARRAY

Il **ksh** permette l'uso di array monodimensionali, identificati da un nome di variabile seguito da un indice. L'indice è un numero racchiuso tra parentesi quadre; per esempio:

```
[48]$ x[3]=20
[49]$ x[4]=30
[50]$ x[5]=${x[3]}+${x[4]}
[51]$ echo ${x[5]}
50
[52]$
```

Notate che quando *assegnate* la variabile indicizzata, potete riferirla direttamente; quando *usate* la variabile indicizzata, dovete racchiuderla fra parentesi graffe, altrimenti **ksh** non sarebbe in grado di interpretarla correttamente.

Le operazioni aritmetiche vengono eseguite in normale aritmetica decimale (base 10) per default, ma **ksh** consente di trattare anche l'aritmetica in basi diverse.

I/O INTERATTIVO DA TERMINALE, I/O DA FILE

Il **ksh** include diversi comandi interni che facilitano la richiesta di dati da terminale a un utente, e la lettura di dati da un file. Il comando **print** sostituisce **echo**, consentendo di stampare linee di output sul terminale (default), nel file storico (opzione **-s**), o su diversi altri obiettivi. Il comando **read** leg-

ge una linea di input; i dati letti vengono assegnati alle variabili di ambiente specificate come argomenti di **read**. Per esempio:

```
[50]$ read a1 a2 a3
uno due tre quattro cinque
[51]$ echo $a1
uno
[52]$ echo $a3
tre quattro cinque
[53]$
```

Se in input compaiono più parole del numero di argomenti previsto in **read**, l'ultimo argomento cattura tutte le parole in eccedenza; se in input compaiono meno parole degli argomenti previsti, gli argomenti in eccedenza restano indefiniti. Se il primo argomento di **read**, dopo il nome di una variabile, contiene **?** (punto interrogativo), i caratteri successivi vengono stampati come avviso di richiesta di input; la variabile di ambiente specificata dalla parola precedente il punto interrogativo riceve la prima parola in input. Per esempio:

```
[54]$ read a1?"ciao, come stai ?" a2 a3
ciao, come stai ? ciao, non male, e tu?
[55]$ echo $a1
ciao,
[56]$ echo $a2
non
[57]$ echo $a3
male, e tu?
[58]$
```

Questo consente un modo efficiente di avvertire l'utente della richiesta di introdurre dati in procedure **ksh**.

TRATTAMENTO DI VARIABILI D'AMBIENTE SOTTO ksh

Il **ksh** supporta una forma migliorata del trattamento delle variabili d'ambiente, che permette: l'uso di un valore di default per variabili non definite; la stampa di un messaggio d'errore se una variabile non è definita; l'uso di una sottostringa del valore di una variabile; oppure l'inverso di ognuna di queste operazioni. Queste forme possono assumere variazioni molto complesse; vedete come esempio la definizione della variabile **ENV** più avanti in questo capitolo.

La forma fondamentale è:

```
${parametro op parola}
```

dove *parametro* è l'usuale nome della variabile di ambiente, *op* è un operatore speciale, *parola* è il valore di default voluto, la sottostringa, e così via. Per esempio:

```
VAR = ${OVAR: - parola}
```

assegna a **VAR** il valore di **OVAR** se **OVAR** è definito, ma assegna a **VAR** il valore *parola* se **OVAR** non è definito. Altre forme sono:

```
VAR = ${OVAR # parola}
VAR = ${OVAR%parola}
```

Queste forme comparano la stringa *parola* con l'inizio (#) o la fine (%) di **OVAR**; se risultano uguali, la parte corrispondente viene cancellata. Come esempio, potete convertire il nome **file.o** in **file.c** con:

```
OLD = file.o
NEW = ${OLD%.o}.c
```

Queste operazioni possono risultare molto utili, ma anche molto complicate; l'elenco completo degli operatori si trova in man page **ksh(1)**.

16.13 C shell

Il C shell, così chiamato a causa della sua sintassi considerata simile a quella del linguaggio C, è il terzo tra gli shell oggi maggiormente usati. In realtà, il C shell è probabilmente più usato degli shell Bourne e Korn, grazie alla sua lunga storia come parte della versione BSD del sistema operativo UNIX. D'altra parte, il C shell è diverso in molti punti dagli altri e molti dei principi di shell presentati finora non sono applicabili. Il C shell non supporta le funzioni di shell.

16.14 Lancio di C shell

Potete lanciare il C shell dalla linea di comando con:

```
$ csh
my_sys%
```

Normalmente il prompt di **csh** è il nome del sistema seguito da % (percento), ma potete cambiare il prompt, come vedremo fra poco. Potete concludere la vostra sessione con **csh** mediante **exit** oppure CTRL-D; in questo caso ritornerete nel vostro precedente shell. In alternativa, potete usare:

```
$ exec csh
```

In questo caso, quando terminate la vostra sessione **csh**, verrete disconnessi dal sistema.

Quando **csh** entra in esecuzione, legge il file **.cshrc** (csh run commands) dal vostro home directory; questo file è una procedura **csh** che può contenere qualsiasi comando. Di solito, **.cshrc** contiene comandi d'inizializzazione che personalizzano il vostro ambiente **csh**. Potete forzare **csh** a trascurare molte delle sue onerose operazioni d'inizializzazione (compresa la lettura del file **.cshrc**) con l'opzione **–f** (fast); poiché molte variabili e altro materiale di **csh** vengono di solito esportati di default, l'opzione consente un apprezzabile risparmio di tempo nelle prove delle procedure **csh**.

16.15 La linea di comando csh

Il **csh** usa alcuni metacaratteri diversi da quelli di **sh** e **ksh**. Dovete fare molta attenzione ai caratteri da proteggere nella linea di comando; un caso importante è **!**. Per esempio, potreste cercare di indirizzare la posta con:

```
% mail yoursys!pat
pat: event not found.
%
```

Per usare questa linea di comando dovete proteggere **!**; per esempio;

```
% mail yoursys\!pat
```

Fate molta attenzione a tutte le differenze fra gli shell; possono procurarvi problemi.

16.16 Variabili csh

Il **csh** differisce da **sh** e **ksh** in molti aspetti; uno dei più importanti è il modo di trattamento delle variabili di ambiente. Il valore di **PS1** non viene usato da **csh**; la stessa funzione viene svolta dalla variabile **prompt**. Usate il comando **set** per cambiare il vostro prompt:

```
my_sys% set prompt = "hello: "
hello:
```

Notate che in **csh** i nomi di variabili sono abitualmente in lettere minuscole. Altre importanti variabili **csh** sono **home** e **term**, che rimpiazzano le variabili **HOME** e **TERM** usate in Bourne shell.

Potete cambiare l'equivalente di **PATH** con un'assegnazione alla variabile **path**; il valore è un elenco di directory, separati da spazi, racchiuso tra parentesi; per esempio:

```
hello: set path = ( /usr/bin /bin /usr/ucb/bin . /usr/local/bin )
hello:
```

Non è necessario **export** per variabili assegnate in **csh**, in quanto l'operazione viene eseguita automaticamente da **csh**. La stessa forma viene usata anche per **cdpath**, l'equivalente di **CDPATH** negli shell Bourne e Korn; per esempio:

```
my-sys% set cdpath = ( $home . /usr/src )
```

Notate l'uso della variabile **home**; per usare una variabile **csh** dovete far precedere il nome dall'operatore **$**. I comandi di questi esempi vengono di solito inclusi nel file **.cshrc** in modo che sono già predisposti quando **csh** viene eseguito.

Molte altre variabili sono supportate da **csh** e gestite con il comando **set**. Alcune delle più importanti sono: **noclobber**, che inibisce a **csh** la sovrascrittura di un file esistente; **ignoreeof**, che inibisce la disconnessione con CTRL-D; **cdpath**. Per esaminare l'elenco delle variabili correntemente attive, usate il comando:

```
my_sys% set
```

Per informazioni complete sulle variabili, consultate la man page **csh(1)**.

16.17 Memoria storica dei comandi in csh

Il **csh** include il supporto per la memoria storica dei comandi e per l'editing di comandi dalla memoria storica. Risulta invece difficile l'editing del comando in corso di introduzione da tastiera e non ancora immesso nella memoria storica. La memoria storica è gestita internamente: non esiste quindi un file storico come in **ksh**. La variabile **history** di **csh** definisce la dimensione della memoria storica; per esempio:

```
my_sys% set history = 40
```

Il comando **history** visualizza il contenuto della memoria storica.

Potete utilizzare la numerazione dei comandi nella memoria storica; includendo l'operatore **!** nel vostro **prompt** ottenete la visualizzazione del *numero di evento* di ogni comando; per esempio:

```
my_sys% set prompt = "'hostname'{'whoami'}!: "
my_sys{giorgio}36:
```

Quando conoscete il numero di evento dei vostri comandi, potete usare l'operatore ! in un punto qualsiasi nella vostra linea di comando per fare riferimento a uno specifico comando nella memoria storica; per esempio:

```
my_sys{giorgio}36: echo !35
echo set prompt = "'hostname'{'whoami'}!: "
set prompt = my_sys{giorgio}!:
my_sys{giorgio}37:
```

Quando usate ! nei vostri comandi, **csh** visualizza il comando dopo la sostituzione di !, ma prima dell'esecuzione. La forma !*n* permette di fare riferimento al comando *n*-esimo nella memoria storica, precedente al comando corrente. Potete ugualmente usare ! per rieseguire un comando dalla memoria storica; per esempio:

```
my_sys{giorgio}37: !35
set prompt = "'hostname'{'whoami'}!: "
my_sys{giorgio}38:
```

Anche questa volta, il comando viene prima visualizzato, poi eseguito.

Invece di usare un numero di evento per fare riferimento a un comando precedente, potete usare una stringa; la forma !*stringa* fa riferimento a un precedente comando che inizia con *stringa*. La stringa deve essere il primo carattere sulla linea di comando nella memoria storica; **csh** troverà il comando eseguito più di recente che inizia con quella stringa.

Una forma semplificata del comando ! permette di fare riferimento al comando immediatamente precedente, l'operatore !! fa riferimento all'ultimo comando eseguito; per esempio:

```
my_sys{giorgio}39: echo per una prova
per una prova
my_sys{giorgio}40: !!
echo per una prova
per una prova
my_sys{giorgio}41:
```

Notate ancora che quando usate **csh** non potete usare ! come un normale carattere senza proteggerlo con \, altrimenti **csh** lo tratterebbe sempre come un riferimento in memoria storica. Se non volete un riferimento alla memoria storica, dovete ricorrere alla protezione del metacarattere; per esempio:

```
my_sys{giorgio}41: echo \!35
!35
my_sys{giorgio}42:
```

Il **csh** fornisce un completo, anche se un poco scomodo, meccanismo di editing dei comandi recuperati dalla memoria storica. Dopo avere selezionato un comando con l'operatore !, potete sostituire parti del comando con operatori di ricerca e sostituzione, per modificarlo prima dell'esecuzione; includete : dopo il numero di evento, seguito dalla normale forma **s**/*vecchio*/*nuovo*/ per sostituire nel comando recuperato. Per esempio:

```
my_sys{giorgio}42: !39:s/una/un'altra/
echo per un'altra prova
per un'altra prova
my_sys{giorgio}43:
```

Sono consentite solo sostituzioni di semplici stringhe; le espressioni regolari complesse non possono essere trattate.

Potete anche fare riferimento a singole parole in comandi recuperati dalla memoria storica; includete : dopo il numero di evento, seguito da una cifra per indicare la posizione della parola che volete usare, considerando che il nome del comando è la parola zero. Per esempio:

```
my_sys{giorgio}43: echo !39:3
echo prova
prova
my_sys{giorgio}44:
```

Potete combinare queste forme, con operatori : addizionali, secondo le necessità; per esempio:

```
my_sys{giorgio}44: !39:3:s/ov/XX/
echo prXXa
prXXa
my_sys{giorgio}45:
```

Inoltre, $ fa riferimento all'ultima parola nella linea di comando recuperata (di solito un nome di file), # fa riferimento all'intera linea di comando; sono possibili anche molte altre operazioni simili. Potete usare un numero qualsiasi di questi modificatori, ovunque nella linea di comando, per ricostruire e cambiare i vostri comandi.

Nel trattamento della memoria storica e delle sostituzioni, **csh** agisce in questa sequenza: primo, l'evento viene recuperato dalla memoria storica, e quindi viene applicata la sostituzione o suddivisione; secondo, la stringa modificata viene sostituita nella linea di comando corrente; terzo, il comando completato viene visualizzato; infine, il comando viene eseguito.

16.18 Tabella interna di csh

Quando **csh** entra in esecuzione, ricerca tutti i comandi nei directory elencati nella variabile **path** e costruisce una tabella interna dei relativi pathname completi. Questa tabella (*hash table*) incrementa la velocità di esecuzione, in quanto **csh**, quando deve eseguire un comando, effettua una ricerca nella tabella anziché direttamente nei singoli percorsi di accesso nel file system. Tuttavia, se cambiate la vostra **path** oppure aggiungete un comando in un directory, **csh** non conosce questi cambiamenti; in questi casi, dovete forzare **csh** a ricostruire la sua tabella col comando **rehash**.

16.19 Alias in csh

Il **csh** fornisce una caratteristica **alias** per consentire di cambiare dei comandi prima di eseguirli. Si crea un alias (pseudonimo) mediante il comando **alias**, seguito dal nome dello pseudonimo e quindi da una stringa (protetta) sostitutiva; per esempio:

```
my_sys{giorgio}45: alias xx "echo salve"
my_sys{giorgio}46: xx
salve
my_sys{giorgio}47: xx gente
ciao gente
my_sys{giorgio}48: /xx
/xx: Command not found.
my_sys{giorgio}49:
```

La sostituzione di alias si effettua solo nella prima parola della vostra linea di comando, all'interno di ciascun elemento di un pipeline, e inoltre solo quando il nome del comando viene dato come pathname relativo, ovvero che non inizia con / (barra).

Potete eliminare un alias col comando **unalias**; per esempio:

```
my_sys{giorgio}49: unalias xx
my_sys{giorgio}50: xx
xx: Command not found.
my_sys{giorgio}51:
```

16.20 Ridirezione di I/O con csh

Il formato **csh** per la ridirezione del file standard error differisce da quello usato negli altri shell. Potete usare > per ridirigere lo standard output come al solito, ma per lo standard error, invece della forma **2>** usata dagli

shell Bourne e Korn, dovete usare >& per ridirigere standard output e standard error assieme nello stesso file; per esempio:

```
my_sys{giorgio}59: cat /etc/passwd >& outfile
```

Se è selezionata l'opzione **noclobber**, potete forzare **csh** a sovrascrivere un file esistente con >! e >&!; in modo simile, potete usare >>! o >>&! per appendere il vostro output alla fine di un file esistente, senza troncare il file prima di scrivere l'output.

Non è facile ridirigere lo standard output e lo standard error su file diversi, ma può essere fatto usando un subshell; per esempio:

```
my_sys{giorgio}60: (cat /etc/passwd > outfile ) >& errorfile
```

16.21 Programmazione con csh

Il **csh** supporta la programmazione shell, ma il suo linguaggio di programmazione differisce in maniera significativa da quello usato negli altri shell. Tuttavia, sono previste le stesse funzioni fondamentali; invece dell'operazione:

```
for var in list
do
done
```

il **csh** prevede l'operatore **foreach**, che è terminato da **end**; per esempio:

```
my_sys{giorgio}51: foreach var ( hello goodbye )
? echo $var
? end
hello
goodbye
my_sys{giorgio}52:
```

In questo esempio vi sono alcuni punti da notare: primo, la lista è racchiusa tra parentesi (potete usare l'output di un comando invece di una lista "letterale"); secondo, l'equivalente in **csh** di PS2 è un **?**; terzo, non esiste il comando **do...done**.

Potete esaminare una condizione ed eseguire dei comandi se la condizione è vera, mediante **if...then...endif**. **then** deve essere sulla stessa linea di **if**; la condizione deve essere racchiusa tra parentesi come in questo esempio:

```
my_sys{giorgio}51: cat isfile
if ( -e $1 ) then
        echo file $1 esiste
```

```
else
        echo file $1 NON esiste
endif
my_sys{giorgio}52: csh -f isfile /etc/passwd
file /etc/passwd esiste
my_sys{giorgio}53:
```

L'operatore **-e** restituisce "vero" (true) se il file esiste. Sono disponibili molti altri operatori; l'elenco completo compare in man page **csh(1)**. Alcuni operatori d'uso più comune sono: **-r**, che restituisce vero se avete diritto di lettura del file; **-f**, che restituisce vero se il file è un file ordinario; **-d**, che restituisce vero se il file è un directory. Potete anche comparare valori numerici, all'interno delle parentesi, usando i normali operatori aritmetici; per esempio:

```
my_sys{giorgio}53: cat numtest
set x=20
if ( $x - 20 ) then
        echo VERO o valore non-zero
else
        echo FALSO o valore zero
endif
my_sys{giorgio}54: csh -f numtest
FALSO o valore zero
my_sys{giorgio}55:
```

Il **csh** supporta operazioni numeriche all'interno delle parentesi, senza l'uso di **expr**.

D'altra parte, **csh** non supporta l'operatore ‘ (accento grave o apice inverso); in sostituzione, è possibile esaminare il codice di uscita di un comando, con { e } (parentesi graffe); per esempio:

```
if { comando argomenti } then
```

In questo caso, le parentesi in **if** sono sostituite dalle graffe.

Semplici operazioni numeriche possono essere trattate in procedure **csh**, ma non in una linea di comando. Per eseguire un'operazione aritmetica su variabili dovete usare il segno @ invece di **set**; per esempio:

```
my_sys{giorgio}62: cat foo
set x = 5
echo $x
@ x++
echo $x
@ x+=32
echo $x
@ x = $x + 32
echo $x
```

```
my_sys{giorgio}63: csh -f foo
5
6
38
70
my_sys{giorgio}64:
```

Dovete racchiudere gli elementi di frasi di assegnazione tra spazi, come per il comando **expr**; sono ammessi i normali operatori aritmetici. Ricordate che l'operatore @ non funziona in C shell interattivi.

In **csh**, la forma di Bourne shell **case...esac** è rimpiazzata da **switch...endsw**; ogni caso separato è identificato da **case** e termina con **breaksw**; per esempio:

```
my_sys{giorgio}51: cat csh_prog
switch ( $1 )
case 43:
        echo hello
        breaksw
case 34:
        echo goodbye
        breaksw
default:
        echo none!
        breaksw
endsw
my_sys{giorgio}52: csh -f csh_prog 20
none!
my_sys{giorgio}53: csh -f csh_prog 34
goodbye
my_sys{giorgio}54:
```

Le parentesi dopo l'operatore **switch** sono obbligatorie e possono contenere un'espressione che viene valutata in una stringa; se tale valutazione avviene, ogni **case** viene comparato con la stringa; i comandi successivi al **case** che uguaglia la stringa vengono eseguiti fino al comando **breaksw**, che provoca l'uscita da **endsw**. Se nessun **case** uguaglia la stringa, viene eseguito il caso **default**; l'istruzione **switch** termina con **endsw** (end switch).

Potete gestire i cicli con la coppia **while...end**; la condizione deve essere racchiusa tra parentesi; per esempio:

```
my_sys{giorgio}56: cat ss
set x = 1
while ( $x < 5 )
        echo $x
        @ x = $x + 1
end
my_sys{giorgio}57: csh -f ss
```

```
1
2
3
4
my_sys{giorgio}58:
```

La programmazione **csh** richiederebbe una trattazione molto più estesa. Anche se **csh** è molto simile al Bourne shell, presenta anche differenze sostanziali; questo fatto è causa di equivoci anche per gli esperti. Il modo migliore per imparare il linguaggio di **csh** sembra quello di studiare procedure esistenti e cercare di dedurne di proprie.

16.22 Identificazione di comandi con csh

Il **csh** non include un comando interno equivalente all'operatore **whence** di **ksh**, che identifica il percorso completo (o l'alias) di un comando; tuttavia, il comando **which** svolge la stessa funzione. In sistemi SVR4, **which** è inserito come **/usr/ucb/which**; questo directory è installato come parte del pacchetto di compatibilità BSD, pertanto può non essere presente in tutti i sistemi SVR4.

Per identificare la locazione di un comando, usate:

```
my_sys{giorgio}58: /usr/ucb/which csh
/usr/bin/csh
my_sys{giorgio}59:
```

16.23 Scelta di shell per eseguire procedure di shell

Quando eseguite una procedura, per default la procedura viene interpretata dallo shell corrente; in altre parole, lo shell interattivo crea una copia di se stesso per l'interpretazione della procedura. Naturalmente questo comportamento può creare dei problemi se la procedura è scritta in linguaggio **csh**, ma viene interpretata da **ksh**; questa situazione può crearsi facilmente in occasione di scambio di procedure di shell tra sistemi diversi.

Per risolvere questo problema, SVR4 include una speciale caratteristica, ripresa dalla versione BSD, che consente di specificare all'interno della procedura da quale shell deve essere interpretata; in questo modo, è sempre lo shell adatto a interpretare il file di comandi, indipendentemente dallo shell usato per introdurre la linea di comando.

Per usare questa caratteristica, inserite come prima linea della procedura la linea:

```
#!  /percorso/assoluto
```

Per una procedura **csh** potreste usare la linea:

```
#!  /usr/bin/csh
```

Per una procedura Bourne shell, invece:

```
#!  /sbin/sh
```

Infine, per una procedura **ksh**:

```
#!  /usr/bin/ksh
```

Il sistema ricerca questa linea prima di lanciare il subshell; poiché una linea che inizia con # viene considerata come commento da tutti e tre gli shell, l'inserimento di queste linee non ha effetto sull'esecuzione della procedura, dopo che il sistema ha selezionato lo shell adeguato.

16.24 Approfondimenti

Gli shell C e Korn sono progettati da esperti per gli esperti, perciò sono molto più potenti e versatili del Bourne shell; comunque, le estensioni si pagano con una maggiore complessità e difficoltà, specialmente quando lo shell interfaccia il sistema UNIX.

SELEZIONE DELLO SHELL DI LOGIN

Normalmente, quando entrate nel sistema, il sistema lancia il Bourne shell. Il Bourne shell inizia interpretando il vostro **.profile**, quindi visualizza un PS1 per richiedere l'introduzione di comandi. Questo shell in ascolto dei comandi viene detto *shell di login*, perché viene creato dal sistema quando vi collegate col login; quando usate un comando come **ksh** o **exec csh** nella linea di comando o nel vostro **.profile**, lanciate un subshell, che non è uno shell di login. Se usate sempre **csh** o **ksh**, potete migliorare le prestazioni del sistema e ridurre il numero di comandi nella vostra procedura d'inizializzazione, stabilendo come shell di login il vostro shell abituale. Per fare ciò dovete possedere i diritti di superuser, oppure richiedere l'intervento del gestore del sistema. Può essere utilizzata la procedura di gestione, oppure possono essere eseguite le semplici operazioni qui descritte.

L'ultimo campo della linea in **/etc/passwd** relativa al vostro login contiene il percorso assoluto dello shell da lanciare come shell di login. Per esempio:

```
[42]$ grep /etc/passwd
giorgio:x:102:1:Giorgio:/home/giorgio:/usr/bin/ksh
```

In questo caso, viene lanciato lo shell **ksh** invece del Bourne shell. Se preferite, potete lanciare lo shell **csh** con una linea di questo tipo:

```
giorgio:x:102:1:Giorgio:/home/giorgio:/usr/bin/csh
```

Non dovete copiare le linee di questi esempi; modificate con un editor la linea nel vostro file **/etc/passwd** per inserire il vostro shell prescelto.

Quando **ksh** è lo shell di login, legge il vostro **.profile** già esistente, quindi legge il file specificato nella variabile **ENV**, come già detto in precedenza; invece, quando **ksh** non è lo shell di login, legge il file in **ENV**, ma non il file **.profile**.

Il **csh**, al contrario, non usa **.profile**; legge, invece, un file **.login** nel vostro home directory, e lo interpreta come una procedura **csh**. Questo è necessario perché il linguaggio di programmazione **csh** è talmente diverso dal linguaggio degli shell Bourne e Korn, che **.profile** non potrebbe essere eseguito da **csh**. Il **csh** come shell di login, dopo il file **.login** legge il file **.cshrc**, già descritto in precedenza; se non è uno shell di login trascura il file **.login**. Quando usate **csh** come shell di login, potete creare un file col nome **.logout** nel vostro home directory; questo file viene letto ed eseguito da **csh** quando uscite dal sistema. Il file **.logout** può essere usato per cancellare file di lavoro, eseguire operazioni di manutenzione del software, fornire l'indicazione della durata della sessione, ecc. Gli shell Bourne e Korn non hanno questa caratteristica.

COMPLETAMENTO DI NOMI DI FILE

Gli shell C e Korn consentono la specificazione parziale di nomi di file nella linea di comando; lo shell ricerca nel directory specificato i file i cui nomi iniziano con la stringa introdotta, elenca tutti questi nomi al terminale, quindi ritorna al comando parzialmente introdotto. Questo consente di vedere tutti i file su cui il comando potrebbe agire, o di selezionare un gruppo di quei file per il comando. In **csh**, mentre state introducendo un nome di file, premete CTRL-D; **csh** visualizzerà tutti i file i cui nomi corrispondono per la parte che avete digitato; per esempio:

```
my_sys{giorgio}70: ls
ball        cribbage    go          greed       tetris
chess       gnu.go      gotool      netgo
my_sys{giorgio}71: ls g CTRL-D
gnu.go/     go/         gotool/     greed/
my_sys{giorgio}71: ls g
```

Quando premete CTRL-D, il **csh** sospende l'operazione in corso, elenca i nomi dei file identificati, quindi ritorna alla stessa linea di comando; potete continuare, completando il nome del file, aggiungendo un *, o altro, oppure annullare il comando.

In **ksh** la stessa caratteristica è disponibile tramite i modi di editor **vi** o **emacs**. Se state usando il modo **vi**, entrate in modalità comando con ESC, quindi premete il tasto =; vengono elencati i nomi dei file identificati e **ksh** ritorna alla linea di comando in corso; per esempio:

```
[66]$ ls g ESC =
1) gnu.go/
2) go/
3) gotool/
4) greed/
[66]$ ls g
```

Se state usando il modo **emacs**, usate il comando **M-=** (ESC uguale).

In **ksh** potete disabilitare il completamento dei nomi di file, con l'opzione "no global":

```
[72]$ set -o noglob
```

Per abilitarlo in **csh** usate:

```
my_sys{giorgio}71: set filec
```

e per disabilitarlo:

```
my_sys{giorgio}71: unset filec
```

ALTRI DETTAGLI SUL FILE ENV DI ksh

Quando la variabile **ENV** contiene il nome di un file che può essere letto, il **ksh** all'inizio dell'esecuzione legge e interpreta quel file. Il file di solito contiene definizioni di alias e altre caratterizzazioni per la vostra sessione interattiva; tuttavia, questo file non dovrà essere usato quando viene eseguita una procedura di shell, perché gli alias in esso definiti potrebbero non essere compatibili con la procedura. Per questo motivo, è opportuno predisporre le cose in modo che il file **ENV** non venga usato per i subshell, ma solo per gli shell interattivi (compreso lo shell di login). Questo può essere ottenuto definendo la variabile **ENV** in questo modo:

```
[73]$ ENV="\${-:+$HOME/.ksh.aliases\${- # # *i*}}"
```

Non dimenticate **export** della variabile **ENV**, dopo averla definita. Questa complessa operazione **ksh** stabilisce la variabile **ENV** al valore **$HOME/.ksh.aliases** per shell interattivi, ma la lascia indefinita per subshell; in questo modo il file d'inizializzazione verrà letto solo quando gli

shell sono interattivi. Questa linea di comando usa caratteristiche di **ksh** che non abbiamo descritto; consultate la man page **ksh(1)** se volete approfondire l'argomento.

DIRECTORY CORRENTE NEL PS1

L'indicazione del directory corrente nel prompt PS1 può risultare utile, in modo che quando cambiate directory con **cd**, il cambiamento viene evidenziato nel prompt. Questo non è possibile con il Bourne shell, mentre è molto facile con la versione SVR4 di **ksh**; semplicemente usate:

```
$ PS1 = '$PWD: '
```

Notate che sono obbligatori gli apici di protezione. La variabile **PWD** è gestita da **ksh** per quest'uso.

In **csh**, l'operazione è un poco più complicata; occorre ridefinire il comando **cd** con un alias che contiene una chiamata esplicita del comando **pwd**.

JOB CONTROL

In SVR4, sia **csh** che **ksh** includono il *job control* (controllo del lavoro), ovvero la capacità di spostare i comandi in esecuzione tra il *background* (attività in bassa priorità) e il *foreground* (attività interattiva), e di arrestarli e rilanciarli a seconda delle necessità. Il job control è stato in qualche maniera superato e duplicato dalle caratteristiche *shell layer*, *virtual console* e *X Window System* introdotte in SVR4; tuttavia, molti utenti considerano ancora il job control il modo più conveniente di gestire lavori di lunga durata lanciati dal terminale. Il Bourne shell supporta il job control se viene eseguito sotto il nome **jsh** (job shell), invece del normale **sh**.

Quando eseguite una linea di comando o lanciate un lavoro interattivo, l'input del terminale è bloccato fino a che quel lavoro non è completato, e nell'attesa non potete introdurre ulteriori comandi. Spesso è conveniente eseguire quel lavoro in background e riottenere il controllo del terminale; ugualmente, può essere opportuno riportare in foreground un lavoro in background, per controllarne l'esecuzione oppure introdurre dati dal terminale. Il job control consente queste operazioni all'interno del livello o finestra corrente. Tenete presente che in molte release SVR4, il job control **csh** non funziona sotto X Window System, mentre **ksh** e **jsh** funzionano come qui descritto.

Mentre avete in esecuzione un lavoro interattivo, potete premere i tasti CTRL-Z in sequenza, per sospendere, o arrestare, quel lavoro, e restituire il controllo allo shell, come in questo esempio:

```
[80]$ sleep 40
CTRL-Z
[1] + Stopped                    sleep 40
[81]$
```

Il lavoro viene arrestato; viene stampato un messaggio che include il numero di lavoro, **[1]** in questo caso (questo è il lavoro numero uno), lo stato del lavoro (**Stopped**) e la linea di comando (**sleep 40** in questo semplice esempio); quindi lo shell ritorna a chiedere altri comandi. Notate che mentre il comando è sospeso, il temporizzatore di **sleep** non viene arrestato, per cui se rilanciate questo lavoro dopo che sono trascorsi i 40 secondi, il lavoro termina immediatamente. Dopo avere sospeso il lavoro, potete introdurre altri comandi, lanciare altri lavori, o far ripartire il lavoro sia in background che in foreground.

Per ottenere l'elenco dei lavori in corso, inclusi tutti quelli in background e anche quelli sospesi, usate il comando **jobs**; esso elenca i vostri lavori nello stesso formato dell'esempio precedente:

```
[81]$ jobs
[1] + Stopped                    sleep 40
[82]$
```

Potete adesso usare i numeri di lavoro per controllare i vostri lavori.

Potete rilanciare un lavoro sospeso, con **bg** (background), se volete rilanciarlo come lavoro in background; con **fg** (foreground), se volete rilanciarlo come lavoro in foreground. Questi comandi si aspettano un argomento con un numero di lavoro preceduto dal segno %; per esempio:

```
[82]$ fg %1
sleep 40
```

Qui, viene visualizzata la linea di comando e il lavoro impegna il terminale, come lavoro interattivo. Il comando **bg** è simile, eccetto che mette il lavoro in background, come se fosse stato lanciato con **&** alla fine della linea di comando.

Potete usare la forma *%stringa*, invece di *%numero*, dove stringa identifica una linea di comando nell'elenco dei lavori, che comincia con stringa; oppure, la forma *%?stringa*, che identifica una linea di comando che contiene la stringa.

Potete sospendere un lavoro in esecuzione in background col comando **stop**, e rilanciarlo con **fg** oppure **bg**, a piacere; per esempio:

```
[83]$ (sleep 40;echo hello)&
[1]       12032
[84]$ jobs
[1] + Running                    (sleep 40;echo hello)&
```

```
[85]$ stop %?sleep
[1] + Stopped (signal)          (sleep 40;echo hello)&
[86]$ jobs
[1] + Stopped (signal)          (sleep 40;echo hello)&
[87]$ fg %1
(sleep 40;echo hello)
hello
[88]$ fg %1
ksh: fg: no such job
[89]$
```

Se chiedete di terminare la vostra sessione mentre avete dei lavori sospesi, gli shell rifiutano l'azione; per esempio:

```
[86]$ jobs
[1] + Stopped (signal)          (sleep 40;echo hello)&
[87]$ exit
You have stopped jobs
[88]$
```

Se questo accade, dovete esaminare l'elenco dei lavori e disporre o il completamento o la cancellazione dei lavori; se immediatamente richiedete una seconda volta di uscire dalla sessione, il **csh** o **ksh** cancellano i lavori sospesi e vi sconnettono dal sistema. Potete cancellare esplicitamente un lavoro con:

```
kill -signal %numero
```

dove *segnale* è il segnale che volete inviare al processo (di solito è omesso), e *%numero* specifica il numero di lavoro.

COPROCESSI SOTTO ksh

Il **ksh** fornisce il supporto per i *coprocessi*, ovvero processi interagenti che sono in esecuzione simultanea e possono comunicare e sincronizzarsi fra di loro. Questo consente di realizzare programmi shell molto sofisticati che possono lavorare efficientemente in ambienti di elaborazione parallela. Qui viene dato un esempio molto semplice.

Potete lanciare un coprocesso concludendo la linea di comando con **|&** (pipe, e commerciale). Come default, lo standard input di un coprocesso è connesso con lo standard output dello shell padre e il suo standard output è connesso con lo standard input dello shell padre; queste connessioni possono essere cambiate nella procedura di shell, se necessario. Un modo usuale di accedere all'input e output di coprocessi è l'utilizzo di **read -p**, che legge una linea dallo standard output del coprocesso, e di **print -p**, che

scrive una linea nello standard input del coprocesso. Sono supportate anche molte altre caratteristiche.

Per esempio, in una procedura di shell è spesso utile un ciclo **while** o **for** che legge una linea alla volta, ed esegue delle azioni su ciascuna linea, come nei comandi **grep** o **sed**. Se voi scrivete una procedura come la seguente:

```
while [ $x < 10 ] ; do
        read FLINE < file.in
        ...
        (elaborazione)
        ...
done
```

troverete che la linea **read** legge ogni volta la prima linea del file; questo non è esattamente quello che volevate. Di fatto, non vi è modo di risolvere questo problema con **/sbin/sh** o **csh**; tuttavia, la caratteristica dei coprocessi lo risolve perfettamente. La procedura seguente raggiunge lo scopo:

```
cat /etc/passwd |&
while true ; do
        read -p FLINE
        if [ -z "$FLINE" ] ; then
                exit 0
        fi
        echo $FLINE
done
```

Il coprocesso:

```
cat /etc/passwd |&
```

viene lanciato per primo, ma rimane bloccato in attesa fino a che lo shell corrente non effettua la lettura, con il comando:

```
read -p FLINE
```

che legge una linea dal coprocesso nella variabile **FLINE**; quando tutto l'output è stato letto, **read** restituisce una stringa vuota e la condizione di **if** provoca l'uscita dalla procedura.

Di fatto, questa procedura è una dispendiosa forma di **echo**; tuttavia, l'uso di coprocessi in una simile maniera è molto comune in programmi **ksh**.

Capitolo 17

Elaborazione di testi

Il sistema UNIX è stato uno dei primi sistemi operativi a disporre di programmi per l'elaborazione di testi e la preparazione di documenti. Durante gli anni Settanta il sistema UNIX si è dimostrato principalmente utile per l'elaborazione di documenti, anche perché in quel periodo non esistevano ancora i personal computer che hanno completamente rivoluzionato questo settore dell'informatica.

Il pacchetto software **troff** (typesetting run off) rappresentava, a quei tempi, uno strumento molto efficiente, che poteva essere usato per creare diversi tipi di documenti, da semplici lettere d'ufficio a prodotti editoriali di buon livello. In particolare, **troff** permette di pilotare una fotocompositrice e realizzare quindi originali su pellicola pronti per la stampa. Lo *UNIX User's Manual* è un esempio di documentazione realizzata interamente con il programma **troff** e con una fotocompositrice.

Ancora oggi gli strumenti **troff** sono insuperati per il trattamento di documenti complessi e di estese dimensioni, come i libri. Negli anni Ottanta, invece, con lo sviluppo dei personal computer e la crescente popolarità dell'automazione d'ufficio e di sofisticati word processor per personal computer, molti dei primi strumenti di elaborazione testi sotto UNIX sono stati abbandonati, specialmente nel campo della creazione di piccoli documenti. Tuttavia, molti esperti di **troff** lo considerano ancora lo strumento di elaborazione di testi più potente e flessibile disponibile in commercio e non pensano minimamente di adottare un altro elaboratore di testi.

Il maggiore inconveniente (e uno dei maggiori vantaggi) del pacchetto **troff** è insito nella sua filosofia di progetto: si tratta di un linguaggio di programmazione per la gestione dei testi e per il controllo dell'unità di stampa, che non impiega un modello WYSIWYG (what you see is what you get) come interfaccia utente; ovvero, non visualizza esattamente l'aspetto finale del testo. Al contrario, il testo deve essere preparato mediante un normale editor, come **vi**, e i comandi **troff** devono essere intercalati nel testo; il risultato è un file ASCII che contiene sia il testo che i comandi che determinano il formato e l'aspetto del testo finale. In una seconda fase, il file di testo viene elaborato (compilato) dall'impaginatore **troff** per produrre il testo finale. In questo caso, dunque, le operazioni di battitura del testo e di impaginazione sono due fasi separate e indipendenti, al contrario dei moderni word processor.

Come molti linguaggi di programmazione, **troff** è molto potente, ma può risultare difficile da apprendere, per cui è indispensabile un periodo di esperienza per utilizzare le direttive di impaginazione con i risultati migliori.

In molte versioni di sistema UNIX, gli strumenti per l'impaginazione dei testi sono disponibili come optional; se non vi occorrono le particolari caratteristiche di **troff**, o se preferite utilizzare uno strumento più moderno come quelli disponibili sotto MS-DOS, potete escludere l'acquisto del *Documenter's Workbench*. Nelle versioni di SVR4 che comprendono il pacchetto di compatibilità BSD, potrete spesso trovare una vecchia versione di **troff** sotto il directory **/usr/ucb**. I più popolari word processor sotto MS-DOS sono disponibili anche sotto UNIX e sono generalmente superiori a **troff**, specie nella grafica; d'altra parte, le più recenti versioni di **troff** sono state migliorate e offrono il supporto delle stampanti laser in linguaggio PostScript, oltre a consentire l'inclusione in un documento di elaborati grafici prodotti a parte.

17.1 Il comando spell

Il comando **spell**, presente generalmente nella versione standard di sistema UNIX, è un analizzatore di testo molto potente e intelligente. Dispone di un vasto database di radicali di parole e utilizza sofisticati algoritmi per produrre la voce plurale e altre desinenze della lingua inglese. Inoltre, consente di includere elenchi di parole, acronimi, nomi di persone e termini tecnici, accettati in base al sistema o all'utente, per limitare il numero di parole che **spell** considera sbagliate. Il comando **spell** non fa distinzione tra i caratteri maiuscoli e i caratteri minuscoli; è una procedura shell, contenuta nel directory **/usr/bin/spell**, il cui esame dettagliato può risultare molto istruttivo.

Il comando **spell** legge lo standard input o un elenco di nomi di file specificati come argomento e produce in uscita sullo standard output la lista di parole che ritiene scritte in modo errato; per esempio:

```
$ spell file
```

L'ingresso deve essere un normale file ASCII, per cui **spell** non è utilizzabile con la maggior parte degli elaboratori di testo WYSIWYG, che di solito intercalano nei file caratteri di controllo non in formato ASCII. Il comando **spell** suddivide l'ingresso in un elenco di parole e ricerca ogni parola in un elenco di parole ortograficamente corrette; ignora le direttive di impaginazione di **troff** e utilizza un algoritmo relativamente intelligente per determinare se una parola è scritta correttamente; per esempio:

```
echo behavior behaviour | spell
behaviour
$
```

Il comando **spell** adotta le regole ortografiche americane nell'analisi di testo, ma un'opzione attiva lo spelling secondo l'uso dell'inglese britannico; il seguente comando utilizza l'opzione **-b** (british):

```
$ echo behavior behaviour | spell -b
behavior
$
```

Oltre al vocabolario di parole predefinite, **spell** può riconoscere un elenco personale di altre parole, specificato sulla linea di comando. A questo scopo, utilizzate l'opzione **+** (più) seguita dal nome di un file che contiene un elenco di parole che, pur non rispondendo agli standard di **spell** non devono essere considerate errori:

```
$ cat spell.locale
behaviour
$ echo behavior behaviour | spell +spell.locale
$
```

Il file **spell.locale** può contenere un elenco, di qualsiasi dimensione, di parole disposte una per linea. Il comando **spell** accetta una sola opzione **+**; se intendete ricorrere a più elenchi, dovete utilizzare una struttura a pipeline:

```
$ spell +spell.locale in.file | spell +secondo.locale
```

Il secondo comando **spell** filtra la lista di parole che il primo comando **spell** considera scorrette da un punto di vista ortografico, per cui l'uscita non contiene le parole specificate in entrambi i file locali.

Mediante l'opzione **-v** (verbose), potete ricavare, oltre alla lista di parole che non sono comprese nel dizionario di **spell**, le tecniche che **spell** utilizza per ricostruire le parole derivate:

```
$ echo derive derivate derivative deriving | spell -v
-e+ion -ion+ive  derivative
-e+ing  deriving
$
```

I radicali *derive* e *derivate* appartengono al dizionario di **spell**, per cui non compaiono in uscita. La parola *derivative* viene costruita da *derivate* rimuovendo prima la lettera *e* (**– e**) e aggiungendo la stringa *ion* (**+** *ion*), poi rimuovendo la stringa *ion* e aggiungendo la stringa *ive*, ottenendo la parola *derivative*. Regole analoghe vengono adottate per derivare la parola *deriving* dal radicale *derive*.

17.2 Il pacchetto impaginatore di documenti troff

Il principale strumento di elaborazione di testi disponibile in ambiente di sistema UNIX è il pacchetto **troff**. Il termine *troff* si riferisce sia al linguaggio di programmazione che al comando del sistema UNIX di elaborazione di testi. Il linguaggio **troff** produce un'uscita adatta a pilotare direttamente le più popolari fotocompositrici e stampanti laser, essendo un linguaggio di programmazione ottimizzato per la fotocomposizione. Riceve in ingresso un file nel formato ASCII che contiene al suo interno le direttive di impaginazione e poi compila questo codice sorgente in un insieme di istruzioni che pilotano l'attività dell'unità di stampa. Il file prodotto in uscita non ha il formato ASCII perché consiste di un insieme di comandi nel linguaggio di controllo del dispositivo di stampa.

Il linguaggio **troff** non può generare un'uscita ASCII adatta per le stampanti a matrice di punti o per altre comuni stampanti perché produce dati di controllo per fotocompositrici; in alternativa, uno strumento denominato **nroff** (nontypesetting runoff) impagina i file in ingresso realizzando un'uscita compatibile con i terminali e le stampanti più semplici. La qualità dei risultati in uscita risulta necessariamente più scadente rispetto a quelli ottenuti con **troff**, ma **nroff** accetta in ingresso gli stessi file e la maggior parte degli argomenti di **troff**; perciò viene spesso impiegato per rivedere un documento al terminale. In ogni caso, l'impaginatore di testi **nroff** occupa un posto di rilievo nella famiglia **troff**, dal momento che le unità di stampa ASCII prevalgono ancora numericamente sulle nuove stampanti laser e molti calcolatori di ridotta capacità supportano solo **nroff**.

17.3 La linea di comando troff

Il comando **troff** legge il testo sorgente dallo standard input o da una lista di nomi di file indicati come argomento.

```
$ troff testo.sorg
```

L'uscita del comando **troff** avviene sullo standard output; non risulta però di grande utilità perché ha un formato adatto per pilotare una macchina

compositrice. Generalmente l'uscita viene ridiretta a un file o inserita in un pipeline in cui un altro comando è predisposto al controllo dell'unità di stampa; potete utilizzare **lp** se la stampante è collegata a una porta seriale o parallela.

Il comando **troff** accetta diverse opzioni, tra cui la più importante specifica il tipo di stampante. Le moderne versioni di **troff** seguono uno schema di filtraggio in cui l'uscita viene inviata a un programma specifico del dispositivo che converte i comandi di composizione nei comandi che l'unità di stampa può interpretare. Questa versione di **troff** è conosciuta come **ditroff** (*device-indipendent* **troff**), ma le sue opzioni non cambiano rispetto a quelle dell'altra versione. L'opzione **-T** (Typesetter) accetta un argomento che indica il nome del dispositivo:

```
$ troff -Tps file.sorg
```

Il termine **ps** indica in questo caso una stampante PostScript. Il comando **troff** esegue direttamente l'operazione di filtraggio, a partire dall'opzione **-T**, per cui gli utenti non devono creare nessun pipeline. Altre opzioni permettono di specificare il valore di alcuni dei registri interni **troff** e di limitare l'uscita ad alcune parti del documento. L'opzione **-a** (ASCII) genera un'approssimazione ASCII dell'uscita e si dimostra molto utile per visionare il documento su terminale prima di inviarlo alla macchina compositrice. Queste opzioni valgono anche in **nroff**, anche se il tipo di stampante specificato con l'opzione **-T** differisce per i due impaginatori di testi.

17.4 Il linguaggio di comando troff

In generale, le funzioni più importanti di **troff** sono le sue capacità di riempire (*fill*) e sistemare (*flow*) più linee di testo in più linee in uscita, per completare una pagina o una colonna, usando determinati tipi e dimensioni di caratteri. In altre parole si tratta di adattare la dimensione dei caratteri, la spaziatura tra le parole e altre direttive di impaginazione alle specifiche di stampa, modificando così in uscita la dimensione delle linee del file in input. Il processo di riempimento continua fino al verificarsi di un'interruzione (*break*), come per esempio la presenza nel testo in input di una linea vuota o di comandi che separano i paragrafi o introducono particolari intestazioni. Inoltre, **troff** riesce a effettuare la sillabazione e la giustificazione a pacchetto per la migliore presentazione della pagina.

Il programma **troff** permette agli utenti di crearsi proprie *macro* per le direttive di impaginazione di uso frequente. Una macro è una notazione abbreviata, dove un'unica semplice direttiva viene usata come sinonimo di una sequenza complessa di altre direttive. In realtà, poche persone adesso usano direttamente i comandi bruti di **troff**, in quanto è molto diffuso l'uso

di popolari pacchetti di macro già sviluppate e reperibili. Tratteremo di questi pacchetti di macro dopo avere illustrato le funzioni fondamentali di **troff**.

CONCETTI BASILARI DI troff

Una direttiva di **troff** è un comando intercalato al testo da impaginare. Le direttive servono a controllare il formato generale del testo da impaginare o a definire condizioni di lavoro per il programma stesso di impaginazione. Le direttive di **troff** appaiono all'inizio di una linea del file di ingresso, sono inframmezzate da linee che contengono il testo da impaginare e vengono identificate dal carattere . (punto) che occupa la prima posizione sulla linea. Tutti i comandi di **troff** si compongono di caratteri minuscoli che seguono immediatamente il punto e alcuni di essi accettano altri argomenti, separati tra loro e dal nome del comando da uno spazio.

Per esempio, per ottenere una linea vuota in uscita potete usare il comando **.sp** (space) nel punto ove volete la linea di spazi; se volete tre linee di spazi, usate il comando:

```
.sp 3
```

La maggior parte dei comandi **troff** è elencata nelle Tabelle 17.1, 17.2, 17.3.

UNITÀ DI troff

Il precedente comando istruiva **troff** di lasciare tre linee vuote in uscita; il pacchetto riconosce tre differenti forme di misurazione: linea, pollice, punto. Un *punto* è 1/72 di pollice; i punti vengono normalmente usati per specificare il corpo, o dimensione dei caratteri: un carattere di 12 punti, ovvero corpo 12, ha un'altezza di 12/72 di pollice. Potete specificare queste maniere alternative di dimensione aggiungendo uno dei suffissi – **i** per inch, **p** per punto, **l** (elle, per linea) – dopo l'argomento numerico, senza spazi intercalati. Per esempio, usando una spaziatura di linea di 12 pollici, questi comandi sono equivalenti:

```
.sp 2i
.sp 144p
.sp 12l
```

I vari comandi **troff** differiscono nei valori di default delle unità di misura, pertanto è buona regola specificare sempre le unità di misura quando usate questi comandi di definizione di distanze.

Le unità consentono anche l'indicazione di decimali, potete così specificare uno spazio di 6.5 pollici; per esempio:

```
.sp 6.5i
```

Tabella 17.1 Comandi **troff** principali: spaziatura, controllo pagina, riempimento.

Comando	Valore iniziale	Senza argom.	Causa break?	Spiegazione
.ps N	10 pt	prec	n	Corpo del carattere in punti
.ss N	12/36 em	ign	n	Dimensiona spazio-carattere
.cs F N M	off		n	Modo spazio-carattere costante
.bd F N	off		n	Annerisce il font F di N – 1 unità
.bd S F N	off		n	Annerisce il font speciale S con il font F corrente
.ft F	tondo	prec	n	Cambia al font F
.fp N F	R,I,B,S	ign	n	Monta il font F in posizione N (1-4)
.pl N	11 in	11 in	n	Dimensiona lunghezza pagina
.bp N	N = 1		s	Emette la pagina corrente, pagina seguente numero N
.pn N	N = 1	ign	n	Pagina seguente numero N
.po N	0	prec	n	Offset della pagina
.ne N		N = 1V	n	Necessita N spazi verticali (V = spazio verticale corrente)
.mk R	nullo	interno	n	Marca la posizione verticale corrente nel registro R
.rt N	nullo	interno	n	Ritorna (su) alla posizione verticale marcata
.br			s	Break di linea
.fi	riempimento		s	Riempie linee in uscita
.nf	riempimento		s	Non riempie, non giustifica, linee in output
.ad c	giustifica ambedue	giustifica	n	Giustifica linee in uscita con modo c
.na	giustifica		n	Non giustifica linee in uscita
.ce N	off	N = 1	s	Centra le successive N linee di testo in ingresso
.ll N	6.5 in	prec	n	Lunghezza della linea
.in N	N = 0	prec	s	Indenta
.ti N		ign	s	Indenta temporaneamente
.vs N	12 pt	prec	n	Spazio di linea verticale (V)
.ls N	N = 1	prec	n	Emette N – 1 V dopo ogni linea di testo in uscita
.sp N		N = 1V	s	Spazia la distanza verticale N nelle due direzioni
.sv N		N = 1V	n	Salva la distanza verticale N
.os			n	Emette in uscita la distanza verticale salvata
.ns	spazio		n	Disabilita la spaziatura
.rs			n	Risabilita la spaziatura

Tabella 17.2 Macro, registri, transcodifiche di **troff**.

Comando	Valore iniziale	Senza argom.	Causa break?	Spiegazione
.ta Nt	0.5 in	nullo	n	Dispone tab sinistra, eccetto se t = R (right) o C (centered)
.tc c	nullo	nullo	n	Carattere di ripetizione tab
.lc c		nullo	n	Carattere di ripetizione iniziale
.fc a b	off	off	n	Definisce a delimitatore di campo, b carattere di riempimento
.de xx yy		.yy = ..	n	Definisce o ridefinisce macro xx; termina alla chiamata di yy
.am xx yy		.yy = ..	n	Accoda a una macro
.ds ss stringa		ign	n	Definisce una stringa ss contenente stringa
.as xx stringa		ign	n	Accoda stringa alla stringa xx
.rm xx		ign	n	Rimuove la richiesta, macro, o stringa
.rn xx yy		ign	n	Rinomina la richiesta, macro o stringa xx in yy
.di xx		end	n	Dirige l'uscita a macro xx
.da xx			n	Dirige l'uscita e accoda a macro xx
.wh N xx		off	n	Definisce posizione di una trappola; se negativo significa posizione da fine pagina
.ch xx N		off	n	Cambia la posizione della trappola
.dt N xx		off	n	Definisce una trappola di diversione
.it N xx		off	n	Definisce una trappola di contatore linee in entrata
.em xx	nullo	nullo	n	Definisce xx fine di macro
.nr R N M			n	Definisce registro R e carica con N; autoincremento M
.af R c	arabo		n	Assegna il formato a R (c = 1,i,I,a,A)
.rr R			n	Rimuove il registro R
.ec c	\	\	n	Definisce il carattere escape
.eo	on		n	Disabilita il meccanismo di escape
.lg N	on	on	n	Modo legatura abilitato se N > 0
.ul N	off	N = 1	n	Sottolinea o corsivizza N linee input
.cu N	off	N = 1	n	Sottolineatura continua in **nroff**, come .ul in **troff**
.uf F	corsivo	corsivo	n	Sottolinea font definito con F
.cc c	;	;	n	Definisce carattere di controllo c
.c2 c			n	Definisce carattere di non-interruzione c
.tr abcd...	nullo		n	Transcodifica a in b, ecc. in uscita

Tabella 17.3 Sillabazione, titoli, condizioni, varie di **troff**.

Comando	Valore iniziale	Senza argom.	Causa break?	Spiegazione
.nh	sì		n	Non suddivide le parole
.hy N	sì	sì	n	Suddivide con modo N
.hc c			n	Carattere di suddivisione c
.hw parola		ign	n	Parole di eccezione
.tl 'sinistra' centro'destra'			n	Titolo in tre parti
.pc c	%	off	n	Carattere per il numero pagina
.lt N	6.5 in	prec	n	Lunghezza del titolo
.nm N M S I		off	n	Modo numero on/off; definisce i parametri
.nn N		N = 1	n	Non numera le N linee successive in uscita
.if c any			n	Se la condizione c è vera accetta in ingresso "any"
.if !c any			n	Se la condizione c è falsa accetta in ingresso "any"
.if N any			n	Se espressione N > 0 accetta "any"
.if !N any			n	Se espressione N < = 0 accetta "any"
.if 'stringa1' stringa2' any			n	Se stringa1 uguaglia stringa2 accetta "any"
.if !'stringa1' stringa2' any			n	Se stringa1 non uguaglia stringa2 accetta "any"
.ie c any			n	Parte if di **if-else**; tutte le forme precedenti
.el any			n	Parte else di **if-else**
.ev N	N = 0	prec	n	Switch di ambiente (push down)
.rd prompt		BELL	n	Legge dati da stdin
.ex			n	Esce da **troff**
.so file			n	Accetta input da file
.nx file		eof	n	File successivo
.pi programma			n	Uscita in pipe a programma (solo **nroff**)
.mc c N		off	n	Definisce carattere margine c e separazione N
.tm stringa		newline	n	Stampa stringa in stdout
.ig yy		.yy = ..	n	Ignora fino alla chiamata yy
.pm t		tutte	n	Stampa nomi e dimensioni di macro (t = solo totale)
.fl			n	Svuota il buffer in uscita
\!			n	Passa la linea invariata in uscita

Tuttavia, tutte queste misure vengono convertite in *unità di base* interne a **troff**, di conseguenza piccole distanze (inferiori a un punto) possono non essere esatte.

FORMATO DI PAGINA

Il programma **troff** prevede comandi per il controllo della dimensione totale della pagina, normalmente posti all'inizio del documento, anche se non è vietato cambiare le dimensioni della pagina all'interno del documento. Il comando **.ll** (line length) definisce la lunghezza della vostra linea di testo e il comando **.po** (page offset) definisce l'ampiezza del margine sinistro; assieme questi determinano l'ampiezza orizzontale della vostra linea di testo; per esempio:

```
.ll 6i
.po 1i
```

produce linee di testo di 6 pollici, con un margine sinistro di 1 pollice. Potete definire l'ampiezza verticale della pagina col comando **.ps** (page size); spesso non necessario, perché il default è di 11 pollici.

Potete sostituire questi valori di lunghezza della linea per una parte del vostro documento con l'indentazione (o rientro) di una linea o di un blocco di testo; viene modificato l'effettivo margine sinistro, ma non il destro. Il comando **.in** (indent) seguito da una misura indenta le linee di testo seguenti; potete usare anche un valore negativo, se occorre; per esempio:

```
.in 1i
il testo viene indentato di 1 pollice.
.in -1i
ritorna al margine sinistro precedente.
```

Potete usare **.ti** (temporary indent), per indentare solo la linea successiva al comando; potete specificare più di una linea con un numero dopo **.ti**; per esempio:

```
.ti 5
```

indenta le successive cinque linee, dopodiché ritorna al precedente margine di sinistra. Notate che questi comandi di indentazione causano un'interruzione (break), cosicché **troff** inizia una nuova linea quando li incontra.

Potete centrare la linea o le linee successive nella pagina con **.ce** (center), che, come **.ti**, può essere seguito da un argomento che specifica il numero di linee da centrare. In modo simile, il comando **.ul** (underline) predispone l'uso del carattere *corsivo* per la linea o gruppo di linee successivo; in **nroff** viene invece eseguita la sottolineatura. Il comando **.ce** causa un'interruzione, al contrario di **.ul**.

Potete sopprimere l'interruzione causata da molti comandi . usando nei comandi il carattere ' (apice) anziché il punto; per esempio:

```
.ce
```

causa l'interruzione, mentre:

```
'ce
```

non causa interruzione, con interessanti risultati che potrete verificare con un vostro testo di prova.

Potete includere linee di commento nel vostro documento sorgente **troff** ponendo la stringa \" all'inizio di una linea, oppure dopo un comando . (punto'); verrà ignorato tutto ciò che segue questa stringa fino alla fine della riga. Potete inserire commenti estesi su più righe dopo il comando **.ig** (ignore); tutte le linee seguenti saranno ignorate, fino al primo comando .. (punto punto). I commenti sono molto utili per documentare i programmi **troff**, come in tutti i linguaggi di programmazione.

CONTROLLO DEI CARATTERI E DEI FONT

Potete determinare la dimensione dei caratteri con **.ps** (point size) e lo spazio di interlinea con – **vs** (vertical spacing). Per esempio, per selezionare un font di carattere del corpo di 12 punti e uno spazio di interlinea di 14 punti, usate:

```
.ps 12p
.vs 14p
```

Notate che il valore di **vs** deve essere maggiore del valore di **ps**, altrimenti le vostre linee si sovrappongono e non risultano leggibili.

Il comando **.vs** consente un accurato controllo dello spazio di interlinea. Potete inoltre includere **.ls** (line spacing) all'inizio del documento, per lasciare linee vuote, della dimensione corrente, fra le linee del testo; per esempio:

```
.ls 2l
```

produrrà il testo con spazio d'interlinea doppio, mentre:

```
.ls 3l
```

produrrà il testo con spazio d'interlinea triplo.

Potete selezionare il font da usare con **.ft** (font); di solito sono disponibili quattro font, ma potete *montare* altri font quando occorre. Il comando **.ft R** produce un normale font tondo, o font 1; **.ft B** produce il font neretto, o font 2; **.ft I** produce il font corsivo, o font 3; **.ft CW** (constant-width) produce il font

4, un font a spaziatura costante utile per stampare linee di comando UNIX od output di terminale. Può essere usato anche il comando **.ft***n*, in cui *n* specifica il numero del font da usare (da 1 a 4). Il comando **.ft P** (previous) consente di ritornare al font usato in precedenza; ciò permette di scambiare temporaneamente font e ritornare al font precedente senza necessità di tenere conto del font corrente.

CONTROLLO DEL RIEMPIMENTO

Normalmente **troff**, prima di cominciare una nuova linea, riempie, o completa, la linea corrente, eseguendo i cambi di font, dimensioni di caratteri e così via secondo le richieste. Quando viene raggiunta la fine della linea (definita con il parametro **.ll**), **troff** avanza del valore di **vs** e ricomincia a riempire una nuova linea; tuttavia, il comando **.sp** causa un'interruzione, cosicché in presenza di questo comando **troff** inizia una nuova linea (dopo avere spaziato in avanti della corretta misura). È possibile causare un'interruzione senza spaziatura col comando **.br** (break). È possibile disabilitare il riempimento col comando **.nf** (no fill) e riabilitarlo dopo un qualsiasi numero di linee di testo con **.fi** (fill); questo consente di formattare manualmente una sezione del documento, escludendo ogni intervento di riorganizzazione da parte di **troff**. Il comando **.bp** (break page) provoca l'abbandono del resto della pagina corrente e l'inizio di una nuova pagina.

Potete richiedere a **troff** la giustificazione a destra del testo con **.ad b** (adjust), in modo che ogni pagina presenti un margine destro allineato; questo viene ottenuto inserendo spazi tra le parole e tra i caratteri, estendendo la lunghezza della linea fino a raggiungere lo spazio disponibile. In assenza di questo comando **troff** produce un margine destro sfrangiato, riempiendo ogni linea fino a che un'ulteriore parola non possa essere contenuta nello spazio restante. Potete annullare la giustificazione a destra col comando **.ad l** (adjust left margin) o con l'equivalente **.na** (no adjust). Quando la giustificazione a destra è abilitata, potrete ottenere un aspetto migliore delle linee abilitando anche la suddivisione automatica delle parole più lunghe col comando **.hy** (hyphenation); la suddivisione automatica può essere disabilitata con **.nh** (no hyphenation).

Questi pochi comandi sono già sufficienti per produrre molti documenti semplici, consentendo un controllo completo della pagina e dei caratteri usati. La Figura 17.1 mostra un breve file di testo che contiene un documento sorgente **troff**, la Figura 17.2 mostra il risultato dopo il trattamento con **troff**; potete consultare l'elenco dei comandi **troff** nelle tabelle per comprendere come è stato ottenuto questo risultato.

La maggior parte dei comandi esegue cambiamenti permanenti nell'ambiente **troff**, come è il caso di quelli che aggiornano lo stile e la dimensione dei caratteri. Altre direttive, invece, si applicano solo alla linea successiva e modificano l'interpretazione solo di quella linea, com'è il caso dei comandi **.ul**, **.ce** e **.ti**.

```
.sp 12
.ll 5i
.ps 14
.vs 16
Questo è un esempio di testo sorgente \fBtroff\f
.ps 6
e dei risultati prodotti.
.ps 14
\fItroff\f è in realtà
un \s+6linguaggio di programmazione\s-6 che
.ul
scrive
testo automaticamente,
e definisce i \fBfont\f e la spaziatura.
Consideriamo una semplice equazione:
.ps 11
\(*S(\*a\(mul\(*b)\(->\(if
.sp 4
e altre cose:
\s8\z\(sq\s14\z\(sq\s22\z\(sq\s36\(sq
\b'\(lt\(lk\(lb'\b'\(lc\(lf' x
\b'\(rc\(rf'\b'\(rt\(rk\(rb'
.sp 3
.ps 8
\fItroff\f può generare risultati su più colonne, numera le pagine, inserisce le
intestazioni,
.de bx
\(br\¦\\$1\¦\(br\l'¦0\(rn'\l'¦0\(ul'
..
.br
.ce
.bx "e può anche incorniciare le parole."
.vs 14
.br
\fItroff\f consente l'estensione attraverso
.ps 24
le macro
.ps 8
per cui la cornice viene creata definendo una macro.
.ps 10
Elenchiamo alcuni caratteri speciali:
\(34  \(ct  \(co  \(bu  \(dg
.ps 6
.br
Gli strumenti \fBtbl\f e \fBeqn\f permettono rispettivamente di formattare
tabelle ed equazioni matematiche.
```

Figura 17.1 Esempio di **troff**: documento sorgente.

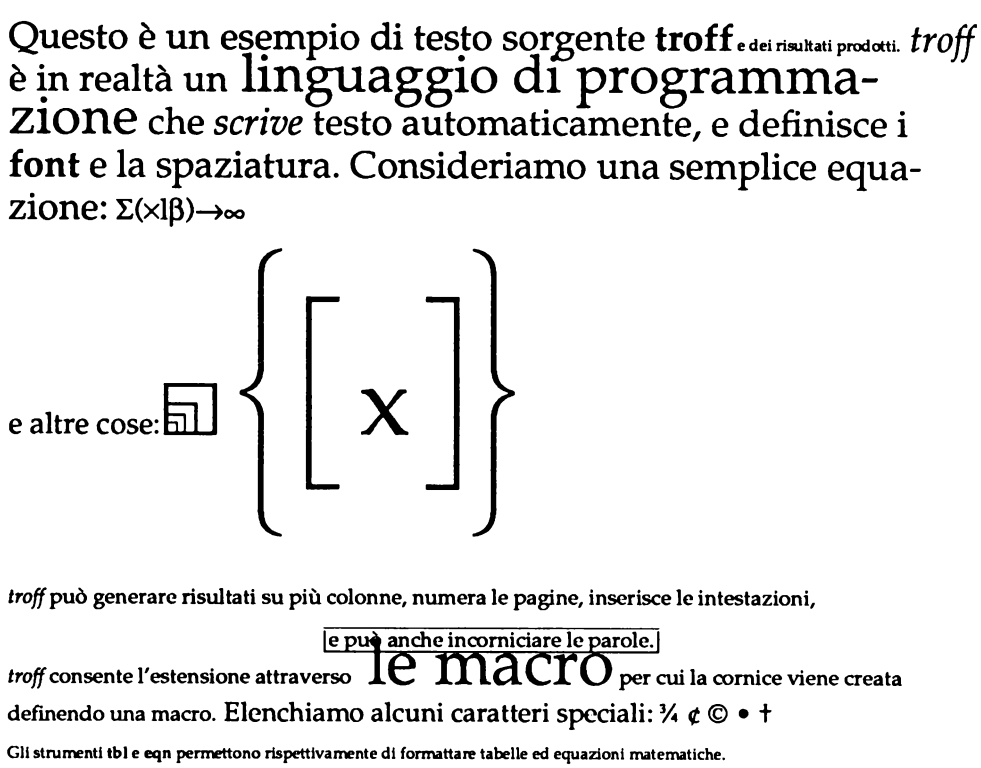


Figura 17.2 Esempio di **troff**: risultato in uscita.

Inoltre, alcuni comandi possono essere inseriti in una linea per agire su parti di testo interne alla linea; questi sono identificati da particolari sequenze di escape che iniziano con \ (barra inversa), per distinguerli dai normali caratteri di testo. Molte istruzioni di **troff** hanno anche una forma interna alla linea, per consentire di realizzare alcuni cambiamenti all'interno di una parola, senza causare spaziature o interruzioni. Questi comandi nella linea sono usati raramente nei semplici documenti e non saranno trattati ancora, tuttavia vengono usati in parte anche negli esempi in questo capitolo.

REGISTRI DI troff

Durante la sua esecuzione il programma **troff** memorizza informazioni numeriche e stringhe in *registri*; per esempio, la data corrente, il numero di pagina e una quantità di altre informazioni, sono disponibili in registri per vo-

stro uso. Alcuni registri sono solo in lettura, nel senso che potete usare il valore contenuto nel registro, ma non potete modificarlo; per esempio, potete leggere il valore corrente del corpo dei caratteri, in punti, col registro **.s**, ma non potete modificare direttamente il registro, dovete usare il comando **.ps**. Altri registri sono predefiniti da **troff**, ma potete modificarli quando vi occorre; per esempio, il numero di pagina corrente è contenuto nel registro %, che viene aggiornato correttamente da **troff** mentre riempie le pagine, tuttavia voi potete sempre modificarlo al valore che volete. Inoltre, **troff** prevede molti registri inutilizzati, che potete inizializzare e usare secondo le vostre necessità.

Se fate uso personale di vostri registri, dovete curare di non usare i registri *predefiniti*, ma di scegliere i vostri dal gruppo di quelli inutilizzati. La Tabella 17.4 elenca tutti i registri predefiniti.

I registri hanno nomi composti da uno o due caratteri minuscoli. Potete fare riferimento a un registro che ha il nome di un carattere con la sequenza speciale **\nx** (barra inversa n nome) e a un registro che ha il nome di due caratteri con la sequenza speciale **\n(xy** (barra inversa n parentesi aperta nome). Per assegnare un valore a un registro usate il comando **.nr** (number register). Per esempio, per assegnare il numero di pagina corrente al registro xx, usate:

```
.nr \n(xx \n%
```

Il nome del registro è il primo argomento di **.nr**, il secondo argomento è il valore da assegnare. Il linguaggio **troff** consente operazioni aritmetiche nel comando **.nr**; per assegnare a xx il numero di pagina corrente più due, potete usare:

```
.nr \n(xx \n% + 2
```

Sono supportate molte operazioni su interi, come incrementi e decrementi. Le operazioni aritmetiche sono eseguite da sinistra a destra, *non* tenendo conto della usuale precedenza aritmetica; potete usare le parentesi se occorre forzare le precedenze.

È possibile inizializzare un registro con un valore dalla linea di comando **troff**, usando l'opzione **–ra***N*, dove **a** è il registro (nome di un carattere) e *N* è il valore da assegnare. Per esempio, potete assegnare il valore 40 al numero di pagina iniziale del vostro documento, con:

```
troff –r%40 file
```

Quest'opzione è molto utile se nel vostro documento sorgente fate uso di registri personali.

Tabella 17.4 Registri numerici generali predefiniti di **troff**.

Nome	Sola lettura	Descrizione
%	n	Numero pagina corrente
ct	n	Tipo di carattere (definito da ct)
dl	n	Ampiezza massima dell'ultima diversione completa
dn	n	Dimensione verticale dell'ultima diversione completa
dw	n	Giorno della settimana corrente (1-7)
dy	n	Giorno del mese corrente (1-31)
hp	n	Locazione orizzontale corrente nella linea in ingresso
ln	n	Numero di linea in uscita
mo	n	Mese corrente (1-12)
nl	n	Posizione verticale dell'ultima linea-base di testo stampata
sb	n	Profondità della stringa sotto linea-base (generata da funzione ampiezza)
st	n	Altezza della stringa sopra linea-base (generata da funzione ampiezza)
yr	n	Ultime due cifre dell'anno corrente
.$	s	Numero di argomenti al livello macro corrente
.A	s	Definito a 1 in **nroff** e **troff** se argomento −a
.H	s	Risoluzione orizzontale disponibile, in unità base
.T	s	Definito a 1 in **nroff** se opzione −T; sempre 0 in **troff**
.V	s	Risoluzione verticale disponibile, in unità base
.a	s	Aumento di interlinea più recente con \x'N'
.c	s	Numero di linee lette dal file in ingresso corrente
.d	s	Locazione verticale corrente nella diversione corrente; uguale a nl se non c'è diversione
.f	s	Font corrente (1-4)
.h	s	Marca di limite della linea-base del testo nella pagina o diversione corrente
.i	s	Indentazione corrente
.l	s	Lunghezza di linea corrente
.n	s	Lunghezza della parte testo nella precedente linea in uscita
.o	s	Offset di pagina corrente
.p	s	Lunghezza di pagina corrente
.s	s	Corpo del carattere corrente in punti
.t	s	Distanza dalla prossima trappola
.u	s	In modo riempimento 1; in non-riempimento 0
.v	s	Spaziatura di linea verticale corrente
.w	s	Ampiezza del carattere precedente
.x	s	Registro riservato, dipende dalla versione
.y	s	Registro riservato, dipende dalla versione
.z	s	Nome della diversione corrente

17.5 I pacchetti di macro per troff

L'impiego diretto dei comandi di impaginazione di **troff** di fatto non è molto frequente; il linguaggio è difficile e tratta il documento a un livello così dettagliato che solo pochi esperti lo utilizzano. La maggior parte degli utenti, invece, si affida a pacchetti di macro, che mettono a disposizione semplici direttive da usare come sinonimo di sequenze di comandi **troff** che svolgono funzioni predefinite. I pacchetti di macro sono basati sulla possibilità di definizione di macro offerta da **troff**, trattata alla fine del capitolo. La definizione delle macro viene letta da **troff** quando inizia l'esecuzione; successivamente, ogni volta che il nome di una macro appare in un documento, **troff** lo sostituisce con la sequenza di direttive di cui quella macro rappresenta il sinonimo. Per esempio, l'intestazione dei documenti, che comporta la numerazione dei capitoli e il cambio di font, richiede l'esecuzione di molti comandi in **troff**; può invece essere risolta con poche direttive di semplice uso di un pacchetto di macro. Molti utenti utilizzano macro specifiche per i loro tipi di documenti, senza avere mai la necessità di usare direttamente comandi **troff**.

Esistono molti pacchetti di macro, ma solo tre hanno raggiunto grande diffusione. Il pacchetto **mm** (memorandum macros) permette di scrivere promemoria e lettere d'ufficio, il pacchetto **me** (macros for education) viene utilizzato nelle università per produrre rapporti tecnici a uso interno e, infine, il pacchetto **man** (manual) serve a produrre lo *UNIX User's Manual* e la documentazione associata. Un quarto pacchetto, **ms**, largamente utilizzato in passato, è stato in seguito sostituito da **mm**. Ogni pacchetto definisce una serie di comandi che possono essere usati in sostituzione della codifica diretta dei comandi **troff**, ma un documento può contenere, oltre alle istruzioni di richiamo di macro, anche normali direttive di **troff**. Per distinguere una macro da una normale direttiva si è adottata la convenzione di usare le lettere maiuscole per le macro, dato che i comandi **troff** hanno tutti nomi con lettere minuscole.

Il comando **troff** accetta come argomento il nome di un solo pacchetto di macro, preceduto dal carattere – (segno meno):

```
$ troff -man cmd.1
```

Questo esempio seleziona le macro **man** per impaginare la pagina di manuale **cmd.1**. Le macro differiscono sensibilmente nei diversi pacchetti e in un'esecuzione del comando **troff** o **nroff** potete selezionare solo un pacchetto di macro. Inoltre, i comandi dei diversi pacchetti di macro svolgono spesso funzioni equivalenti ed è facile perciò confondere i comandi dei vari pacchetti. Per esempio, il pacchetto **mm** utilizza il comando **.P** per iniziare un nuovo paragrafo, mentre la stessa funzione nel pacchetto **man** viene svolta dal comando **.PP**.

17.6 Le macro mm

Il miglior pacchetto di macro è il pacchetto **mm** (memorandum macro); consente la facile generazione di intestazioni e piè pagina, intestazioni di capitoli e di paragrafi, elenchi e schemi. Uno "schema" è un blocco di materiale che può "scorrere" nel testo fino al punto in cui **troff** decide di inviarlo in uscita; vale a dire che potete specificare che uno schema non deve causare un break immediato nel testo, ma deve piuttosto essere inserito in modo da apparire nel documento col migliore aspetto. Questa caratteristica può essere usata per posizionare tabelle e figure nelle pagine sinistre o all'inizio della pagina, oppure per occupare una pagina completa. Il pacchetto supporta anche l'indirizzamento di memo e lettere d'ufficio, estratti, copertine e indici del contenuto.

Il pacchetto **mm** è molto completo e ricco di possibilità; l'elenco completo delle macro e delle stringhe appare nelle Tabelle 17.5 e 17.6; la Figura 17.3 contiene un semplice modello per documenti che utilizzano le caratteristiche più importanti di **mm**.

COMANDI FONDAMENTALI

Con le macro **mm** potete specificare l'inizio di un paragrafo con **.P** (paragraph), creare una linea vuota con **.SP** (space), usare **.SP** *n* per saltare *n* linee vuote, iniziare una nuova pagina con **.SK** (skip).

Il tipo di paragrafo viene selezionato con un valore nel registro **.Pt** (paragraph type), come nell'esempio:

```
.nr Pt 1
```

Questo valore definisce un paragrafo indentato; usate il valore **2** per indentare ogni paragrafo, eccetto il primo successivo a un'intestazione; il default **0** definisce paragrafi non indentati.

La macro **.S** (size) sceglie il corpo del carattere in punti e la spaziatura verticale, che sono rispettivamente il primo e il secondo argomento della macro; per esempio:

```
.S 12 14
```

è equivalente a:

```
.ps 12
.vs 14
```

Selezionate la giustificazione a destra con **.SA 1**, oppure un margine destro non allineato con **.SA 0**; potete cambiare font con **.B** (bold=neretto), **.R** (roman=tondo), **.I** (italic=corsivo), **.CW** (constant-width), o **.P** (previous).

Tabella 17.5 Macro fondamentali di **mm**.

Nome	Descrizione
.P [tipo]	Paragrafo
.H livello [intestazione]	Intestazione, numerata
.HU intestazione	Intestazione, non numerata
.HM [arg1]...[arg7]	Stile di intestazione (numerazione araba o romana, lettere)
.HX dliv rliv intestazione	Uscita d'utente X prima della stampa dell'intestazione
.HZ dliv rliv intestazione	Uscita d'utente Z dopo la stampa dell'intestazione
.ND data	Aggiorna la data
.TL [incarico] [archivio]	Titolo di memorandum
.AF [società]	Formato alternativo del blocco "soggetto-data-mittente"
.AU nome [iniz] [loc] [dir] [int] [stanza]	Informazioni dell'autore
.TM [numero]	Numero del memorandum tecnico
.AS [arg] [indent]	Inizio di estratto
.AE	Fine di estratto
.OK [parole]	Altre parole per copertina di TM
.MT [tipo]	Tipo di memorandum
.SG [arg]	Linea della firma
.NS [arg]	Inizio di nota
.NE	Fine di nota
.CS [pag] [altro] [tot] [fig] [tab] [ref]	Copertina
.TC [slev] [spaz] [tlev] [tab] [int1]...[int5]	Indice del contenuto
.TX	Uscita d'utente dell'indice del contenuto
.PH 'sinistra'centro'destra'	Intestazione pagina
.OH 'sinistra'centro'destra'	Intestazione pagina dispari
.EH 'sinistra'centro'destra'	Intestazione pagina pari
.PF 'sinistra'centro'destra'	Piè pagina
.OF 'sinistra'centro'destra'	Piè pagina dispari
.EF 'sinistra'centro'destra'	Piè pagina pari
.BS	Inizio blocco finale
.BE	Fine blocco finale
.PX	Uscita d'utente per intestazione pagina
.TP	Macro d'inizio pagina
.I [arg corsivo] [arg font prec]	Corsivo (sottolinea in **nroff**)
.R	Ritorna a font tondo
.B [arg neretto] [arg font prec]	Neretto (sottolinea in **troff**)
.SP [linee]	Spaziatura verticale
.SK [pagine]	Salta pagine
.OP	Inizio di pagina con numero dispari
.2C	Uscita su due colonne
.1C	Uscita su una colonna
.SA [arg]	Definisce giustificazione del margine destro
.HC [ind div]	Carattere di suddivisione sillabica
.S [arg] [arg]	Corpo del carattere in punti e dimensioni della spaziatura verticale

Tabella 17.6 Liste, visualizzazioni, stringhe, di **mm**.

Nome	Descrizione
.AL [tipo] [indent.testo]	Inizio elenco con autoincremento
.BL [indent.testo]	Inizio elenco con pallino
.DL [indent.testo]	Inizio elenco con trattino
.ML marca [indent.testo]	Inizio elenco marcato
.RL [indent.testo]	Inizio elenco riferimenti
.VL indent.testo [indent.marca]	Inizio elenco d'elementi variabili
.LI [marca]	Elemento d'elenco
.LE	Fine elenco
.LB indent.testo indent.marca pad [marca]	Inizio elenco
.LC [liv.elenco]	Azzera stato dell'elenco
.DS [formato] [riemp.]	Inizio visualizzazione
.DF [formato] [riemp.]	Inizio visualizzazione variabile
.DE	Fine visualizzazione
.FG [titolo] [soppr.] [indic.]	Titolo di figura
.TS	Inizio tabella
.TE	Fine tabella
.TB [titolo] [soppr.] [indic.]	Titolo di tabella
.EQ [etich.]	Inizio visualizzazione d'equazione
.EN	Fine visualizzazione d'equazione
.EC [titolo] [soppr.] [indic.]	Intestazione d'equazione
.FS [etich.]	Inizio nota a fine pagina
.FE	Fine nota a fine pagina
.FD [arg]	Formato di default di nota a fine pagina
*(BU	Pallino
*F	Numeratore di nota a fine pagina
*(DT	Data corrente
*(HF	Elenco di font dell'intestazione (fino a sette codici per livelli 1-7)
*(HP	Corpo del carattere in punti dell'intestazione (fino a sette corpi in punti per livelli 1-7)
*(RE	Livello di release di MM

FORMATI DI MEMO PREDEFINITI IN mm

Il pacchetto **mm** fornisce una serie di macro per l'intestazione dei memo, e altre informazioni che compaiono all'inizio di molti documenti come: autore, titolo, estratto, copertina, ecc. Queste macro, se presenti, debbono apparire all'inizio del documento sorgente **mm**, con gli argomenti necessari

```
argomenti in linea di comando: -rC3 "DRAFT" e data a piè pagina
                               -rA1 simula macro .AF (stampa su lettera
                                    intestata)
                               -rB1 indice del contenuto [occorre .TC alla
                                    fine]

.so /usr/share/lib/tmac/tmac.m (oppure usare "troff -mm")
.ND "5 marzo 1991" (se omesso, data corrente)
 (usate \*(DT per inserire la data corrente ovunque nel testo)
= = = = = = = = = = = =
Questi non sono per memo; buoni per intestazione e piè pagina:
.PH "''SC* \\\\n(dy/ \\\\n(mo/ \\\\n(yr"
.PF "'- \\\\nP -''"
= = = = = = = = = = = =
.AF "stringa per logo" (Se omesso, stampa "AT&T" in alto a sinistra)
.TL "ZW 52040" "" (primo "" per modifica, secondo "" per riempimento)
Titolo del memo (Titolo del memo)
.AU "Giorgio" RP USWAT 9040301 555-1234 giorgio@my_sys.com
.SA 1 (causa giustificazione sinistra e destra del testo inserito)
.nr Df 4
.nr De 1 - - Questi due registri sono usati per ottenere una razionale disposi-
             zione di schemi variabili; forzano l'inserimento di ogni schema
             esattamente in una pagina (o più, se più lunghi) e in posizioni
             razionali.
.nr Hb 3
.nr Hs 3 - - Questi due registri definiscono il livello più basso di intestazione
             non incluso nella seguente linea di testo (default 2)
.ds HF 3 3 2 2 2 2 2 - - questo definisce il tipo di font per ciascun livello d'in-
                          testazione
                        1 = tondo
                        2 = corsivo (sottolineatura)
                        3 = neretto
.PM - - per indicazione diritti di proprietà:
        .PM P  Riservata; riproduzione vietata
        .PM N  Proprietà; firma di direttore
        .PM BP Proprietà; società consociate
        .PM BR Registrata; restrizioni in copertina
.AS 2
   questo pone una pagina di estratto all'inizio;
   il testo dell'estratto deve essere posto qui.
.AE (termina l'estratto)
.NS "solo estratto"          [linea "Copy (abstract only) to"]
oppure .NS                      [linea "Copy to"]
oppure .NS 1                    [linea "Copy (wiht att.) to"]
oppure .NS 2                    [linea "Copy (without att.) to"]
nome1
nome2
.NE      (termine elenco destinatari copie)
```

Figura 17.3 Modello per documenti con **mm**.

```
.MT "Memorandum per archivio" (tipo di memo, esattamente come scritto)
        .MT memo per archivio
        .MT "" tipo non memo e lettere interne
        .MT 3 note tecniche
        .MT 4 stile documenti distribuiti
        .MT 5 lettere esterne
IL TESTO NELLA FORMA ABITUALE VIENE INSERITO QUI
.H 1 "Questo è il titolo dell'intestazione"
Nel testo, le note a piè pagina (e le referenze bibliografiche in certe pagine) possono
venire generate e numerate automaticamente
utilizzando [1]
.FS
qui il testo delle note a piè pagina
.FE
Inserite "\*F" nel punto del testo dove deve apparire il numero della nota a piè pagi-
na. Usate ".FS *" per iniziare la nota a piè pagina con "*" invece di un numero (il
carattere di nota può essere usato nel testo come ogni altro carattere) e usate anche
"\*RF".
.RS AA
inserite qui il testo per le referenze da inserire alla fine del documento. "AA", "AB",
ecc. sono indicatori di sequenza dell'ordine delle referenze nell'elenco finale.
.RE
.SP (aggiunge una linea di spazi)
Inoltre, l'utente può accedere ai registri numerici, in questo modo:
.nr X 1 (registro X, inizializzato a 1)
(evitate di riusare nomi di registri predefiniti e nomi di più di una lettera)
quindi richiamare i registri nel testo con \nX (stampa il valore)
e modificarne il valore con
.nr X \nX+1 (questo assegna al registro X il valore X+1; sono ammesse anche altre
operazioni aritmetiche)

.SG dattilo (linea della firma; l'argomento è dattilografo/a)
```

Figura 17.3 Modello per documenti con **mm** (*continua*).

nell'ordine opportuno. La Figura 17.4 contiene un esempio di sorgente **mm**; la Figura 17.5 mostra il risultato dopo il trattamento con:

```
$ troff -mm lettera
```

La macro **.TL** (title) deve apparire nel documento prima delle macro .AU o .MT, ma può essere preceduta da comandi **troff**. Sulla stessa linea di **.TL** potete includere uno o due argomenti numerici opzionali, usati, nell'ordine, come *numero di modifica* e *numero di archiviazione*; se presenti, questi numeri generano in uscita linee aggiuntive. La linea successiva a **.TL** contiene il titolo del memo o della lettera.

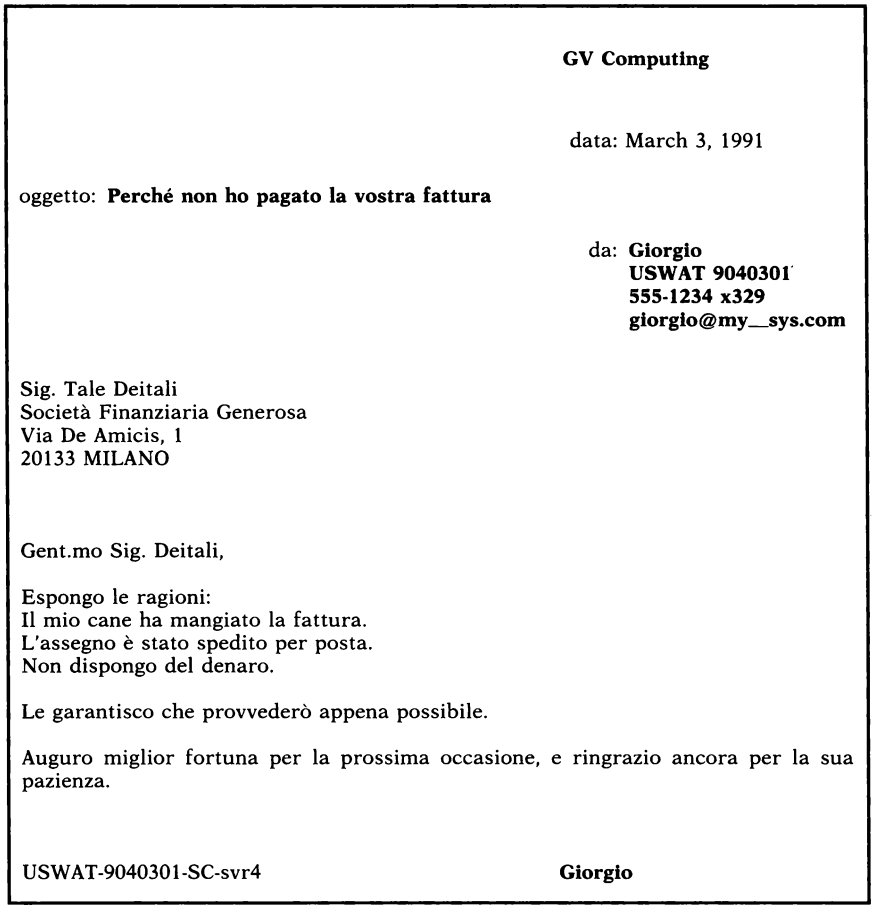
La linea **.AU** (author) specifica il nome dell'autore, le iniziali, la località, sigla dell'organizzazione, numero di telefono, stanza, e fino ad altri tre argo-

```
.AF "GV Computing"
.S 12
.TL
Perché non ho pagato la vostra fattura
.AU "Stefano" SC USWAT 9040301 329 555-1234 giorgio@my_sys.com
.SA 1
.MT ""
.SP 2
Sig. Tale Deitali
.br
Società Finanziaria Generosa
.br
Via De Amicis, 1
.br
20133 MILANO
.SP 2
Gent.mo Sig. Tale,
.SP
Espongo le ragioni:
.AL
.LI
Il mio cane ha mangiato la fattura.
.LI
L'assegno è stato spedito per posta.
.LI
Non dispongo del denaro.
.LE
.SP
Le garantisco che provvederò appena possibile.
.SP
Auguro miglior fortuna per la prossima occasione, e ringrazio ancora per la sua
pazienza.
.SG svr4
```

Figura 17.4 Semplice documento sorgente **mm**.

menti; se presenti, questi dati appariranno in varie parti del documento, specialmente nella copertina, nell'intestazione e nella riga della firma. Solo il nome è obbligatorio, tutti gli altri argomenti sono opzionali.

La linea **.MT** (memo type) specifica in argomento il formato o tipo di memo che verrà usato da **mm** per organizzare il documento; questa macro deve essere sempre presente se sono presenti le macro **.TL** e **.AU**. Il tipo di memo viene stampato all'inizio del documento; se l'argomento è una stringa, viene stampata la stringa stessa; se l'argomento è una stringa vuota ("") il tipo di memo non viene stampato. Se l'argomento di **.MT** è il numero **1**, viene stampato "Memorandum for file", se l'argomento è **2** viene stampato "Programmer's Notes", se è **3**, "Engineer's Notes"; l'argomento **4** produce



GV Computing

data: March 3, 1991

oggetto: **Perché non ho pagato la vostra fattura**

da: **Giorgio**
USWAT 9040301
555-1234 x329
giorgio@my_sys.com

Sig. Tale Deitali
Società Finanziaria Generosa
Via De Amicis, 1
20133 MILANO

Gent.mo Sig. Deitali,

Espongo le ragioni:
Il mio cane ha mangiato la fattura.
L'assegno è stato spedito per posta.
Non dispongo del denaro.

Le garantisco che provvederò appena possibile.

Auguro miglior fortuna per la prossima occasione, e ringrazio ancora per la sua pazienza.

USWAT-9040301-SC-svr4 **Giorgio**

Figura 17.5 Semplice documento **mm**: risultato in uscita.

un memo in stile *documento in distribuzione*; l'argomento **5** produce un documento nello stile di lettera esterna; ciascuna di queste scelte produce un memo o una lettera con un aspetto diverso.

Nel testo potete usare le macro **.P** o **.SP**, od ogni altra macro **mm**, o anche comandi **troff** necessari per dare al vostro documento il formato voluto; se volete, potete omettere la sequenza **.TL .AU .MT** e formattare l'inizio del documento manualmente con comandi di base **troff** o **mm**. Questo è quanto è stato fatto in Figura 17.5 per ottenere un indirizzo con break e linee di spaziatura; tuttavia, è sempre preferibile usare per quanto possibile uno dei formati di memo, perché si hanno disponibili utili caratteristiche e si riduce la possibilità di errori.

Se avete utilizzato le macro **.TL .AU .MT** potete inserire **.SG** (signature) alla fine del documento, per la linea della firma; la macro può avere un argomento con la sigla abitualmente usata per indicare nei memo o lettere la/il dattilografa/o.

NOTAZIONE PER ESTRATTI E "Copy to"

Potete delimitare parti di testo con **.AS** (abstract start) e **.AE** (abstract end) per produrre un blocco separato col titolo **Abstract**; tali estratti appaiono spesso all'inizio di documenti scientifici e tecnici. La macro **.AS** di solito non richiede argomenti, ma può accettare argomenti opzionali che interagiscono con la macro **.MT**; questi argomenti (**0 1 2**) determinano se l'estratto deve comparire sulla pagina uno del documento, sulla copertina, o su entrambi.

Racchiudendo un elenco di nomi tra **.NS** (notation start) e **.NE** (notation end) otterrete un'intestazione "Copy to:" seguita dall'elenco dei nomi. Fornendo un argomento a **.NS** potete modificare l'intestazione: con l'argomento **1** otterrete "Copy (with att.) to:"; con **2** otterrete "Copy (without att.) to:"; con

```
.NS "stringa"
```

otterrete "Copy (*stringa*) to:". Sono ammessi anche vari altri argomenti.

INTESTAZIONI

All'interno di un memo, **mm** consente di formattare intestazioni, come titoli di capitoli, paragrafi, sezioni; le macro possono rinumerare automaticamente le intestazioni, in modo che non le dobbiate rinumerare manualmente se dovete riorganizzare un documento. Il sistema consente di stabilire all'inizio di un documento convenzioni di font e dimensione dei caratteri, che verranno applicate al momento della stampa delle intestazioni.

Usate la macro **.H** (heading) per identificare un'intestazione nel vostro documento; la macro richiede un argomento numerico per il livello, seguito da una stringa protetta tra apici, che costituisce il testo dell'intestazione; per esempio:

```
.H 1 "Word processing"
.H 2 "Il linguaggio comandi di troff"
.H 3 "Concetti basilari di troff"
.H 3 "Unità troff"
.H 2 "La linea di comando di troff"
.H 2 "Inizio di un memo"
.H 2 "Approfondimenti"
.H 3 "Struttura del directory di troff"
.H 3 "Writer's Workbench
```

Naturalmente, tra queste intestazioni saranno incluse parti di testo. Questo esempio produrrà una gerarchia di sottointestazioni, ciascuna numerata in base al livello; vale a dire, se queste intestazioni sono nel Capitolo 17, la linea "Unità troff" dopo la formattazione apparirà come:

```
17.1.2 Unità troff
```

Le intestazioni sono organizzate gerarchicamente, dal primo livello che è il più importante, fino al settimo livello che è il meno importante; tutte le intestazioni allo stesso livello vengono numerate sequenzialmente a partire da uno, fino al primo comando **.H** con un livello inferiore.

Potete specificare il formato di ciascun livello di intestazione caricando i valori in vari registri **mm**, come per esempio:

```
.nr Hb 3
.nr Hs 3
.ds HF 3 3 2 2 2 2 2
.ds HP 14 12 12 12 12 12 12
```

I registri **Hb Hs** insieme stabiliscono il più basso livello di intestazione da separare dal testo contiguo; intestazioni di livello inferiore saranno unite alla successiva linea di testo. Il registro **HF** specifica il tipo di font per ciascuno dei sette possibili livelli d'intestazione; il valore **1** indica il font tondo, **2** il corsivo o sottolineato, **3** il neretto. Il registro **HP** (heading points) stabilisce il corpo del carattere per ciascun livello di intestazione, nell'esempio 14 punti per il livello uno, 12 punti per tutti gli altri livelli.

La macro **.HU** permette di specificare un'intestazione non numerata; è richiesto l'unico argomento col testo dell'intestazione protetto fra apici.

DISPLAY

Un *display* (in seguito schema) è una sezione di un documento che deve essere tenuta in un unico blocco, ma che non necessariamente deve apparire in uscita nel punto esatto dove è stata definita in input. Generalmente, si richiede che **troff** completi correttamente le righe sia prima che dopo lo schema, senza forzare un break allo schema stesso; ugualmente, si può richiedere che lo schema compaia in un'unica pagina e non sia mai spezzato tra due pagine in uscita. Il pacchetto **mm** permette di definire diversi tipi di schemi.

Racchiudete gli schemi fra **.DS** (display static) oppure **.DF** (display float) e **.DE** (display end); il materiale racchiuso fra i due comandi verrà mantenuto assieme in un'unica pagina in uscita. La macro **.DS** definisce uno schema statico che viene inviato al più presto possibile in uscita, comparendo nella stessa posizione relativa rispetto all'input; poiché uno schema non deve

uscire dai limiti della pagina, si potrà avere come risultato uno spazio vuoto a inizio o fine pagina, se lo schema non occupa l'intera pagina. Il comando **.DS** accetta argomenti per definire il formato e l'indentazione dello schema:

.DS [*formato*] [*riempimento*]

formato è un numero: **0** non produce indentazione,.**1** produce l'indentazione standard, **2** centra ciascuna linea dello schema, **3** centra l'intero schema in un unico blocco; *riempimento* può essere **0** per il modo non-riempimento all'interno dello schema, **1** per il normale modo riempimento.

La macro **.DF** definisce l'inizio di uno schema mobile, che può apparire in uscita dopo il testo che lo precede nel file sorgente, in dipendenza degli argomenti di **.DF** e del conteggio da parte di **troff** dello spazio disponibile nella pagina corrente. Sono accettati gli stessi argomenti di **.DS**, ma il comportamento di **.DF** dipende da due registri che vengono abitualmente caricati all'inizio del documento. Per esempio, con i comandi

```
.nr De 1
.nr Df 4
```

potete richiedere che ogni schema appaia da solo in una pagina separata, senza altro testo nella stessa pagina. Caricate il registro **De** con **1** per ottenere un salto pagina dopo ogni schema; caricate **0** se non richiedete nessun trattamento particolare. Il registro **Df** può essere caricato con: **0**, che trattiene tutti gli schemi fino alla fine della sezione con intestazione di primo livello; **2**, che produce lo schema all'inizio della pagina successiva; **3**, che produce lo schema nella pagina corrente se lo spazio è sufficiente, in caso contrario nella pagina successiva; **5**, che produce tanti schemi quanti ne possono essere contenuti nella pagina corrente.

In una coppia **.DF .DE** o **.DS .DE** può essere incluso materiale di qualunqe tipo, ossia testo, tabelle, grafici, ecc., inoltre **troff** può gestire qualunque numero di schemi sia necessario, e produrli in uscita secondo le regole esposte, senza particolare intervento da parte dell'utilizzatore.

ELENCHI

Il pacchetto **mm** comprende un eccellente supporto degli elenchi, inclusa la capacità di gestire la formattazione degli elenchi e la numerazione degli elementi di un elenco senza intervento dell'utilizzatore. Questo significa che non occorre numerare o formattare manualmente gli elenchi, e che è possibile aggiungere o cancellare elementi; automaticamente **mm** formatta e numera l'elenco.

Un elenco può iniziare con **.AL** (alphabetic list), **.BL** (bulleted list), **.DL** (dashed list); i tipi differenti si riferiscono al contrassegno che compare davan-

ti a ogni elemento dell'elenco: un carattere alfabetico o un numero, un pallino, o un trattino, rispettivamente. Ogni elemento dell'elenco deve essere preceduto da **.LI** (list item); un elenco, di qualunque tipo, deve terminare con **.LE** (list end). Per esempio:

```
.AL A
.LI
Elemento dell'elenco – – può comprendere un qualsiasi numero di linee.
.LI
Altro elemento – – mm riempie le linee all'interno di ciascun elemento.
.LI
Ancora un altro elemento – – all'interno dell'elemento può essere inserito
qualunque normale comando troff o macro mm.
.LE
```

La macro **.AL** richiede un argomento che specifica il tipo di contrassegno, l'argomento indica il primo elemento della sequenza prescelta; nell'esempio precedente, il primo elemento sarebbe contrassegnato *A*, il secondo *B* e così via. Se l'argomento di **.AL** fosse stato **6**, il primo elemento della lista sarebbe stato contrassegnato con *6*; se l'argomento viene omesso, gli elementi vengono contrassegnati a partire da *1*. Le macro **.BL** e **.DL** non richiedono un argomento per il contrassegno, ma accettano un argomento che stabilisce l'indentazione dell'elenco. Anche la macro **.LI** accetta un argomento che sostituisce il normale contrassegno; per esempio:

```
.LI +
```

contrassegna il corrispondente elemento della lista con **+**, indipendentemente dal tipo di elenco.

NOTE A PIÈ PAGINA, RIFERIMENTI, INDICE DEL CONTENUTO

Le macro **mm** gestiscono le note e i riferimenti in maniera simile agli schemi; quando nel corpo del testo compare una nota o un riferimento, **mm** ne tiene conto in un registro, producendo le note alla fine della pagina stessa e memorizzando i riferimenti per la fine del documento. La notazione ***F** viene usata per reperire il numero di nota corrente, la stringa può essere usata anche nel testo per fare riferimento a una nota; il testo della nota può essere collocato nel normale documento sorgente in **troff**, racchiuso tra **.FS** (footnote start) e **.FE** (footnote end). La macro **.FS** accetta un argomento come *etichetta* da apporre all'inizio della nota, in assenza dell'argomento viene assunto il numero di nota corrente. La coppia **.FS .FE** col testo della nota interposto deve seguire immediatamente l'etichetta o il numero della nota nel documento sorgente (dalla prima riga successiva), per garantire il cor-

retto trattamento della nota, specie nel caso dovesse cadere vicino alla fine di una pagina del documento in uscita.

Le referenze vengono trattate in modo simile, eccetto che vengono conservate fino alla fine del documento, e a quel momento prodotte in uscita in un unico gruppo, nell'ordine corretto. I riferimenti vanno racchiusi tra **.RS** (reference start) e **.RE** (reference end). Ciascun riferimento viene numerato automaticamente; potete fare riferimento al numero di riferimento successivo con la stringa ***(Rf**; introducete il testo del riferimento immediatamente dopo avere menzionato il riferimento nel testo del documento.

Mentre **troff** tratta ciascuna intestazione in un documento, può mantenere un elenco delle intestazioni per l'indice del contenuto del documento. Potete richiedere la stampa dell'indice del contenuto includendo la macro **.TC** alla fine del documento, se avete incluso l'argomento **–rB1** nella linea di comando di **troff**, per richiedere la memorizzazione delle intestazioni. La macro **.TC** accetta diversi argomenti che specificano il formato e la spaziatura dell'indice del contenuto; raramente sono utili. Come default, **.TC** tratta tutte le intestazioni del documento, ma potete controllare il livello delle intestazioni da trattare caricando il registro **Cl** (contents level); per esempio:

```
.nr Cl 3
```

include nell'indice del contenuto solo le intestazioni dal livello uno al livello tre inclusi.

INTESTAZIONI DI INIZIO E DI FINE PAGINA

Con le macro **mm** potete agevolmente generare il contenuto delle intestazioni di inizio e di fine pagina. La macro **.PH** genera l'intestazione d'inizio pagina, la macro **.PF** l'intestazione di fine pagina; sono disponibili anche le macro **.EH** (even-page header) e **.OH** (odd-page header) per l'inizio pagina pari e dispari, **.EF** (even-page footer) e **.OF** (odd-page footer) per la fine pagina pari e dispari, rispettivamente.

Ogni macro di questo gruppo accetta un argomento con un formato in tre parti; le tre parti stabiliscono il contenuto delle sezioni sinistra, centrale e destra delle intestazioni d'inizio e fine pagina. La parte sinistra viene giustificata a sinistra, la parte centrale viene centrata e la parte destra viene giustificata a destra. Le tre parti dell'argomento di tutte queste macro sono separate da un apice ('); per esempio:

```
.EH 'Capitolo \n(H1'' YABU'
.OH '\n(dy/\n(mo/\n(yr''Revisione 2'
.PF '' – % – ''
```

La parte sinistra dell'intestazione delle pagine con numero pari conterrà **Capitolo** *n*, dove *n* è il numero del capitolo corrente; ricordate che la stringa

\n(H1 in **mm** richiama il valore del registro **H1**, che contiene il valore corrente dell'intestazione di primo livello. La parte sinistra dell'intestazione delle pagine dispari conterrà la data corrente; la parte centrale dell'intestazione di fine pagina conterrà il numero della pagina corrente, richiamato da %.

17.7 Le macro man

Il pacchetto di macro **man** ha permesso di produrre l'intero *UNIX User's Manual*. Quasi tutto il nuovo software per il sistema UNIX, sia quello professionale sia quello di dominio pubblico, include almeno una man page che ne descrive la funzione. Quando il software viene installato sulla macchina, queste man page vengono spesso collocate su disco rigido, generalmente sotto il directory **/usr/man** o in un subdirectory. Le man page hanno di solito un formato che si adatta a **troff**, ma alcuni progettisti forniscono anche delle versioni che possono essere lette dal comando **cat**. Per stampare questa documentazione in formato sorgente sono necessari il pacchetto **troff** e le macro **man**; se la vostra macchina non dispone di un pacchetto **troff**, ma contiene le man page, potete cancellarle per rendere disponibile lo spazio su disco.

Per apprendere l'uso delle macro **man** la maniera ormai tradizionale è quella di assumere a modello una man page esistente e modificarla secondo le proprie necessità. La Tabella 17.7 contiene l'elenco delle macro **man**.

17.8 Approfondimenti

Come la maggior parte dei linguaggi di programmazione, gli strumenti di **troff** forniscono l'opportunità per sviluppare applicazioni di elevata complessità, ma quanto abbiamo trattato finora non pretende di essere una guida all'uso di **troff**. Per conoscere a fondo **troff**, dovete studiare il dettagliato manuale di utente che è fornito col pacchetto *Documenter's Workbench*. In questo paragrafo esamineremo alcuni problemi che sono associati con l'elaborazione di testi.

IL COMANDO tbl

Il pacchetto *Documenter's Workbench* comprende numerosi strumenti, tra cui spicca il comando **tbl** (tables), un preprocessore dei file di **troff** che produce le direttive di impaginazione per la preparazione di tabelle multicolonna con titoli, cornici e didascalie. Il comando **tbl** stabilisce automaticamente il corretto posizionamento delle colonne della tabella, sollevando così l'utente dal compito di tracciare manualmente la tabella. Esiste a questo sco-

Tabella 17.7 Macro di **man**.

Comando	Senza argom.	Causa break?	Spiegazione
.TH n c x v m		s	Inizio pagina n del capitolo c. x = testo centro piè pagina v = testo sinistra piè pagina m = testo centro intestazione pagina
.TX t p		n	Risolve l'abbreviazione di titolo t; unisce alla punteggiatura p
.SH t	linea succ.	s	Sotto intestazione
.PD d	.4v	n	Definisce distanza tra paragrafi d
.PP		s	Inizio paragrafo; definisce indentazione a .5i
.LP		s	Identico a .PP
.RS i	indentazione	s	Inizio indentazione relativa
.RE		s	Fine indentazione relativa
.TP i	indentazione	s	Definisce indentazione i; inizio paragrafo indentato
.HP i	indentazione	s	Definisce indentazione i; inizio paragrafo pendente
.IP x i	indentazione	s	Identico a .TP con identificatore x
.DT	.5i	n	Ripristina tabulazioni di default
.SM t	linea succ.	s	Stampa il testo t più piccolo di due punti
.SB t		n	Stampa t in neretto piccolo
.B t	linea succ.	n	Stampa il testo t in neretto
.I t	linea succ.	n	Stampa il testo t in corsivo
.BI t	linea succ.	n	Stampa t alternativamente in neretto e in corsivo
.BR t	linea succ.	n	Stampa t alternativamente in neretto e in tondo
.IB t	linea succ.	n	Stampa t alternativamente in corsivo e in neretto
.IR t	linea succ.	n	Stampa t alternativamente in corsivo e in tondo
.RB t	linea succ.	n	Stampa t alternativamente in tondo e in neretto
.RI t	linea succ.	n	Stampa t alternativamente in tondo e in corsivo

po uno speciale insieme di direttive di definizione del formato delle tabelle, che il comando filtro **tbl** traduce in sequenze di direttive elementari **troff**. L'uscita di **tbl** viene passata alla stampa attraverso **troff** o **nroff**:

```
$ tbl doc.sorg ¦ troff -mm -Tps ¦ lp
```

I comandi **tbl** sono inclusi nei file sorgenti **troff**; **tbl** legge ed elabora solo i propri comandi e passa il resto del documento sorgente invariato a **troff**.

Il programma di utilità **tbl** è un normale comando e non un pacchetto di macro in quanto **troff** può di fatto utilizzare un solo pacchetto di macro alla volta; invece di inserire il comando **tbl** in ogni pacchetto di macro, risulta più semplice impiegare **tbl** come programma filtro.

Se usate **mm**, troverete conveniente collocare le tabelle all'interno di uno "schema", per controllarne la posizione nel documento stampato. I comandi **mm** includono le macro **.TS** (table start) e **.TE** (table end) per migliorare il trattamento delle macro, specialmente di quelle che possono estendersi su più pagine in uscita; usando **mm** potrete codificare le tabelle con:

```
.DF 2
.TS
(contenuto di tbl)
.TE
.DE
```

Tuttavia, le macro **mm** non invocano **tbl** esplicitamente, dovrete quindi usare sempre la linea di comando già mostrata.

Le tabelle vengono definite da: una sezione opzioni, che contiene opzioni globali della tabella; una sezione formato, che specifica l'aspetto della tabella; una sezione dati, che contiene i contenuti della tabella. La sezione opzioni è la prima e definisce lo stile della tabella. La sezione formato contiene i comandi che definiscono il numero di colonne che compongono la tabella, l'allineamento dei dati, ecc. La sezione dati contiene una linea per il contenuto di ogni riga della tabella; i singoli elementi di dati nella riga (ovvero il contenuto di ogni colonna nella tabella) sono separati da caratteri *tab*. Il numero di colonne è dichiarato nella sezione formato; questa dichiarazione deve trovare corrispondenza nel numero di elementi in ciascuna riga della sezione dati.

USO DI tbl

La Tabella 17.8 elenca la serie completa dei comandi **tbl**, e la Figura 17.6 contiene il codice sorgente **tbl** per una semplice tabella a tre colonne. Avete definito lo stile della tabella con le opzioni **center** e **box**, per specificare che la tabella deve essere incorniciata e centrata nella pagina; tutte le altre possibili opzioni sono elencate nella Tabella 17.8. Potete usare un numero qualsiasi di queste opzioni, quante ne occorrono per definire la vostra tabella, ma tutte le opzioni devono comparire nella stessa linea che segue il comando **.TS** e terminare col carattere ;. Potete omettere l'intera linea delle opzioni, se non avete esigenze particolari.

La sezione formato determina l'aspetto della tabella; definisce il numero delle colonne, il corpo dei caratteri, se i dati debbono essere giustificati a

Tabella 17.8 Comandi e parole di **tbl**.

Nome	Tipo	Descrizione
allbox	opzioni	Incornicia tutti gli elementi
box	opzioni	Incornicia la tabella
center	opzioni	Centra la tabella nella pagina
doublebox	opzioni	Doppia incorniciatura della tabella
expand	opzioni	Tabella di ampiezza pari alla linea
tab(x)	opzioni	Definisce separatore dati x
|	formato	Linea verticale
||	formato	Doppia linea verticale
^	formato	Distanza verticale
\^	formato	Distanza verticale
a A	formato	Sottocolonna alfabetica
b B	formato	Elemento neretto
c C	formato	Colonna centrata
e E	formato	Colonne di uguale ampiezza
f F	formato	Cambio di font
i I	formato	Elemento corsivo
l L	formato	Colonna allineata a sinistra
n N	formato	Colonna numerica
nnn	formato	Separazione colonne
p P	formato	Cambio di corpo in punti
r R	formato	Colonna allineata a destra
s S	formato	Elemento spaziato
t T	formato	Distanza verticale in alto
v V	formato	Cambio di distanza verticale
w W	formato	Valore di ampiezza minima
T{T}	dati	Blocco testo
.xx	dati	Comando **troff**
=	dati	Doppia linea orizzontale
_	dati	Linea orizzontale
_	dati	Linea orizzontale breve

sinistra o centrati nella colonna, ecc. In questa sezione viene usato un codice per specificare ciascun elemento della tabella, riga per riga; ovvero, se la tabella contiene tre colonne, la sezione formato deve consistere di linee che contengono tre codici. Inoltre, le righe di formato possono includere separatori per richiedere a **tbl** di separare le colonne con righe verticali, o di usare solo spazi. Ogni linea della sezione dati della tabella deve avere una linea corrispondente nella sezione formato; tuttavia, se vi sono più linee dati che linee formato, l'ultima linea nella sezione formato viene associata a tutte le linee dati rimanenti. La sezione formato termina con un punto alla fine dell'ultima linea.

```
.Ts
center box;
c s s
c s s
c s s
cb ¦ cb ¦ cb
l ¦ l ¦ l.

Comandi e parole tbl

=
Nome        Tipo        Descrizione
=
allbox      opzioni     Incornicia tutti gli elementi
box         opzioni     Incornicia la tabella
center      opzioni     Centra la tabella nella pagina
doublebox   opzioni     Doppia incorniciatura della tabella
expand      opzioni     Tabella di ampiezza pari alla linea
tab(x)      opzioni     Definisce separatore dati x
=
```

Figura 17.6 Esempio di codice sorgente **tbl**.

La sezione formato consente di definire una vasta gamma di formati: potete specificare che un elemento dei dati deve essere allineato a sinistra nella sua colonna con il codice **L** o **l** nella relativa posizione della sezione formato (maiuscole e minuscole sono equivalenti in sezione formato); potete usare **c** per centrare l'elemento nella sua colonna, ecc. Potete combinare diversi codici, se necessario; per esempio, usate **cb** se volete un elemento dati centrato e in neretto; ugualmente, potete specificare anche cambi nel corpo dei caratteri, in punti, usando il codice **p** seguito dalla dimensione, per esempio, potete specificare **cp14** per un elemento centrato con un corpo di 14 punti.

Usate **s** (spanned) quando volete che un elemento dei dati si estenda su più di una colonna; nella Figura 17.6 il titolo "Comandi e parole tbl" è controllato dalle linee:

```
c s s
c s s
c s s
```

Nella sezione dati il titolo è incluso tra due linee vuote, questo spiega la necessità della presenza di *tre* linee di formato. Il carattere **c** specifica che il testo deve essere centrato nella colonna e i due caratteri **s** stabiliscono che lo stesso testo può estendersi nelle due colonne successive; il risultato è un titolo centrato nell'intera linea, incluso tra due linee di spazi.

Potete separare gli elementi in sezione formato col carattere | (barra verticale) oppure || (doppia barra verticale), per richiedere a **tbl** di separare le colonne con una linea singola o doppia; se questi caratteri sono omessi, le colonne verranno separate con il carattere spazio.

Ciascun elemento dei dati della tabella appare nella sezione dati; ogni linea della sezione dati corrisponde a una linea della tabella in uscita, e gli elementi di dati in ogni linea sono separati tra loro da caratteri *tab*. Un carattere *tab* nella linea dati provoca da parte di **tbl** la collocazione dell'elemento successivo nella colonna successiva; dovete pertanto curare che ciascuna linea dati contenga esattamente lo stesso numero di colonne dichiarate nella sezione formato. Oltre alle linee dati la sezione dati può contenere linee che contengono un solo carattere = (uguale) oppure _ (sottolineatura); questi caratteri provocano la generazione di una linea orizzontale semplice o doppia da parte di **tbl**. Tali linee vengono usate spesso per separare sezioni in una tabella. Potete includere nella sezione dati dei normali comandi **troff**, per controllare la spaziatura, o altro, se necessario. Anche un blocco di testo può essere trattato come elemento di una tabella, racchiuso all'inizio da **T**{ (T graffa aperta) e alla fine da **T**} (T graffa chiusa); questo consente di trattare come un elemento di tabella del materiale che non potrebbe essere convenientemente contenuto nel formato normale fra tab. Questi blocchi di testo vengono trattati dai normali comandi **troff** prima di essere trattati da **tbl**, potete perciò includervi anche complessi cambi di font o di spaziatura.

La separazione degli elementi nelle righe della sezione dati, normalmente ottenuta con il carattere tab, può essere ottenuta con un diverso delimitatore, utilizzando il comando **tab**(*x*) nelle opzioni, dove *x* è il separatore prescelto.

Con **tbl** è possibile realizzare molti tipi di tabelle complesse, incluse sotto-tabelle in una posizione di dati, elementi di dati multilinea e altre variazioni. Per esempio, potete interrompere il trattamento degli elementi di dati, e cambiare il formato della tabella col comando **.T&** (table continue), quindi includere una nuova sezione formato e una nuova sezione dati; questo modifica il formato all'interno della tabella, mantenendo tuttavia le stesse opzioni.

IMPAGINAZIONE DI EQUAZIONI MATEMATICHE E DI GRAFICI

Il comando **eqn** (equations) è uno strumento che consente di produrre testi contenenti equazioni e formule matematiche; si tratta di un preprocessore per **troff** specializzato che dispone di un insieme proprio di comandi. Tuttavia, l'utilizzo di **eqn** è ristretto a utenti particolarmente esperti, per documenti comprendenti un gran numero di formule ed equazioni.

Il programma **troff** non è stato progettato per il supporto all'inserimento di grafici nei documenti; tuttavia, alcune fotocompositrici e stampanti laser

permettono l'utilizzazione di un insieme speciale di caratteri grafici che **troff** può utilizzare per produrre una semplice grafica, tipo diagrammi a blocchi e diagrammi di flusso moderatamente complessi. In particolare, i comandi **pic** (pictures) e **grap** (graphs) sono altri preprocessori per **troff** specializzati che consentono di produrre queste figure; come **eqn**, dispongono di un proprio insieme di comandi.

Se usate una stampante PostScript potete includere del codice PostScript in vostri documenti sorgenti **troff** usando il comando **.so** (source). Il comando **.so** accetta un argomento che permette di specificare un nome di percorso e legge quel file nella locazione corrente del documento; potete includere con questo mezzo nei vostri documenti qualsiasi materiale sorgente **troff** contenuto in un file separato.

Ogni linea che inizia con la stringa \! (barra inversa punto esclamativo spazio) viene passata da **troff** in uscita invariata e non inclusa nel testo in uscita; potete in questa maniera includere nei vostri documenti programmi PostScript (o altro materiale) che verranno interpretati direttamente dal dispositivo di stampa. Di preferenza, questi programmi verranno gestiti in file separati e inclusi nei vostri documenti col comando **.so**. Per esempio, questa sequenza disegna una singola linea all'interno di uno schema:

```
.DF 2
.ne 50i     \" chiede spazio sufficiente nella pagina corrente
\!  /inchXX { 72 mul } def
\!  % traccia una semplice linea retta
\!  /dolineXX {     % hstart vstart hend vend dolineXX -
\!    newpath moveto lineto stroke
\!  } def
\!  gsave
\!  3 inchXX 3 inchXX 6 inchXX 7 inchXX dolineXX
\!  grestore
.DE
```

Curate che ogni funzione PostScript da voi definita abbia un nome univoco, non già usato internamente dal sistema **ditroff**; in caso di incertezza usate nomi insoliti o complessi

LA STRUTTURA DEL DIRECTORY DI troff

Il pacchetto **troff** e i comandi associati risiedono in diversi directory del file system. Le librerie di macro risiedono nel directory **/usr/share/lib/macros**, tuttavia il comando **troff** frequentemente non accede alla libreria direttamente, bensì tramite brevi file di collegamento contenuti in **/usr/share/lib/tmac**. Lo scopo di questo collegamento indiretto non è chiaro ma, se uno dei due directory viene alterato, i programmi di utility di **troff** probabilmente non funzionano correttamente. Inoltre, le descrizioni delle stam-

panti e dei terminali utilizzati da **troff** risiedono in uno dei directory **/usr/lib/term** o **/usr/lib/nterm**, a seconda di quale versione di **troff** viene impiegata. Tenete presente che i file di descrizione dei terminali di **troff** non sono in correlazione con il database **terminfo** utilizzato dalle applicazioni video come **vi**.

OPERATORI CONDIZIONALI E MACRO IN troff

Il sistema **troff** è veramente un linguaggio di programmazione a tutti gli effetti (anche se difficile), include esecuzione condizionale, salti e la possibilità di creare sottoprogrammi; le macro come **mm** e **man** utilizzano queste capacità per incrementare grandemente i comandi basilari di **troff**.

I comandi **.if** (if) e **.ie** (if-else) sono usati per esaminare il valore di alcune espressioni e decidere azioni diverse in base al risultato; per esempio, la linea:

```
'if o 'tl '\n(dy/\n(mo/\n(yr'- % -'\n(H1'
```

è interpretata come segue: se la pagina corrente è dispari, predispone il titolo, nelle tre parti, alla data corrente, la pagina corrente e il capitolo corrente. Ugualmente, la linea:

```
'ie o 'tl '\n(dy/\n(mo/\n(yr'- % -'\n(H1'
'el 'tl ''- % -''
```

se (**ie**) la pagina corrente è dispari, predispone il titolo come prima, altrimenti (**el**) lo predispone in maniera diversa. Potete usare condizioni per esaminare il valore di qualsiasi stringa o registro e scegliere le azioni appropriate; nella parte condizionale dei comandi **if** e **ie** sono ammesse operazioni aritmetiche e logiche.

La definizione di una macro viene effettuata col comando **.de X**, in cui X è un nome di uno o due caratteri; nel corpo della macro può essere incluso qualunque comando **troff**, o richiami di altre macro; la definizione termina con una linea contenente solo .. (punto punto). Dopo avere definito una macro potete richiamarla come un qualsiasi altro comando **troff**. La Figura 17.7 mostra la definizione di una semplice macro per il trattamento del fine pagina. L'esempio fa uso di varie caratteristiche di **troff** che non sono state descritte; principalmente crea un ambiente separato, stabilisce il corpo dei caratteri in punti e il titolo, quindi spazia alcune linee per il margine finale. Il programma **troff** usa ambienti separati per gestire contemporaneamente due o più processi di trattamento; per esempio, gli schemi sono trattati in un ambiente diverso dal normale testo. Nella Figura 17.7 la macro **BP** richiede una nuova pagina, il che provoca la stampa del titolo corrente con le dimensioni e il font correnti; quindi la macro spazia alcune linee all'inizio

```
de BP
.ev 2
.ft B
.ps 12
'if o 'tl '\n(dy/\n(mo/\n(yr' - % - '\n(H1'
'if e 'tl '' - % - ''
.br
.ft P
.ps
'sp 21
.bp
'sp 1.5i
.ev
..
.wh ¦9.25i BP
```

Figura 17.7 Semplice definizione di macro fine-pagina.

della pagina e ripristina l'ambiente precedente, consentendo a **troff** di riprendere la creazione delle linee di testo in uscita. Il comando **.wh** (when) istruisce **troff** di richiamare la macro **BP** e quindi di eseguirne le istruzioni definite al suo interno, quando vengono raggiunti 9.25 pollici dall'inizio della pagina.

Le macro possono accettare degli argomenti, se necessario; all'interno della macro potete fare riferimento agli argomenti con i simboli **$1** per il primo argomento, **$2** per il secondo, ecc.

Tenete presente che all'interno delle condizioni, nei titoli in tre parti e nelle macro, possono essere necessari caratteri \ (barra inversa) quando usate i valori dei registri; il numero esatto di questi caratteri nelle diverse situazioni può essere determinato sperimentalmente.

LOGO IN mm

Il pacchetto **mm** fornisce la possibilità di stampare un logo aziendale nell'intestazione di lettere e memo; tuttavia la versione originale distribuita da AT&T, e anche la versione di molti altri produttori, include solo il logo AT&T. Se anche la vostra versione produce solo il logo AT&T inserite il comando **.AF** *stringa* (alternate form) prima di **.TL** nei vostri documenti sorgenti; *stringa* verrà stampata in caratteri di maggiori dimensioni; in assenza di *stringa* verrà lasciato uno spazio per consentire la stampa su carta intestata.

IL PACCHETTO WRITER'S WORKBENCH

Uno dei più interessanti software addizionali per **troff** è il pacchetto *Writer's Workbench* che effettua un'analisi critica di documenti scritti con

troff; il pacchetto, basato su **troff**, **spell** e filtri dedicati, consente l'analisi della struttura delle frasi, dell'uso delle parole, dello stile e di molte altre caratteristiche dei documenti. I dati prodotti in uscita sono di facile interpretazione e possono aiutare a migliorare lo stile di scrittura sia di studenti che di scrittori di professione; queste informazioni possono essere dettagliate (tutorial) oppure concise (terse) a seconda del grado di preparazione dell'utilizzatore. La Tabella 17.9 elenca alcuni dati riassuntivi prodotti dal programma di analisi dello stile **style**, uno degli strumenti del pacchetto *Writer's Workbench.*

I dati sono ovviamente riferiti a documenti originali in lingua inglese.

IL FILE STORICO DI spell

Quando il programma **spell** è in esecuzione, memorizza nel file storico **/var/adm/spellhist** tutte le parole che identifica come sbagliate. Questo file può servire a un gestore di sistema per includere nell'elenco delle parole accettate nuove parole e acronimi che risultano di impiego comune; in un ambiente didattico può consentire un'analisi statistica delle parole che più spesso vengono sbagliate. Tuttavia, la dimensione del file può crescere notevolmente perché non esistono strumenti di gestione automatica in grado di limitarla. Il gestore di sistema può impedire l'incremento eccessivo del file

Tabella 17.9 Analisi di leggibilità di alcuni documenti di **troff**.

Caratteristica	Indicazioni tecniche dei Bell Laboratories	Documentazione di riferimento AT&T	Questo libro
Grado di leggibilità Kincaid	da 10.1 a 15.0	da 7.8 a 12.4	10.5
Lunghezza media di frase	da 16.7 a 25.3 parole	da 12.3 a 20.2	20.9
Lunghezza media di parola	da 5.8 a 7.0 lettere	da 5.5 a 6.8	5.85
Percentuale di frasi brevi	da 29.2% a 38.0%	da 23.1% a 31.4%	33%
Percentuale di frasi lunghe	da 11.7% a 18.9%	da 7.3% a 12.8%	13%
Percentuale di frasi semplici meno percentuale di frasi complesse	da −24.2% a 30.1%	da −28.4% a 56.0%	−3.0%
Percentuale di frasi composte più percentuale di frasi complesse e composte	da 5.7% a 35.2%	da 4.7% a 25.7%	26%
I verbi passivi devono avere una presenza inferiore a	28.6%	28.7%	20%
I simbolismi devono avere una presenza inferiore a	4.2%	3.4%	2%
Le particelle espletive devono avere una presenza inferiore a	5.7%	7.2%	2%

spellhist, riducendo manualmente a intervalli regolari il file storico delle parole, includendo questo tipo di operazione in una procedura di pulizia di file attivata con una regolarità prestabilita, oppure modificando il comando **/usr/bin/spell** per inibire la funzione di registrazione storica.

DEFINIZIONE DI UN NUOVO DATABASE DI spell

Se utilizzate **spell** per documenti che adottano una specifica terminologia, potete avere la necessità di aggiornare gli elenchi delle parole di **spell**. Inoltre, potete derivare molti comandi di controllo da altri elaboratori di testo, diversi da **troff**, se aggiungete i codici corrispondenti al database di **spell**. Analizzate il file **spellhist** per stabilire quali parole mancano per garantire un impiego dei programmi di **spell** adeguato alle vostre esigenze.

L'elenco delle parole utilizzate da **spell** è memorizzato in un database di radicali di parole, estremamente efficiente, che risiede nel directory **/usr/share/lib/spell**:

```
$ ls -F /usr/share/lib/spell
compress*       hlista       hlistb       hstop
$
```

Gli elenchi delle parole in lingua americana e britannica sono contenuti rispettivamente nei file **hlista** e **hlistb**; non potete esaminare i file a causa del loro formato binario. Il file **hstop** contiene un elenco di parole scorrette nell'ortografia (come thier), ma che non verrebbero segnalate perché supererebbero comunque l'analisi ortografica. Il programma **/usr/lib/spell/spellprog**, richiamato da **/usr/lib/spell**, è l'effettivo esecutore dell'analisi ortografica; il directory **/usr/bin/spell** contiene anche gli strumenti per aggiornare gli elenchi delle parole. Il programma **hashmake** legge una lista di parole dallo standard input e scrive sullo standard output le corrispondenti codifiche. Il programma **spellin** legge una lista di codifiche dallo standard input e la riporta sullo standard output in forma compressa, mentre il programma **hashcheck** effettua l'operazione opposta; legge una lista in forma compressa dallo standard input e scrive sullo standard output una lista di corrispondenti codifiche. Il programma **spellin** accetta un argomento numerico, che specifica il numero di codifiche contenute nei dati in ingresso.

Per includere nel file **hlista** una serie di parole, occorre seguire una precisa procedura con diversi passaggi. Primo, dovete preparare la lista delle parole da inserire, una per linea; secondo, sottoporre in ingresso a **hashmake** la lista per ottenere le codifiche; terzo, decomprimere il file **hlista** tramite **hashcheck**; quarto, dovete ordinare insieme questo output decompresso e la lista delle codifiche delle parole da inserire; infine, comprimere con **spellin** l'ultimo output ordinato. I seguenti comandi eseguono quanto descritto:

```
$ PATH = $PATH:/usr/lib/spell
$ hashmake < new.list ¦ sort > new.hash # codifica new.list
$ hashcheck < hlista ¦ sort -mu - new.hash > hash.out # lista finale
$ NUM = 'wc -l<hash.out'                # conteggio per spellin
$ spellin $NUM < hash.out > hlista      # produce il nuovo hlista
$
```

Dovete provvedere a conservare una copia del file originale, fino a che non avrete adeguatamente provato la nuova versione.

Capitolo 18

Supporti di memorizzazione

La gestione dei supporti magnetici costituisce un settore di fondamentale importanza in ogni sistema operativo. Avrete sempre uno stretto rapporto con i supporti di memorizzazione come dischi e nastri magnetici, dal primo momento in cui caricate il sistema operativo su disco rigido, o nella gestione dello spazio su disco, o quando salvate i dati su dischetto. Il sistema UNIX definisce due differenti modalità di gestione di supporti di memorizzazione e consente di utilizzare dischi con formati e dimensioni diverse.

La gestione dei dischi può essere complessa e non esiste di fatto un modo

rapido per conoscerne tutti gli aspetti. I sistemi UNIX differiscono notevolmente nel modo con cui referenziano le unità a disco ed è sempre richiesto un certo impegno per comprendere lo schema utilizzato in una nuova versione. In SVR4 sono stati adottati nuovi tipi di file system, che presentano comportamenti differenti fra loro: possono essere tutti utilizzati sulla stessa macchina; possono anche coesistere nello stesso disco, anche se alcuni sono file system *virtuali* che non sono realmente rappresentati da spazio fisico del disco. Quando richiedete un'azione su un disco dovete di solito specificare il tipo di file system nella linea di comando; i nastri magnetici invece rispondono a regole del tutto diverse.

In questo capitolo vedremo l'accesso manuale agli strumenti di gestione dei dischi ed esamineremo alcuni dei problemi associati con l'utilizzo dei dischi e dei nastri nelle normali attività. Menu semplificati per la gestione dei dischi sono di solito presenti in uno strumento di gestione del sistema, come descritto nel Capitolo 12.

18.1 Blocchi e i-node su disco

Il sistema UNIX gestisce lo spazio su disco in termini di unità dette *blocchi*. In SVR4 i blocchi hanno una dimensione di 512 byte ciascuno. Tutte le operazioni con i file su disco avvengono per blocchi; vale a dire che la dimensione minima di un file o di un directory è di un blocco, anche se il contenuto effettivo è di un solo byte; in altre parole, un file di 510 byte occupa un blocco di disco come un file di 1 byte, mentre un file di 514 byte occupa due blocchi. La maggior parte dei comandi relativi ai dischi riportano le indicazioni di spazio occupato su disco in termini di numero di blocchi; solo pochi comandi, in particolare **ls**, utilizzano direttamente i byte come unità di misura dello spazio su disco. Anche i directory usano blocchi di disco; ancora, lo spazio minimo di un directory è un intero blocco, anche se il directory contiene un file solo, o nessuno.

Il file system contiene un elenco dei nomi di tutti i file presenti sul disco, associati ai puntatori al primo blocco su disco relativo a ogni nome. I nomi dei file e i blocchi associati sono memorizzati in entità chiamate *i-node* (nodo di identificazione). In un directory, per ciascun file esiste un singolo i-node che contiene il nome e il puntatore del file stesso; ogni volta che un file viene aggiornato, il sistema aggiorna il corrispondente i-node. L'elenco di i-node è stabilito quando viene creato un file system; raramente avrete motivo di trattare con gli i-node ma, se non esistono più i-node disponibili, specialmente su dischetto, non potrete creare nessun altro file.

18.2 Il file system

L'elenco dei blocchi su disco, gli i-node e altre informazioni associate con l'uso dei dischi vengono gestiti separatamente per ogni file system presente sulla macchina. Ogni disco fisico collegato alla macchina deve avere il proprio file system; è possibile definire più file system su un unico disco, ma l'operazione inversa non è lecita: un file system non può occupare più dischi. Quando il sistema UNIX viene installato esistono sempre uno o due file system, disponibili in un dispositivo fisico noto, che costituiscono il file system originario e vengono generalmenté indicati con / (root) e **/usr** (user). Anche i dischetti e i nastri magnetici possono avere i propri file system, ma questa non è una condizione indispensabile.

I file system vengono trattati separatamente in ambiente UNIX, per cui non potete eseguire **ln** o **mv** tra file system distinti, a meno di non usare **symlink** (collegamenti simbolici). Per spostare un file da un file system a un altro dovete crearne una copia con il comando **cp**. Se un file system si riempie totalmente, o viene danneggiato gravemente, gli altri file system presenti sulla macchina non risentono minimamente di questi inconvenienti.

18.3 Gestione del disco rigido

Poiché la principale attività del sistema UNIX coinvolge il disco rigido installato sulla macchina, dovrete diventare esperti della sua gestione. Lo spazio sul disco rigido diminuisce con il passare del tempo ed è una tendenza abituale non cancellare il contenuto non più indispensabile. Se pianificate i vostri fabbisogni in termini di spazio su disco, potete ottimizzare il funzionamento della macchina e garantire la disponibilità di spazio quando vi occorre.

18.4 Spazio disponibile su disco: il comando df

Tramite il comando **df** (disk free), potete determinare per ogni file system sulla macchina, lo spazio occupato sul disco e lo spazio disponibile:

```
$ df
/          (/dev/root      ):  140331 blocks    63540 files
/proc      (/proc          ):       0 blocks       75 files
/dev/fd    (/dev/fd        ):       0 blocks        0 files
/stand     (/dev/dsk/0s10  ):    7307 blocks       97 files
$
```

La colonna all'estrema sinistra identifica il file system, segue il nome del dispositivo nel directory **/dev** che contiene il file system, poi il numero di blocchi liberi e, infine, il numero di i-node liberi. Possono esistere alcuni file system *virtuali* per scopi speciali, come **/proc** e **/dev/fd**; questi sono usati da software di gestione di file, ma non sono destinati a contenere file effettivi, perciò i valori di blocchi e file per questi file system non sono significativi. Il numero di i-node, o file, indica il numero totale di nuovi file che potete creare prima che il file system si riempia. Se create nel file system root ancora 63.540 file senza cancellarne nessun'altro, il file system si satura anche se rimangono ancora blocchi liberi.

Il comando **df** riporta anche la quantità totale di spazio su disco nei file system, se utilizzate l'opzione **-t** (total):

```
$ df -t
/               (/dev/root       ):  140323 blocks  63540 files
                          total:    294986 blocks  73632 files
/proc           (/proc           ):       0 blocks     75 files
                          total:         0 blocks    102 files
/dev/fd         (/dev/fd         ):       0 blocks      0 files
                          total:         0 blocks     66 files
/stand          (/dev/dsk/0s10   ):    7307 blocks     97 files
                          total:    10710 blocks    104 files
$
```

Potete facilmente calcolare la percentuale di spazio su disco realmente utilizzato e tenere sotto controllo la velocità di occupazione dello spazio su disco. Alcune versioni di SVR4 dispongono del comando **dfspace** (disk free space) che permette di calcolare direttamente queste percentuali; il comando viene spesso eseguito automaticamente al login. Per esempio:

```
$ dfspace
/               : Disk space:  68.51 MB of 144.03 MB available (47.57%).
/proc           : Disk space:   0.00 MB of   0.00 MB available ( 0.00%).
/dev/fd         : Disk space:   0.00 MB of   0.00 MB available ( 0.00%).
/stand          : Disk space:   3.56 MB of   5.22 MB available (68.23%).
Total Disk Space : 72.08 MB of 149.26 MB available (48.29%).
$
```

Il comando **dfspace** accetta una lista di nomi di file system, per consentire di limitare le informazioni ottenute a una parte dei file system presenti nel sistema.

18.5 Spazio utilizzato su disco: il comando du

Il comando **df** restituisce lo spazio su disco disponibile nell'intero file system, ma potete anche determinare lo spazio occupato da ogni file o directo-

ry. Il comando **du** (disk usage) restituisce queste informazioni per i directory specificati come argomento:

```
$ du /var/spool/cron
1       /var/spool/cron/atjobs
4       /var/spool/cron/crontabs
6       /var/spool/cron
$
```

Il comando **du** riporta il numero di blocchi fisici di 512 byte occupati da ogni file o directory; per determinare la quantità effettiva di byte utilizzati su disco, moltiplicate i dati forniti da **du** per 512, come potete fare anche per **df**.

Il comando **du** non accetta il nome di un file come argomento, ma solo il nome di un directory e riporta in uscita le informazioni dell'alberatura di subdirectory contenuti nel directory indicato. L'opzione **-a** (all) permette di visualizzare su linee distinte anche i file oltre ai directory, mentre l'opzione **-s** (summary) visualizza solo il numero totale di blocchi fisici che il directory indicato occupa:

```
$ du -s /etc
9159    /etc
$
```

Questa opzione si dimostra utile per ottenere delle brevi notizie sull'uso del disco.

Il comando **du** ignora i file e i directory che non può aprire senza riportare nessun messaggio di avvertimento, a meno che utilizziate l'opzione **-r** (report).

I comandi che abbiamo descritto – **df**, **dfspace** e **du** – sono inestimabili strumenti per gestire lo spazio su disco; usateli con una certa regolarità, manualmente o tramite un processo attivato periodicamente, per controllare l'evolversi dell'occupazione di disco sulla vostra macchina.

18.6 Dimensione dei file e variabile ulimit

Nel sistema UNIX, la maggior parte dei file ha una dimensione relativamente ridotta, ed è raro che esistano file di un megabyte o di dimensione maggiore; tuttavia, il sistema permette ai programmi di usare tutto lo spazio su disco che occorre. In questo modo, processi fuori controllo o applicazioni errate possono utilizzare indebitamente tutto lo spazio disponibile su disco. Per evitare questa situazione, il sistema UNIX prevede il comando **ulimit** (user limit) e definisce la variabile **ulimit**, il cui valore stabilisce la dimen-

sione massima in blocchi di un file creato da un utente. Nella maggior parte dei sistemi UNIX il valore iniziale di **ulimit** è abbastanza ridotto; per esempio:

```
$ ulimit
4096
$
```

Quando cercate di creare un file che supera la dimensione indicata da **ulimit**, la scrittura del file si arresta al valore di **ulimit** e il comando che crea il file termina con una condizione d'errore. Nell'esempio precedente, la dimensione massima dei file è di 2 MB.

Il valore iniziale di **ulimit** viene assegnato per l'intero sistema, ma può essere cambiato in una singola sessione di login. Naturalmente, voi non potete aumentare il vostro **ulimit**, potete solo ridurlo; solo il superuser può aumentare **ulimit** per una sessione. Per assegnare il nuovo valore a **ulimit**, viene indicato come argomento:

```
$ ulimit
4096
$ ulimit 1000
$ ulimit
1000
$
```

Questa modifica risulta effettiva solo nella sessione corrente; se la modifica ha luogo in un subshell, non influenza lo shell padre. Spesso il gestore di sistema inizializza la variabile **ulimit** a un valore ragionevole considerando le diverse necessità degli utenti. Si tratta generalmente di 1 o 2 megabyte per le macchine di piccole dimensioni, per cui se utilizzare grandi database o file di grandi dimensioni, accertatevi che **ulimit** non sia troppo piccola per le vostre esigenze.

18.7 Controllo dell'occupazione del disco rigido

Lo spazio su disco rigido è limitato e sembra esaurirsi sempre con una rapidità superiore alle previsioni. Se non esiste più spazio disponibile per i file, la macchina cessa il funzionamento, sia interrompendo brutalmente l'esecuzione, sia diminuendo le prestazioni così drasticamente che il sistema diventa non operativo. Se accade una situazione di questo tipo, dovete reinizializzare il sistema da dischetto e rimuovere manualmente dal disco rigido i file che non servono, operazione che può risultare difficile e complessa. In qualche caso, può addirittura essere necessario ricaricare il software di sistema, causando la perdita di tutti i file sul disco rigido.

In conclusione, per la gestione del sistema è un obiettivo indispensabile mantenere sempre almeno il **10%** di spazio disponibile sul disco rigido principale della macchina. Per stimare al meglio questa percentuale dovrete utilizzare con regolarità il comando **df** e organizzare procedure per cancellare i file e i directory che non servono. Potete utilizzare alcuni accorgimenti per ridurre l'impiego di spazio su disco. Innanzitutto, salvate su dischetto (o nastro) i file e i directory che utilizzate raramente e ricaricate i dischetti solo quando occorre. Inoltre, potete usare il directory **/tmp** per memorizzare file temporanei e dati intermedi prodotti dalle vostre procedure di shell; il contenuto di **/tmp** viene cancellato ogni volta che la macchina viene resettata e quindi non rimane nel file system per troppo tempo. Infine, potete lanciare il comando **find** per cercare periodicamente i file che hanno una dimensione superiore a un valore predefinito e cancellare i file che si sono incrementati oltre il previsto. Per esempio, il comando

```
$ find / -size +200 -exec ls -l { } \;
```

visualizza il nome e la dimensione di tutti i file che hanno una dimensione maggiore di 200 blocchi. Fate comunque attenzione a non cancellare file che sono legittimamente più grandi della vostra dimensione limite, come per esempio **/stand/unix** e molti altri nel sistema. Tenete sotto controllo file che incrementano la loro dimensione in un arco di tempo di giorni o settimane, considerando comunque che i file contenuti nei directory **/sbin**, **/lib**, **/usr/sbin**, **/usr/bin** e **/usr/lib** sono di solito legittimi, mentre altri possono essere sospetti.

Inoltre, ricercate la presenza nel sistema di file col nome **core**, col comando:

```
$ find / -name core -print
```

Questi file **core** vengono prodotti quando un comando o una applicazione terminano con una condizione d'errore; occupano molto spazio e generalmente sono inutili. In effetti la loro presenza segnala il verificarsi nella macchina o nelle applicazioni di qualche condizione patologica di malfunzionamento che deve essere indagata; qualche volta la posizione che i file **core** occupano nei directory può essere una traccia per risalire alla causa del guasto. Infine, verificate le aree utente, come home directory e subdirectory, per individuare situazioni di esagerata occupazione di spazio dei dischi da parte degli utenti, che dovrebbero invece mostrarsi rispettosi delle risorse della macchina e delle esigenze degli altri utenti collegati. In certi casi potrete usare lo strumento **quota** per limitare l'utilizzazione di spazio su disco da parte dei singoli utenti; questa tecnica viene trattata alla fine del capitolo.

COMPRESSIONE DI FILE

Potete ridurre le dimensioni dei file usati raramente sottoponendoli a *compressione*, una tecnica che consente di risparmiare dal 30 al 50 per cento dello spazio occupato dal file originale. I file compressi devono essere *decompressi* prima di essere utilizzati; quindi, prima di comprimere un file dovete essere certi che non debba essere utilizzato dal sistema o da un altro utente.

SVR4 fornisce due programmi per comprimere file; differiscono nell'algoritmo usato per la compressione, ma per il resto operano in maniera simile. Il programma **compress** richiede un elenco di nomi di file come argomento e produce una versione compressa di ciascun file; per default, non fornisce alcun messaggio:

```
$ compress prova.file
$ ls -l
-rwx------ 1 giorgio     30202 Mar  8 16:23 prova.file.Z
$
```

Per ottenere un dettaglio sull'entità della compressione dovete aggiungere l'opzione **.v** (verbose).

Il programma **compress** cambia il nome del file aggiungendo il suffisso **.Z** (Z maiuscola); perciò, se i vostri file hanno il limite di 14 caratteri per il nome, potete sottoporre a compressione solo file con nomi lunghi al massimo 12 caratteri.

Per decomprimere un file prodotto da **compress** dovete usare il comando **uncompress**:

```
$ uncompress -v *Z
prova.file.Z: -- replaced with prova.file
$ls -l
-rwx------ 1 giorgio     72465  Mar  8 16:23 prova.file
$
```

L'indicazione data-ora del file non viene alterata dalle operazioni **compress** e **uncompress**.

L'altro strumento, **pack**, accetta anch'esso un elenco di nomi di file e produce una versione impaccata di ciascun file:

```
$ pack prova.file
pack: prova.file: 40.9% Compression
$
```

Il programma **pack** cambia il nome dei file aggiungendo il suffisso **.z** (z minuscola); anche in questo caso, se i vostri file hanno il limite di 14 caratteri per il nome, potete sottoporre a compressione solo file con nomi lunghi al

massimo 12 caratteri. Per default, il programma informa della riduzione di spazio realizzata per ogni file compresso. Il programma **unpack** ricostituisce il file nella sua forma originaria.

Generalmente viene preferito il programma **compress**, in quanto produce un'uscita di dimensioni inferiori rispetto a **pack**.

È anche possibile ottenere una versione decompressa di un file su standard output, invece di rimpiazzare il file compresso con la corrispondente versione normale; il programma **zcat**, simile a **cat**, fornisce questa possibilità per file prodotti con **compress**. Come esempio:

```
$ zcat prova.file.Z ¦ wc
    2195     12844     72465
$
```

Questo lascia invariato il file compresso. La stessa funzione per programmi compressi con **pack** viene fornita dal comando **pcat**.

18.8 Tipi di file system

SVR4 introduce un nuovo meccanismo di trattamento dei file system, che consente agli sviluppatori di software di creare nuovi tipi di file system con caratteristiche speciali ed estranee al sistema UNIX. Esempi comuni di questo meccanismo sono il directory **/proc** e la possibilità di montare file system MS-DOS sotto il sistema UNIX. Inoltre, anche file system reali che compaiono nel normale albero di directory possono trarre vantaggio di questa nuova caratteristica. In SVR4 sono previsti due tipi principali di file system, dai comportamenti differenti; dovete comprenderne le caratteristiche per trattare adeguatamente i dischetti.

Il file system "vecchio stile" System V, usato da molti anni nei sistemi UNIX AT&T, è uno di questi importanti tipi; usa una tabella i-node di dimensione fissa per l'allocazione dei file e relativi blocchi su disco, e limita i nomi di file e directory a 14 caratteri. In SVR4 questo tipo di file system è denominato **s5**. In molte macchine i file system fondamentali a livello utente sui dischi rigidi sono del tipo **s5**, e dipendono da come la macchina fu configurata all'installazione del software UNIX.

L'altro principale tipo di file system è un adattamento dalla versione BSD del sistema UNIX; usa una tabella flessibile di i-node per i directory definita alla configurazione, e accetta nomi di file e directory di lunghezza fino a 256 caratteri; inoltre, fornisce prestazioni superiori in velocità e risulta maggiormente affidabile del tipo **s5** in molte situazioni critiche. In SVR4 questo tipo di file system è denominato **ufs** (UNIX file system).

A causa della complessità dei modi di configurazione del sistema UNIX, ogni particolare directory visibile all'utente può rientrare in ambedue i tipi di file system. Poiché ogni directory può essere un symlink a qualche altra

locazione nell'intera alberatura di directory, non potete presumere che un subdirectory appartenga allo stesso tipo del suo directory padre; tuttavia, normalmente il gestore di sistema procura di costruire dischi che contengono un unico tipo di file system, **s5** o **ufs**. In linea di massima, il tipo **ufs** è preferibile a **s5**, perché consente nomi di maggiore lunghezza, non ha limitazioni nelle dimensioni della tabella i-node, e fornisce prestazioni più veloci. I sistemi SVR3 e quelli precedenti usano esclusivamente il tipo **s5**, pertanto i dischetti d'interscambio dati con sistemi più vecchi debbono essere del tipo **s5**.

Il comando **df** **-n** (name) riporta il tipo di file system per ogni directory; per esempio:

```
$ df -n $HOME
/home/giorgio        : ufs
$
```

Le differenze fra i file system diventano importanti nella creazione di file system su dischetti

18.9 Gestione dei dischetti

Prima di utilizzare un dischetto, dovete formattarlo. L'operazione di formattazione predispone il disco per l'uso, per cui dovete formattare tutti i nuovi dischi acquistati. Potete anche formattare un disco già usato per cancellarne il contenuto e predisporlo a un nuovo utilizzo.

Le modalità di formattazione dipendono dal tipo di dischetto. Un disco da $5^{1/4}$ pollici a *doppia faccia* e *doppia densità*, usato nelle macchine PC-compatibili meno recenti, può memorizzare 360 kB di dati; i dischetti da $5^{1/4}$ pollici ad *alta densità*, utilizzati nelle macchine AT-compatibili, possono memorizzare 1.2 MB di dati. I dischetti da $3^{1/2}$ pollici con custodia in plastica contengono 720 kB a doppia faccia e doppia densità; oppure 1.4 MB ad alta densità.

Quelli descritti sono i tipi più comuni di dischi; dovete sempre specificare il tipo di disco che vi occorre all'atto dell'acquisto.

Anche se i dischetti sono fisicamente identici, i formati dei dischetti usati dal sistema UNIX non coincidono con quelli usati in ambiente MS-DOS o in altri sistemi operativi. Molti sistemi SVR4 consentono di leggere e scrivere dischetti in formato MS-DOS sotto il sistema UNIX, mediante strumenti software speciali. Quanto detto in questo capitolo è valido esclusivamente per dischetti usati totalmente in sistema UNIX; l'utilizzo di dischetti MS-DOS è trattato nel Capitolo 19.

Una volta formattato il dischetto, il sistema UNIX prevede due differenti modi di utilizzazione. Con il primo modo, potete creare un file system nel dischetto e utilizzarlo come un disco rigido, cioè creare directory, passare

da un directory a un altro, copiare file, ecc. Per usare questo modo di accesso, dovete montare il file system, che avete creato sul dischetto, in un punto determinato del file system del disco rigido, denominato *mount point*; dopo questa operazione, potete utilizzare il dischetto come un normale directory di sistema.

Il secondo metodo di accesso ai dischetti previsto dal sistema UNIX è denominato *raw access*. In questo caso i dischetti non dispongono di file system e quindi non possono essere utilizzati come un disco rigido; d'altra parte, questo tipo di accesso permette di riempire totalmente di dati più dischi, perché quando un disco si è riempito, potete passare semplicemente a un altro disco. L'accesso al dischetto nel formato *raw* viene generalmente utilizzato per salvare o archiviare i dati.

Un'altra complicazione nasce se volete la possibilità di inizializzare la macchina da un dischetto invece che dal disco rigido. In ambiente UNIX l'inizializzazione da dischetti viene usata raramente; tuttavia, il primo disco del System Installation Set è un esempio di dischetto inizializzante. I dischi inizializzanti devono contenere un *boot block* (cioè un blocco riservato al bootstrap) e un kernel eseguibile perché l'intero sistema operativo deve andare in esecuzione dal dischetto. Questi requisiti aggiuntivi riducono notevolmente lo spazio su disco disponibile per i dati, per cui, a meno di reali necessità, non è conveniente utilizzare dischetti inizializzanti. La scelta se rendere inizializzante un dischetto deve essere presa in fase di formattazione del disco, ma i normali strumenti di formattazione non creano dischi di inizializzazione. Generalmente occorre copiare il primo disco (talvolta i primi due dischi) del System Installation Set e poi modificarne il contenuto con un editor per creare un nuovo disco di inizializzazione.

18.10 Device file dei dischi

I dischetti (e anche i dischi rigidi) vengono gestiti tramite i *device file* che individuano il tipo di disco e specificano sia l'unità disco da usare per l'operazione sia il tipo di formato del disco. Quando utilizzate un dischetto, indicate il device file che individua quel tipo di disco su quel drive. Le Tabelle da 18.1 a 18.3 elencano i device file per i sistemi SVR4, sia per i dischi rigidi (HD) che per i floppy disk (FD). La Tabella 18.1 fornisce informazioni sui device file per le varie forme di accesso per i dischetti.

Tutte le unità a disco si trovano nel directory **/dev**: in particolare, il subdirectory **dsk** (disk) contiene i dischi caricabili con *mount*, mentre il subdirectory **rdsk** (raw disk) contiene i dischi usati in modo *raw*. In altre parole, ogni file presente in questi directory si riferisce a un particolare tipo di dispositivo.

I nomi dei file identificano uno specifico dispositivo per un tipo di accesso al disco; per esempio, nel nome **f0q15dt**, **f0** si riferisce all'unità a disco

Tabella 18.1 Device file per i dischetti dei sistemi UNIX SVR4.

Device file	Device file a blocchi	Device	Dimensione	Commento
/dev/rdsk/f0t	/dev/dsk/f0t	FD 0	*	Qualunque drive installato; intero disco
/dev/rdsk/f0	/dev/dsk/f0	FD 0	*	Qualunque drive installato; escluso il blocco di boot
/dev/rdsk/f1t	/dev/dsk/f1t	FD 1	*	Qualunque drive installato; intero disco
/dev/rdsk/f1	/dev/dsk/f1	FD 1	*	Qualunque drive installato; escluso il blocco di boot
/dev/rdsk/f0d8dt /dev/rdsk/f05d8t	/dev/dsk/f0d8dt /dev/dsk/f05d8t	FD 0	320 kB	Intero disco
/dev/rdsk/f0d8d /dev/rdsk/f05d8 /dev/rdsk/f05d8u	/dev/dsk/f0d8d /dev/dsk/f05d8 /dev/dsk/f05d8u	FD 0	320 kB	Escluso il blocco di boot
/dev/rdsk/f1d8dt /dev/rdsk/f15d8t	/dev/dsk/f1d8dt /dev/dsk/f15d8t	FD 1	320 kB	Intero disco
/dev/rdsk/f1d8d /dev/rdsk/f15d8	/dev/dsk/f1d8d /dev/dsk/f15d8	FD 1	320 kB	Escluso il blocco di boot
/dev/rdsk/f0d9dt /dev/rdsk/f05d9t	/dev/dsk/f0d9dt /dev/dsk/f05d9t	FD 0	360 kB	Intero disco
/dev/rdsk/f0d9d /dev/rdsk/f05d9	/dev/dsk/f0d9d /dev/dsk/f05d9	FD 0	360 kB	Escluso il blocco di boot
/dev/rdsk/f1d9dt /dev/rdsk/f15d9t	/dev/dsk/f1d9dt /dev/dsk/f15d9t	FD 1	360 kB	Intero disco
/dev/rdsk/f1d9d /dev/rdsk/f15d9	/dev/dsk/f1d9d /dev/dsk/f15d9	FD 1	360 kB	Escluso il blocco di boot
/dev/rdsk/f0q15dt /dev/rdsk/f05ht	/dev/dsk/f0q15dt /dev/dsk/f05ht	FD 0	1.2 MB	Intero disco
/dev/rdsk/f0q15d /dev/rdsk/f05h	/dev/dsk/f0q15d /dev/dsk/f05h	FD 0	1.2 MB	Escluso il blocco di boot
/dev/rdsk/f1q15dt /dev/rdsk/f15ht	/dev/dsk/f1q15dt /dev/dsk/f15ht	FD 1	1.2 MB	Intero disco
/dev/rdsk/f1q15d /dev/rdsk/f15h	/dev/dsk/f1q15d /dev/dsk/f15h	FD 1	1.2 MB	Escluso il blocco di boot
/dev/rdsk/f03ht	/dev/dsk/f03ht	FD 0	1.4 MB	$3^{1/2}$ pollici; intero disco

Tabella 18.1 Device file per i dischetti dei sistemi UNIX SVR4 (*continua*).

Device file	Device file a blocchi	Device	Dimensione	Commento
/dev/rdsk/f03h	/dev/dsk/f03h	FD 0	1.4 MB	$3^{1/2}$ pollici; escluso il blocco di boot
/dev/rdsk/f13ht	/dev/dsk/f13ht	FD 1	1.4 MB	$3^{1/2}$ pollici; intero disco
/dev/rdsk/f13h	/dev/dsk/f13h	FD 1	1.4 MB	$3^{1/2}$ pollici; escluso il blocco di boot
/dev/rdsk/f03dt	/dev/dsk/f03dt	FD 0	720 kB	$3^{1/2}$ pollici; intero disco
/dev/rdsk/f03d	/dev/dsk/f03d	FD 0	720 kB	$3^{1/2}$ pollici; escluso il blocco di boot
/dev/rdsk/f13dt	/dev/dsk/f13dt	FD 1	720 kB	$3^{1/2}$ pollici; intero disco
/dev/rdsk/f13d	/dev/dsk/f13d	FD 1	720 kB	$3^{1/2}$ pollici; escluso il blocco di boot

Tabella 18.2 Device file per i dischi rigidi dei sistemi UNIX SVR4.

Device file	Device file a blocchi	Device	Commento
/dev/rdsk/0s0	dev/dsk/0s0	HD 0	Intero disco
/dev/rdsk/1s0	dev/dsk/1s0	HD 1	Intero disco
/dev/rdsk/0s1	dev/dsk/0s1	HD 0	File system root
/dev/rdsk/1s1	dev/dsk/1s1	HD 1	File system addizionale 1
/dev/rdsk/0s2	dev/dsk/0s2	HD 0	Area di scambio
/dev/rdsk/1s2	dev/dsk/1s2	HD 1	Area di scambio
/dev/rdsk/0s3	dev/dsk/0s3	HD 0	File system **usr**
/dev/rdsk/1s3	dev/dsk/1s3	HD 1	File system addizionale 2
/dev/rdsk/0s4	dev/dsk/0s4	HD 0	File system **home**
/dev/rdsk/1s4	dev/dsk/1s4	HD 1	File system addizionale 3
/dev/rdsk/0s5	dev/dsk/0s5	HD 0	Solo partizione DOS
/dev/rdsk/1s5	dev/dsk/1s5	HD 1	Solo partizione DOS
/dev/rdsk/0s6	dev/dsk/0s6	HD 0	Dump device
/dev/rdsk/1s6	dev/dsk/1s6	HD 1	
/dev/rdsk/0s7	dev/dsk/0s7	HD 0	Boot device
/dev/rdsk/1s7	dev/dsk/1s7	HD 1	
/dev/rdsk/0s8	dev/dsk/0s8	HD 0	Settori alternativi
/dev/rdsk/1s8	dev/dsk/1s8	HD 1	
/dev/rdsk/0s9	dev/dsk/0s9	HD 0	Tracce alternative
/dev/rdsk/1s9	dev/dsk/1s9	HD 1	
/dev/rdsk/0s10	dev/dsk/0s10	HD 0	File system **stand**
/dev/rdsk/0s11	dev/dsk/0s11	HD 0	File system **var**
/dev/rdsk/0s12	dev/dsk/0s12	HD 0	File system **home2**
/dev/rdsk/0s13	dev/dsk/0s13	HD 0	File system **tmp**

Tabella 18.3 Device file per i nastri dei sistemi UNIX SVR4.

Device file	Tipo	Commento
/dev/rmt/c0s0	Streaming	Riavvolge dopo operazioni I/O
/dev/rmt/c0s0n	Streaming	Non riavvolge dopo operazioni I/O
/dev/rmt/c0s0r	Streaming	Ritensiona prima di I/O; riavvolge dopo I/O
/dev/rmt/c0s0nr	Streaming	Ritensiona prima di I/O; non riavvolge dopo I/O
/dev/rmt/f1q80	Floppy	Riavvolge dopo operazioni I/O
/dev/rmt/f1q80n	Floppy	Non riavvolge dopo operazioni I/O
/dev/rmt/f1q80r	Floppy	Ritensiona prima di I/O; riavvolge dopo I/O
/dev/rmt/f1q80nr	Floppy	Ritensiona prima di I/O; non riavvolge dopo I/O

0 (**f1** si riferisce all'unità a disco 1, se è presente sulla macchina) e **q15** specifica un dischetto a quadrupla densità con 15 tracce per cilindro, mentre nel nome **f0d9dt**, **d9** referenzia un dischetto a doppia densità con 9 tracce per cilindro. Per i dischetti $3^{1/2}$ pollici, nei nomi si usa di solito **d** per dischi a 720 kB a doppia densità e **h** per dischi ad alta densità da 1.4 MB.

Infine, **dt**, oppure **ht**, indicano che il disco non contiene il *boot block*, mentre **d** o **h** indicano un disco con il boot block. Generalmente si usa il formato **dt** (o **ht**), perché assicura la disponibilità di una maggiore quantità di spazio sul dischetto per i vostri usi. Tuttavia, alcune applicazioni sono distribuite su dischi nel formato **d**, e in questo caso per poterli leggere dovete utilizzare quel formato. Provate comunque con altri formati se non riuscite a leggere un disco in un determinato formato.

Quindi, il nome **f0q15dt** identifica un disco da 1.2 MB contenuto nell'unità 0, mentre **f13ht** inividua un dischetto da $3^{1/2}$ pollici da 1.4 MB contenuto nell'unità 1. Di solito, le unità che possono utilizzare dischi da 1.2 MB possono anche formattare e leggere dischi da 360 kB, ma non è possibile il contrario; ugualmente, dispositivi da $3^{1/2}$ pollici che possono usare dischi da 1.4 MB possono usare anche dischi da 720 kB, mentre dispositivi $3^{1/2}$ a doppia densità non possono usare dischetti da 1.4 MB.

I dispositivi di dischi rigidi seguono la stessa base di organizzazione, tuttavia ciascun disco rigido ha di solito file system indipendenti, denominati *slice*; ogni slice ha associato un file device a blocco e un file device raw.

Per formare i nomi identificativi degli slice vengono utilizzati diversi schemi; il più comunemente usato ha la forma */dev/dsk/nsm*, dove *n* è il numero del disco (**0** per il primo, **1** per il secondo) e *m* il numero di slice in quel disco. La Tabella 18.2 elenca i nomi per tutti gli slice, tuttavia gli slice effettivamente esistenti su un determinato sistema dipendono dalla configurazione della macchina. Non tutti i dispositivi elencati nella Tabella 18.2 saranno presenti su tutte le macchine, perché se non installate un file system alla configurazione del sistema, il suo device (e il suo slice sul disco) non compariranno nel directory **/dev/dsk**.

Uno schema alternativo di formazione dei nomi, usato principalmente

per dischi SCSI, utilizza la forma *c0t0d0s0*, in cui: *c0* si riferisce al controller numero **0**; *t0* si riferisce al target id **0**; *d0* si riferisce al dispositivo numero **0** sul target id **0**; *s0* si riferisce allo slice **0** sul dispositivo **0**. Il numero di slice, e la funzione di file system su quello slice corrispondono alla parte *s0* del nome in Tabella 18.2. I nomi con questo schema non sempre includono la parte *t0*; perciò, sono possibili file system con nomi del tipo **c0d0s3** oppure **c0t0d0s3**. Una determinata release di SVR4 userà comunque solo uno di questi schemi, mai ambedue.

Dopo questa introduzione, possiamo trattare l'uso dei device file.

18.11 Formattazione di dischetti

Per formattare un disco manualmente usate il comando **format**. Il comando **format** richiede come argomento il pathname completo del dispositivo che intendete usare per formattare:

```
$ format /dev/rdsk/f0q15dt
```

Questo formato richiede un disco ad alta densità e un drive da 1.2 MB. Inserite il disco nel drive e chiudete la porta del drive prima di lanciare il comando; accertatevi anche che a lato del dischetto non ci sia l'etichetta di protezione dalla scrittura. Al termine di queste operazioni avviene la formattazione del dischetto, che cancella ogni eventuale dato precedente.

Per le operazioni di formattazione si deve utilizzare sempre il dispositivo nel formato originale (**/dev/rdsk/...**) perché la formattazione è un'operazione di basso livello che richiede di accedere al disco a un livello inferiore del file system.

Il comando **format** formatta il tipo di disco indicato e verifica che il formato sia corretto:

```
$ format /dev/rdsk/f0q15dt
formatting.....
Formatted 160 tracks: 0 thru 159, interleave 2.
$
```

Altre unità a disco generano risultati leggermente diversi nell'uscita di **format**, perché il numero delle tracce può essere diverso:

```
$ /bin/format /dev/rdsk/f0d9dt
format limited to track: 80
formatting.....
Formatted 80 tracks: 0 thru 79, interleave 4.
$
```

Il comando **format** segnala se il disco è difettoso, o non ha il formato adatto, e non esegue la formattazione. In questo caso, dovete assicurarvi che il disco sia del tipo richiesto e si trovi nel drive corretto, in base al nome del dispositivo che state utilizzando. Se non riuscite a formattare un dischetto dopo due o tre tentativi, scartatelo; non usate dischi difettosi, perché i dati che potete perdere hanno generalmente un valore ben superiore al costo di un nuovo dischetto.

Una volta formattato, il disco non va riformattato, a meno che il contenuto non sia stato alterato per qualche motivo, o intendiate cancellare completamente i dati sul disco. Parte dell'operazione di formattazione consiste in un'azione di verifica che ha lo scopo di individuare i blocchi difettosi presenti sul dischetto, che vengono marcati come blocchi inutilizzabili. Talvolta, riformattando un disco che provoca frequenti errori se ne migliora nettamente il comportamento, perché sul dischetto viene creata una nuova tabella di blocchi difettosi.

18.12 Installazione di un file system su disco

Una volta che il disco è stato formattato correttamente, potete utilizzarlo nel formato originale oppure installarvi un file system. Il comando **/sbin/mkfs** (make file system) consente di creare un file system sul dischetto formattato; non è possibile montare un dischetto che non contiene un file system.

Quando create un file system dovete specificare il tipo di file system; usate **ufs** a meno di particolari condizioni che richiedano il tipo **s5**. Il tipo di file system deve essere specificato dopo l'opzione **-F** (File) nella linea di comando, inoltre, dovete specificare come argomento il nome del dispositivo:

```
$ mkfs -F ufs /dev/rdsk/f0q15dt 2400
```

Dovete utilizzare il dispositivo originale per creare un file system su dischetto. La creazione di un file system su dischetto ne cancella il contenuto precedente perché viene ricostruita la tabella dei file e dei directory.

DIMENSIONI DEL FILE SYSTEM

Oltre al nome del dispositivo, **mkfs** richiede di specificare il numero di blocchi da utilizzare per il file system. Questo numero segue il nome del dispositivo nella linea di comando **mkfs** ed è espresso in unità fisiche da 512 byte. Normalmente si utilizza per il file system l'intero disco, per cui potete ricavare il numero di blocchi dalla dimensione del disco: 1.2 MB diviso 512 byte determina che vengono riservati 2400 blocchi al file system su un disco ad alta densità.

Potete specificare un numero minore di blocchi, ma in questo modo viene sprecato lo spazio su disco non riservato al file system; viceversa, se specificate un numero di blocchi superiore, il comando **mkfs** segnala l'errore e non crea il file system.

Il comando **mkfs** riporta i valori utilizzati nella creazione del file system, come in questo esempio:

```
$ mkfs -F ufs /dev/rdsk/f13dt 1440
Warning: 18 sector(s) in last cylinder unallocated
/dev/rdsk/f13dt: 1440 sectors in 9 cylinders of 9 tracks, 18 sectors
        0.7Mb in 1 cyl groups (16 c/g, 1.33Mb/g, 256 i/g)
super-block backups (for fsck -b#) at:
  32,
$
```

Questo output di **mkfs** può differire leggermente in caso di creazione di un file system **s5** anziché **ufs**.

Eseguite un accurato controllo del dispositivo che specificate, perché in caso di un errore nell'identificare il device potete distruggere vostri importanti dati sul disco rigido o di un altro file system.

18.13 Come montare un dischetto

Una volta che il disco è stato formattato e il file system creato, potete montare il dischetto. Con "montare" un dischetto si intende collegare il suo file system al file system presente sul disco rigido della macchina. Dopo che avete montato un dischetto (o qualunque altro file system), potete passare da un directory a un altro, copiare i file tra i directory e utilizzare tutti i normali comandi del file system. Di solito per montare un file system è richiesto il privilegio di superuser.

Potete pensare al file system come a una parte della gerarchia di directory che si estende sotto un preciso directory. Ogni file system è una struttura completa di directory che comprende un directory radice (root) e un insieme di subdirectory che si aprono a ventaglio da questa radice. Quando montate un dischetto, inserite di fatto il suo directory radice in una particolare posizione del file system del disco rigido. Questa posizione, chiamata *mount point*, consiste in un normale directory presente nel file system del disco rigido. In altre parole, potete creare un directory sul vostro disco rigido e montare il dischetto in quel punto, poi potete passare a quel directory del disco rigido e vi troverete nel directory radice del file system che avete montato. Per esempio, la maggior parte dei sistemi fornisce un mount point predefinito nel directory /, in corrispondenza di **/mnt** (mount). Normalmente questo directory è vuoto:

```
$ ls /mnt
$
```

Quando montate un dischetto in quel punto, il directory radice del dischetto diventa il contenuto del directory **/mnt**:

```
# mount -F ufs /dev/dsk/f13dt /mnt
# ls /mnt
dati        esempio
#
```

Notate che nel comando **mount** occorre specificare il tipo di file system con l'opzione **-F**.

Se nel directory che corrisponde al mount point esistono già dei file prima di montare il dischetto, questi file vengono nascosti dall'operazione di montaggio e non sono accessibili fino a quando il file system montato non viene smontato. Anche se i file non vengono persi o modificati, è preferibile utilizzare come mount point un directory vuoto.

Il mount point può essere qualunque directory all'interno del file system, ma si utilizza frequentemente un mount point standard. A questo scopo, i sistemi SVR4 riservano il directory **/mnt** per caricare i dischetti o i nastri. In un mount point potete montare un unico file system, per cui ne occorre un secondo se avete un altro file system da montare, per esempio da due unità dischetto. Create il directory **/mnt2** come secondo mount point se non esiste già nel directory root.

Quando create un file system **s5** sul dischetto con l'interfaccia utente di gestione, potete specificare un nome di file system. Se non caricate il disco in questo mount point, il comando **mount** visualizza un messaggio di errore:

```
mount: warning: <nomefs> mounted at <mount-point>
```

dove *nomefs* è il nome del file system e *mount-point* è il mount point. Questo è un messaggio di avvertimento e non un effettivo errore.

Quando il dischetto è stato montato, il sistema UNIX segue le normali procedure per tenerlo aggiornato con i cambiamenti che effettuate; il dischetto agisce in tutto come un disco rigido. Il comando **mount** aggiunge un elemento alla *system mount table* contenuta in **/etc/mnttab**; questa tabella viene letta dai programmi che lavorano con le memorie di massa montate, come per esempio **df**. In ogni caso, non esiste una registrazione permanente dei file system che vengono montati, per cui se reinizializzate il sistema dopo che un dischetto o un nastro è stato montato, questa informazione viene persa. Per concludere regolarmente l'attività della macchina quando un dischetto è stato montato deve essere utilizzato il comando **shutdown**. È preferibile montare supporti rimovibili solo quando vengono impiegati e smontarli quando non servono più.

IL COMANDO mount

Il comando **mount** permette di montare un file system in uno specifico mount point. Indicate il nome di block device del dispositivo da montare come primo argomento e il mount point come secondo argomento:

```
# mount -F ufs /dev/dsk/f0q15dt /mnt
```

Questo comando carica il dischetto da 1.2 MB posto in drive 0 nel directory **/mnt**. Non potete rimuovere il dischetto dal drive fino a quando non lo avete smontato.

Una volta che avete caricato il dispositivo, potete passare da un directory a un altro attraverso il mount point, che è a tutti gli effetti un normale directory:

```
$ cd /mnt
$ ls -F
dati        file1        file2
$
```

Il file system che avete montato è accessibile in scrittura e in lettura, per cui dovete rimuovere dal dischetto l'etichetta di protezione dalla scrittura, prima di caricarlo. Il montaggio non viene accettato se il disco è protetto in scrittura, a meno che non abbiate specificato che il dischetto viene caricato in sola lettura; in questo caso potete utilizzarlo in lettura ma non in scrittura. Dovreste sempre montare i vostri più importanti dischi di archivio solo in lettura per evitare che vengano accidentalmente rovinati. Utilizzate l'opzione **-r** (read) con **mount** per montare un dischetto accessibile solo in lettura:

```
# mount -r -F ufs /dev/dsk/f0q15dt /mnt
```

Non potete scrivere un file né modificare un file system che è stato montato come accessibile solo in lettura.

Il comando **df** può riportare le caratteristiche di un dischetto che è montato:

```
# mount -F ufs /dev/dsk/f0q15dt /mnt
# df -t
/            (/dev/root       ):  140248 blocks  63537 files
                          total:  294986 blocks  73632 files
/proc        (/proc           ):       0 blocks     73 files
                          total:       0 blocks    102 files
/dev/fd      (/dev/fd         ):       0 blocks      0 files
                          total:       0 blocks     66 files
```

```
/stand          (/dev/dsk/0s10 ):    7307 blocks     97 files
                           total:   10710 blocks    104 files
/mnt            (/dev/dsk/f0q15d-
                              t):    2356 blocks    285 files
                           total:    2400 blocks    288 files
#
```

Queste informazioni permettono di tenere sotto controllo l'occupazione di spazio sui supporti rimovibili. Il comando **du** fornisce le previste informazioni sui directory dei file system montati.

SMONTARE UN DISCHETTO

Una volta terminato di utilizzare un dischetto o un nastro che avete caricato, dovete smontarlo prima di rimuoverlo dal drive. A questo scopo, esiste il comando **umount**, che richiede come argomento il nome del dispositivo:

```
# umount /dev/dsk/f0q15dt
#
```

In ambiente SVR4 potete anche utilizzare come argomento il mount point invece del nome del dispositivo:

```
# umount /mnt
#
```

Questi comandi smontano il dischetto o il nastro, che possono essere rimossi dal drive. Dovete smontare lo stesso dispositivo che avete precedentemente montato; se l'operazione ha successo, il comando **umount** restituisce silenziosamente il controllo allo shell.

Se qualche utente ha un file ancora aperto nel file system che è stato montato, oppure ha utilizzato il comando **cd** per entrare nel file system, **umount** non accetta la funzione. Tutti i file e i directory su disco devono essere rilasciati prima di lanciare il comando **umount**:

```
# mount -F s5 /dev/dsk/f0q15dt /mnt
mount: warning: <> mounted as </mnt>
# cd /mnt
# umount /dev/dsk/f0q15dt
umount: /dev/dsk/f0q15dt busy
# cd /
# umount /dev/dsk/f0q15dt
#
```

Tutti gli utenti devono rilasciare la risorsa prima che possa essere smontata; in questo modo, fortunatamente, non correte il rischio di smontare il disco rigido di sistema.

INFORMAZIONI SUI SUPPORTI MONTATI

Il comando **mount** senza argomenti fornisce informazioni sui dispositivi correntemente montati sulla macchina:

```
# mount
/ on /dev/root read/write/setuid on Wed Aug 26 10:45:03 1991
/proc on /proc read/write on Wed Aug 26 10:45:03 1991
/dev/fd on /dev/fd read/write on Wed Aug 26 10:45:03 1991
/stand on /dev/dsk/0s10 read/write on Wed Aug 26 10:45:03 1991
# mount -r -F ufs /dev/dsk/f0q15dt /mnt
# mount
/ on /dev/root read/write/setuid on Wed Aug 26 10:45:03 1991
/proc on /proc read/write on Wed Aug 26 10:45:03 1991
/dev/fd on /dev/fd read/write on Wed Aug 26 10:45:03 1991
/stand on /dev/dsk/0s10 read/write on Wed Aug 26 10:45:03 1991
/mnt on /dev/dsk/f0q15dt read only/setuid on Wed Aug 26 12:46:11 1991
# umount /mnt
# mount
/ on /dev/root read/write/setuid on Wed Aug 26 10:45:03 1991
/proc on /proc read/write on Wed Aug 26 10:45:03 1991
/dev/fd on /dev/fd read/write on Wed Aug 26 10:45:03 1991
/stand on /dev/dsk/0s10 read/write on Wed Aug 26 10:45:03 1991
$
```

Queste informazioni segnalano la condizione di sola lettura per il dischetto montato con l'opzione **-r**; l'indicazione *read only* cambia in *read/write* se non viene usata l'opzione **-r**.

18.14 Copie di dischetti

Il sistema UNIX fornisce due diverse procedure per copiare i dischetti. In entrambi i casi, dovete formattare il nuovo dischetto prima di realizzare la copia e assicurarvi che il tipo e il formato siano compatibili con quelli del disco originale.

COPIA MANUALE CON cp DI UN DISCO MONTATO

Dopo aver montato il dischetto secondo le modalità descritte, potete copiare singolarmente tutti i file dal dischetto in un directory temporaneo del disco rigido. Allo stesso scopo, potete anche usare il comando **cp** per copiare diversi file in una volta sola, oppure realizzare per l'occasione un file di comandi per facilitare l'operazione, oppure utilizzare il comando **cpio -p**, che vedremo al termine di questo capitolo.

Dopo di questo, caricate un nuovo disco formattato, con incluso il file system, e copiate i file dal directory temporaneo al nuovo disco con gli stessi comandi che avete utilizzato per copiare nel directory temporaneo. Se disponete di un secondo drive, potete inserirvi il nuovo disco formattato (con il file system), montarlo in un diverso punto di montaggio e copiare i file direttamente dal directory sorgente al directory destinazione.

Questa procedura risulta noiosa, è origine di errori e funziona solo per supporti rimovibili. Accertatevi di copiare esattamente tutti i subdirectory e i file i cui nomi iniziano con il carattere . (punto). Comunque, se intendete cancellare o modificare i file durante il trasferimento, questa è la sola procedura da seguire.

IL COMANDO dd

Il sistema UNIX dispone di un altro comando che realizza una copia completa di un dischetto, indipendentemente dal fatto che sia stato montato o no. Il nuovo disco deve essere già formattato e deve avere la stessa capacità e lo stesso tipo del disco originale che intendete copiare. Invece, non occorre installare un file system sul nuovo dischetto, perché la procedura di copiatura lo crea automaticamente come parte dell'operazione di copia.

Il comando **dd** esegue la copia esatta del contenuto dei supporti di memorizzazione; senza argomenti, copia lo standard input nello standard output, per cui potete copiare un file con la seguente linea di comando:

```
$ dd < in.file > out.file
```

Se uno dei file identifica un dispositivo, potete indicarne il pathname:

```
$ dd < /dev/rdsk/f0q15dt>/tmp/out.file
2400+0 records in
2400+0 records out
$
```

Il comando **dd** restituisce il numero di blocchi letti e scritti; il numero che segue il segno + indica il numero di blocchi parziali copiati.

Per copiare un dischetto, lanciate il comando **dd** per copiare il contenuto del dispositivo in un file temporaneo, come nell'esempio precedente, quindi sostituite il dischetto con un nuovo disco formattato dello stesso tipo e lanciate ancora **dd** per copiare il file temporaneo sul dischetto:

```
$ dd < /tmp/out.file>/dev/rdsk/f0q15dt
```

Questa procedura utilizza il nome originale del dispositivo per garantire che la copia dell'intero dischetto sia esatta e completa. Come ultima operazione, cancellate il file temporaneo.

La procedura descritta funziona indipendentemente dal tipo di dischetto o dal numero di file e directory in esso contenuti, perché considera il dispositivo come un unico grande file. Quando l'intero disco è stato copiato, viene letto il segnale di fine file e la procedura si conclude.

LA LINEA DI COMANDO dd

La sintassi della linea di comando **dd** differisce notevolmente da quella di molti altri comandi. Le opzioni a **dd** vanno specificate nella forma *parolachiave = valore*, dove *parolachiave* è l'opzione da assegnare, mentre *valore* corrisponde al valore da assegnare a quell'opzione. Per esempio, potete specificare il nome del file di ingresso dopo l'opzione **if=** (input file) e il nome del file di uscita dopo l'opzione **of=** (output file). Se queste opzioni sono specificate sostituiscono lo standard input e lo standard output:

```
$ dd if=/dev/rdsk/f0q15dt of=/tmp/out.file
```

Il comando **dd**, di default, per effettuare una copia legge un blocco di 512 byte, lo scrive, ne legge un altro e così via; questa operazione può risultare lenta, perciò sono previste opzioni che permettono di aumentare la dimensione del blocco (in multipli di 512 byte), migliorando le prestazioni. Una dimensione di blocco di 5120 byte consente un'esecuzione significativamente più rapida. L'opzione **bs** = (block size) permette di specificare la dimensione del blocco:

```
$ dd bs=5120 </dev/rdsk/f0q15dt >/tmp/out.file
240+0 records in
240+0 records out
$
```

In questo caso, la nuova dimensione di blocco viene assunta sia per le operazioni di ingresso che per le operazioni di uscita; il conteggio del numero di record copiati cambia essendo cambiata la dimensione del blocco. Questa dimensione di blocco è la più efficiente del comando **dd** per copiare dischetti; per copiare nastri magnetici conviene dichiarare **bs= 1024k**.

Potete anche specificare due diverse dimensioni di blocco per le operazioni di ingresso e di uscita; a questo scopo, le opzioni **ibs=** (input block size) e **obs=** (output block size) stabiliscono rispettivamente la dimensione di blocco in ingresso e la dimensione di blocco in uscita:

```
$ dd if=/dev/rdsk/f0q15dt ibs=5120 of=/tmp/out.file obs=51200
240+0 records in
24+0 records out
$
```

Quando il valore di **ibs** differisce da quello di **obs**, il conteggio dei record in ingresso e in uscita non coincide. Questo utilizzo delle opzioni risulta relativamente inefficiente perché **dd** deve costruire i blocchi in uscita prima di scriverli. Per fare la copia di un dischetto, utilizzate la stessa dimensione di blocco per tutte le operazioni di lettura e scrittura, preferibilmente con **bs=**.

Inoltre, il comando **dd** fornisce strumenti di conversione per cambiare i formati dei dati durante la copia. L'opzione **conv=** (convert) stabilisce un algoritmo di conversione: potete specificare **ascii** per convertire file da EBCDIC ad ASCII; **ebcdic** per convertire da ASCII a EBCDIC; **lcase** per avere una copia con soli caratteri minuscoli; **ucase** per avere una copia con soli caratteri maiuscoli. Inoltre, l'opzione **swab** consente di scambiare l'ordine di ogni coppia di byte del file e deve essere usata per trasportare file tra macchine che non utilizzano lo stesso sistema di ordine dei byte. Tutte le opzioni descritte possono essere combinate nell'opzione **conv**, separate dal carattere , (virgola).

```
$ dd bs=5120 conv=ascii,swab < in.file > out.file
```

Queste opzioni di conversione sono utili per le copie da file a file, ma poco utilizzabili per le copie multifile su disco.

Infine, il comando **dd** permette anche di trascurare alcuni blocchi in ingresso o in uscita prima di iniziare la copia. L'opzione **skip=***n* permette di ignorare i primi *n* blocchi del file di ingresso, prima di iniziare la copia e **seek=***n* consente di conservare i primi *n* blocchi nel file di uscita prima di cominciare a copiare i dati. Utilizzate con attenzione queste opzioni perché, se male impiegate possono rovinare un file system.

18.15 Accesso ai dispositivi: il comando cpio

Il comando **mount** consente di accedere tramite file system ai dischetti e ai nastri. Tuttavia, montare un dischetto è un'operazione relativamente lenta e il file system standard occupa su disco spazio che potreste altrimenti riservare ai vostri dati. Inoltre, non potete estendere i file oltre lo spazio disponibile sul disco che avete montato o suddividere un file tra più file system: il disco montato deve contenere interamente il file.

Per risolvere questi problemi, il sistema UNIX dispone del comando **cpio** (copy in/out) che consente di accedere semplicemente al dischetto senza necessità di eseguire **mount**.

Il comando è di fatto un programma di archiviazione che copia un elenco di file in un unico grande file di uscita, inserendo delle intestazioni che consentono di rintracciare i singoli file. Le opzioni del comando **cpio** permettono di creare gli archivi, oppure rileggere gli archivi e ricaricare i file in essi

contenuti. Il programma **cpio** viene di solito utilizzato per archiviare file su dischetti o nastri; tuttavia, può anche creare gli archivi direttamente sul disco rigido della macchina.

Gli archivi creati da **cpio** possono estendersi su più dischetti, rendendo possibili salvataggi efficienti di grandi gerarchie di directory. Inoltre, il comando **cpio** conserva le indicazioni di proprietà del file e le date di modifica e può archiviare sia file di testo che file binari. Il comando **cpio** è la maniera più efficiente per memorizzare file su un dischetto, inoltre un archivio di **cpio** risulta più compatto dei file originali che lo compongono. Il software di sistema e i dischi di molte altre applicazioni installabili hanno generalmente il formato del comando **cpio**.

I SUPPORTI DI MEMORIZZAZIONE NELLE OPERAZIONI DI cpio

Prima di lanciare il comando **cpio** con un dischetto o un nastro, dovete formattare il supporto di memorizzazione; potete invece non usare il comando **mkfs**, perché **cpio** distrugge il file system sul dischetto quando crea l'archivio. In realtà, se avete utilizzato un dischetto con **cpio**, dovete ricreare un file system sul disco prima di caricarlo con **mount**; i dischi utilizzati dal comando **cpio** non sono compatibili con i dischetti di file system e, se create un file system con **mkfs**, distruggete ogni dato memorizzato nel formato **cpio**.

Dovete utilizzare i dispositivi nel formato raw originale perché il comando **cpio** non impiega il file system. Poiché **cpio** invia l'archivio sullo standard output, il comando assume la seguente forma per scrivere su un dischetto:

```
$ echo nomefile ¦ cpio -o > /dev/rdsk/f0q15dt
```

Prima di lanciare qualsiasi comando **cpio**, accertatevi di avere introdotto un dischetto formattato nel drive adatto e chiudete la porta del drive. Il dischetto deve essere accessibile in scrittura per operazioni in uscita, ma può essere protetto in scrittura durante operazioni di lettura.

CREAZIONE DI ARCHIVI CON cpio

Quando create un archivio, il comando **cpio** accetta una lista di file o pathname sullo standard input e scrive l'archivio sullo standard output. I dati in uscita vengono quasi sempre ridiretti a un file oppure a un dispositivo. Dovete specificare i nomi dei file, uno per linea, sullo standard input; l'opzione **-o** (output) istruisce **cpio** a creare un archivio da questa lista di file:

```
$ echo "$HOME/miofile\n$HOME/tuofile" ¦ cpio -o > output.file
```

Spesso si utilizza il comando **ls** per archiviare un intero directory:

```
$ ls | cpio -o > output.file
```

Potete anche lanciare il comando **cpio** ridirigendo da un file la lista di nomi dei file:

```
$ cpio -o > output.file < lista.file
```

Si ottiene in uscita un unico file che contiene tutti i file specificati nel file in ingresso.

Il comando **cpio -o** accetta diverse altre opzioni. L'opzione **-a** (access) aggiorna l'indicazione delle date di modifica dei file, che altrimenti vengono preservate identiche ai file originali. L'opzione **-c** (character) produce le intestazioni interne nel formato carattere invece del formato binario, per una migliore portabilità dei dati. Dovreste utilizzare abitualmente il comando **cpio** sempre con l'opzione **-c**: la maggior parte dei file prodotti da **cpio**, che potrete ricevere, la utilizzano.

COMPATIBILITÀ DI cpio

In SVR4 il comando **cpio** è stato cambiato in maniera tale che archivi creati con la versione SVR4 possono non essere leggibili da sistemi UNIX precedenti. Per evitare questo e altri problemi, **cpio** consente di creare archivi con differenti formati di intestazioni, oltre al formato **-c**; l'opzione **-H** (Header) permette di selezionare i diversi formati. Per compatibilità con le versioni di UNIX precedenti a SVR4, usate questa linea di comando, omettendo l'opzione **-c**:

```
$ cpio -o -H odc > output.archiv < lista.file
```

L'opzione **-H** accetta anche: **crc** per controlli di ridondanza addizionali; **tar** per compatibilità col comando **tar** trattato più oltre in questo capitolo. Quando usate un'opzione **-H** nella creazione di un archivio, dovrete probabilmente usarla anche nella rilettura dell'archivio.

L'opzione **-B** (block) permette di creare blocchi separati di dati al posto di un flusso unico. Se il file archivio del comando **cpio** risiede sul disco rigido della macchina o viene trasferito tramite il sottosistema **uucp**, non occorre specificare l'opzione **-B**, al contrario, se l'archivio risiede su un dischetto o su un nastro, l'opzione **-B** è conveniente per accelerare l'accesso al disco.

L'opzione **-v** (verbose) consente di visualizzare su standard error i nomi di tutti i file che il comando **cpio** legge.

Per default, **cpio** non risolve i symlink; questo è di solito opportuno, perché gli obiettivi di collegamenti simbolici probabilmente non esistono quan-

do l'archivio viene ricostituito e **cpio** non potrebbe crearli. Se volete includere anche questi file nel vostro archivio, dovete specificare l'opzione **-L** (Link) quando create gli archivi e assicurarvi che i collegamenti esistano quando ricaricate gli archivi.

COME USARE IL COMANDO cpio

Per creare gli archivi, il comando **cpio** viene di solito abbinato al comando **find**. Il comando **find** ricerca in un directory i file che soddisfano gli argomenti nella sua linea di comando e scrive i pathname sullo standard output. Questi dati in uscita vengono convogliati, mediante una struttura pipeline, al comando **cpio** che archivia:

```
$ cd
$ find . -print | cpio -oc >/tmp/home.cpio
113 blocks
$
```

Il comando **cpio** restituisce il numero di blocchi scritti, se non si verificano errori. L'esempio precedente crea un archivio che contiene tutti i file e i subdirectory dell'home directory e lo memorizza in **/tmp/home.cpio**.

Potete riportare i dati in uscita su un dischetto, specificando il nome corretto del dispositivo:

```
$ find . -print | cpio -ocvB >/dev/rdsk/f0q15dt
```

È possibile ovviamente cambiare gli argomenti del comando **find** per selezionare altri file da includere nell'archivio. Per esempio, il comando **cpio** può creare un archivio che contiene solo i file che avete modificato nell'ultima settimana:

```
$ find . -mtime -7 -print | cpio -ocv>/tmp/home.cpio
```

Accertatevi di inserire l'opzione **-print** nella linea di comando **find**, altrimenti nessun nome di file viene trasmesso a **cpio** per essere archiviato.

Il comando **cpio** può anche archiviare un file che è già un archivio e tenere traccia dei livelli di archiviazione. In ogni caso, non dovete creare un archivio ridirigendo l'uscita in un file che si trova nello stesso directory che state archiviando, poiché si genera un ciclo infinito e un archivio che può raggiungere dimensioni incontrollabili. Per esempio, il seguente comando, anche se formalmente accettabile, dal punto di vista pratico genera seri problemi:

```
$ find . -print | cpio -oc > ./arch.cpio
```

LETTURA DI UN ARCHIVIO PRODOTTO DA cpio

Il comando **cpio** accetta l'opzione **–i** (in) per leggere gli archivi che ha creato con l'opzione **–o**. L'archivio da leggere è lo standard input di **cpio**, che ricrea i file in accordo ai pathname assegnati quando l'archivio è stato creato:

```
$ cpio –icv < /tmp/home.cpio
```

Se, creando l'archivio, sono stati utilizzati i pathname relativi, come nel caso di **find . –print**, il comando **cpio –i** tratta i file in ingresso come un albero di directory interno al directory corrente. Se, al contrario, l'archivio è stato creato utilizzando i pathname assoluti (che iniziano con il carattere /), il comando impiega lo stesso path assoluto quando ricrea il file. L'utilizzo dei pathname assoluti può risultare pericoloso perché non è possibile trasferire facilmente in una diversa locazione l'albero di directory in ingresso e così il comando **cpio –i** può cercare di riscrivere sui file originali.

Le opzioni **–c**, **–v**, **–B** e **–H** hanno in input lo stesso significato che hanno in output; se avete creato un archivio con le opzioni **–c**, **–B** o **–H**, dovete riusarle quando rileggete l'archivio, altrimenti il comando **cpio** non può eseguire l'operazione e segnala l'errore:

```
$ find . –print ¦ cpio –o >/tmp/home.cpio
113 blocks
$ cpio –ic </tmp/home.cpio
cpio: ERROR: This is not a cpio file. Bad magic number.
$
```

Se non conoscete le opzioni che il comando **cpio** ha utilizzato per creare l'archivio, dovrete fare delle prove con diverse combinazioni di **–c**, **–H** e **–B** fino a quando **cpio** legge il file correttamente.

Quando dirigete a un dispositivo l'archivio prodotto da **cpio**, dovete specificare il nome del dispositivo nel formato raw originale, corrispondente al formato di dischetto usato. Analogamente, quando rileggete l'archivio, dovete utilizzare lo stesso dispositivo:

```
$ find . –print ¦ cpio –ocB >/dev/rdsk/f0q15dt
113 blocks
$ cpio –icB </dev/rdsk/f0q15dt
113 blocks
$
```

Quando i file vengono ricaricati, conservano i loro diritti di accesso originali. Il proprietario e il gruppo di utenti del file diventano rispettivamente il vostro identificatore utente e il gruppo a cui appartenete, a meno che non siate collegati come supervisore; in questo caso, non cambiano né il gruppo né il proprietario originale. Gli identificatori di utente e di gruppo sono

espressi da codifiche numeriche, per cui se trasferite file da una macchina a un'altra come supervisore, dovete fare attenzione, perché gli stessi identificatori di utente e di gruppo possono appartenere anche a un utente dell'altra macchina. Dopo il caricamento di un archivio con **cpio** **-i**, verificate i diritti di accesso dei file con il comando **ls** **-l** per essere sicuri che siano adeguati e se necessario correggeteli. Si può creare grande confusione se gli identificatori di utente e di gruppo non sono corretti, specialmente se state trasferendo importanti file di sistema come **/etc/passwd**.

OPZIONI DEL COMANDO cpio IN INPUT

Il comando **cpio** dispone di molte opzioni per controllare la lettura dei file dall'archivio. Il comando, per default, non crea automaticamente i directory necessari per ricostruire i pathname dei file contenuti nell'archivio; l'opzione **-d** (directory) invece forza **cpio** a creare i directory necessari per i file che sta leggendo:

```
$ cpio -icBd </dev/rdsk/f0q15dt
```

I directory esistenti vengono utilizzati ma, se necessario, il comando **cpio** realizza nuovi directory per completare il percorso specificato.

Analogamente, **cpio** non riscrive un file esistente con lo stesso nome di un file letto da un archivio, se non specificate l'opzione **-u** (unconditional). Utilizzate questa opzione con grande attenzione perché l'archivio potrebbe contenere una versione rovinata o non aggiornata dei file. Generalmente conviene creare archivi con pathname relativi e poi ricaricarli in un directory temporaneo per evitare conflitti con i file esistenti.

Potete anche evitare questo problema utilizzando l'opzione **-r** (rename); quando **cpio** legge un file dall'archivio, richiede un nuovo nome di file da terminale:

```
$ cpio -icBr </dev/rdsk/f0q15dt
Rename <dati1>
```

Potete inserire un pathname oppure battere il tasto di ritorno a capo per ignorare il file. Questa procedura viene ripetuta per ogni file dell'archivio, per cui può risultare noioso ricaricare un archivio di grandi dimensioni.

L'opzione **-m** (modification) permette di conservare l'indicazione della data di aggiornamento del file originale (corrisponde alla data e ora corrente in cui il file originale venne creato o modificato); per default il file viene ricreato con data e ora corrente del ricaricamento.

VISUALIZZARE LA TABELLA DEL CONTENUTO DI UN ARCHIVIO

Potete esaminare il contenuto di un archivio prodotto da **cpio** senza di fatto ricaricarlo. Questo procedimento permette di risparmiare lo spazio su disco, ma non risulta più rapido di ricaricare completamente l'archivio perché occorre comunque esaminare l'intero archivio. Utilizzate l'opzione **-t** (table), insieme all'opzione **-i**, per elencare i nomi di file e altre informazioni, senza tuttavia creare nessun file. Anche in questo caso le opzioni **-c** e **-B** devono essere esatte per leggere l'archivio:

```
$ ls -l
total 128
-rw-r--r--    1 root    utenti     526 Aug 27 18:20 dati1
-rw-r--r--    1 root    utenti    6404 Aug 27 18:21 dati2
-rw-r--r--    1 root    utenti   57856 Aug 27 18:21 xx
$ find . -print | cpio -ocv >/tmp/arch.cpio
xx
dati1
dati2
128 blocks
$ cpio -ict </tmp/arch.cpio
.
xx
dati1
dati2
128 blocks
$ cpio -icvt </tmp/arch.cpio
drwxr-xr-x    2 root    utenti  0       Aug 27 18:20 1991 .
-rw-r--r--    1 root    utenti  526     Aug 27 18:20 1991 dati1
-rw-r--r--    1 root    utenti  6404    Aug 27 18:21 1991 dati2
-rw-r--r--    1 root    utenti  57856   Aug 27 18:21 1991 xx
128 blocks
$
```

L'output differisce sensibilmente quando utilizzate l'opzione **-v** con **-t**; assume l'aspetto della lista di **ls -l**. La colonna all'estrema sinistra indica i diritti di accesso ai file, le colonne successive riportano rispettivamente il proprietario, la dimensione in byte, la data e l'ora di modifica e il nome dei file. Questa informazione è memorizzata nell'archivio e appare quando leggete il suo contenuto.

SELEZIONARE UN SOTTOINSIEME DI FILE ARCHIVIATI

Potete chiedere al comando **cpio** di ricaricare solo un sottoinsieme dei file presenti nell'archivio, specificando un'espressione nel formato di metacaratteri di shell dopo le opzioni della linea di comando. Il programma **cpio** cerca tutti i file i cui nomi soddisfano l'espressione e carica solo quei file. Dovete delimitare tra virgolette le espressioni per evitare che lo shell li espanda prima che il comando **cpio** li interpreti:

```
$ cpio -icvB "*file" < /dev/rdsk/f0q15dt
```

Questo comando ricarica tutti i file i cui pathname terminano con la stringa **file**, dall'archivio che si trova sul dischetto da 1.2 MB.

Potete specificare più espressioni, ma ciascuna deve essere delimitata con le virgolette:

```
$ cpio -icvB "*file" "*[0-9]ciao*" <dev/rdsk/f0q15dt
```

Oltre ai file i cui pathname terminano con **file**, questo comando ricarica tutti i file i cui pathname contengono le cifre da 0 a 9 seguite dalla stringa "ciao", in qualunque posizione.

18.16 Archiviazione su dischetto o su nastro

Quando il comando **cpio** crea un archivio su un dischetto o su un nastro, l'archivio può superare la capacità di memorizzazione del supporto. Il comando **cpio** gestisce questa situazione e vi chiede di sostituire il dischetto con un altro (già formattato), per proseguire il vostro archivio su altri dischetti.

Quando un dischetto è pieno, il comando **cpio** chiede di sostituirlo con un altro; in questo modo:

```
$ ls | cpio -ocB >/dev/rdsk/f0q15dt
Reached end of medium on "output".
To continue, type device/file name when ready.
```

Dovete introdurre il pathname completo del dispositivo da usare, generalmente quello con cui avete iniziato. In questo caso, sostituite il dischetto con un altro dello stesso tipo, formattato correttamente, poi battete il pathname completo del dispositivo e premete il tasto di ritorno a capo; a questo punto il comando **cpio** riprende l'esecuzione. Con questo modo di gestione, potete anche cambiare tipo di dispositivo in qualunque momento; potete utilizzare dischetti di diversi formati, alternare da un drive a un altro e così via. In ogni modo, è consigliabile mantenere lo stesso formato dei supporti per un intero archivio. Se cambiate formato nel corso di un'operazione di **cpio**, marcate attentamente ogni dischetto con il corrispondente formato.

Nei sistemi SVR4 potete anche utilizzare l'opzione **-O** (O maiuscola) oppure **-I** (I maiuscola), seguita dal nome del dispositivo, per istruire **cpio** a utilizzare il dispositivo indicato al posto, rispettivamente, dello standard output o dello standard input. Per esempio:

```
$ find . -print | cpio -ocvB -O /dev/rdsk/f0q15dt
```

Questa opzione consente inoltre di visualizzare un messaggio diverso quando il dischetto si riempie:

```
$ find . -print ¦ cpio -ocB -O /dev/rdsk/f0q15dt
Reached end of medium on "output".
Change to part 2 and press RETURN key. [q]
```

Per continuare, inserite un disco formattato dello stesso tipo e poi premete il tasto di ritorno a capo. Potete ripetere questa procedura fino a quando l'archivio è completo, assicurandovi di marcare e numerare correttamente i supporti.

Analogamente, quando leggete un archivio con **cpio**, dovete introdurre i supporti nel corretto ordine, sostituendo un disco con il successivo quando occorre. Il comando **cpio** vi chiede di sostituire il disco o di indicare il nome del dispositivo, con le stesse modalità che abbiamo descritto precedentemente.

```
$ cpio -icBd -I /dev/rdsk/f0q15dt
Reached end of medium on "input".
Change to part 2 and press RETURN key. [q]
```

I dispositivi e l'ordine dei dischetti devono corrispondere esattamente a quelli stabiliti in fase di creazione dell'archivio.

Se volete interrompere l'esecuzione della procedura **cpio -i**, potete premere il tasto **q** (quit) quando appare il messaggio di cambio supporto; l'operazione termina e il controllo ritorna allo shell. I file già copiati vengono conservati; tuttavia, non potete iniziare a leggere l'archivio da un dischetto intermedio, ma dovete partire sempre dal primo dischetto della sequenza. Analogamente, è possibile interrompere l'esecuzione della procedura **cpio -o**, ma non è possibile ripartire in seguito dal punto di interruzione. In ambedue i casi, se il comando **cpio** non completa normalmente l'esecuzione, l'archivio può contenere un file parziale.

18.17 Salvataggio e ripristino di file

Per il salvataggio delle copie di sicurezza e il ripristino di file, sono disponibili strumenti nelle procedure per l'utente gestore del sistema, inoltre sono realizzabili diverse procedure manuali. Le più semplici procedure utilizzano i comandi **cpio** (o **tar**), tramite un elenco di file da archiviare, incluso di solito in una procedura di shell; questo è un metodo efficiente e diretto, che facilita l'esecuzione di salvataggi frequenti. Gli amministratori di sistemi di maggiori dimensioni possono trovare più convenienti i comandi **backup** e **restore**, che consentono salvataggi incrementali; questi stessi comandi ven-

gono utilizzati dalle procedure gestionali in **sysadm.** I file system **ufs** prevedono ulteriori procedure specifiche.

COPIE DI SICUREZZA CON cpio

La Figura 18.1 mostra un file di comandi che salva su un dischetto ad alta densità i file e i directory specificati, e anche i file contenuti nei subdirectory dei directory specificati. Potete ovviamente adattare questo file di comandi per comprendervi i file e i directory che voi intendete salvare periodicamente e per usare un qualunque altro tipo di supporto. Il comando che permette di leggere l'archivio è indicato all'inizio del file.

```
#! /sbin/sh
echo "Salvataggio dei file di sistema..."
echo "Ripristino con: 'cd / ; cpio -icvBdu -I</dev/rdsk/f0q15dt'"
echo "Inserire un dischetto formattato in alta densità e battere return."
read DUMMY
cd /

find \
etc/passwd \
etc/shadow \
etc/inittab \
etc/ttytype \
etc/profile \
etc/uucp/Sys* \
etc/uucp/Permissions \
etc/uucp/D* \
etc/conf/pack.d/kd/space.c \
etc/conf/node.d/osm \
etc/default/login \
home/giorgio/lib \
sbin/cleanup \
etc/saf/ttymon/_pmtab \
usr/spool/cron/crontabs \
.olinitrc \
.olprograms \
.olsetup \
.Xdefaults \
.profile \
etc/conf/node.d/asy \
etc/conf/sdevice.d/asy \
.kshrc \ -print | cpio -ocvB -O /dev/rdsk/f0q15dt
```

Figura 18.1 File di comandi che salva un insieme di file e directory.

Molti utenti predispongono diversi file di comandi, come quello di Figura 18.1, che consentono di salvare diverse aree di lavoro. I file indicati in figura sono un esempio di file di sistema che potrebbero essere modificati, ma sono realizzabili altre procedure adatte a salvare gli home directory o i database degli utenti.

COPIE DI SICUREZZA CON backup E restore

I comandi **backup** e **restore**, assieme ad altri comandi di supporto come **bkhistory**, **bkoper** e **bkreg**, sono usati all'interno delle procedure per l'utente gestore di sistema, per controllare l'esecuzione delle copie di sicurezza incrementali e di sistema. Mediante tabelle di file e di metodi nel directory **/etc/bkup** questi comandi consentono lo sviluppo di strategie e procedure di salvataggio per ogni situazione, includendo anche elenchi di file particolari da escludere nei salvataggi periodici. Le normali procedure per l'utente gestore consentono l'utilizzazione completa di questi strumenti, ma sono consentiti anche interventi manuali per esigenze speciali. Per usare questi strumenti, consultate le relative pagine di manuale ed esaminate anche i metodi previsti in **/etc/bkup**. Se intervenite manualmente nelle procedure, verificate accuratamente i file ripristinati, prima di fare affidamento sul metodo di salvataggio adottato.

COPIE DI SICUREZZA CON ufsdump

Oltre alle procedure già descritte, per i file system **ufs** è disponibile un altro strumento per salvataggi incrementali. I comandi **ufsdump** e **ufsrestore** consentono salvataggi periodici, a differenti livelli di salvataggio (*dump level*), nel senso che ciascun salvataggio a un dato livello copia dal file system solo i file che hanno subito modifiche dal momento dell'ultimo salvataggio a un livello inferiore. Questa procedura è usata principalmente nella gestione di sistemi molto grandi, perché è d'uso generale e perché in salvataggi di grandi dimensioni impiega molto minore spazio nei supporti; tuttavia, ha lo svantaggio che il ripristino dei file risulta più complesso che con le procedure descritte in precedenza. Se usate solo file **ufs** e avete esperienza nell'utilizzazione di questi strumenti in ambienti BSD, potete usarli con profitto; consultate le man page **ufsdump(1)** e **ufsrestore(1)** per le informazioni complete.

18.18 Nota sui salvataggi di file

In generale è meglio eseguire molti salvataggi parziali che un unico grande salvataggio perché salvataggi meno complessi sono più rapidi e più facili

da effettuare; inoltre, e molto più importante, se uno dei primi dischi di un archivio multidisco prodotto da **cpio** si rovina, è molto difficile leggere i restanti dischi (vedere in seguito il paragrafo in questo capitolo sugli archivi danneggiati). La realizzazione di diversi piccoli archivi aumenta notevolmente l'affidabilità della procedura di salvataggio e riduce la perdita di dati, qualora si dovesse verificare un'alterazione o danneggiamento dei supporti.

In ogni caso, vi consigliamo di effettuare il salvataggio dei dati con regolarità e tempestività, senza eccedere nella frequenza per non appesantire la gestione del sistema. Salvate dunque i dati almeno giornalmente, ma anche ogni ora nei giorni di maggiore attività; tuttavia, non occorrerà salvare con questa frequenza i dati che non hanno subito delle modifiche. Dovete comunque periodicamente ricaricare i supporti di salvataggio, per accertarvi che i vostri supporti di sicurezza siano sempre leggibili oltre che scrivibili; potreste essere in grado di scrivere le copie di salvataggio, ma non essere poi in condizioni di rileggerle quando fosse necessario. Tenete anche presente che i dischetti e i nastri si logorano dopo un uso frequente e quindi possono diventare illeggibili dopo un certo tempo.

Su una macchina multiutente in cui non è consentito accedere liberamente alle funzioni di salvataggio, potete copiare i vostri file su un file system di un altro disco fisico, in modo da cautelarvi dal deterioramento del disco originale. Consultate il gestore del sistema per concordare le procedure e le misure di sicurezza.

I guasti di sistema creano realmente problemi solo quando non avete eseguito le periodiche operazioni di salvataggio, in quanto determinano una dannosa perdita dei dati; mentre, se anche un guasto non dovesse mai verificarsi, il salvataggio regolare dei dati contribuisce ad aumentare la sicurezza interna dei vostri dati. Considerate che riscrivere i file e i dati eventualmente persi è sicuramente più costoso del tempo speso per salvarli regolarmente e dei denari necessari per acquistare supporti per le copie. Salvare i dati sistematicamente risulta sicuramente un'operazione tediosa, ma può farvi risparmiare mesi di lavoro.

18.19 Cura dei dischetti

I dischetti sono soggetti a rischi di danneggiamento e i campi magnetici sono il loro nemico più pericoloso, mantenete dunque i vostri dischetti lontani da grandi motori elettrici, magneti di ogni tipo e strumenti magnetizzati. Inoltre, potete alterare i dati su un dischetto avvicinandolo troppo a un moderno CRT a colori, o danneggiare il supporto esponendolo a temperature troppo alte o troppo basse. Non lasciate mai i supporti al sole o in un'auto calda; tenete i dischetti nel loro involucro di carta e metteteli in un posto pulito, senza polvere. Naturalmente non dovete rovesciare cibo, liquidi o ce-

nere su un dischetto e non dovete mai rimuovere il disco dal suo involucro di plastica.

Gli archivi di salvataggio vengono generalmente conservati in un luogo lontano dalla macchina, per evitare che si danneggino in caso di incendio o di altri disastri; le grandi società immagazzinano i supporti di sicurezza in speciali armadi o camere ad ambiente controllato. Un tale livello di sicurezza non è di solito necessario per macchine di piccole dimensioni, ma conviene sempre mantenere una copia dei dati in un ambiente diverso da quello in cui vengono elaborati.

18.20 Ricostruzione di file danneggiati

Se un archivio di sicurezza viene involontariamente danneggiato, è tuttavia possibile recuperare almeno parte dei dati; la versione SVR4 di **cpio** include l'opzione **-k** in input:

```
$ cpio -icdkBI /dev/rdsk/f13ht
```

L'opzione provoca l'abbandono da parte di **cpio** dei dati non leggibili per errori di I/O, a livello di file e anche di singolo blocco; se il danno è limitato a una piccola parte di un disco (come appunto un singolo blocco), è possibile recuperare i restanti dati su disco, prima e dopo il punto dell'errore. Il blocco, il file o i file con errori verranno comunque perduti; questa può essere considerata una soluzione non completa, tuttavia le versioni precedenti di **cpio** in caso di un errore nella lettura dei dati terminavano con abort.

18.21 Approfondimenti

Come prevedibile, vi sono ancora molti problemi relativi alle memorie di massa. Finora abbiamo trattato principalmente i dischetti, ma in ambiente UNIX si utilizzano altri tipi di supporti, soprattutto i nastri magnetici. In ogni caso, a eccezione dei pathname del dispositivo, questi supporti hanno generalmente un comportamento analogo a quello dei dischi, per cui rimangono valide le procedure **cpio** e talvolta anche le operazioni **mount**.

TRASFERIRE UNA INTERA GERARCHIA DI DIRECTORY

Il comando **cpio** consente anche di copiare un insieme di file da una locazione del file system a un'altra. A questo scopo, l'opzione **-p** (pass), utilizzata invece delle opzioni **-i** o **-o**, *passa*, ovvero trasferisce i file tramite il comando **cpio**, anziché realizzare un archivio. In questo caso, nella linea di comando dovete specificare il pathanme del directory finale, invece di ridirigere lo standard output come per le altre due opzioni di **cpio**:

```
$ ls | cpio -p /tmp/myfiles
```

Al solito, anche il comando **cpio -p** accetta un elenco di file sullo standard input e copia nel directory **/tmp/myfiles**, se esiste, tutti i file contenuti nel directory corrente.

Potete predisporre una struttura pipeline tra i comandi **cpio** e **find**, col risultato di copiare un'intera gerarchia di directory con una sola linea di comando:

```
$ find . -print | cpio -p /tmp/myfiles
```

Questo esempio differisce da quello precedente in quanto trasferisce l'intero albero di directory che si trova sotto il directory corrente.

Insieme all'opzione **-p** potete usare le consuete opzioni di **cpio**: **-a**, **-d**, **-u**, **-v** e **-m** per aggiornare le date di accesso al file, creare i subdirectory che occorrono, sovrascrivere i file esistenti con lo stesso nome, visualizzare i nomi di file quando vengono copiati o conservare la data della creazione del file originale.

Tenete presente che **cpio -p** esegue una reale copia dei file, per cui il copiare gerarchie di directory molto estese può richiedere una grande quantità di spazio su disco. Quando ciò è compatibile con le vostre esigenze, utilizzate l'opzione **-l** (link) per indurre **cpio -p** a collegare insieme i file piuttosto che copiarli, riducendo così lo spazio occupato su disco. Naturalmente non è possibile collegare i file tra file system diversi.

CREAZIONE DI FILE SYSTEM OTTIMIZZATI

Oltre a creare file system **s5** e **ufs** standard, il comando **mkfs** permette di cambiare il numero di i-node ammessi nel file system, di scegliere il formato dei gap tra i blocchi, di definire il numero di blocchi per cilindro e di stabilire configurazioni particolari di file system. Il procedimento presuppone una notevole esperienza sistemistica e richiede comunque un certo numero di tentativi, specie per l'ottimizzazione di file system **ufs**. Le opzioni in linea di comando e le configurazioni ammesse differiscono notevolmente tra i file system **s5** e **ufs**; consultate a questo riguardo la man page **mkfs(1)**.

Per file system **s5** potete definire il numero di i-node aggiungendolo al numero di blocchi, nella forma *:n*, senza spazi intermedi, in cui *n* è il numero di i-node da creare; per esempio:

```
# mkfs -F s5 /dev/rdsk/1s1 130000:30000
```

Potete specificare il gap tra i blocchi e il numero di blocchi per cilindro come argomenti addizionali nella linea di comando. Questi parametri possono migliorare l'efficienza delle operazioni di I/O su disco; vengono generalmen-

te utilizzati solo con i dischi rigidi. Dopo gli argomenti *blocchi:i-node*, indicate la dimensione del gap e il numero di blocchi per cilindro:

```
$ /mkfs – F s5/dev/rdsk/ls1 130000:30000 10 144
```

Questo comando crea un gap di 10 unità e suddivide ogni cilindro in 144 blocchi; si tratta di valori ottimali per un disco rigido Micropolis da 67 MB. Consultate la documentazione del vostro hardware per trovare quali sono i valori migliori per il vostro disco.

Potete anche indicare come argomento di **mkfs** un *file prototipo*, in cui avete memorizzato i dettagli del file system che intendete creare, inclusi i nomi dei file e dei directory del nuovo file system e i relativi diritti di accesso. Consultate la pagina di manuale **mkfs** relativa a **s5** per avere maggiori dettagli sull'uso del file prototipo. La configurazione di particolari file system **ufs** si effettua analogamente specificando parametri addizionali nella linea di comando, per esempio:

```
# mkfs – F ufs /dev/rdsk/f0q15dt 18 9 4096
```

Consultate la pagina di manuale **mkfs** relativa a **ufs** per maggiori dettagli sul significato dei parametri specifici.

MONTARE UN SECONDO DISCO RIGIDO

La maggior parte dei sistemi UNIX di piccole dimensioni dispone di un solo disco rigido, ma è probabile che possiate avere necessità di altro spazio su disco. Potete acquistare un disco rigido più grande per sostituire quello originale, oppure potete aggiungere un secondo disco rigido alla macchina. Accertatevi che la scheda del controllore del disco rigido che utilizzate supporti il nuovo disco e consultate il vostro fornitore per ottenere il disco e i cavi adatti, che dovete collegare con cura alla scheda del controllore.

Molti sistemi permettono di configurare i nuovi dischi mediante gli strumenti per l'utente di gestione; se il vostro sistema prevede questa possibilità, utilizzatela, perché la configurazione dei dischi rigidi può essere complicata; in caso contrario, ricercate il comando **/sbin/diskadd**, una procedura shell prevista per guidare l'introduzione dei dati necessari al procedimento di configurazione. Di solito, questi procedimenti consentono di configurare sia dischi SCSI, che dischi a "indirizzo fisso"; le opzioni in linea di comando possono cambiare a seconda del tipo di disco che avete scelto di installare.

Per configurare un secondo disco potete anche seguire procedure manuali. Dopo avere installato il disco, eseguite il comando:

```
# fdsk /dev/rdsk/ls0
```

per creare sul disco una partizione di sistema UNIX; quindi usate il comando **edvtoc** (edit volume table of contents) per creare gli slice sul disco per il sistema UNIX. Il comando **edvtoc** sostituisce il comando **mkpart** usato precedentemente per la creazione delle partizioni. La Tabella 18.2 contiene informazioni sulle partizioni dei dischi. Potete usare il comando **prtvtoc** (print vtoc) per stampare la mappa delle partizioni sui dischi esistenti.

Dopo la creazione delle partizioni, potete creare un file system nella prima partizione:

```
# mkfs -F ufs /dev/rdsk/lsl
```

A questo punto, potete montare le partizioni come spiegato nel paragrafo seguente. Questo procedimento può essere fonte di errori con distruzione delle informazioni sul disco, in compenso consente pieno accesso manuale al disco e alle sue partizioni.

Qualsiasi procedimento decidiate di usare, salvate completamente il materiale del primo disco rigido, prima di installare il secondo. Queste operazioni di configurazione del sistema presentano numerose aree di rischio, per cui talvolta vengono distrutti i contenuti del primo disco, pur facendo uso delle procedure per l'utente di gestione.

MONTAGGIO PERMANENTE DI FILE SYSTEM

Il file **/etc/vfstab** contiene una lista di file system, uno per linea, che sono montati in fase di inizializzazione del sistema. L'elenco include sia le risorse montate localmente, che le risorse remote montate tramite le reti (il Capitolo 24 contiene ulteriori informazioni sulle risorse montate in remoto). Ecco un esempio della lista:

```
# cat /etc/vfstab
/dev/root           /dev/rroot           /         ufs   1   yes   -
/dev/dsk/c0t0d0sa   /dev/rdsk/c0t0d0sa   /stand    bfs   1   yes   -
/proc               -                    /proc     proc  -   no    -
/dev/fd             -                    /dev/fd   fdfs  -   no    -
/dev/dsk/f0t        /dev/rdsk/f0t        /install  s5    -   no    -
/dev/dsk/f1t        /dev/rdsk/f1t        /install  s5    -   no    -
/dev/dsk/f0         /dev/rdsk/f0         /install  s5    -   no    -
/dev/dsk/f1         /dev/rdsk/f1         /install  s5    -   no    -
#
```

I campi di ciascuna linea contengono: block device; raw device; locazione nel file system (mount point); tipo di file system; indicazione se il file system deve essere verificato automaticamente a inizializzazione, **(1)**; indicazione se il file system deve essere montato dal comando **mountall**, **(yes)**; indicazione di opzioni addizionali per montaggio remoto.

Potete aggiungere nel file **/etc/vfstab** nuovi dispositivi che volete montare automaticamente, per esempio la seguente linea per montare un secondo disco rigido in **/usr/src**:

```
/dev/dsk/1s1    /dev/rdsk/1s1    /usr/src ufs    1    yes    -
```

Notate la distinzione tra i file **/etc/vfstab** e **/etc/mnttab**: il file **vfstab** definisce i dispositivi permanentemente montati all'inizializzazione, mentre il file **mnttab** elenca i dispositivi correntemente montati dopo una sequenza di comandi **mount** e **umount**. Una modifica manuale al file **mnttab** non ha alcun effetto sul sistema, mentre una modifica a **vfstab** avrà effetto alla successiva inizializzazione.

VERIFICA DEL FILE SYSTEM

Il comando **/sbin/fsck** (file system check) consente di verificare la correttezza di un file system e di ripristinarla se occorre. Si tratta di uno strumento complesso che può recuperare in parte i dati contenuti su un disco *logicamente* rovinato, ma che, utilizzato scorrettamente, può danneggiare un file system. L'uso di **fsck** è limitato al supervisore; come vedremo nel Capitolo 21, viene di solito usato per ricostruire i dati sul disco rigido dopo un guasto di sistema, ma consente anche di ripristinare il file system su supporti rimovibili, come un dischetto o un nastro.

Se sospettate che i dati di un dischetto contenente un file system siano danneggiati, potete salvare alcuni dei suoi file con **fsck**. Il comando **fsck** lavora solo su un supporto caricabile con **mount** e non su archivi prodotti da **cpio**. Se è possibile, prima di lanciare **fsck** copiate sempre, con il comando **dd**, il dischetto sospetto.

In ogni caso, alcuni dati vengono sempre persi nel ricostruire un file system incongruente, ma **fsck** garantisce di ridurre al minimo queste perdite.

Il comando **fsck** accetta come argomento il nome del dispositivo e ne controlla il file system:

```
# fsck /dev/rdsk/f0q15dt
```

La Figura 21.5 mostra alcune informazioni tipiche che **fsck** produce in uscita.

Quando il comando **fsck** incontra parti di file incoerenti, non collegati al file system in modo corretto, li collega automaticamente nel directory **lost+found** sotto il directory radice del file system verificato; dopo che **fsck** ha terminato, potete esaminare questi file per vedere se ne esiste qualcuno significativo.

Il programma di verifica **fsck**, per default, è uno strumento interattivo, che chiede conferma prima di apportare aggiustamenti al file system. Se

specificate l'opzione **–y** (yes) nella linea di comando, forzate **fsck** a scegliere in ogni occasione le risposte adatte; l'opzione è raccomandabile, in quanto le regole interne del comando forniscono sempre risposte migliori di quelle che potrebbe dare un utente anche esperto. L'opzione **–n** (no) induce **fsck** a controllare il file system, senza tuttavia apportare al file system alcuno degli aggiustamenti necessari; l'opzione è utile per verificare la correttezza di supporti protetti dalla scrittura.

Il comando **fsck** accetta molte altre opzioni, nella maggior parte dei casi non necessarie; sperimentatele solo su un file system che potete compromettere senza danno.

NASTRO MAGNETICO

Con il termine *supporto* o *memoria di massa* sono stati finora indicati tutti i dispositivi che possono essere utilizzati in un sistema UNIX: dischi rigidi, dischetti, nastri. In genere, questo è vero, in quanto molti dispositivi a nastro magnetico possono supportare file system su nastro, e questi nastri possono essere "montati" alla pari dei dischi. La differenza tra i vari dispositivi è data solo dai nomi che vengono utilizzati per il drive dei nastri.

In pratica, alcuni dispositivi a nastro non sono supportati da **mount**; questi sono i nastri *streamer*, che possono essere usati solo come dispositivi originali, in modo *raw*. Troverete indicata questa caratteristica nella documentazione del fabbricante del dispositivo e dovrete utilizzare il modo di accesso specificato. La release standard SVR4 supporta uno streamer a cassetta da 1/4 di pollice in formato QIC e un dispositivo SCSI con nastro ad alta capacità in formato DAT. La capacità dei nastri in formato QIC può variare; le più comuni nei sistemi SVR4 sono di 60 MB (QIC-24) e di 150 MB (QIC-150). Dispositivi a nastro funzionanti con capacità maggiori possono non essere in grado di scrivere nastri con capacità inferiori; se dovete scambiare dati su nastro con altri sistemi, consultate il vostro fornitore per assicurarvi che il vostro dispositivo sia compatibile con quelli degli altri sistemi.

La Tabella 18.3 fornisce un elenco dei nomi di dispositivo per le differenti funzioni supportate su nastri in SVR4. Se usate un intero nastro per un singolo archivio dovrete usare il dispositivo con riavvolgimento, per riposizionare il nastro all'inizio dopo il completamento dell'operazione di I/O; al contrario, potrete aggiungere uno o più archivi addizionali dopo il primo, utilizzando il dispositivo *senza* riavvolgimento, che lascia il nastro posizionato alla fine di un archivio esistente. Per ricaricare uno degli archivi successivi al primo dovrete posizionare correttamente il nastro all'archivio dovuto, usando il comando **tapecntl** trattato nella sezione seguente; potrete poi continuare a leggere file multipli in sequenza usando sempre il dispositivo senza riavvolgimento. Prima dell'ultima operazione su nastro dovrete passare a utilizzare un dispositivo *con* riavvolgimento, per riposizionare corretta-

mente il nastro all'inizio prima di rimuoverlo. Non è possibile rimuovere un nastro non riavvolto e poi ricaricarlo nella posizione in cui era stato lasciato, perché il dispositivo a nastro riavvolge automaticamente ogni nastro quando viene caricato.

Se riscontrate errori di I/O durante operazioni con nastro, potrete ritentare utilizzando il dispositivo con ritensionamento, che svolge il nastro interamente fino alla fine e quindi lo riavvolge fino all'inizio; questo consente un successivo scorrimento più regolare del nastro sotto le testine di lettura e scrittura.

Con unità a nastro streamer potete usare **cpio** per creare e rileggere nastri di archivio, come con i dischetti; per esempio:

```
$ find . -print | cpio -ocvB >/dev/rmt/c0s0
```

Il procedimento può risultare molto lento con archivi di grande estensione, conviene allora aumentare la dimensione del blocco:

```
$ find . -print | cpio -ocv -C 1024000 >/dev/rmt/c0s0
```

L'opzione **-C** rimpiazza l'opzione **-B** per specificare una dimensione di blocco in byte; il trasferimento può procedere molto velocemente utilizzando 1 MB, come in quest'ultimo esempio. Accertatevi di riutilizzare nella lettura dell'archivio la stessa dimensione di blocco utilizzata nella creazione.

Se sul vostro sistema disponete di un'unità a nastro *start-stop*, potete usare i device file della Tabella 18.3 per controllare l'unità a nastro, a livello di dispositivo raw, come un floppy disk. Questi nastri "floppy" possono quindi essere usati come floppy disk; in realtà, la forma più comune di nastro usa l'elemento **f1** in **/dev/dsk** e **/dev/rdsk**. Con il dispositivo a nastro potete usare tutti i comandi relativi al file system, già trattati. Per usare un nastro in questo modo, dovete prima formattare il nastro e quindi crearvi un file system.

IL COMANDO tapecntl

Il comando **tapecntl** consente di ritensionare e riposizionare i nastri. Il comando non esegue operazioni né di lettura né di registrazione, ma esclusivamente operazioni di posizionamento del nastro per successive operazioni di lettura o registrazione. L'opzione **-w** (wind) riavvolge il nastro; l'opzione **-r** (reset) reinizializza il dispositivo a nastro e riavvolge il nastro; l'opzione **-t** (tension) ritensiona il nastro; l'opzione **-e** (erase) cancella totalmente un nastro e ovviamente deve essere sempre usata con molta attenzione.

Il comando **tapecntl** consente inoltre di riposizionare il nastro a un determinato archivio, o file; avendo provveduto a contare, a partire da 1, i file

dall'inizio alla fine del nastro, potete leggere il file numero *n*, con la procedura seguente:

```
# tapecntl -w
# tapecntl -p n
# dd if=/dev/rmt/c0s0 of=/tmp/infile bs=1024k
#
```

L'opzione **-p** è seguita dal numero di file e il comando **tapecntl** posiziona il nastro all'inizio di quel file; quindi, potete leggere (o scrivere) a partire da quel punto con **dd**, **cpio** o **tar**. Usando nel vostro comando di lettura un nome di dispositivo senza riavvolgimento, potete continuare la lettura del file direttamente successivo al file che avete appena letto. Notate che se scrivete un file su un nastro multifile, posizionandovi su un file che non sia l'ultimo, tutti i file successivi verranno perduti; perciò dovrete porre molta attenzione nella registrazione su nastri che contengono file multipli.

IL COMANDO tar

Oltre a **cpio**, è disponibile anche un altro comando, **tar** (tape archive). Il comando **tar** è originario della release BSD, e venne inizialmente realizzato per l'archiviazione su bobine di nastro a nove tracce, tuttavia viene frequentemente utilizzato per registrare archivi sia su nastri in cassette che su dischetti. Molti sistemi più vecchi supportano **tar** anziché **cpio**. Il comando **cpio** non consente di sostituire un file in un archivio con una versione più recente senza ricreare completamente l'archivio; al contrario, il comando **tar** permette di aggiungere nuovi file alla fine di un archivio esistente e consente di sostituire i file nell'archivio. Per realizzare questa operazione di sostituzione con **tar**, è sufficiente scrivere il nuovo file in fondo all'archivio, in modo che quando i file vengono ricaricati dall'archivio, l'ultimo file si sovrascrive a tutti quelli che hanno lo stesso nome. Le modalità di impiego del comando **tar** risultano leggermente più complesse rispetto a quelle del comando **cpio** perché voi stessi dovete gestire gli archivi sui supporti; in altre parole, se create un archivio con tre versioni di un file, dovete accertarvi che l'ultima sia sempre la sola che interessa, perché **tar** può scegliere solo l'ultima occorrenza di un file. Naturalmente nelle operazioni di salvataggio l'ultima versione risulta generalmente quella che interessa perché corrisponde alla più recente.

Il comando **tar** è notevolmente più lento in esecuzione rispetto al comando **cpio**, tuttavia può utilizzare il dispositivo nel formato originale o nel formato a blocchi, e il primo tipo risulta sensibilmente più rapido. Infine, **tar** non può scrivere un secondo disco dopo che il primo disco si è riempito, per cui i suoi archivi sono limitati alla dimensione massima dei supporti da voi utilizzati.

LA LINEA DI COMANDO tar

Il comando **tar** accetta come primo argomento dopo **–f** il nome di un file normale o il nome di un dispositivo, che considera l'archivio da creare; gli argomenti che seguono il nome dell'archivio specificano i file da archiviare. A differenza di **cpio**, il comando **tar** archivia automaticamente tutti i subdirectory dei directory indicati. Per esempio, il seguente comando archivia su un dischetto ad alta densità i directory **/home/giorgio** e **/usr/src**, con tutti i relativi subdirectory:

```
$ cd /
$ tar –cf /dev/rdsk/f0q15dt home/giorgio usr/src
```

Se come nome dell'archivio è indicato il carattere – (segno meno), il comando **tar** utilizza l'I/O standard, che potete quindi ridirigere.

Il comando **tar** può creare, leggere e aggiornare un archivio, se specificate rispettivamente le opzioni **–c** (create), **–x** (extract) e **–r** (replace). Per creare un nuovo archivio, indicate nell'ordine l'opzione **–c** insieme con **–f**, il dispositivo di archiviazione e i file di ingresso. L'esempio mostrato distrugge il precedente contenuto dell'archivio. Se l'operazione ha successo, **tar** termina l'esecuzione senza visualizzare alcun messaggio. Come il comando **cpio**, **tar** memorizza il pathname utilizzato quando viene creato l'archivio; per cui dovete fare attenzione se usate i pathname completi, perché quando i file vengono estratti dall'archivio sovrascrivono i file esistenti. Per evitare questo rischio conviene dunque usare solo pathname relativi nelle linee di comando **tar**.

Per ricaricare l'archivio, utilizzate l'opzione **–x** seguita dal nome dell'archivio che intendete ricaricare:

```
$ tar –xf /dev/rdsk/f0q15dt
Tar: blocksize = 20
$
```

Questa procedura ricarica l'archivio nel directory corrente, a condizione che l'archivio sia stato creato con pathname relativi, come nel primo esempio. Il comando **tar** restituisce la dimensione di blocco utilizzata nella fase di creazione dell'archivio.

Se viene lanciato con una lista di nomi di file di seguito al nome dell'archivio, **tar –x** estrae solo i file specificati:

```
$ tar –xf /dev/rdsk/f0q15dt usr/src/giorgio/bsplit.c
```

Assicuratevi di utilizzare lo stesso pathname che avete impiegato quando l'archivio è stato creato.

Utilizzate l'opzione **–r** per sostituire un file o aggiungere nuovi file senza distruggere l'archivio esistente; per esempio:

```
$ tar –rf /dev/rdsk/f0q15dt usr/src/giorgio/bsplit.c
```

L'opzione **-x** estrae la versione del file più recente quando ne esiste più di una nell'archivio.

Potete visualizzare una tabella di contenuti dell'archivio tramite l'opzione **-t** (table):

```
$ tar -ft /dev/rdsk/f0q15dt
home/giorgio/dati1
home/giorgio/dati2
home/giorgio/cpio.out
usr/src/bsplit.c
usr/src/bsplit
$
```

L'opzione **-v** (verbose) produce un elenco di file scritti in, o letti da, un archivio; utilizzata insieme all'opzione **-t** produce una tabella di contenuti simile a quella che si ottiene con il comando **ls -l**.

Infine, l'opzione **-w** (what) induce **tar** a chiedere conferma all'utente prima di intraprendere qualunque azione; se rispondete **y**, il comando esegue l'azione, mentre se inserite qualunque altro carattere, il comando ignora l'azione e passa oltre.

```
$ tar -fxw /dev/rdsk/f0q15dt usr/src/bsplit.c
x usr/src/bsplit.c:
```

La lettera **x** che precede il nome del file preannuncia l'intenzione di **tar** di estrarre dall'archivio quel file.

Con l'opzione **-w** potete recuperare una versione precedente di un file, se accettate la versione che interessa e rifiutate quelle successive.

MONTARE IL DISCO RIGIDO DA UN DISCHETTO DI AVVIAMENTO

A volte può essere necessario inizializzare il sistema da un dischetto, per esempio quando dimenticate la password di **root**. In questo caso dovete inizializzare con un dischetto, quindi montare il file system del disco rigido e modificare con editor il file **/etc/passwd** o **/etc/shadow** per rimuovere la password **root**. Poi potete avviare di nuovo il sistema dal disco rigido e iniziare a lavorare senza difficoltà. La procedura di avvio da dischetto verrà trattata nel Capitolo 21.

Una volta che la macchina è stata avviata dal dischetto (oppure da un disco RAM), potete montare il disco rigido di default ed esaminare o correggere dati sul disco. Utilizzate il seguente comando:

```
# mount /dev/dsk/0s1 /mnt
```

In questo caso, montate il file system root del disco rigido in corrispondenza di un mount point sul dischetto; tutti i file che si trovano in quella parti-

zione sul disco rigido saranno disponibili all'uso. Per accedere a file in altri file system consultate la Tabella 18.2 per individuare il nome di dispositivo esatto.

QUOTE

Il file system di tipo **ufs** consente di stabilire per ciascun utente *quote* di utilizzo dello spazio su disco. Diversamente da **ulimit** che controlla la dimensione di ciascun file, le quote limitano gli utenti a un utilizzo di una dimensione massima di spazio su disco per tutti i loro file all'interno di un file system. Le quote non sono previste per i file system di tipo **s5**; questa caratteristica è comunque gestita da comandi riservati al superuser.

Il sistema delle quote viene attivato con il comando **quotaon**, che accetta come argomenti un elenco di nomi di file system; potete usare l'opzione **-a** che seleziona tutti i file system; per esempio:

```
# quotaon -v -a
```

L'opzione **-v** elenca tutti i file system interessati dall'operazione. Il meccanismo delle quote viene disabilitato mediante il comando **quotaoff** con le stesse opzioni.

Per abilitare la gestione delle quote per ciascun utente dovete creare un file di nome **quotas** nel directory di massimo livello di ciascun file system del sistema; dovete quindi usare il comando **edquota** (edit quotas) per creare nel file i dati correttamente formattati; per esempio:

```
# > /quotas
# edquota giorgio
```

Il comando **edquota** è necessario per convertire il file binario **quotas** di ciascun file system a un formato a noi comprensibile; il comando usa la variabile **EDITOR** per determinare l'editor da usare, e tratta ciascun utente specificato nella linea di comando. Per limitare gli errori è preferibile trattare con **edquota** un singolo utente alla volta.

Nella sessione di editor compare una linea per ciascun file system della macchina che ha un file **quotas**; per esempio:

```
fs /mnt blocks (soft = 0, hard = 0) inodes (soft = 0, hard = 0)
fs / blocks (soft = 0, hard = 0) inodes (soft = 0, hard = 0)
```

L'elenco di questo esempio specifica per ciascun file system le risorse soggette alle quote per l'utente in questione; per ogni file system che volete controllare dovete modificare i valori zero nei valori di limite di blocchi e i-node. Tenete conto che il valore zero ha il significato speciale di concedere spazio illimitato, curate perciò di modificarlo sempre ai valori opportuni.

Quando la gestione delle quote è attivata, a ogni utente viene impedito di usare più risorse di quelle concesse, come nell'esempio:

```
$ cp /etc/profile ./myfile
cp: cannot create ./myfile
cp: Permission denied
$
```

Potete ottenere un rapporto dello spazio usato da un utente rispetto alle sue quote, col comando **quota** con argomento l'identificatore dell'utente:

```
# quota -v giorgio
Disk quotas for giorgio (uid 104):
Filesystem usage  quota  limit  timeleft  files  quota  limit timeleft
/              86    100    100              14      0      0
/mnt            0      2      2               0      2      2
$
```

Questo comando viene spesso incluso in un rapporto periodico per tenere sotto controllo l'utilizzazione del disco da parte degli utenti; l'opzione **-v** è opportuna per produrre l'elenco nel formato dell'esempio.

Infine, il comando **repquota** produce un rapporto delle quote e dello spazio su disco impegnato da un utente e **quotacheck** verifica la coerenza del sistema di gestione delle quote.

Capitolo 19

Il sistema MS-DOS in ambiente UNIX

Una delle più interessanti innovazioni di UNIX negli ultimi anni riguarda la possibilità di creare un ambiente MS-DOS su macchine UNIX. In pratica, gli elaboratori basati su microprocessori Intel 80286 e 80386 hanno la possibilità intrinseca di eseguire in maniera indipendente i sistemi operativi MS-DOS; tuttavia, molte release recenti di sistemi UNIX per macchine di quel tipo consentono l'esecuzione di MS-DOS contemporaneamente al sistema UNIX.

Le versioni di sistemi UNIX che funzionano su architetture hardware diverse richiedono spesso schede di coprocessore, che contengono il microprocessore compatibile con MS-DOS; altre implementazioni prevedono ambienti di emulazione software, che appaiono alle applicazioni come un processore 80x86, ma funzionano sul microprocessore nativo della macchina.

Queste caratteristiche, in qualunque forma siano realizzate, consentono alla macchina di suddividere le proprie risorse fra i due sistemi operativi

in esecuzione contemporanea. Potete lanciare una sessione MS-DOS da shell UNIX, o in una console virtuale UNIX; mentre MS-DOS è in funzione potete in qualunque momento passare al sistema UNIX, ed eseguire comandi come al solito, in un'altra console virtuale; potete ancora ritornare in MS-DOS per lavorare sotto quel sistema operativo. Quando non avete più necessità del sistema MS-DOS potete chiudere quel processo e ritornare al normale sistema UNIX. Il processo MS-DOS coesistente usa lo stesso file system del sistema UNIX, in questo modo potete agevolmente condividere file e directory fra i due sistemi operativi, e usare nella stessa sessione sia comandi UNIX che MS-DOS.

In molte release, specialmente in quelle per macchine 80386 e 80486, è possibile eseguire contemporaneamente nella macchina più sessioni MS-DOS; sotto X Window System è anche possibile eseguire sessioni MS-DOS in più finestre. In ogni caso, solo una di queste sessioni alla volta può accedere ai dispositivi hardware, come i dischetti. Altre release, specialmente quelle basate sull'emulazione software, consentono l'esecuzione nella macchina di una sola sessione alla volta.

Oltre alla caratteristica della coesistenza MS-DOS e UNIX (*Merge*), quasi tutte le release SVR4 includono in file system strumenti che consentono di leggere o scrivere dischetti MS-DOS direttamente col sistema UNIX. Se la vostra macchina è dotata di dispositivo a dischetti compatibile MS-DOS, potete trasportare file tra i sistemi UNIX e MS-DOS con questo mezzo. Se sul disco rigido della vostra macchina esiste una partizione separata MS-DOS potete copiare file tra questa partizione e le vostre partizioni di sistema UNIX.

La caratteristica di coesistenza MS-DOS sotto UNIX comporta di solito un costo aggiuntivo, specialmente se richiede una scheda di coprocessore; inoltre, può richiedere ulteriore memoria RAM e una maggiore velocità di CPU, rispetto all'utilizzo di solo UNIX. Al contrario, molte release di SVR4 includono già la capacità di leggere e scrivere dischetti MS-DOS; pertanto, se non avete assoluta necessità di eseguire applicazioni MS-DOS, potete evitare l'acquisto del pacchetto *Merge* e di hardware addizionale.

19.1 Uso dei dischi MS-DOS

Se non avete necessità di eseguire le applicazioni MS-DOS potete leggere e scrivere supporti MS-DOS direttamente col sistema UNIX; in questo modo, potete trasportare dati fra i due sistemi operativi senza acquistare il pacchetto addizionale Merge. Alcune release di SVR4 consentono anche di montare supporti MS-DOS direttamente nel file system UNIX. Per usare queste caratteristiche, la vostra macchina deve disporre di un dispositivo a dischetti in grado di accettare dischetti MS-DOS; di solito i dispositivi da $5^{1/4}$ pollici da 1.2 MB e quelli da $3^{1/2}$ pollici da 1.4 MB hanno questa possibilità, che manca invece in altri tipi.

Per accedere ai file system MS-DOS sono disponibili diversi comandi, come per esempio:

```
$ ls /usr/bin/dos*
/usr/bin/doscat
/usr/bin/doscp
/usr/bin/dosdir
/usr/bin/dosformat
/usr/bin/dosls
/usr/bin/dosmkdir
/usr/bin/dosrm
/usr/bin/dosrmdir
$
```

Questi comandi operano come i loro simili; per esempio, **doscp** opera come **cp** per copiare file tra sistemi UNIX e MS-DOS, e **dosformat** formatta dischetti MS-DOS. In alcune versioni di SVR4 questi programmi di accesso a MS-DOS possono essere inclusi nel pacchetto *Xenix Compatibiliy Package*, che sarà fornito a parte.

Ciascun comando accetta un designatore di drive MS-DOS, come **a:** o **b:** per specificare l'obiettivo dell'azione; per esempio, potete formattare un dischetto MS-DOS con:

```
$ dosformat a:
Insert new diskette for /dev/rdsk/f0t
and press <RETURN> when ready

Formatting....Format complete

    1213952 bytes total disk space
    1213952 bytes available on disk
$
```

In questo esempio il drive **a:** è un dispositivo $5^{1/4}$ pollici da 1.2 MB. Se aggiungete l'opzione **–v** (volume) **dosformat** richiede l'introduzione di un'etichetta di volume da assegnare al dischetto formattato.

Il comando **doscp** copia un file tra i file system UNIX e MS-DOS; per esempio:

```
$ doscp /etc/profile a:
```

Potete anche indicare un nome di file e directory di MS-DOS; per esempio:

```
$ doscp a:/profile /tmp/file
```

In questi comandi per separare gli elementi di percorso dei file MS-DOS dovete seguire lo stile UNIX, usando / (barra), invece dello stile MS-DOS, che

usa \ (barra inversa). Tenete anche presente che i nomi dei file creati sui supporti MS-DOS devono rispettare le convenzioni dei nomi MS-DOS: possono essere composti di otto caratteri al massimo e possono avere un'estensione composta da un punto e tre caratteri al massimo. Se eccedete questi limiti **doscp** troncherà il nome di file. Tutti i nomi legali sotto MS-DOS sono legali anche sotto UNIX.

I comandi **dosdir** e **dosls** riportano il contenuto del disco MS-DOS; l'uscita di **dosdir** appare così:

```
$ dosdir a:
  Volume in drive A: has no label
  Directory of A:

PROFILE          1092   6-23-90   12:51a
        1 File(s)   1212416 Bytes free
$
```

Il comando **dosls** riporta i risultati in una forma simile a quella prodotta da **ls**.

I comandi **dosmkdir** e **dosrmdir** operano come i normali comandi MS-DOS **mkdir** e **rmdir** per creare e cancellare directory nel disco MS-DOS; richiedono un nome di directory come argomento:

```
$ dosmkdir a:/subdir
```

Usate sempre / nei pathname di disco MS-DOS. Il comando **dosrm** cancella un file dal disco MS-DOS.

```
$ dosrm a:profile
```

Potete usare **doscat** per scrivere file dal disco MS-DOS allo standard output, senza copiarli.

Tutti questi comandi, come logico, operano con drive **a:** e **b:**, ma operano anche con partizioni standalone MS-DOS su disco rigido; se avete configurato la macchina con una partizione MS-DOS oltre alla partizione UNIX, potete operare con questi strumenti anche sulla partizione MS-DOS. La partizione del disco rigido dedicata a MS-DOS è visibile nel sistema UNIX come **/dev/dsk/0s5** oppure **/dev/dsk/dos**; è visibile nei comandi "DOS" come drive **d:**. In questo modo, potete copiare file tra il sistema UNIX e la partizione standalone MS-DOS specificando nei comandi il drive **d:**. Per vedere il contenuto della partizione MS-DOS potete usare:

```
$ dosdir d:
```

Non è possibile eseguire direttamente nessun comando MS-DOS da nessuno di questi drive, ma è possibile eseguire qualsiasi operazione sui file.

La mappatura tra le lettere dei drive MS-DOS e le locazioni di file system UNIX sono definite nel file **/etc/default/msdos**; potete cambiare il significato delle lettere dei drive MS-DOS o aggiungere nuove lettere modificando con editor questo file.

19.2 Montaggio dei dischi MS-DOS

In aggiunta ai comandi di accesso diretto sopra descritti, alcune release di SVR4 consentono di montare file system MS-DOS nel file system UNIX. In questo modo è possibile percorrere con **cd** l'albero di directory e accedere ai file MS-DOS con i normali comandi UNIX. Per esempio, il comando:

```
# mount -F dos /dev/dsk/f0t /mnt
```

può consentire di montare il dischetto **a:** al mount point **/mnt**; se il sistema ha la possibilità di montare dischi MS-DOS il comando verrà accettato. Potete montare la partizione standalone MS-DOS del disco rigido con **/dev/dsk/0s5**.

Neanche in questa situazione avrete la possibilità di eseguire i comandi MS-DOS direttamente nella partizione montata; tuttavia il directory montato seguirà le regole di gestione dei file del sistema UNIX, e potrete trattare i file normalmente senza la necessità di usare i comandi "DOS" descritti sopra.

19.3 Conversione di file

Il formato dei file su disco differisce nei sistemi MS-DOS e UNIX, a causa delle diverse convenzioni nelle regole di terminazione delle linee di testo. Queste differenze sono principalmente avvertibili nei file di testo, ma coinvolgono anche i file binari e i programmi eseguibili, così come i file binari che contengono linee di testo (per esempio alcuni file documento di word processor).

In ambiente UNIX, ogni linea di testo termina con il *newline*, carattere di ritorno a capo, che di fatto è il carattere ASCII CTRL-J, cioè il *linefeed*, carattere di avanzamento di interlinea; in ambiente MS-DOS ogni linea termina con la coppia ritorno carrello e avanzamento di interlinea, ovvero CTRL-M seguito da CTRL-J. Quindi, quando trattate un file MS-DOS sotto un editor di UNIX, noterete uno spurio CTRL-M al termine di ogni linea; analogamente, quando lanciate un comando MS-DOS, come per esempio **type** per esaminare un file creato in ambiente UNIX, ogni linea terminerà con un avanzamento di interlinea, ma la linea successiva non inizierà al margine sinistro. Per esempio, questo file UNIX:

```
$ cat dati
123 456 789
987 654 321
salve ciao
$
```

sotto MS-DOS apparirà come:

```
C> TYPE DATI
123 456 789
          987 654 321
                    salve ciao
C>
```

Nell'esempio, ogni linea del file occupa una linea di video, ma la mancanza del ritorno carrello non riporta il cursore al margine sinistro.

I comandi "DOS" descritti sopra, quando copiano file tra i due formati di file system compiono automaticamente le conversioni necessarie, in ambedue i sensi; anche i file montati vengono opportunamente convertiti. Tutte queste operazioni di conversione hanno luogo automaticamente, in modo che le differenze di formato nei file di testo di solito non presentano problemi. Tuttavia, i programmi eseguibili e i file di database non possono essere convertiti con le stesse regole; dovrete compiere alcuni esperimenti con file dal formato non usuale per verificare che la conversione sia accettabile.

Per copiare un file senza conversione, potete usare l'opzione **-f** (force) nei comandi **doscat** e **doscp**; per esempio:

```
$ doscp -f /tmp/myfile a:
```

Questo comando esegue una copia esatta del file specificato.

19.4 Una nota sui pacchetti Merge

Per andare oltre al solo accesso ai dischi ed eseguire applicazioni MS-DOS sotto il sistema UNIX, dovete installare il pacchetto Merge disponibile come addizione in molte versioni di SVR4. Con questo pacchetto potete eseguire le Versioni 3.x del sistema MS-DOS; alcuni pacchetti richiedono una particolare versione di MS-DOS, di solito 3.3; invece non è possibile usare MS-DOS in Versione 4 e neanche il sistema OS/2. Il pacchetto Merge, tuttavia, provvede a un completo ambiente MS-DOS; consente il corretto funzionamento di quasi tutte le applicazioni MS-DOS, inclusi i programmi residenti TRS, applicazioni di comunicazione, applicazioni con intenso uso di mouse e video.

Il pacchetto Merge è stato commercializzato indipendentemente da vari produttori, che hanno adattato e personalizzato le varie caratteristiche per

macchine diverse; per questi motivi vi sono notevoli differenze tra le varie implementazioni. Ciascuna delle implementazioni di Merge si presenta all'utente in modo diverso e nella configurazione delle sessioni MS-DOS possono esservi notevoli differenze.

In questo capitolo considereremo le implementazioni che consentono la condivisione di CPU tra MS-DOS e il sistema UNIX; queste implementazioni sono basate principalmente su macchine 80x86, anche se alcuni ambienti di emulazione software realizzano ugualmente la condivisione. Le versioni supportate da una scheda di coprocessore possono risultare notevolmente differenti.

In questo capitolo, esamineremo solo le principali caratteristiche e gli aspetti negativi di un tipico sottosistema Merge. La maggior parte delle informazioni proviene dal pacchetto Locus DOS Merge per 80x86, che viene venduto assieme a molte versioni SVR4; tuttavia, molti dei concetti e delle procedure descritti si applicano a tutte le versioni di MS-DOS sotto sistemi UNIX. Per avere maggiori dettagli, consultate la documentazione relativa al vostro pacchetto Merge.

Queste "vere" capacità MS-DOS sono semplici nei concetti, ma di pesante implementazione.

Semplificando al massimo, viene predisposto un processo che fornisce a MS-DOS un ambiente di lavoro, in cui le applicazioni MS-DOS "credono" di avere il completo controllo della macchina. L'interfaccia tra questo processo e il sottostante hardware avviene attraverso uno speciale strato di software (*linkage* o *bridge*), che garantisce che i due sistemi non entrino in contesa nell'acquisizione di risorse.

19.5 Attivazione e arresto di MS-DOS

Per lanciare le applicazioni MS-DOS, dovete innanzitutto attivare il processo di controllo MS-DOS, che consente di accedere a shell MS-DOS, il normale comando **command.com**. A questo punto, potete eseguire i programmi MS-DOS senza limitazioni, come se vi trovaste in un ambiente MS-DOS indipendente. Quando avete finito, potete concludere il processo MS-DOS e ritornare al sistema UNIX.

Per lanciare il processo MS-DOS dallo shell di UNIX, usate:

```
$ dos
```

Il comando **dos** è un comando di sistema UNIX che attiva il sistema operativo MS-DOS; viene eseguito sotto il sistema UNIX, ma crea una sessione MS-DOS gestita da un normale processo nel sistema UNIX. Dopo qualche linea di intestazione e notizie sui diritti di autore (copyright), appare il prompt del comando **command.com**:

```
$ dos
  .
  .
  .
C>
```

In questo esempio abbiamo omesso le linee di intestazione.

A partire dal prompt **C>** è disponibile l'intero sistema MS-DOS; potete eseguire praticamente tutti i comandi e le applicazioni MS-DOS.

FILE DI INIZIALIZZAZIONE MS-DOS

Quando la sessione MS-DOS viene avviata, legge innanzitutto i vostri file **config.sys** e **autoexec.bat** per inizializzare la vostra sessione MS-DOS come se aveste inizializzato un sistema MS-DOS indipendente. In questo modo, potete personalizzare il file **autoexec.bat**, per adattare alle vostre esigenze l'ambiente di lavoro all'interno della sessione MS-DOS. Consultate il manuale di documentazione MS-DOS per avere maggiori dettagli sulle modalità di inizializzazione del sistema MS-DOS. Potete includere nel file **config.sys** i programmi pilota dei dispositivi MS-DOS (device driver), come **ansi.sys**, o le applicazioni **TSR** (terminate and stay resident). Queste modifiche della configurazione dell'ambiente di lavoro rimangono circoscritte alla vostra sessione MS-DOS e non influenzano nessun'altra sessione in corso nel sistema UNIX.

Di solito, i file **config.sys** e **autoexec.bat** che si trovano nel directory / (root) del file system vengono eseguiti per primi, seguiti poi dai file con identico nome presenti nel vostro home directory.

In questo modo, esistono due origini del materiale che stabilisce il profilo della sessione MS-DOS.

I device driver specificati nel file **config.sys**, come **ansi.sys**, possono risiedere in qualsiasi directory sotto il sistema UNIX, ma ne dovete specificare il pathname completo nella linea **device=** dei file **config.sys**. Nel creare i vostri file **config.sys** dovete seguire le normali regole MS-DOS per i nomi di path; per esempio:

```
$ cat config.sys
device = c:\usr\dbin\ansi.sys
device = c:\usr\lib\merge\mouse.sys
device = c:\usr\lib\merge\emm.sys d000 208
$
```

Fate molta attenzione nel modificare il file **config.sys**, perché alcuni device sono necessari per il regolare funzionamento della sessione Merge; non cancellate né modificate gli elementi di **config.sys** che trovate presenti dopo l'installazione del pacchetto Merge.

FINE DELLA SESSIONE MS-DOS

Per concludere la sessione MS-DOS e ritornare allo shell, usate il comando **quit** al prompt MS-DOS:

```
C> QUIT
$
```

A differenza della sessione di shell, non è possibile terminare la sessione MS-DOS con i tasti CTRL-D, CTRL-C o CTRL-BREAK, perché il sistema UNIX li interpreta in modo diverso dal sistema MS-DOS. In molte implementazioni di Merge potete terminare la sessione MS-DOS con la combinazione CTRL-ALT-DEL, ovvero tenendo abbassati i tasti CTRL-ALT e quindi premendo il tasto DEL. Provate la combinazione per controllare se la vostra macchina offre questa opportunità.

19.6 Esecuzione in background di MS-DOS

Per scambiare le sessioni MS-DOS e UNIX sono disponibili due procedure, supportate nelle diverse versioni di Merge. La più semplice delle due fa uso del servizio di console virtuale. Dopo avere attivato la funzione di console virtuale con il comando **vtlmgr** potete accedere a sessioni multiple con gli *hot key*, in genere la sequenza ALT-SYSREQ seguita da un tasto funzione (di solito da F1 a F8). La sequenza deve essere eseguita tenendo abbassato il tasto ALT, premendo il tasto SYSREQ e quindi il tasto funzione corrispondente alla sessione da selezionare. Alcune release di Merge richiedono la sequenza CTRL-ALT-SYSREQ sequita dal tasto funzione desiderato, mentre altre predispongono uno speciale tasto marcato "window" per rimpiazzare la sequenza multi-tasti (le differenti sessioni di OS vengono chiamate window, anche se occupano l'intero schermo). Ogni tasto funzione attiverà una nuova sessione di shell in una finestra separata, oppure potete anche eseguire il comando **newvt** per lanciare una nuova sessione in una nuova finestra. Potete scambiare le sessioni con la sequenza hot key seguita da un tasto funzione; questo causa lo scambio del video con la sessione logicamente connessa a quel tasto funzione.

Normalmente con questa procedura lancerete una sessione UNIX addizionale, quindi potrete attivare una sessione MS-DOS in quella console virtuale; in seguito, potrete scambiare tra i sistemi UNIX e MS-DOS quando necessario, attivando nuove sessioni quando volete.

La seconda procedura di scambio delle sessioni, supportata spesso nelle precedenti release di Merge, consente di lanciare una sessione MS-DOS in background, e viene affiancata dalla procedura **vtlmgr/newvt**. Se questa possibilità è supportata, potete lanciare la vostra sessione MS-DOS eseguendo il comando **dos** in backgound:

```
$ dos &
```

La sessione MS-DOS impegna lo schermo corrente, ma voi potete usare la sequenza hot key descritta sopra per ritornare nel sistema UNIX, dove potete lanciare altre sessioni MS-DOS oppure eseguire applicazioni nel sistema UNIX.

SCAMBIO DELLO SCHERMO

Quando selezionate una diversa sessione, il contenuto dello schermo diventa quello relativo a quella sessione, ma quando ritornate alla sessione precedente, viene ancora visualizzato il contenuto precedente; i contenuti dello schermo non vengono perduti scambiando le sessioni.

Potete scambiare le sessioni mentre un comando è in esecuzione in uno dei sistemi operativi; non è necessario essere al prompt dei comandi per scambiare le sessioni, perché la sequenza hot key viene interpretata a livello molto basso nella macchina, più basso del livello di shell.

Se un comando è in esecuzione mentre scambiate le sessioni, l'uscita di quel comando può essere bloccata; l'esecuzione del programma può essere interrotta fino a quando rientrate in quella sessione. Spesso il tipo di schermo usato e le necessità dell'applicazione in corso nella sessione MS-DOS, determinano se la sessione sospesa si blocca o continua l'esecuzione. Le applicazioni che utilizzano funzioni EGA o VGA si bloccano fino a che l'applicazione non ritorna in sessione corrente; al contrario, applicazioni che usano solo funzioni CGA, come lo stesso **command.com**, continuano a funzionare in background. Alcune release di Merge consentono di controllare mediante un'opzione all'avvio di MS-DOS se bloccare o no le sessioni in background. Il bloccaggio garantisce contro la perdita di dati in uscita, ma questa precauzione può non essere necessaria in tutte le situazioni.

Comunque, le applicazioni in background si arrestano in ogni caso se attendono dati in ingresso dall'utente; una sessione bloccata o in attesa di dati dalla vostra tastiera riparte se ritornate alla sessione.

19.7 Lancio direttamente da shell di programmi MS-DOS

Oltre alle modalità descritte, che consentono di attivare una sessione MS-DOS con lo scambio dello schermo, è anche possibile eseguire i comandi e le applicazioni MS-DOS direttamente da shell. A questo scopo sono utilizzabili due diverse procedure.

La prima procedura consiste nel fornire il nome dell'applicazione MS-DOS come argomento al comando **dos**:

```
$ dos ws
```

Questo comando attiva il sistema MS-DOS e immediatamente lancia il programma WordStar. Quando uscite dal programma applicativo richiesto, la sessione MS-DOS si conclude e ritornate nel sistema UNIX.

La seconda procedura consiste nell'indicare il nome del comando MS-DOS con le stesse modalità utilizzate per lanciare un normale comando in ambiente UNIX. Viene attivato l'ambiente MS-DOS in cui viene eseguita l'applicazione specificata. Quando il comando MS-DOS termina, la sessione MS-DOS si conclude e ritornate in shell come nel caso precedente. Per esempio:

```
$ ws
```

esegue il programma WordStar.

Queste procedure indirette di attivazione di MS-DOS sono più semplici ma meno efficienti dell'attivazione di una sessione **dos**, in quanto l'ambiente MS-DOS deve essere attivato ogni volta che voi eseguite un comando e non permane tra un comando e il successivo.

Quando utilizzate la seconda procedura, dovete indicare i pathname nel formato del sistema UNIX, cioè utilizzare il carattere / (barra) come separatore di elementi nei directory invece del carattere \ (barra inversa) che si usa nella notazione MS-DOS. Per esempio, potete lanciare il comando MS-DOS **dir** da shell per ottenere un elenco di file in stile MS-DOS:

```
$ dir /home/giorgio/miodosdir
```

Quando lanciate un comando MS-DOS direttamente da shell, accertatevi che i caratteri "speciali" (metacaratteri) siano delimitati da virgolette per evitare che lo shell li interpreti prima del sistema MS-DOS. Questo vale per i caratteri speciali di ambedue i sistemi, MS-DOS e UNIX, perché ciascun carattere può essere interpretato o espanso dallo shell prima di essere passato al sistema MS-DOS. Per esempio, potreste usare il seguente comando

```
$ dir *.bat
```

per listare i nomi dei file batch nel directory corrente. Questo comando è corretto quando viene eseguito direttamente da MS-DOS, ma, quando viene lanciato in ambiente UNIX, lo shell espande il metacarattere * prima di trasferire il comando al sistema MS-DOS; in questo caso, si provoca un errore perché il comando **dir** accetta un unico argomento. Dovete evitare questo tipo di errore proteggendo con virgolette gli argomenti della linea di comando che contengono caratteri speciali o metacaratteri:

```
$ dir "*.bat"
```

Fate molta attenzione a questi caratteri, perché un'errata interpretazione di metacaratteri può facilmente distruggere dei file.

PIPELINE E PROCEDURE DI SHELL

Potete creare strutture pipeline che contengono sia comandi del sistema UNIX sia comandi MS-DOS. Se il comando MS-DOS si comporta correttamente, leggerà lo standard input e scriverà nello standard output, proprio come la maggior parte dei programmi che operano in ambiente UNIX. In questa condizione, potete combinare i comandi di entrambi i sistemi operativi in un singolo pipeline:

```
$ dir a: | sort
```

Questo pipeline esegue il comando UNIX **sort** sul directory prodotto in uscita dal comando MS-DOS **dir**. I programmi dal comportamento illegale, che non utilizzano lo standard I/O MS-DOS, non devono essere utilizzati nei pipeline. Ogni programma che legge o scrive direttamente sull'hardware della macchina, senza utilizzare le convenzioni BIOS MS-DOS, ha un comportamento illegale.

Alcune implementazioni di Merge consentono di attivare sulla macchina una sola sessione MS-DOS, per cui permettono di lanciare un solo comando MS-DOS alla volta; di conseguenza, il pipeline può contenere un solo comando MS-DOS e può essere lanciato da un solo utente. In molte implementazioni SVR4 i pipeline possono comprendere più comandi MS-DOS. In ogni caso, pipeline di questo tipo risultano inefficienti in quanto l'ambiente **dos** deve essere attivato ogni volta che un comando MS-DOS viene eseguito. Potete anche includere comandi MS-DOS in procedure di shell; le regole descritte in questa sezione e nelle successive si applicano anche alle procedure di shell.

ESTENSIONI DEI COMANDI MS-DOS

Diversamente dai comandi nel sistema UNIX, i comandi MS-DOS hanno una estensione del comando. L'estensione segue il punto alla fine del nome del comando, come **exe** (executable) nel nome **ws.exe**. Altre estensioni possibili sono **.com** (command) e **.bat** (batch). Nella normale esecuzione sotto MS-DOS non è obbligatoria l'indicazione dell'estensione dei comandi; per esempio, i comandi:

```
C> WS
C> WS.EXE
```

sono equivalenti. Alcuni comandi come **del** e **dir** sono interni e inclusi direttamente in **command.com**, per cui non hanno estensione.

Quando eseguite un comando MS-DOS sotto il sistema UNIX, il processo di **dos** ne determina l'estensione nello stesso modo di **command.com**, cosic-

ché non avete necessità di preoccuparvi di aggiungere l'estensione. Potete quindi eseguire un comando dal sistema UNIX nello stesso esatto modo che usereste sotto MS-DOS.

Di solito anche il software aggiuntivo e applicazioni come WordStar o Lotus 1-2-3 possono essere eseguiti dal sistema UNIX come **ws** e **123** senza la relativa estensione MS-DOS, *a condizione che siano state installate con gli strumenti previsti nel pacchetto Merge*. Se installate delle applicazioni sotto una sessione MS-DOS usando la procedura di installazione dell'applicazione, potreste non essere in grado di eseguirle da shell, a meno di aggiungere l'estensione nella vostra linea di comando, e a condizione che l'applicazione risieda nel normale directory bin di MS-DOS. Se possibile, procurate di installare software applicativo MS-DOS mediante strumenti Merge, per evitare di incorrere in questi problemi. Gli aspetti di installazione sono trattati alla fine di questo capitolo.

19.8 PATH e altre variabili di ambiente

Entrambi i sistemi MS-DOS e UNIX utilizzano una variabile di ambiente PATH che definisce i directory in cui cercare il comando che viene richiesto. Poiché i comandi MS-DOS devono essere accessibili sotto il sistema UNIX, e viceversa, spesso viene usata la medesima PATH per ambedue le sessioni; questa PATH deve includere i directory Merge bin, di solito **/usr/dbin** e **/usr/ldbin**. Accertatevi che questi directory siano inclusi nella vostra PATH sotto il sistema UNIX; per esempio:

```
$ echo $PATH
/sbin:/usr/sbin:/usr/bin:/usr/X/bin:/usr/dbin:/usr/ldbin:
$
```

Quando avviate una sessione MS-DOS il comando **dos** di solito deriva la variabile PATH per l'ambiente MS-DOS dalla variabile PATH di UNIX, trasformandola in formato MS-DOS.

In alternativa, potete definire la variabile di ambiente PATH nel vostro file **autoexec.bat**, in modo da assegnarla all'avvio della sessione MS-DOS. Inoltre, alcune release di Merge prevedono sotto il sistema UNIX la variabile di ambiente *DOSPATH*, che diventa la vostra variabile PATH sotto una sessione MS-DOS. Fate attenzione di non rimuovere dalla vostra PATH MS-DOS i directory Merge e i normali bin di UNIX, altrimenti diverse parti di Merge non funzionerebbero

In alcune release è consentito passare altre variabili di ambiente alla sessione MS-DOS se i loro nomi sono i valori della variabile di ambiente *DOSENV*. In altre parole, il comando **dos** esamina il contenuto della variabile DOSENV e assegna le variabili specificate all'ambiente MS-DOS:

```
$ PROMPT = "salve:"      ; export PROMPT
$ TERM = AT386                    ; export TERM
$ DOSENV = PROMPT,TERM            ; export DOSENV
$
```

Le variabili PROMPT e TERM vengono aggiunte all'ambiente MS-DOS quando viene lanciato il comando **dos**. Usate le possibilità della variabile DOSENV con parsimonia, perché nella sessione MS-DOS lo spazio a disposizione delle variabili di ambiente può essere limitato.

Il sistema Merge crea anche le variabili di ambiente *TEMP* e *TMP* sotto la sessione MS-DOS a uso di applicazioni che le richiedano; queste variabili sono di solito assegnate col valore **C:**, ma potete modificarle (ed esportarle) sotto il sistema UNIX, o mediante il meccanismo DOSENV.

19.9 Esecuzione di programmi UNIX in ambiente MS-DOS

È possibile eseguire programmi UNIX dalla sessione Merge, col comando **on unix** (**rununix** in alcuni sistemi). Questa caratteristica differisce notevolmente nelle varie implementazioni di Merge; in particolare, nella gestione dei dati in uscita, nell'uso dei nomi dei comandi e nella condizione di bloccaggio o meno della sessione. Alcune versioni di Merge permettono di specificare direttamente alcuni comandi UNIX, omettendo il comando **on unix**; di solito sono supportati in questo modo i comandi più comuni, come **cat**, **ls**, **cp**, **lp**, **mv**, **pr**, **grep**, **chmod**. Comunque, per dettagli più completi, consultate la documentazione fornita con la vostra versione del programma.

La linea di comando di sistema UNIX deve essere specificata di seguito al comando **on unix**:

```
C> ON UNIX SORT /TMP/DATI.FILE > ORD.FILE
```

Per indicare i comandi e gli argomenti dovete seguire le consuete regole sintattiche del sistema UNIX. L'uscita viene inviata alla sessione MS-DOS. Potete usare CTRL-C per interrompere l'esecuzione dei comandi lanciati con **on unix**.

In alcune versioni, la procedura **on unix** può essere utilizzata solo con comandi e pipeline che non sono interattivi. Per eseguire qualunque comando che richiede dati all'utente, potete essere costretti a ritornare a una normale sessione UNIX con hot key. Altre versioni prevedono una speciale forma del comando **on unix**, chiamata **ion unix** (interactive on), che supporta applicazioni interattive.

CONTROLLO DI JOB

Con il comando **on unix** potete eseguire lavori in background, aggiungendo l'operatore **&** alla fine della linea di comando:

```
C> ON UNIX CAT DATI.FILE &
```

Dovete inserire uno spazio prima di **&**, altrimenti il comando **on unix** lo interpreta in modo errato.

Questi lavori in background vengono denominati processi *distaccati*.

Quando create questi processi distaccati, riappare il prompt MS-DOS "C>" e potete continuare con la vostra sessione o creare altre applicazioni distaccate. In alcune versioni di Merge, è disponibile una guida a menu che consente di rintracciare un'applicazione distaccata. Altre versioni prevedono il comando **jobs**, che può essere lanciato dalla sessione MS-DOS, per elencare le applicazioni distaccate che avete creato, insieme al loro stato corrente:

```
C> JOBS
JOB     HOST       STATE   EXIT STATUS     COMMAND
[ 1]    c my_sys   Done    exit(0)         cat /etc/profile
[ 2]    c my_sys   Running                  ps -ef
C>
```

Il comando **jobs** termina senza visualizzare alcun messaggio se in assenza di lavori non ha nulla da riferire.

La colonna JOB contiene gli identificatori numerici delle applicazioni, simili a quelli usati nel sistema UNIX; la colonna STATE indica lo stato corrente del lavoro, che può essere **running, done, unknown, signal, coredump** o **err3**. Gli stati **running** e **done** sono le condizioni normali delle applicazioni; **signal** significa che il comando ha terminato dopo aver ricevuto un segnale; **coredump** segnala che si è verificato un errore nel comando; mentre **unknown** ed **err3** indicano errori interni al sottosistema **on unix**.

Le applicazioni che sono terminate rimangono nella lista fino a quando non le eliminate; per cancellare tutte le applicazioni concluse utilizzate il carattere – (segno meno), come argomento di **jobs**:

```
C> JOBS -
```

Una volta che avete cancellato un'applicazione dalla lista di **jobs**, non potete più avere accesso né all'applicazione né ai suoi dati in uscita.

Esiste la possibilità di "riagganciare" un'applicazione distaccata di cui interessa esaminare i risultati in uscita, usando il comando **jobs** e specificando come argomento l'identificatore numerico dell'applicazione per richiedere che venga riagganciata. L'identificatore numerico deve essere pre-

ceduto dall'operatore % (percento), come nel sistema di controllo lavori di UNIX:

```
C> JOBS %1
```

Per salvare i dati prodotti da un'applicazione che ha terminato l'esecuzione, ridirigete su un file l'uscita del comando **jobs**:

```
C> JOBS %2 > OUTPUT
```

Potete interrompere l'esecuzione di un'applicazione distaccata prima della conclusione, lanciando dalla sessione MS-DOS il comando **kill**, indicando come argomento l'identificatore numerico dell'applicazione:

```
C> KILL %1
```

Questo comando opera come il comando **kill** in ambiente UNIX e accetta come argomento un numero di segnale:

```
C> KILL -9 %1
```

on unix ha molte altre possibilità che potrete conoscere consultando la relativa documentazione. In particolare, esiste la possibilità di rimpiazzare la parola chiave *unix* con il nome di una macchina connessa alla vostra rete LAN; in questo caso, il comando verrà eseguito sulla macchina remota, e i risultati restituiti alla vostra macchina.

19.10 Condivisione del disco con le sessioni Merge

In condizioni normali i due sistemi operativi condividono il disco rigido, per cui entrambe le sessioni possono utilizzarlo contemporaneamente e tutto il file system è visibile a entrambe le sessioni. La designazione dei drive cambia sensibilmente nelle varie versioni di Merge, tuttavia, generalmente il disco rigido di sistema viene indicato come drive C, il primo dischetto come drive A e il secondo come drive B.

In ambiente UNIX potete percorrere il file system utilizzando il comando **cd** oppure il comando equivalente MS-DOS; tuttavia, se cambiate il directory di lavoro sotto un sistema operativo e poi utilizzate il tasto di switch per passare all'altro sistema, il cambio di directory non è più valido. Ricordate, sotto il sistema UNIX dovete usare / per separare gli elementi dell'albero del directory, mentre dovete usare \ sotto MS-DOS.

Nella maggior parte delle versioni, il directory di partenza per una nuova

sessione MS-DOS coincide con il directory corrente quando iniziate la sessione **dos**. Nel file MS-DOS **autoexec.bat** potete includere un comando **cd** che sposta nell'home directory o in qualche altro directory.

Il comando **dos** si riferisce per default al disco rigido di sistema per ogni comando specificato come suo argomento. Il seguente comando

```
$ dos ws
```

equivale a

```
$ dos c:ws.exe
```

Potete eseguire anche programmi che si trovano su dischetto, indicando esplicitamente il drive in cui risiede il dischetto. Questo comando

```
$ dos a:123
```

esegue per esempio il programma Lotus 1-2-3 memorizzato sul dischetto residente nel drive A. Potete utilizzare questa procedura per eseguire applicazioni protette che non possono essere copiate e che si trovano su un dischetto chiave, come per esempio Lotus 1-2-3. Qualche volta potete anche copiare l'applicazione nel disco rigido e poi inserire il disco chiave nel drive prima di lanciare l'applicazione. Alcune applicazioni leggono infatti correttamente il disco chiave quando entrano in esecuzione.

CONVENZIONI SUI NOMI DI FILE E DIRECTORY

Le diverse regole di formazione dei nomi di file utilizzate nei due ambienti possono costituire un problema. I nomi dei file in ambiente UNIX possono avere una lunghezza qualsiasi, mentre i nomi dei file MS-DOS non possono superare 11 caratteri, con un limite di 8 caratteri per la parte nome e di 3 caratteri per la parte estensione; inoltre, alcuni caratteri ammessi in ambiente UNIX non sono permessi nei nomi dei file MS-DOS. Di conseguenza, quando utilizzate per esempio il comando MS-DOS **dir** per visionare un directory creato in ambiente UNIX, i nomi dei file possono risultare inaccettabili per le regole MS-DOS. Tutti i nomi di file che sono legali in ambiente MS-DOS sono invece accettabili anche in ambiente UNIX.

In alcune versioni di Merge, la sessione MS-DOS ignora i nomi di file che non rispettano le regole MS-DOS; nelle versioni più accurate, si applicano *regole di conversione*, per garantire che i nomi di file validi in ambiente UNIX siano unici anche in ambiente MS-DOS. In questo modo, tutti i file sono accessibili in entrambi gli ambienti MS-DOS e UNIX. Quando elencate file o directory sotto MS-DOS potrete a volte vedere nomi differenti da quelli usati sotto il sistema UNIX. Per utilizzare un file o un directory in am-

biente MS-DOS dovrete usare lo stesso nome visualizzato dal comando **dir**, anche se appare diverso dal nome noto sotto il sistema UNIX. Qualche volta i nomi che vengono convertiti contengono alcuni caratteri o numeri inconsueti. Le regole di conversione che seguono si applicano ai sistemi SVR4, ma le versioni meno recenti possono avere regole differenti.

Innanzitutto, tutti i nomi di file che contengono caratteri minuscoli vengono convertiti in caratteri maiuscoli. I nomi dei file UNIX minori di otto caratteri rimangono invariati nel sistema MS-DOS. Se il nome ha una lunghezza non maggiore di otto caratteri, è seguito dal carattere . (punto) e poi da non più di tre caratteri, rimane invariato per le regole MS-DOS, tranne per il fatto che i caratteri sono maiuscoli; al contrario, i nomi dei file UNIX che superano i limiti subiscono delle modifiche. Se il nome da convertire contiene il carattere . (punto), le regole si applicano separatamente alla parte che precede e alla parte che segue il punto. Il comando **dos** infine esamina tutti i nomi dei file contenuti nel directory alla ricerca di duplicati dei nomi; se ne trova, crea una nuova combinazione di caratteri che sia unica e soddisfi le regole MS-DOS.

Per esempio, se nel sistema UNIX è presente questo directory:

```
$ ls /tmp/SS
1234567890.dat      2234567890.dat      short.txt
1234567891.dat      longerthan8         small.large
$
```

in ambiente MS-DOS, lo stesso directory può risultare:

```
C> DIR

Volume in drive C is giorgio
Directory of C:\TMP\SS'F8U

.            <DIR>        9-25-91   9:59a
..           <DIR>        9-25-91   9:57a
1234'F84 DAT         0    9-25-91   9:45a
1234'F9T DAT         0    9-25-91   9:45a
2234'F9U DAT         0    9-25-91   9:59a
LONG'F9V             0    9-25-91   9:44a
SHORT    TXT         0    9-25-91   9:44a
SMAL'F95 LAR         0    9-25-91   9:45a
       8 File(s) 3567616 bytes free
```

Come potete vedere, i nomi che violano le regole MS-DOS possono subire delle modifiche sostanziali.

Accertatevi che i directory che create all'interno del sistema UNIX siano leciti anche in ambiente MS-DOS; altrimenti potete incontrare dei problemi a utilizzare il comando **cd** sotto MS-DOS per cambiare directory, se il nome è soggetto alle regole di conversione. Nel precedente esempio, il comando

dos cambia il nome del directory **SS** in **SS'F8U**; questo accade perché le regole di conversione cambiano i caratteri minuscoli del sistema UNIX in caratteri maiuscoli per il sistema MS-DOS, per cui un nome che nel sistema UNIX contiene caratteri maiuscoli deve essere convertito. Se create un directory o un file di nome **SS** sotto MS-DOS, viene visualizzato come **ss** in ambiente UNIX.

Se definite un nome di file che segue le regole MS-DOS, il suo nome non differisce quando utilizzate il comando **ls**; analogamente, se utilizzate o trattate con un editor un file il cui nome è stato cambiato sotto la sessione MS-DOS, il nome rimane invariato quando ritornate al sistema UNIX. Comunque, se cancellate un file sotto MS-DOS e poi lo ricreate sotto il nome utilizzato in **dos**, sotto il sistema UNIX appare il nuovo nome anziché il nome precedente.

IL COMANDO udir

Il nuovo comando MS-DOS **udir** (UNIX system **dir**) elenca sia i nomi convertiti che quelli non convertiti, per cui potete verificare le conversioni eseguite:

```
C> UDIR

Volume is drive C is giorgio
Directory of c:/tmp/SS

.                       root  drwxr-xr-x  <DIR>     7-21-91  3:02p
..          ..          sys   drwxrwxrwt  <DIR>     7-21-91  3:03p
1234567890.dat 1234'F84.DAT root  -rw-r--r--        0 7-21-91  3:01p
1234567891.dat 1234'F9T.DAT root  -rw-r--r--        0 7-21-91  3:01p
2234567891.dat 2234'F9U.DAT root  -rw-r--r--        0 7-21-91  3:01p
longerthan8 LONG'F9V    root  -rw-r--r--        0 9-25-91  9:44a
short.txt   SHORT.TXT   root  -rw-r--r--        0 9-25-91  9:44a
small.large SMAL'F95.LAR root  -rw-r--r--        0 9-25-91  9:45a
      8 File(s)          3567616 bytes free
C>
```

Il comando **udir** accetta argomenti con metacaratteri per restringere l'elenco in uscita a una parte dei file presenti; tenete presente che le regole di metacaratteri seguono le regole MS-DOS, non le convenzioni UNIX. L'opzione **-h** (hidden) consente di visualizzare i file UNIX con nomi che iniziano con punto. Notate che **udir** è un comando del sistema UNIX e pertanto le opzioni richiedono il - (meno) invece di / (barra) usato sotto MS-DOS.

ACCESSO CONCORRENTE AI FILE

I file che si trovano sul disco rigido della macchina sono di solito visibili a entrambi i programmi MS-DOS e UNIX. Nessuno dei due sistemi operativi può impedirvi di modificare un file in un sistema e cambiare sessione prima che i cambiamenti vengano effettivamente registrati; se poi leggete lo stesso file con un editor nell'altro sistema operativo, vi accorgete che è rimasto invariato. In altre parole, non esiste nessun controllo di concorrenza del file durante le modifiche che si effettuano nelle due sessioni, per cui la garanzia della coerenza dei file è sotto la vostra responsabilità. Nel caso degli editor generalmente è sufficiente assicurarsi di salvare il file su disco, prima di cambiare sessione. In casi più complessi può accadere di mettere in esecuzione in entrambi i sistemi operativi programmi applicativi che processano un file nello stesso momento; è facile dimenticare di avere installato questi strumenti, e in questi casi i file subiscono delle alterazioni non controllate.

19.11 Conversione di file

Ricordate che le convenzioni di fine-linea differiscono fra i sistemi UNIX e MS-DOS. Il pacchetto Merge fornisce due programmi di utilità che consentono di convertire i file nel formato adatto all'altro sistema operativo: **unix2dos** e **dos2unix** convertono rispettivamente un file dal formato di sistema UNIX al formato MS-DOS, e viceversa; i nomi di questi comandi e le relative opzioni possono differire. L'operazione di conversione è di solito richiesta per trasferire file tra i sistemi operativi. Entrambi i programmi sono disponibili in ambiente del sistema MS-DOS o UNIX.

Sia **unix2dos** che **dos2unix** per default leggono il proprio standard input e scrivono i dati convertiti sullo standard output:

```
$ unix2dos < unix.file > dos.txt
$
```

Se non si verificano errori questi filtri terminano senza visualizzare alcun messaggio.

Se specificate un nome di file come argomento, quel file viene preso come file di ingresso e l'uscita va sullo standard output:

```
$ dos2unix dos.txt > unix.file
```

Se specificate due nomi di file, il primo è preso come file di ingresso e il secondo è il file di uscita:

```
$ unix2dos unix.file dos.txt
```

Se utilizzate uno dei filtri su un file che ha già il formato adatto, il file non viene modificato.

La linea di comando di queste utility prevede molte opzioni; ricordate che in ambiente UNIX le opzioni iniziano con il carattere – (segno meno), mentre gli argomenti in ambiente MS-DOS iniziano con il carattere / (barra).

Questi programmi per default generano in uscita caratteri ASCII a 7 bit; l'opzione **–b** (binary) conserva i dati in caratteri a 8 bit. Potete convertire i file in caratteri tutti minuscoli o tutti maiuscoli, rispettivamente con le opzioni **–l** (lowercase) o **–u** (uppercase). Le opzioni **–b**, **–l** e **–u** sono mutualmente esclusive, cioè ne potete utilizzare solo una nella linea di comando. Con l'opzione **–?** ottenete un elenco di tutte le opzioni della linea di comando.

Entrambi i programmi rimuovono i caratteri di ritorno a capo multipli alla fine di ogni linea; perciò, quando convertite i file di UNIX, si ottengono file a spaziatura singola. Se intendete conservare la spaziatura originale del file in entrata convertendo tutti i caratteri di ritorno a capo nelle coppie "ritorno-carrello-avanzamento-interlinea", utilizzate l'opzione **–f** (force):

```
$ unix2dos –f doppio.spazio > doppio.spz
```

Abbiate cura di utilizzare questi programmi solo su file di testo ASCII a 7 o 8 bit, e non su file binari; non è facile convertire i file binari, come quelli prodotti da alcuni elaboratori di testi dei due sistemi operativi. I comandi **unix2dos** e **dos2unix** consentono comunque di specificare delle tabelle di conversione per trasformare ogni singolo carattere; utilizzando opportune tabelle è possibile convertire correttamente qualsiasi file binario. Eseguite comunque alcune prove su copie dei file trattati dagli elaboratori di testo, prima di confidare di poterli convertire con successo da un formato all'altro.

I DIRECTORY MS-DOS

I comandi e i programmi di utilità MS-DOS dispongono di un proprio directory **bin** all'interno del file system condiviso; di solito **/usr/dbin** o **/usr/vpix/dosbin**, a seconda della versione Merge che possedete. Le applicazioni locali si possono trovare in **/usr/ldbin** o **/usr/vpix/dosapps**; altri programmi di supporto possono essere contenuti in subdirectory di **/usr/vpix** o di **/usr/lib/merge**. Il directory / generalmente contiene i file primari **autoexec.bat** e **config.sys**, mentre i driver dei dispositivi installati da **config.sys** possono risiedere in **/usr/dbin** o in **/usr/lib/merge**.

19.12 Allocazione di memoria alla sessione MS-DOS

Nel momento in cui viene attivata, la sessione MS-DOS occupa una parte della memoria (RAM) della macchina. In molte implementazioni di Merge non è possibile rimuovere dalla memoria una sessione MS-DOS prima che abbia terminato, di conseguenza il sistema UNIX non può utilizzare la memoria assegnata a MS-DOS. Questo può causare dei problemi se la macchina dispone di poca memoria reale, perché il sistema UNIX diminuisce notevolmente le sue prestazioni quando è in esecuzione la sessione MS-DOS; nei casi peggiori appare evidente la lentezza di esecuzione dei comandi. In questi casi, utilizzate il tasto di switch per ritornare alla sessione MS-DOS e concluderla con **quit**; i programmi sotto il sistema UNIX riacquisteranno la consueta velocità. Per risolvere questi problemi occorre aggiungere memoria alla macchina per migliorarne le prestazioni.

La quantità di memoria reale acquisita dalla sessione MS-DOS può essere specificata come argomento del comando **dos** in base alle reali necessità, ma in ogni caso non può superare il valore limite MS-DOS di 640 kB. Utilizzate l'opzione **+m** (memory) sulla linea di comando **dos**:

```
$ dos +m400
```

Questo comando alloca 400 kB alla sessione MS-DOS. Potete chiedere l'assegnazione di una quantità di memoria compresa tra 64 kB e 640 kB; se la quantità specificata non è disponibile, il comando **dos** acquisisce la massima dimensione possibile. Potete ottenere più memoria per la sessione MS-DOS riducendo il numero e la dimensione dei processi in esecuzione nel sistema UNIX quando attivate la sessione MS-DOS. Se avete ampia disponibilità di memoria e chiedete 640 kB, il programma **dos** probabilmente fornisce circa 600 kB, perché il collegamento tra i sistemi MS-DOS e UNIX richiede un po' di memoria.

La maggior parte delle applicazioni MS-DOS può determinare se la quantità di memoria a disposizione è inadeguata e, nel caso, visualizza un messaggio di errore; utilizzate questa informazione per richiedere il corretto valore con l'opzione **+m**. Abbiate cura di utilizzare il minimo indispensabile, per evitare spreco di memoria inutilizzata.

L'allocazione di default della memoria, applicata in assenza dell'opzione **+m**, è un parametro di configurazione che può essere definito in base alle vostre esigenze abituali. La configurazione della sessione MS-DOS è descritta nel seguito del capitolo.

Il comando **dos** consente di usare un'espansione di memoria, se un'applicazione la richiede. Potete allocare memoria per simulare un'espansione di memoria includendo l'opzione **+aems** nella linea comando **dos**; per esempio:

```
$ dos +aems
```

Questa linea di comando alloca 1 MB di espansione di memoria. Potete allocare ulteriore memoria di espansione in blocchi di 1 MB accodando una cifra all'opzione **+aems**; per esempio:

```
$ dos +aems4
```

Questa linea di comando alloca 4 MB di memoria d'espansione.

La memoria d'espansione deve essere supportata da una linea nel file **config.sys**; per esempio:

```
device=c:\usr\lib\merge\emm.sys D000 208
```

Se volete usare l'espansione di memoria non dovete cancellare questa linea.

Molte versioni di Merge non supportano i dischi RAM MS-DOS, come **ramdrive.sys**; in tal caso non dovete includerli nel vostro file **config.sys**.

19.13 Altre opzioni per il comando dos

Oltre all'opzione **+m**, il comando **dos** supporta molte altre opzioni per configurare la sessione MS-DOS e definire l'ambiente di lavoro. Le opzioni variano ampiamente secondo l'implementazione di Merge disponibile sulla macchina, per cui dovrete consultate la documentazione di man page **dos** relativa alla vostra versione. Generalmente, queste opzioni seguono il formato **—** *x* (segno meno) per disattivare l'opzione *x*, mentre il formato **+***x* (segno più) permette di attivarla. Potete di solito utilizzare **+h** (help) per visualizzare l'elenco delle opzioni disponibili per la vostra versione di comando **dos**.

Potete acquisire le unità hardware (vedi paragrafo seguente) con l'opzione **+a** (assign) e rilasciarle con l'opzione **−a**. Potete inviare gli argomenti di **dos** direttamente a **command.com**, senza interpretazione da parte del comando **dos** , utilizzando l'opzione **+c** (command). L'opzione **+u** (UNIX system) cambia il carattere separatore di pathname e il carattere indicatore di argomenti negli equivalenti del sistema UNIX. Nelle linee di comando il carattere MS-DOS / diventa quindi il carattere UNIX **−** , mentre il carattere \ diventa il carattere /. Questa opzione non funziona con tutti i programmi MS-DOS, per cui la dovrete utilizzare con attenzione. Sono disponibili molte altre opzioni; l'opzione **+h** dovrebbe fornire l'elenco completo per il vostro sistema.

19.14 Condivisione di periferiche tra MS-DOS e UNIX

Quando uno dei due sistemi operativi ha il pieno controllo sulla macchina, vi consente di accedere a qualunque unità hardware collegata. Poiché le

modalità di gestione delle periferiche differiscono tra i due sistemi, può sorgere qualche problema nell'utilizzo delle periferiche quando entrambi i sistemi sono contemporaneamente attivi. Questi problemi non si presentano con il disco rigido, ma i dischetti, le stampanti e le porte di comunicazione qualche volta possono avere un comportamento anomalo.

GESTIONE DEI DISCHETTI

Entrambi i sistemi UNIX e MS-DOS consentono di utilizzare i drive, ma non possono condividere gli stessi dischetti. Infatti, non solo i dischetti stessi differiscono nel formato, ma anche i driver di device presentano delle differenze a basso livello nella lettura e scrittura dei dischetti, per cui non è possibile l'utilizzazione dei drive da parte dei due sistemi.

Quando attivate il sistema MS-DOS, i drive non vengono immediatamente acquisiti, fino a quando non occorre utilizzarli sotto la sessione MS-DOS. In altre parole, potete ritornare dalla sessione MS-DOS al sistema UNIX e utilizzare i comandi previsti per accedere ai dischetti in ambiente UNIX. Se invece utilizzate un drive in ambiente MS-DOS, la sessione MS-DOS lo acquisisce e lo rende inaccessibile all'altro sistema; analogamente, se un drive è usato sotto il sistema UNIX, il comando MS-DOS che tenta di accedere al drive non viene eseguito; al contrario, se il sistema UNIX non utilizza il drive, la sessione MS-DOS può impiegarlo come previsto.

In alcune versioni di Merge, il drive acquisito viene rilasciato dopo un breve tempo di mancato utilizzo (in genere cinque secondi); in altre implementazioni il drive rimane di proprietà di MS-DOS, una volta che lo ha utilizzato, e il sistema UNIX può accedervi solo quando la sessione MS-DOS è stata conclusa. Per rilasciare il drive e renderlo disponibile al sistema UNIX una volta che è stato impiegato sotto una sessione MS-DOS, può essere necessario uscire dalla sessione MS-DOS; quando rientrate ancora nella sessione MS-DOS, i drive ritornano accessibili al sistema UNIX.

STAMPA IN AMBIENTE MS-DOS

Una situazione analoga a quella precedente si verifica per le stampanti collegate alla macchina, perché entrambi i sistemi possono cercare di stampare contemporaneamente i risultati delle loro elaborazioni. Molte implementazioni di Merge prevedono due possibili soluzioni al problema.

Se è attivo il sottosistema di stampa **lp**, il comando MS-DOS **print** invia l'uscita alla coda di stampa **lp** invece di stampare direttamente dallo spooler di stampa MS-DOS; in questo modo, si evita ogni conflitto, perché il sistema **lp** gestisce lo spool intelligentemente. Analogamente, anche la funzione MS-DOS **print screen** invia l'immagine dello schermo nella coda di stampa **lp**. In entrambi i casi, l'uscita viene indirizzata alla stampante designata

doslp in fase di configurazione del sistema **lp**; per questa riassegnazione del nome **doslp** alla stampante che scegliete per le stampe di MS-DOS userete i normali strumenti **lp**. Alcune versioni di Merge prevedono anche un comando chiamato **lpinstall** per cambiare la configurazione della stampante.

In alternativa, potete richiedere di utilizzare i normali driver di stampa MS-DOS al posto della coda **lp**. Uno speciale comando MS-DOS consente di acquisire la stampante dallo spooler **lp** e riservarla alle utility di stampa MS-DOS. In ambiente MS-DOS, utilizzate questo comando

```
C> PRINTER DOS
```

per acquisire la stampante per la sessione MS-DOS. Potete restituire la stampante al sistema UNIX con il seguente comando:

```
C> PRINTER UNIX
```

Dopo l'assegnazione della stampante alla sessione MS-DOS, può essere necessario utilizzare il comando **mode** per inizializzare correttamente la configurazione della stampante, specialmente se impiegate una stampante seriale.

Quando il sistema MS-DOS acquisisce la stampante, lo spool **lp** viene disabilitato, per cui la coda può crescere mentre l'uscita non viene stampata.

USO DELLE PORTE DI COMUNICAZIONE

La maggior parte delle implementazioni di Merge richiede che quando attivate una sessione Merge vengano acquisite le porte seriali COM1 e COM2. Non è possibile usare le porte seriali "al volo" come i dischetti; dovete riservare questi dispositivi quando attivate la sessione MS-DOS. Quando acquisite una porta seriale in questo modo, la porta non sarà disponibile al sistema UNIX fino a che la sessione Merge non viene conclusa. Prima di usare una porta COM sotto una sessione Merge dovete interrompere ogni monitor di comunicazione o processo **getty** attivo su quella porta.

Per acquisire il dispositivo COM1 usate il comando:

```
$ dos +acom1
```

Questo comando assegna il device COM1 (tty0) alla sessione Merge. Analogamente, usate **+acom2** per acquisire il device COM2; potete anche usare più di un'opzione **+a**, se necessario.

Se volete usare le porte seriali in una sessione Merge già attiva dovete concludere la sessione **dos** e riattivarla con l'opzione **+a**.

19.15 Approfondimenti

Come potete immaginare, esistono molti altri problemi connessi alla condivisione di una macchina tra due diversi sistemi operativi.

COMANDI MS-DOS INUTILIZZABILI

È impossibile utilizzare i comandi MS-DOS che possono impedire la coesistenza dei due sistemi operativi. Non lanciate i comandi MS-DOS **fdisk, ship** o qualsiasi altro comando che parcheggia le testine del disco rigido. Inoltre, non usate i comandi MS-DOS **chkdsk, format** o **sys** sulla parte di disco rigido condiviso, ma utilizzateli solo sui dischetti.

ESECUZIONE DI MERGE DA TERMINALE

Molte versioni di Merge consentono di eseguire la sessione MS-DOS da un terminale, al pari della console. Potete lanciare la sessione MS-DOS come al solito e lo strato "bridge" di Merge invierà l'uscita al terminale, se avete predisposto correttamente la variabile TERM. Tuttavia, solo le applicazioni MS-DOS regolari funzionano su un normale terminale ASCII; applicazioni non regolari, che scrivono direttamente sull'hardware del display non funzionano su un terminale.

I tasti speciali disponibili sulle tastiere stile AT non sono di solito presenti sui terminali; possono mancare i tasti funzione, PRTSC e anche i tasti di controllo del cursore. Per ovviare a queste carenze, il sistema Merge prevede speciali sequenze di ESCAPE per simulare sulla tastiera solo ASCII i tasti mancanti. Per esempio, il tasto F1 è simulato premendo la sequenza di tasti ESC seguita dalla cifra 1; il tasto PRTSC è simulato premendo ESC seguito dalla lettera *r*. La sequenza ESC-CTRL-R causa la rivisualizzazione dello schermo; la sequenza ESC-CTRL-F crea una nuova istanza di shell UNIX. La sequenza ESC-CTRL-K termina (kill) la sessione MS-DOS e vi fa ritornare nel sistema UNIX; deve essere usata quando non è possibile usare il comando **quit** perché un programma MS-DOS si è bloccato.

Se il vostro terminale supporta solo 24 linee non potete vedere l'intero display a 25 linee di MS-DOS; la sequenza ESC-CTRL-U causa da parte di **dos** la rivisualizzazione dello schermo per mostrare le *ultime* 24 linee, mentre la prima linea non viene visualizzata; usando di nuovo ESC-CTRL-U vengono di nuovo rivisualizzate le *prime* 24 linee.

Nella vostra documentazione di Merge troverete l'elenco di tutte le sequenze di escape che vi sono necessarie se volete usare Merge da terminale.

ESECUZIONE DI MERGE SOTTO X WINDOW SYSTEM

Potete eseguire il sistema Merge sotto X Window System, se è installato sulla vostra macchina. Quando lanciate una sessione MS-DOS il comando **dos**

determina che state usando X, e crea un normale display MS-DOS all'interno di una nuova finestra X; quando concludete la sessione MS-DOS con **quit**, la finestra X scompare. Molte release di Merge gestiscono finestre MS-DOS multiple nella vostra sessione di X.

Per usare **dos** sotto l'ambiente X Window System dovete assegnare ed esportare la variabile di ambiente XMERGE, col valore del tipo di display che state usando; per esempio:

```
$ XMERGE = vga
$ export XMERGE
```

I tipi di display accettabili sono: *mono, cga, ega, herc, vga*. Il valore della variabile deve essere definito nel file di predisposizione di X (di solito **.olintrc** oppure **.xinitrc**), non nel normale **.profile**, perché alcune release di Merge non funzionano correttamente se la variabile esiste al di fuori dell'ambiente di X.

La finestra MS-DOS di default è una normale finestra di console 25-per-80; le applicazioni MS-DOS regolari funzioneranno in questa finestra come previsto. La sessione X supporta inoltre applicazioni grafiche che di solito usano il mouse e un video grafico; per usare queste caratteristiche può essere necessario assegnare il mouse alla sessione MS-DOS, ovvero il processo detto *focus* del mouse su una finestra MS-DOS. Se l'applicazione richiede la piena risoluzione del display, potete *zoom*are la finestra MS-DOS per estenderla a tutta l'ampiezza del video; il sistema Merge vi avvertirà se state eseguendo un'applicazione che necessita dello zoom.

Queste due funzioni sono trattate con un menu temporaneo disponibile sotto il sistema X; richiamate il menu premendo ALT-D dalla finestra MS-DOS; selezionate l'opzione Focus per assegnare il mouse alla sessione MS-DOS. Quest'azione rende il mouse inutilizzabile in altre finestre di X fino a che non richiamate di nuovo il menu temporaneo e selezionate l'opzione Unfocus. In pratica, questa è un'operazione di acquisizione di una risorsa, alla pari dell'acquisizione della stampante o del dispositivo a dischetti.

Selezionate Zoom dal menu per assegnare lo schermo interamente alla sessione MS-DOS, dopodiché potete eseguire applicazioni grafiche ad alta risoluzione; quando avete terminato l'applicazione, premete ancora ALT-D per ridurre lo zoom e ripristinare il vostro display di X. Ovviamente, potete selezionare le opzioni Focus o Zoom su una sola finestra alla volta; tuttavia, potete eseguire applicazioni multiple in console virtuali al di fuori di X Window System.

I colori dello schermo nelle finestre MS-DOS possono differire da quelli usati nelle normali applicazioni MS-DOS, perché le *colormap* di X e di MS-DOS sono in conflitto. Potete risolvere il problema includendo una nuova risorsa nel file **.Xdefaults**; a questo scopo, aggiungete la linea:

```
dos*installcolormap: true
```

Quando spostate il puntatore del mouse nella finestra MS-DOS i colori dello schermo verranno cambiati in quelli standard di MS-DOS (probabilmente anche il resto dello schermo cambierà colori); i colori sono corretti anche per finestre di console e finestre con zoom. Diverse altre normali risorse di X, come i colori dei bordi, i font, ecc., possono essere predisposte per la finestra **dos**.

MICROSOFT WINDOW SOTTO MERGE

Molte release di Merge consentono di eseguire gestori di sessioni MS-DOS, quali Microsoft Windows, e le applicazioni disponibili per Windows. Dovete tuttavia usare solo versioni di Windows designate per *real mode*; non potete usare le versioni *protected mode* quali Windows/380 o la versione estesa di Windows 3. Questa limitazione è imposta dal fatto che il sistema UNIX utilizza il modo potetto dei processori 80386 e 80486, in conflitto con l'utilizzazione similare da parte dei prodotti Windows. In una sessione Merge dovete usare versioni di Windows in real mode; anche altre versioni di gestori di sessioni devono rispettare questa limitazione.

COMUNICAZIONI E INTERRUZIONI

Poiché la sessione MS-DOS viene eseguita come un processo sotto il sistema UNIX, non ha realmente pieno accesso alle risorse hardware della macchina. Infatti, il sistema UNIX intercetta e prende in gestione gli eventi hardware e le interruzioni anche quando è attiva una sessione MS-DOS. Il sistema UNIX passa le interruzioni alla sessione MS-DOS, ma le applicazioni MS-DOS che cancellano le interruzioni in realtà agiscono solo su loro interruzioni virtuali. Questa situazione comporta diverse implicazioni. Prima fra tutte, programmi MS-DOS fuori controllo non possono distruggere l'ambiente UNIX; tuttavia alcune release Merge terminano la sessione MS-DOS quando le interruzioni sono disabilitate per troppo tempo. Applicazioni MS-DOS malfunzionanti sono quindi abbastanza ben protette da non poter danneggiare il sistema UNIX in esecuzione. Secondariamente, le periferiche di comunicazione che generano frequenti interruzioni non possono raggiungere sotto la sessione Merge le stesse prestazioni di velocità che avrebbero in una sessione MS-DOS indipendente. Di solito 4800 o 9600 baud è la velocità massima di una periferica seriale di comunicazione in esecuzione sotto una sessione MS-DOS.

MS-DOS INDIPENDENTE SU UNA MACCHINA UNIX

Se avete configurato la vostra macchina con una partizione separata riservata al sistema MS-DOS, come descriveremo nel Capitolo 25, potete usare

MS-DOS come un sistema indipendente, senza la funzione Merge e senza condividere le risorse con UNIX. Quando MS-DOS è in esecuzione come sistema indipendente, non potete accedere alla partizione di disco rigido utilizzata dal sistema UNIX, ma dovete usare solo la partizione di disco riservata a MS-DOS durante la formattazione del disco rigido. In questo modo, quando occorre potete sfruttare completamente la velocità e la potenza del sistema MS-DOS, senza che UNIX condivida la CPU.

Sono possibili due metodi per attivare un ambiente MS-DOS indipendente. Con il primo, potete caricare la macchina da un dischetto MS-DOS di sistema. In questo caso, la partizione del disco rigido riservata solo a MS-DOS è nota come **drive C**, che non coincide con l'altra partizione indicata come drive C, nota a MS-DOS quando è in esecuzione sotto la sessione Merge. Quest'ultimo è infatti il normale file system condiviso dai sistemi MS-DOS e UNIX, mentre il primo si riferisce alla partizione MS-DOS indipendente.

Prima di utilizzare il drive C indipendente, lo dovete formattare come uno dei normali dischi MS-DOS. Questa operazione di formattazione viene eseguita solo la prima volta che utilizzate inizialmente la partizione. Dopo l'inizializzazione del sistema da un dischetto MS-DOS, usate il comando:

```
A> FORMAT C: /S
```

per predisporre l'uso della partizione. Questo comando di formattazione può visualizzare un messaggio di avviso, come nel caso seguente:

```
Warning: this will destroy all the data on the hard disk.
```

In realtà, il comando cancella solo i file nella partizione riservata a MS-DOS, non tutti i dati della partizione condivisa.

Dopo la formattazione, potete caricare applicazioni o il sistema operativo MS-DOS nel drive C con le abituali modalità; i file non entrano in conflitto con il sistema UNIX sul file system condiviso. In questo modo, quando reinizializzate la macchina dal disco rigido, attivate il sistema UNIX come al solito.

Il secondo modo per accedere alla partizione riservata solo a MS-DOS consiste nel lanciare in ambiente UNIX il comando **fdisk** per cambiare la partizione attiva nella partizione destinata solo a MS-DOS. A questo scopo, eseguite **fdisk** dal sistema UNIX, non dalla sessione MS-DOS. Non dovete mai usare **fdisk** dalla sessione MS-DOS all'interno del sistema Merge. Quando reinizializzate la macchina, è attiva la partizione MS-DOS e la macchina si inizializza in ambiente MS-DOS; tutte le volte che reinizializzate la macchina dopo quest'operazione, è attiva la partizione MS-DOS. Quando desiderate ritornare all'uso del sistema UNIX, lanciate il comando **fdisk** dal sistema MS-DOS indipendente per attivare la partizione condivisa, dopodiché alla reinizializzazione partirà il sistema UNIX.

IL DRIVE E

Quando usate la sessione condivisa Merge, potete accedere alla partizione riservata solo a MS-DOS utilizzando un indicatore logico di drive. Alcune versioni utilizzano il *drive E* come notazione standard, mentre in altre versioni potete scegliere la notazione preferita. Il drive E è accessibile solo sotto una sessione MS-DOS all'interno del sistema Merge; per accedere alla partizione MS-DOS indipendente con i comandi **dosdir** e **doscp** dovete usare il drive D. Non è possibile accedere alla partizione condivisa dal sistema MS-DOS indipendente, e neppure accedere da UNIX alla partizione riservata solo a MS-DOS. Diverse sessioni MS-DOS possono leggere contemporaneamente dal drive E; al contrario, la scrittura sul drive E causa l'acquisizione del dispositivo, di modo che può scrivere una sola sessione alla volta; il dispositivo ritorna libero quando il primo processo termina.

IL DRIVE D

Quando attivate la sessione MS-DOS sotto Merge è disponibile un altro indicatore logico di drive, il *drive D*. Il drive D differisce dal drive C in un solo aspetto: la radice del drive D è il vostro home directory e non l'effettiva radice del file system condiviso. Non è possibile vedere nel drive D al di sopra dell'home directory. Per passare al drive D battete da tastiera:

```
C> D:
D>
```

Il drive D si dimostra utile per le applicazioni che intendono creare i file nel directory / (root); una volta installate nel drive D, queste applicazioni memorizzano i loro file nel vostro home directory e non nel vero directory /, evitando conflitti quando più utenti adoperano queste applicazioni.

IL DRIVE J

Oltre ai drive D e E, il comando **dos** può fornire un drive J, che risulta disponibile quando lanciate un comando MS-DOS direttamente da shell:

```
$ ws document
```

Il drive J utilizza come directory di lavoro quello in cui si trovava il programma eseguibile. Si tratta di un directory che può differire dal directory di lavoro disponibile in ambiente UNIX, che è il drive C. Alcune versioni di applicazioni MS-DOS, specialmente i programmi meno recenti, richiedono che i file di dati siano nello stesso directory del programma eseguibile. Il drive J referenzia il programma eseguibile, mentre il drive C referenzia il

file di dati, in modo che l'esecuzione dei programmi proceda correttamente, altrimenti il drive J non si dimostra di grande utilità e generalmente referenzia il directory bin locale MS-DOS, **/usr/ldbin**.

FILE DI DISCHI VIRTUALI

Potete creare nel file system condiviso dei file che vengono trattati come dischetti o come una partizione MS-DOS su disco rigido. A questi file si può quindi accedere con lettere di drive MS-DOS. Vengono detti *dischi virtuali*, poiché sono in realtà singoli file nel file system condiviso, ma sotto MS-DOS ognuno di essi può contenere numerosi file ed essere visto come un disco. I dischi virtuali non sono abitualmente molto usati, tuttavia danno la possibilità di copiare il contenuto di un dischetto in un disco virtuale e quindi di accedervi come se fosse un dischetto. Questo può accelerare il lancio di applicazioni che devono risiedere su dischetto, come alcune applicazioni protette contro la copiatura. Alcune applicazioni protette potrebbero non funzionare con questa procedura; per sicurezza dovrete effettuare delle prove.

Potete copiare il contenuto di un dischetto MS-DOS (anche inizializzante) nel file system condiviso con il comando **dd** (sotto il sistema UNIX), come nell'esempio:

```
$ dd if=/dev/rdsk/f13dt of=/home/giorgio/dosappl
```

Questo comando consente di copiare un dischetto da $3^1/_2$ pollici, 720 kB, inserito nel drive B. Quando il comando è completato, il contenuto del dischetto risiede in un file nel disco rigido; a questo punto potete lanciare la sessione MS-DOS, assegnando il file disco virtuale come disco B, con:

```
$ dos +ab:=/home/giorgio/dosappl
```

Questa linea di comando assegna il file al drive B sotto la sessione MS-DOS, in modo che potete accedere al contenuto del drive B con i normali strumenti MS-DOS, per esempio:

```
C> DIR B:
```

Potete leggere, scrivere o creare nuovi file nel dischetto virtuale secondo le vostre necessità. Quando avete assegnato il drive A o B a un disco virtuale in questo modo, non potete accedere al dischetto reale se non avviando una nuova sessione **dos**. Notate che i dischetti virtuali devono usare le lettere A o B, come nelle normali sessioni MS-DOS, le altre lettere sono riservate alle partizioni su disco rigido.

Una procedura similare consente di creare partizioni su disco rigido

virtuali, potete creare o copiare un file e assegnarlo a una lettera di drive virtuale, come nell'esempio:

```
$ dos +ak:=/home/giorgio/virthd
```

Quando la sessione MS-DOS viene avviata, dovete formattare la partizione come se fosse una qualunque partizione di disco rigido sotto MS-DOS; quindi, potete utilizzare il drive assegnato (in questo caso il drive K) come una partizione MS-DOS. Questo può essere utile quando diversi utenti desiderano avere partizioni private nella sessione MS-DOS, in quanto i singoli file all'interno di queste partizioni virtuali non sono visibili dal sistema UNIX. Spesso una procedura di gestione orientata a menu inclusa nel pacchetto Merge fornisce strumenti per la creazione di file di disco virtuale.

Potete lanciare le applicazioni che risiedono nel disco virtuale indicandone il nome alla fine della linea di comando **dos**:

```
$ dos +ab:=/home/giorgio/dosappl b:appl
```

Questo comando lancia l'applicazione **appl** esistente sul disco virtuale **dosappl**.

AVVIARE MS-DOS DA DISCHETTO

Quando lanciate il comando **dos**, la sessione MS-DOS viene lanciata dall'*immagine MS-DOS* esistente sul disco rigido di sistema. Questo modo di procedere si dimostra il più rapido e il più utilizzato. Qualche volta occorre invece caricare il sistema MS-DOS da dischetto, sia perché alcune applicazioni non possono essere caricate sul disco rigido, sia perché altre applicazioni sono configurate su dischetti indipendenti che non contengono un vero e proprio sistema MS-DOS. Potete utilizzare queste applicazioni in un ambiente Merge condiviso caricando la sessione MS-DOS direttamente da dischetto.

Inserite il dischetto inizializzante nel drive A ed eseguite il comando **dosboot** da UNIX:

```
$ dosboot
```

Il sistema operativo viene lanciato dal dischetto. Quando inizializzate da dischetto con questa procedura, il drive C su disco rigido non è disponibile; potete utilizzare solo il sistema operativo indipendente su dischetto. Con questa caratteristica, potete inizializzare anche con altri sistemi operativi, come CP/M-86 o anche OS/2.

Quando volete concludere la sessione **dosboot**, premete CTRL-ALT-DEL per

chiudere la sessione e ritornare al prompt di sistema UNIX; poiché il drive C non è disponibile, non è disponibile il normale comando **quit**.

MS-DOS COME SHELL DI login

Potete predisporre gli identificatori di login di utente in modo da attivare una sessione MS-DOS invece di un normale shell quando si collegano alla macchina. Inserite il comando **dos** al termine del normale **.profile** di utente, come per esempio:

```
$ tail -2 /home/dosuser/.profile
dos
exit
$
```

Con questa predisposizione, gli utenti vengono introdotti in ambiente MS-DOS quando si collegano al sistema. Il comando **exit** dopo **dos** è necessario per garantire che gli utenti siano scollegati dal sistema quando concludono la loro sessione MS-DOS con **quit**.

IL COMANDO dosopt

Quando lanciate un'applicazione MS-DOS direttamente dal sistema UNIX, il programma **dos** provvede come prima azione a creare l'ambiente MS-DOS. Poiché in questo caso **dos** non viene lanciato direttamente, non è possibile specificare nessuna opzione tramite linea di comando, attivando la sessione MS-DOS. Il comando **dosopt** (MS-DOS option) consente di associare permanentemente a un programma applicativo alcune opzioni **dos**, che si attivano ogni volta che il comando viene lanciato direttamente dal sistema UNIX. Alcuni sistemi includono parte delle funzioni **dosopt** in uno strumento di gestione orientato a menu, chiamato **dosadmin**. Utilizzate **dosadmin**, se è presente, altrimenti scrivete

```
$ dosopt [opzioni] comando
```

per assegnare al comando le opzioni specificate; sono disponibili le stesse opzioni del comando **dos**. Per esempio, per inizializzare la dimensione della memoria al valore maggiore consentito, quando eseguite l'applicazione Lotus 1-2-3, utilizzate il seguente comando:

```
$ dosopt +m640 123
```

Quindi, quando eseguite

```
$ 123
```

viene riservata la massima quantità di memoria disponibile. Questo comando può variare tra le versioni di Merge, per cui prima di usarlo consultate la documentazione del vostro sistema.

Le opzioni memorizzate possono essere rintracciate nei file **/etc/dosenv.def**, **/etc/dosapp.def**, **$HOME/dosenv.def** e **/$HOME/dosapp.def**.

INSTALLAZIONE DI Merge

L'installazione di Merge è un'area che presenta grande variabilità. Nella maggior parte dei casi, il sistema UNIX fornitovi disporrà di una speciale interfaccia che guida l'utente nel processo d'installazione. Consultate la documentazione allegata al vostro sistema Merge per conoscere le esatte procedure da seguire.

Di solito, l'installazione di Merge si effettua con le normali procedure di installazione del software del sistema UNIX, dopodiché si esegue uno speciale comando **dosinstall** per caricare il sistema operativo MS-DOS. Per poter utilizzare Merge occorre prima eseguire il comando **dosinstall**; dovrete avere a disposizione i necessari dischetti del sistema operativo MS-DOS da caricare quando vi verrà richiesto da **dosinstall**. Parte di questa procedura provvede alla creazione dell'immagine MS-DOS per la vostra configurazione hardware.

Dopo il caricamento dei programmi di utilità Merge e del software di sistema MS-DOS, potete caricare i programmi applicativi MS-DOS nel disco rigido. È opportuno installare le applicazioni usando le caratteristiche di Merge, per evitare problemi con le estensioni dei comandi MS-DOS e per avere la possibilità di eseguire le applicazioni direttamente da shell UNIX. Non ricorrete alle procedure d'installazione fornite con il software applicativo, a meno che non carichiate l'applicazione nella partizione MS-DOS indipendente. Piuttosto, utilizzate invece l'interfaccia di gestione a menu **dosadmin** fornita con il sistema Merge, che prevede strumenti per l'installazione di applicazioni MS-DOS. Con questi strumenti le applicazioni vengono caricate nel normale directory bin di **dos**, dove possono accedere sia Merge che il sistema UNIX. Successivamente, potete usare **dosopt** per associare opzioni speciali a un'applicazione.

L'IMMAGINE MS-DOS

L'immagine MS-DOS è un file contenente l'esatta configurazione della sessione MS-DOS che viene attivata quando lanciate il comando **dos**. Si tratta di un'immagine, memorizzata su disco, della configurazione **dos**. Esiste un'immagine per ogni tipo di display esistente nella macchina; le immagini di solito risiedono in **/usr/lib/merge/*.img**. La presenza di questi file consen-

te una rapida partenza del sistema MS-DOS, in quanto ogni immagine è già configurata e rapidamente caricabile nella memoria della macchina quando viene eseguito il comando **dos**.

L'immagine MS-DOS viene creata quando caricate Merge, tuttavia l'immagine cambia ogni volta che aggiornate la configurazione hardware della macchina aggiungendo o rimuovendo dispositivi vari, come dischi, schede display, ROM di sistema, ecc. Potete rimuovere Merge prima di cambiare la configurazione hardware e ricaricarla dopo che avete eseguito le estensioni del sistema.

Come alternativa, potete utilizzare il comando **dosadmin** per creare una nuova immagine MS-DOS. Quando siete sicuri di avere completato le modifiche hardware, eseguite

```
$ dosadmin
```

dal sistema UNIX. La voce di menu **Create MS-DOS Image** consentirà di creare la nuova immagine; se trascurate di effettuare questa operazione, la sessione MS-DOS non funzionerà correttamente dopo le modifiche di hardware alla macchina.

La maggior parte dei dati specifici del sistema usati dal processo di creazione dell'immagine è memorizzata nel file **/etc/dosdev**; potete esaminare questo file per rendervi conto di come il sistema Merge utilizza le risorse hardware della macchina.

Capitolo 20

Temporizzazione e scheduling

La selezione scadenzata (scheduling) del processo prioritario cui cedere il controllo dell'unità centrale è uno degli aspetti di maggiore importanza in ogni sistema operativo multiprocesso. Per garantire a tutti i processi un accesso imparziale alla singola CPU della macchina, il sistema operativo deve prevedere meccanismi di temporizzazione (timing) per lanciare un nuovo processo quando il processo corrente ha esaurito la risorsa di tempo consentito di CPU. In effetti, il sistema UNIX dispone di eccellenti strumenti di temporizzazione e scheduling, con una scala di tempi compresa tra i millisecondi e gli anni. I meccanismi base di temporizzazione sono realizzati nel sistema operativo stesso, e i comandi e gli strumenti basati su questi meccanismi eseguono certe funzioni amministrative con regolarità, senza richiedere grande attenzione da parte degli utenti o del gestore di sistema.

La temporizzazione è talmente importante in ambiente di sistema UNIX che quasi tutte le macchine UNIX dispongono di un hardware orologio/calendario alimentato con batteria tampone che mantiene l'ora esatta anche a macchina spenta. In ogni caso, le funzioni di gestione, per operare correttamente richiedono che la macchina sia in funzione almeno per l'1% del tempo. Il sistema può attivare funzioni scadenzate per essere eseguite una sola volta al giorno, generalmente nelle ore notturne, altre una volta alla settimana, altre ancora una volta o due all'anno. In questo capitolo, esamineremo la temporizzazione a livello utente e accenneremo ad alcune operazioni di gestione eseguite su scadenzazione dal sistema UNIX.

20.1 Confronto di prestazioni timesharing/real time

Quando molti utenti o processi condividono la macchina, la responsabilità dell'allocazione delle risorse di CPU è demandata al sistema operativo. In circostanze normali gli utenti non possono controllare l'ammontare di tempo della CPU loro assegnato in un certo intervallo; questo comporta che possono sorgere problemi con processi real time. Per esempio, una periferica di I/O ad alta velocità può rimanere troppo tempo in attesa di dati, o al contrario ricevere più dati di quanti ne possa trattare, se il sistema risulta impegnato pesantemente. In passato, questo si è rivelato un serio problema che impediva l'utilizzazione del sistema UNIX in molte applicazioni real time. In SVR4 è stato introdotto un nuovo sistema di scheduling che consente all'utente (o al gestore) di designare alcuni o tutti i processi di tipo real time. Questo significa che alcune parti del sistema acquisiscono una maggiore quantità di risorse di sistema. Rivedete il Capitolo 11 per altre informazioni sul controllo della priorità e relative caratteristiche.

20.2 Impiego di UNIX a tempo pieno

Nei primi tempi, il sistema UNIX veniva utilizzato su sistemi timesharing, con una disponibilità di circa l'1% del tempo macchina. In queste macchine la CPU risultava pienamente disponibile solo a notte inoltrata, quando pochi erano gli utenti collegati al sistema. Di conseguenza, molti compiti di gestione a bassa priorità venivano lanciati durante le ore notturne; nel corso degli anni gli strumenti si sono evoluti per consentire una loro attivazione automatica. In realtà, molte macchine venivano spente solo in caso di occasionale manutenzione hardware. Col passare del tempo molti importanti concetti di alta disponibilità sono stati lentamente introdotti nel sistema UNIX, per cui oggi è preferibile mantenere le macchine accese e funzionanti il più possibile. Inoltre, questa elevata disponibilità consente di trasferire la posta elettronica e i dati con **uucp** durante le ore notturne, quando le tariffe telefoniche sono basse e la macchina è poco impegnata.

Naturalmente il sistema UNIX funziona correttamente anche se viene attivato solo occasionalmente, in quanto mantenere costantemente la macchina attiva e funzionante non è una esigenza assoluta. Comunque, se la macchina viene utilizzata di giorno, conviene mantenerla sempre attiva piuttosto che disattivarla nel corso della notte, specie se è possibile spegnere il video di console indipendentemente dalla CPU. Nella trattazione della funzione **cron** più avanti in questo capitolo, vi risulterà chiaro come e quando le attività vengono scadenzate nella macchina e potrete fare in modo che il lavoro di gestione venga effettuato in un periodo della giornata a vostra scelta, quando la macchina è abitualmente accesa.

20.3 Il comando date

Il sistema UNIX mantiene un calendario interno a disposizione degli utenti. Nel formato base, questo calendario è di fatto un contatore del numero totale di secondi trascorsi dal 1 gennaio 1970, mitica data di inizio dell'era UNIX. Non esiste un modo semplice per richiamare dalla linea di comando questa indicazione di tempo nel formato originale, trattandosi di un numero in rapidissima crescita, composto da molte cifre (ha un valore oltre 660 000 000).

La maggior parte dei comandi che utilizza l'ora corrente converte il valore in un formato comprensibile. Abbiamo già visto nel Capitolo 7 le funzioni base del comando **date**, che consentono di accedere a livello utente alle funzioni di data e ora. Questo comando

```
$ date
Wed Jun 10 18:50:06 MDT 1991
$
```

visualizza la data e l'ora correnti secondo l'interpretazione della macchina.

Inoltre, il comando **date** è in grado di presentare la data con molti altri formati. Il formato desiderato va specificato come argomento di **date**, subito dopo il carattere **+**; è possibile ottenere praticamente qualunque formato definendo dei campi nell'argomento. L'operatore **%**, seguito da un carattere, consente di definire un campo; il carattere successivo descrive l'elemento della data desiderato. Per esempio, il carattere **m** richiede il numero del mese corrente (da 1 a 12):

```
$ date +%m
06
$
```

Il comando visualizza il mese 06, cioè giugno. Altri caratteri che possono seguire l'operatore % sono **d** per il giorno del mese (da 1 a 31), **y** per le ultime due cifre dell'anno, **H** per l'ora del giorno, **M** per i minuti, **S** per i secondi, **w** per il giorno della settimana (da Sunday a Saturday), **h** per il mese (da January a December) e **r** per l'ora nella notazione AM/PM. Questi caratteri assumono questi significati solo se seguono l'operatore %, in caso contrario vengono visualizzati normalmente. In questo modo **date** può visualizzare espressioni complesse che includono la data e l'ora. Per esempio:

```
$ date +"Oggi è : %h %dth. L'ora corrente è : %r"
Oggi è : Jun 10th. L'ora corrente è : 06:26:32 PM
$
```

Inoltre, potete utilizzare la notazione **%n** per richiedere un carattere di ritorno a capo, in modo da ottenere un'uscita multilinea.

```
$ date +"Mese: %m%nGiorno: %d%nOra: %T"
Mese: 06
Giorno: 10
Ora: 19:02:58
$
```

Il comando **date** in forme simili viene spesso utilizzato in file di comandi per produrre una visualizzazione o assegnare variabili di ambiente. Le possibilità sono limitate solo dalla vostra immaginazione, in quanto per il comando **date** sono disponibili 29 diversi operatori %.

IMPOSTAZIONE DELLA DATA DI SISTEMA

Il superuser può utilizzare il comando **date** per impostare la data e l'ora correnti del sistema. In questo caso, l'argomento (la data e l'ora correnti) di **date** non deve essere preceduto dal carattere + e deve avere il formato **mmddhhmm**, dove il primo **mm** rappresenta il mese espresso con due cifre, **dd** rappresenta il giorno espresso con due cifre, **hh** indica le due cifre dell'ora nella forma su 24 ore, e infine, il secondo **mm** specifica il minuto. Il comando

```
# date 06071647
Sun Jun 7 16:47:00 MDT 1990
#
```

imposta la data corrente al 7 giugno, ore 16 e 47 e la visualizza per verificarne la correttezza. Se necessario, potete inizializzare anche l'anno, aggiungendo la relativa indicazione con altre due cifre aggiunte all'argomento. Questo comando

```
# date 0607164791
```

imposta la stessa data nell'anno 1991.

Nella pratica la data deve essere modificata solo quando indispensabile, in quanto interruzioni nel conteggio del tempo possono causare problemi con il software. È consigliabile aggiornare la data quando il sistema non è impegnato pesantemente; il momento migliore è immediatamente dopo l'inizializzazione. Nelle macchine che dispongono di un hardware dedicato di orologio/calendario occorre di rado inizializzare nuovamente la data di sistema. In ogni caso, per il regolare funzionamento del sistema, è molto importante mantenere la data e l'ora del sistema sempre coerenti.

FUSI ORARI E ORA LEGALE

Il comando **date** gestisce l'ora locale e converte la data e l'ora in GMT (l'ora universale di Greenwich), che è il formato interno del sistema. Il fuso orario

locale e la differenza in ore da GMT sono memorizzati nel file **/etc/TIMEZONE** e sono disponibili di solito anche nella variabile di ambiente **TZ**. Questi riferimenti hanno un formato simile a **EST5EDT**, dove compare prima il nome del fuso orario USA (EST), seguito dal numero di ore di differenza da GMT (5) e dall'indicazione se viene utilizzata l'ora legale (EDT). Generalmente, il sistema inizializza la variabile di ambiente **TZ** dal file **/etc/TIMEZONE**, per cui se trasferite (di fuso orario!) la macchina potete cambiare il file **/etc/TIMEZONE** e poi effettuare l'inizializzazione. Comunque, per sicurezza, inizializzate il fuso orario mediante l'interfaccia di gestione che viene fornita con la macchina.

Il sistema prevede un algoritmo predefinito per passare dall'ora standard all'ora legale, che non garantisce però regolarità assoluta, dal momento che ogni anno il periodo di validità dell'ora legale può cambiare per decisione gevernativa. Abbiate cura di controllare il calendario del vostro sistema dopo l'inizio e la fine del periodo di ora legale.

20.4 Indicazioni di data e ora per i file

Il sistema UNIX mantiene diverse indicazioni di data e ora per tutti i file presenti nel sistema e una di queste informazioni viene visualizzata dal comando **ls -l**. Si tratta dell'*ora* o *tempo di modifica* del file, cioè dell'ora e data dell'ultimo aggiornamento del file. Diversi comandi utilizzano questo tempo per determinare se un file è aggiornato, oppure se è stato cambiato dall'ultima volta in cui il comando ha esaminato il file. Inoltre, il sistema mantiene due ulteriori indicazioni di tempo associate a un file: *l'ora* o *tempo di creazione* e *l'ora* o *tempo di accesso*. L'ora di creazione registra quando il file è stato originariamente creato, mentre l'ora di accesso registra l'ultima volta che un utente o un programma ha letto il file. I tempi di accesso e di creazione non vengono facilmente visualizzati dai comandi utente, ma alcuni comandi, come per esempio **find** e **test**, sono in grado di trattare queste indicazioni di tempi. Generalmente risulta molto importante e significativo il tempo di modifica, o ultimo aggiornamento.

IL COMANDO touch

Il comando **touch** consente di modificare l'indicazione di tempo associata al file. Accetta come argomento un elenco di nomi di file e per default aggiorna l'ora di accesso e di modifica del file:

```
$ ls -l old.file
-rw-rw-rw-1 giorgio     utenti      539 Jan 15 04:15 old.file
$ touch old.file
$ ls -l old.file
-rw-rw-rw-1 giorgio     utenti      539 Jun 10 21:02 old.file
$
```

L'opzione **-m** (modification) induce **touch** a cambiare solo l'ora di modifica, mentre l'opzione **-a** (access) consente di cambiare solo il tempo di accesso.

Il comando **touch** può anche modificare una indicazione di tempo di un file a un qualunque altro valore, specificato con un ulteriore argomento:

```
$ ls -l old.file
-rw-rw-rw-   l1 root     utenti     539 Jun 10 21:02 old.file
$ touch 03211541 old.file
$ ls -l old.file
-rw-rw-rw-   1 root      utenti     539 Mar 21 15:41 old.file
$
```

La nuova data segue le opzioni eventualmente presenti, ma precede l'elenco dei nomi dei file; viene specificata nello stesso formato della data di sistema, cioè **mmddhhmm**, dove il primo **mm** corrisponde alle due cifre del mese, **dd** rappresenta le due cifre del giorno, **hh** rappresenta le due cifre dell'ora nella forma su 24 ore, e il secondo **mm** indica i minuti. Potete anche specificare l'anno, aggiungendo due altre cifre alla stringa della data.

Il comando **touch** consente anche di creare un file vuoto non preesistente:

```
$ ls -l new.file
new.file not found
$ touch new.file
$ ls-l new.file
-rw-rw-rw-   1 root      utenti     0 Jun 10 21:06 new.file
$
```

I file di comandi utilizzano spesso queste istruzioni per verificare l'esistenza di un file. In alternativa, potete creare un file vuoto tramite un editor oppure lanciando il comando di shell > in una linea di comando:

```
$ >new.file
```

L'opzione **-c** (create) evita che **touch** crei un file se non esisteva in precedenza.

Come per altri aspetti di controllo dell'ora in ambiente UNIX, non è conveniente cambiare le indicazioni di tempi associati con i file a meno che non sia realmente indispensabile. Alcune procedure, come per esempio gli strumenti di salvataggio automatico, possono avere un funzionamento diverso dal previsto se il tempo di modifica dei file è stato cambiato.

20.5 I comandi at e batch

Il comando **at** consente di scadenzare (in gergo schedulare, dall'inglese schedule) i lavori da eseguire in un tempo successivo. Questa caratteristica

permette di accodare linee di comando o file di comandi. Se occorre, un file di comandi può includere al suo interno un altro comando **at** che consente di rischedulare l'esecuzione di se stesso. Il comando **at** è accessibile ai normali utenti del sistema e dispone di adeguate misure interne di sicurezza che autorizzano o negano il suo impiego ai singoli utenti. Il comando **at** opera in unione col meccanismo di scheduling a livello di sistema denominato **cron**, che solo i processi di sistema e le utility di gestione del sistema possono utilizzare. Il comando **at** è presente nella versione di sistema UNIX SVR4 e in altre recenti versioni, ma può mancare in alcune versioni meno recenti.

Utilizzate **at** specificando come argomenti nella linea di comando un'ora e una data; il comando **at** legge dallo standard input il testo del comando o procedura da eseguire all'ora indicata; per esempio:

```
$ at 11:45 < testo
job 660414980.a at Thu Jun 14 11:45:00 1991
$
```

In questo caso i comandi specificati nel file **testo** vengono eseguiti in data corrente alle ore 11:45. Il comando **at** risponde con un identificatore di lavoro e l'ora in cui eseguirà il lavoro, scritto sullo standard error. Il lavoro verrà eseguito in quell'ora anche se uscite dal sistema o se la macchina viene reinizializzata prima dell'ora programmata. Il numero di identificazione del lavoro è espresso con il valore del calendario della macchina nel formato interno, in numero di secondi trascorsi dal 1 gennaio 1970. Ovviamente, se la macchina non è in funzione all'ora programmata, il comando non viene eseguito.

Il comando **at** si comporta intelligentemente riguardo all'ambiente di shell in cui eseguire la procedura; fornisce lo stesso ambiente di variabili d'ambiente esportate al momento del lancio di **at** e ristabilisce per la procedura il directory corrente, **umask** e **ulimit**. Lo standard output e lo standard error della procedura di shell vengono inviati tramite mail all'utente dopo l'esecuzione della procedura; se la procedura durante l'esecuzione legge dal suo standard input riceve immediatamente un'indicazione di end-of-file.

INDICAZIONE DELLA DATA NEL COMANDO at

Nel comando **at** è possibile specificare un tempo in diversi formati, che consentono anche l'indicazione di date future. Un tempo può essere specificato come un'ora, con una o due cifre; per esempio, con:

```
$ at 17
```

il comando verrà eseguito alle 17:00. A meno che dopo il tempo sia indicato *am* o *pm* il tempo espresso viene presunto in formato 24 ore. Questo comando

```
$ at 5pm
```

eseguirà la procedura ugualmente alle 17:00. Non si deve inserire alcuno spazio tra il valore numerico dell'ora e la stringa **am** o **pm**; è possibile anche specificare un'ora con quattro cifre inserendo il carattere : tra l'indicazione dell'ora e quella dei minuti:

```
$ at 4:35pm
```

Se la stringa **am** o **pm** viene omessa, l'ora viene ancora interpretata nel formato 24 ore. Altre stringhe speciali permesse sono: **now** (adesso), **noon** (mezzogiorno) e **midnight** (mezzanotte). Per esempio:

```
$ at noon
```

È possibile stabilire l'esecuzione di un lavoro dopo un incremento in tempo, con l'operatore + dopo l'indicazione del tempo. Questo comando

```
$ at now + 15 minutes
```

eseguirà il lavoro fra 15 minuti. Oltre a **minutes**, esistono altri specificatori temporali che possono essere usati con l'operatore +: **hours** (ore), **days** (giorni), **weeks** (settimane), **months** (mesi) e **years** (anni). Per esempio, questo comando

```
$ at 3:15pm + 6 months
```

fornirà un risveglio a distanza di sei mesi dalle ore 15:15 odierne.

Oltre all'ora, potete specificare anche una data per l'esecuzione del comando, come nel caso di

```
$ at 2:15pm Jul 16
```

che stabilisce l'esecuzione del comando alle 14:15 del 16 luglio. Dopo l'ora, l'eventuale data si compone del nome del mese e del giorno numerico oppure del giorno della settimana. Il mese e il giorno della settimana possono essere indicati integralmente o abbreviati a tre lettere; dopo la data, potete specificare anche l'anno, separato da una virgola; per esempio:

```
$ at 2:15pm Jul 16, 1991
```

Anche dopo queste complesse forme di specificazione della data può essere utilizzato l'operatore +; per esempio, questo comando:

```
$ at 2:15pm Jul 16, 1991 + 3 years
```

eseguirà il lavoro tre anni dopo il 16 luglio 1991.

Queste specifiche di date sembrano complesse, ma in realtà il formato è abbastanza intuitivo, per cui nella maggior parte dei casi la vostra linea di comando funzionerà correttamente. Se questo non si verifica, **at** restituisce un messaggio di errore conciso:

```
$ at 6:15 pm Satur
at: bad date specification
$
```

VISUALIZZAZIONE DELLA CODA DI LAVORI at

Il comando **at** prevede due opzioni per gestire la lista di lavori che avete scadenzato. Con l'opzione **-l** (list) ottenete l'elenco di tutti i lavori con il relativo numero di identificazione e la data e l'ora di esecuzione scadenzata:

```
$ at -l
660503300.a     Fri     Jun 15 09:35:00 1991
660503360.a     Fri     Jun 15 09:36:00 1991
$
```

Utilizzate l'opzione **-r** (remove) con un elenco di identificatori di lavoro specificati come argomenti, per rimuovare dalla coda i lavori così identificati:

```
$ at -r 660503300.a
$ at -l
660503360.a     Fri     Jun 15 09:36:00 1991
$
```

Naturalmente, potete rimuovere solo i lavori che voi stessi avete creato, mentre solo il superuser può rimuovere qualunque lavoro dalla coda. I lavori creati dal comando **at** vengono temporaneamnete memorizzati nel directory **/usr/spool/cron/atjobs** fino a quando non vengono eseguiti; se necessario, potete cancellare un particolare lavoro direttamente dal directory invece di utilizzare **at -r**.

Il contenuto del file di procedura shell che **at** esegue dipende dalle vostre personali esigenze, tuttavia sono permesse solo le procedure che si possono normalmente eseguire da shell. In altre parole, poiché il comando **at** esegue la procedura nell'ambiente di esecuzione che era in vigore quando il coman-

do stesso è stato specificato, non è possibile fare con **at** più di quanto sia permesso operando al terminale.

Il comando **at** esegue la procedura una sola volta, al momento specificato; se volete realizzare una procedura che vada in esecuzione con regolarità, per esempio una volta al giorno o una volta ogni ora, inserite un nuovo comando **at** nella procedura, in modo che la procedura stessa si riscadenzi per entrare in esecuzione. È possibile utilizzare gli operatori di tempo relativo disponibili per il comando **at**, come appare nel seguente esempio:

```
$ at now + 1 day
```

Questo comando eseguirà la procedura tra ventiquattro ore esatte. Se inserite questo comando nella procedura eseguita da **at**, avrete creato un comando che è in esecuzione una volta al giorno. Un semplice modo per raggiungere questo obiettivo è di scrivere

```
at now + 1 day < $0
```

al termine della procedura. Lo shell interpreta la variabile **$0** come il nome della procedura. Il valore **$0** tuttavia è accettabile solo se il comando **at** che scadenza la procedura viene eseguito dal directory in cui risiede la stessa procedura; in caso diverso, il valore di **$0** non è utilizzabile e occorre specificare il pathname completo.

I lavori che si autoscadenzano possono causare molti problemi nel sistema se non sono corretti; controllate attentamente il funzionamento di una procedura prima di attivarne l'autoscadenzazione. Inoltre, tenete conto che se la procedura non è più presente nel disco o se il suo pathname cambia, **at** abortisce e il lavoro non viene più scadenzato, producendo talvolta file delle registrazioni storiche estremamente grandi. L'utility **crontab**, che tratteremo tra breve, è generalmente preferibile per scadenzare i lavori che devono essere messi in esecuzione con regolarità.

IL COMANDO batch

Un comando strettamente connesso ad **at** è **batch**, che agisce come

```
$ at now
```

con la differenza che il lavoro viene scadenzato per essere eseguito quando il carico di lavoro del sistema è ridotto. Inoltre, **at now** ha dei tempi di risposta superiori, mentre **batch** risponde come previsto. Il comando **batch** utilizza un algoritmo interno per decidere quando eseguire il lavoro; questo è utile quando intendete eseguire molti lavori di lunga durata nello stesso momento. Se questi lavori venissero semplicemente eseguiti in background

utilizzando l'operatore **&**, richiederebbero tutti di andare subito in esecuzione, sovraccaricando probabilmente la CPU, e prolungando la loro esecuzione perché ognuno otterrebbe dal sistema operativo solo una piccola frazione del tempo macchina. Il comando **batch** evita che si verifichi questa situazione, in quanto consente di scadenzare subito tutti i lavori, ma ne mette in esecuzione immediata solo una piccola parte. In questo modo i primi lavori concludono la loro attività in un tempo relativamente ridotto e l'esecuzione di quelli successivi viene spaziata in base al carico del sistema. Il tempo totale necessario al completamento di tutti i lavori risulterà inferiore con il comando **batch** e anche l'utilizzo complessivo di CPU risulterà ridotto.

Il comando **batch** non accetta argomenti perché il lavoro viene sempre eseguito appena è conveniente. Come il comando **at**, **batch** legge la procedura dallo standard input e invia all'utente tramite mail lo standard output e lo standard error della procedura. Per esempio:

```
$ batch < testo
job 660501560.b at Fri Jun 12 09:06:00 1991
$
```

Il comando **batch** scrive il numero di lavoro sullo standard error nello stesso formato utilizzato da **at**. Potete usare il comando **at −r** per cancellare anche un lavoro selezionato da **batch**, a condizione che lo facciate prima che vada in esecuzione.

CONSIDERAZIONI DELLA SICUREZZA CON at E batch

Sia **at** che **batch** prevedono meccanismi di sicurezza che impediscono agli utenti (a eccezione del supervisore) di scadenzare lavori senza una preventiva autorizzazione. Se sperimentate i precedenti comandi sul vostro sistema, potreste non essere autorizzati a utilizzarli, come in questo caso:

```
$ at 16:35 < testo
at: you are not authorized to use at. Sorry
$
```

Ogni singolo utente può essere autorizzato o inibito a utilizzare questi strumenti; l'elenco degli utenti autorizzati si trova nel directory **/etc/cron.d**.

```
$ cd /etc/cron.d
$ ls −F
FIFO        at.deny      cron.deny      queuedefs
at.allow    cron.allow   logchecker*
$
```

FIFO viene utilizzato come canale di comunicazione tra i programmi **crontab** e **cron**, mentre **queuedefs** contiene le informazioni utilizzate internamente dall'utility **cron**. **logchecker** è uno strumento usato per troncare i file storici troppo lunghi. I file di interesse per **at** e **batch** in questo directory sono **at.allow** (autorizza) e **at.deny** (vieta), che contengono rispettivamente l'elenco di identificatori degli utenti che hanno l'autorizzazione o il divieto di usare questi comandi:

```
$ cat at.allow
root
sys
adm
uucp
giorgio
leo
pat
$
```

Gli identificatori di utente sono elencati linea per linea nel file.

Solo il superuser può modificare il file **at.allow** e il file **at.deny**, ed è compito del gestore di sistema mantenere questi file aggiornati con le variazioni negli identificatori degli utenti della macchina. Tuttavia, la gestione è abbastanza semplice, con le regole esposte qui di seguito. Se il file **at.allow** è presente, solo gli utenti elencati nel file possono lanciare i comandi **at** e **batch**, indipendentemente dalla presenza o meno del file **at.deny**. Se invece **at.allow** non è presente, **at.deny** viene esaminato per verificare se a un utente è esplicitamente vietato l'utilizzo dei comandi; per cui solo gli utenti che non sono in elenco **at.allow** possono eseguire i comandi, se manca il file **at.allow**. Se infine nessuno dei due file è presente, l'unico abilitato a lanciare i comandi è il superuser. In conclusione, per permettere l'impiego di **at** a tutti gli utenti, cancellate il file **at.allow** e assicuratevi che **at.deny** sia presente, ma vuoto. Se il file **at.allow** è presente, deve sempre comprendere gli identificatori di utente **root**, **sys**, **adm** e **uucp**.

In aggiunta i comandi **at** e **batch** usano un file *prototipo* che viene unito alla procedura dell'utente per generare la procedura di shell "reale" eseguita quando si raggiunge la data di scadenza.

Il file **etc/cron.d/.proto** contiene questo file prototipo. Il superuser può modificare questa procedura per imporre maggiore sicurezza a tutti i job scadenzati. Consultate la man page **proto(4)** per ulteriori dettagli sulla personalizzazione dell'ambiente per i lavori scadenzati.

20.6 Il servizio cron

I comandi **at** e **batch** utilizzano i servizi messi a disposizione da un meccanismo di scheduling sempre attivo all'interno del sistema UNIX. Si tratta del programma **cron** (chronograph), già incontrato quando abbiamo descritto

il comando **ps –ef**; il nome **cron** identifica un demon o processo di sistema che viene immediatamente lanciato all'inizializzazione del sistema UNIX e rimane costantemente presente finché il sistema è in funzione.

Una volta ogni minuto, il processo si risveglia, esamina un file di controllo per vedere se esistono lavori da eseguire in quel momento e, in caso positivo, li esegue. Se non è scadenzato alcun lavoro per quel preciso istante, **cron** si sospende fino al minuto successivo. Questo potente meccanismo di scheduling si basa sulla natura multiprocesso del sistema UNIX; sebbene sia sempre attivo, non utilizza molte risorse di CPU.

Il codice eseguibile di **cron** si trova nel file **/etc/sbin/cron**; tuttavia, non essendo un comando utente, non deve mai essere eseguito direttamente, neanche dal superuser. Se nella macchina sono attive due versioni di **cron**, il sistema non può funzionare; se invece **cron** non è in esecuzione nella macchina, reinizializzate il sistema piuttosto che eseguire il comando direttamente. Se **cron** non entra in esecuzione neppure dopo la reinizializzazione esistono dei seri problemi nel sistema, che di solito potete risolvere ricaricando il sistema operativo dai dischi originali. Comunque, alcune versioni di sistema UNIX hanno meccanismi di temporizzazione con nome diverso da **cron**, per cui se il sistema operativo della macchina non è SVR4, il comando **ps –ef** può non visualizzare processi di nome **cron**. D'altra parte, le funzioni eseguite da **cron** sono fondamentali per il corretto funzionamento di tutti i sistemi UNIX, per cui il processo è sempre presente sotto nome diverso, a meno di malfunzionamenti del sistema.

Oltre alla utilizzazione a livello di sistema delle funzioni di **cron**, vari meccanismi di temporizzazione permettono ai singoli utenti e al superuser di scadenzare i lavori da mandare in esecuzione a intervalli regolari di un minuto o più. A differenza di **at**, che scadenza i lavori per una sola esecuzione, **cron** scadenza i lavori per l'esecuzione ripetuta a intervalli regolari specificati.

IL FORMATO DEL FILE crontab

Il file di controllo utilizzato da **cron** è conosciuto con il nome di **crontab** (cron table). Era originariamente un singolo file col pathname **/etc/crontab**; solamente il supervisore aveva l'autorizzazione a modificare questo file e a modificare quindi la programmazione dei lavori nel sistema; alcune versioni meno recenti di sistema UNIX utilizzano ancora questa locazione di file. Attualmente SVR4 e altre recenti versioni di UNIX prevedono possibilità **crontab** più estese all'interno del directory **/var/spool/cron/crontabs**, come:

```
$ ls -F /var/spool/cron/crontabs
adm     root     sys
$
```

Gli utenti che hanno scadenzato dei lavori tramite **cron** possiedono un file in questo directory, con il nome associato all'identificatore di login dell'utente che ha creato il file. I file elencati nell'esempio sono presenti su quasi tutti i sistemi SVR4 e ognuno viene designato per una applicazione specifica di scadenzazione a livello di sistema.

Il file **sys** viene generalmente impiegato per raccogliere i dati relativi alle prestazioni del sistema, se la macchina dispone degli strumenti **sar** di analisi delle prestazioni; il file **adm** viene di solito usato per scadenzare l'esame delle prestazioni a partire dai dati **sar**.

La maggior parte delle schedulazioni a livello di sistema è di solito collocata nel file **root**, che spesso contiene diversi commenti che descrivono il formato di file di tutti i file **crontab**.

La Figura 20.1 mostra un tipico file **root crontab** per un piccolo sistema SVR4; anche se i file del vostro sistema probabilmente differiscono da questo esempio, la figura illustra il formato di file richiesto e alcuni tra i pro-

```
# Il contenuto di questo file viene scadenzato tramite il comando cron
#
#     Formato di linea:
#min ora giorme mese giorse lineacomando
#
#     min - minuto/i
#     ora - ora/e
#     giorme - giorno/i del mese (1,2,...,31)
#     mese - mese/i dell'anno (1,2,...,12)
#     giorse - giorno/i della settimana (0-6, 0 dom)
#
#     Esempio:
#17 5 * * 0 /bin/su root -c "/sbin/cleanup > /dev/null"
#
#     Alle 5:17 di domenica di ogni mese, verificare
#     se esiste qualche file system da salvare.
#
#
#=====================================
=
#
17 5 * * 0 /bin/su root -c "/sbin/cleanup > /dev/null"
48 11,14 * * 1-5 /bin/su uucp -c "/usr/lib/uucp/uudemon.admin \
    > /dev/null 2>&1"
45 23 * * * ulimit 5000; /bin/su uucp -c "/usr/lib/uucp/uudemon.cleanup \
    > /dev/null 2>&1"
40 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.poll > /dev/null"
26,56 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.hour > /dev/null"
0 2 * * 0,4 /etc/cron.d/logchecker
```

Figura 20.1 Tipico file **crontab** di root per un piccolo sistema UNIX SVR4.

cessi di gestione scadenzati nell'intero sistema. Esamineremo più avanti nel capitolo le modalità con cui gli utenti possono creare i lavori scadenzati.

Nei file **crontab** le linee di commento iniziano con il carattere # e vengono ignorate quando **cron** legge il file. Le linee che non iniziano con # indicano i comandi che devono essere scadenzati, uno per linea; i comandi possono estendersi su più linee se le linee intermedie terminano col carattere \ (barra inversa), che precede il carattere di ritorno a capo e ne annulla il significato di fine linea. I primi cinque campi della linea, separati uno dall'altro da uno spazio, stabiliscono la data e l'ora in cui il comando deve essere eseguito; in ognuno di questi campi la presenza di un numero specifica una data o un momento, la presenza del carattere * (asterisco) ha per **cron** il significato *ogni*. Il primo campo della linea specifica il minuto in cui il lavoro viene eseguito, per cui 00 indica l'ora, 30 significa 30 minuti dopo l'ora e così via; il secondo campo della linea specifica l'ora del giorno, nel formato 24 ore; il terzo campo indica il giorno del mese, da 1 a 31; il quarto campo specifica il mese dell'anno, da 1 a 12, mentre il quinto campo indica il giorno della settimana, da 0 per la domenica a 6 per il sabato.

Potete combinare queste specifiche, per cui

```
30 5 * * 1
```

viene interpretato come 5:30 di ogni lunedì, mentre

```
30 5 * * *
```

significa 5:30 di ogni giorno della settimana e

```
30 * * * *
```

indica 30 minuti dopo ogni ora, di ogni giorno.

All'interno di ogni campo è possibile definire una serie di momenti, separati da una virgola, senza spazi interposti:

```
00 17 * * 1,2,3,4,5
```

Questo esempio indica che l'esecuzione del comando deve avvenire diciassette minuti dopo la mezzanotte, da lunedì a venerdì.

Il resto della linea che segue l'ora indica il nome del comando da eseguire, insieme a una ridirezione dell'ingresso o dell'uscita. Nella prima linea di comando di Figura 20.1 **/sbin/cleanup** viene eseguito 17 minuti dopo la mezzanotte, con lo standard output e lo standard error ridiretti al device nullo, quindi trascurati; questo comando viene eseguito come se fosse stato lanciato dall'utente **root**. Nella figura il comando **su** viene ancora usato per cambiare utente con quello indicato come primo argomento, di solito **uucp**.

L'operatore **%** ha uno speciale significato per **cron**: induce **cron** a consi-

derare il resto della linea come lo standard input del comando specificato prima del %. Utilizzando l'operatore \ prima del ritorno a capo, potete inserire nei file **crontab** procedure shell di grandi dimensioni, tuttavia questo procedimento viene raramente seguito, e la maggior parte delle procedure rimane contenuta nei rispettivi file.

Il primo comando in Figura 20.1 esegue alcune regolari operazioni di pulizia di file a livello di sistema; il suo output viene scartato. Se la ridirezione alla console viene cancellata, questo output viene inviato per posta all'utente **root**. Il comando **cron**, come **at**, utilizza la posta elettronica per restituire all'utente lo standard output e lo standard error dei comandi.

I successivi quattro comandi sono delle attività di gestione che i programmi di comunicazione **uucp** eseguono regolarmente in tempi diversi, a seconda della loro funzione. Il comando **uudemon.admin** genera un rapporto sullo stato della macchina e viene attivato alle 11:48 AM e alle 2:48 PM ogni giorno della settimana; in particolare, riporta le eventuali violazioni di sicurezza **uucp** e i lavori che sono stati accodati nella settimana passata e non sono stati ancora trasmessi.

La linea successiva, attivata ogni giorno alle 11:45 PM, è una procedura di ripulitura per il sistema **uucp**, che generalmente elimina vecchi lavori e file storici delle sessioni dal sistema **uucp**. Le successive due linee in Figura 20.1 sono i processi di scadenzazione oraria di **uucp**, che ogni ora ricercano i lavori da spedire non ancora trasmessi e, se occorre, cercano di spedirli. Poiché **uucp** utilizza un algoritmo di tipo incrementale per determinare se deve effettuare un collegamento con una macchina remota, il processo orario non sempre tenta di stabilire la connessione, anche se ha del lavoro in attesa. Queste scadenzazioni per **uucp** devono essere presenti in un file **crontab** di sistema se utilizzate l'utility **uucp** o **mail** per comunicare con altre macchine; se non sono presenti, la vostra posta non può essere inviata correttamente.

L'ultima linea in Figura 20.1 viene eseguita ogni domenica e giovedì alle 2:00 AM, e consiste in una procedura che ripulisce i file storici prodotti dalla stessa funzione **cron**.

Queste voci, o altre simili, sono di solito presenti nei sistemi attivi UNIX, ma il loro contenuto può differire in relazione al lavoro che viene normalmente eseguito su quella macchina. In ogni caso, le attività ripetitive di gestione, come per esempio la cancellazione di vecchi file storici in **/var/adm**, la cancellazione di vecchi file **core**, e così via, vengono generalmente delegati alla funzione **cron**.

IL COMANDO crontab

I gestori esperti spesso modificano i file **crontab** direttamente con un editor per realizzare cambiamenti nello scadenzario dei programmi. Comunque,

gli utenti possono creare lavori a esecuzione programmata utilizzando il comando **crontab**, che consente di creare e di modificare file **crontab**.

Per definire un nuovo piano di scadenzazione **crontab**, create un file **crontab** con l'editor o copiate quello esistente in un nuovo file per modificarlo; quando il file è corretto, potete aggiungere il file al directory **crontab** con il comando:

```
$ crontab nomefile
$
```

dove *nomefile* è il nome del nuovo file che avete creato. Come sempre, il comando non restituisce nessun messaggio se l'operazione di installazione del piano ha avuto successo. Il piano viene creato sotto l'identificatore dell'utente che esegue il comando, per cui gli utenti possono avere piani di scadenzazione personali che non interferiscono con altri. È permesso un solo file **crontab** per utente. Il comando **crontab** può anche leggere il piano dallo standard input cosicché è ammessa la ridirezione. Comunque, se introducete il comando **crontab** senza nomi di file e senza ridirezione dell'input, dovete fare attenzione a uscire con il tasto DEL invece che con CTRL-D perché con il carattere di fine-file **crontab** carica un file vuoto nel vostro piano di scadenzazione: certo non quello che intendevate.

Il comando **crontab** rimuove un piano di scadenzazione se specificate l'opzione **-r** (remove):

```
$ crontab -r
$
```

Il comando **crontab** visualizza un messaggio di errore se non esiste alcun file da rimuovere. Infine, l'opzione **-l** (list) consente di elencare il contenuto del file **crontab** se è stato creato.

SICUREZZA E DIRITTI DI ACCESSO DI cron

I dati necessari per utilizzare la funzione **cron** sono presenti in due directory. Il directory **/var/spool/cron/crontabs** contiene il file **crontab**, il directory **/etc/cron.d** contiene le informazioni di controllo relative ai meccanismi di scadenzazione. I file **cron.allow** e **cron.deny**, contenuti in questo directory, svolgono le stesse funzioni di **at.allow** e **at.deny** che abbiamo visto in precedenza. Se il file **cron.allow** è presente viene esaminato per ricavare la lista degli utenti autorizzati a utilizzare il comando **crontab**; se il file **cron.allow** non è presente, **cron.deny** consente di verificare se a un utente è proibito utilizzare il comando **crontab**. Se non sono presenti né il file **cron.allow** né il file **cron.deny**, solo il superuser può utilizzare il comando **crontab**. In entrambi i file **cron.allow** e **cron.deny**, gli identificatori di utente sono elencati uno per linea.

20.7 Approfondimenti

Molte utility di temporizzazione di sistema UNIX sono dedicate agli implementatori di sistema per consentire l'ottimizzazione del sistema e delle sue applicazioni; poiché per la maggior parte degli utenti sono di interesse limitato, parleremo solo delle più significative.

IL FILE log DI cron

cron tiene una traccia storica di tutti i lavori eseguiti con **at**, **batch** o **cron**. Nei sistemi SVR4, questo file si chiama **log** e si trova nel directory **/var/cron**. Tuttavia, le versioni meno recenti del sistema possono collocare questo file in altri directory, di solito **/usr/lib/cron/cronlog**. Questo file storico può risultare utile per determinare i motivi di vari problemi associati con la scadenzazione dei lavori; il formato di questo file non è molto esplicito ma può essere facilmente compreso. Ecco un esempio:

```
$ tail log
< sysadm 229 c Wed Jun 13 17:00:02 1991
< sys 230 c Wed Jun 13 17:00:02 1991
! *** cron started *** pid = 75 Thu Jun 14 08:18:45 1991
> CMD: /usr/lib/sa/sa1
> sys 86 c Thu Jun 14 08:20:00 1991
< sys 86 c Thu Jun 14 08:20:01 1991
! *** cron started *** pid = 75 Thu Jun 14 08:41:45 1991
> CMD: 660414080.a
> root 98 a Thu Jun 14 08:48:00 1991
< root 98 a Thu Jun 14 08:48:03 1991
$
```

Nelle versioni meno recenti, il formato del file può differire dall'esempio, che si riferisce ai sistemi SVR4. Le linee che contengono "*** cron started ***" vengono prodotte da **cron** quando il sistema viene reinizializzato e il programma **cron** parte. Viene inclusa anche l'indicazione della data. Ogni comando eseguito dal processo **cron**, tramite il comando **at** o il file **crontab**, ha una sezione che inizia con "CMD:" e il nome del comando da eseguire. L'ultima voce dell'esempio – "CMD: 660414080.a" – contiene un nome familiare di un comando selezionato da **at**. Tutte le linee che seguono "CMD:", fino al comando successivo, sono voci che **cron** produce quando esegue i comandi. In realtà, **cron** esegue i comandi definendo una copia di se stesso (*fork*) e delega al processo figlio il compito di eseguire i comandi. Tuttavia, ambedue i processi scrivono nel file log; le linee che iniziano con il carattere > vengono scritte dal programma originale **cron** prima che il comando sia eseguito, mentre le linee che iniziano con il carattere < vengono prodotte dal processo figlio, una per ogni ulteriore programma eseguito da **cron**.

L'identificatore effettivo di utente del processo appare per primo, seguito dal PID del processo, dal tipo di lavoro, **a** per un lavoro **at** o **c** per un lavoro selezionato da **cron**, infine la data e l'ora in cui il processo è stato eseguito chiudono la linea.

Queste informazioni di log possono essere utili se sospettate che **cron** crei o modifichi file scorrettamente. Spesso la data o il proprietario di un file, come indicati dal comando **ls −l**, possono essere rintracciati nelle informazioni di **cron** per identificare il **CMD** che ha cambiato il file. Queste ricerche sono necessarie solo nei rari casi in cui i lavori scadenzati causano problemi, per esempio quando si verificano con una frequenza regolare nel tempo inspiegabili cambiamenti nella vostra macchina o nel file system, ovviamente fuori dal controllo dell'utente.

Il file log viene regolarmente trasferito in **/var/cron/olog** dal programma **logchecker** già citato, e viene cancellato nella successiva esecuzione di **logchecker**.

MISURA DEL TEMPO DI ESECUZIONE DI UN COMANDO

Quando create nuove procedure di shell o programmi eseguibili, spesso è opportuno valutare le risorse di sistema utilizzate dal comando. Questa informazione può aiutarvi a capire meglio il costo relativo dei comandi per ottimizzarli. Comunque, in un sistema multiprocesso il tempo reale (clock time) che un comando impiega può non essere una buona stima delle risorse di sistema che consuma. Conseguentemente, il sistema UNIX fornisce molti strumenti per calcolare il tempo di esecuzione di un comando. Il più semplice di questi strumenti è il comando **time**. Specificate un comando o un pipeline di cui intendete valutare il tempo di esecuzione come argomento di **time**:

```
$ time sleep 100
real     1:39.4
user        0.0
sys         0.1
$
```

Questo esempio "dorme" per 100 secondi. Il comando **time** scrive sullo standard error i dati in uscita che indicano la quantità di risorse di sistema, espressa in secondi e decimi di secondo, che sono utilizzate dal comando **sleep**. Per tempo reale (*real*) si intende il tempo totale intercorso tra l'inizio e la fine dell'esecuzione del comando. In questo caso è trascorso 1 minuto e 39.4 secondi, con un margine di imprecisione di 0.6 secondi. Il comando **sleep** ha un margine di imprecisione che arriva a un secondo. Il tempo utente (*user*) indica la quantità di tempo che il programma impiega a eseguire il suo codice e in questo esempio è al di sotto della precisione degli strumen-

ti di misura. Il tempo *sys* rappresenta la quantità di tempo utilizzato direttamente dal sistema UNIX per servire quel comando, 0.1 secondi in questo caso.

Il tempo di CPU utilizzato da un comando è in realtà la somma delle misure dei tempi *sys* e utente, ma il comando **time** li indica separatamente per cui i progettisti software possono stabilire se il programma utilizza risorse di kernel o se impiega il tempo all'interno del codice. La differenza tra il tempo reale e questa somma rappresenta una misura del tempo di CPU destinato agli altri programmi in esecuzione sulla macchina mentre si effettua la misura. È evidente che il tempo di CPU utilizzato da un comando può essere molto minore del tempo reale trascorso. In condizioni di poco carico di macchina, i programmi hanno un tempo reale che si avvicina molto al totale, come in questo caso:

```
$ time cat /unix >/dev/null
real    6.5
user    4.5
sys     1.0
$
```

Questo comando impiega 5.5 secondi di tempo di CPU e termina in 6.5 secondi, e un solo secondo viene assegnato a un altro lavoro nel sistema. In un sistema molto più impegnato, questo stesso comando impiegherebbe all'incirca la stessa quantità di tempo di utente e di tempo **sys**, ma indubbiamente il tempo reale sarebbe molto maggiore.

IL PROGRAMMA sync

Per considerazioni di efficienza interna, il sistema UNIX mantiene i dati di I/O dei dischi nei buffer della sua memoria e aggiorna i blocchi su disco solo quando risulta necessario. Questo vuol dire che potete scrivere un file e ritornare in shell senza che il file sia effettivamente scritto su disco: in questo caso, se l'alimentazione del sistema cade, le vostre modifiche possono non esistere quando riaccendete la macchina. I sistemi SVR4 dispongono di particolari processi per aggiornare periodicamente i buffer modificati sul disco rigido di sistema. Questi processi, denominati **pageout** (page output) e **fsflush** (file system flush), visualizzati dal comando **ps** **-e**, aggiornano il disco regolarmente, di solito ogni 30 secondi. Le versioni meno recenti del sistema UNIX possono non includere questa procedura automatica di aggiornamento, per cui dovete fare attenzione che i buffer di sistema siano riportati sul disco prima di spegnere la macchina. Uno speciale programma denominato **sync** (synchronize) consente di forzare manualmente l'aggiornamento dei buffer di memoria sul disco. Potete eseguire questo programma

ogni volta che occorre forzare un aggiornamento di buffer, tuttavia nei sistemi SVR4 quest'operazione è necessaria raramente:

```
$ sync
$
```

Il programma **sync** può restituire il controllo a shell, prima di avere compiuto l'operazione, per cui dovete eseguire di solito **sync** due o tre volte per assicurarvi che il disco venga effettivamente aggiornato correttamente, prima di togliere tensione al sistema:

```
$ sync
$ sync
$
```

ADDEBITO DEL TEMPO MACCHINA AGLI UTENTI

I grandi sistemi UNIX spesso dispongono di procedure di contabilizzazione che consentono di addebitare ai singoli utenti il tempo impiegato per il collegamento, l'utilizzo di CPU e l'utilizzo di spazio su disco. Questi strumenti, se installati, si trovano generalmente nel directory **/usr/lib/acct**; i file di storicizzazione di tutti i processi eseguiti sono contenuti in **/var/adm/acct**; altri dati sono memorizzati in **/var/adm/utmp** e **/var/adm/wtmp**. Se la macchina prevede questi procedimenti di conteggio, il gestore del sistema deve accertarsi che le procedure siano regolarmente messe in esecuzione, magari tramite la funzione **cron**, altrimenti questi file di dati possono raggiungere dimensioni eccessive. Dovete verificare periodicamente questi file e troncarli (ma non cancellarli) quando si sono estesi oltre i limiti ragionevoli.

Sono disponibili molti comandi che consentono di addebitare agli utenti professionali l'uso delle risorse di sistema, come risulta da questo elenco:

```
$ ls -F /usr/lib/acct
acctcms*    acctmerg*   chargefee*  lastlogin*  ptecms.awk*  turnacct*
acctcon*    accton*     ckpacct*    monacct*    ptelus.awk*  utmp2wtmp*
acctcon1*   acctprc*    closewtmp*  nulladm*    remove*      wtmpfix*
acctcon2*   acctprc1*   diskusg*    prctmp*     runacct*
acctdisk*   acctprc2*   dodisk*     prdaily*    shutacct*
acctdusg*   acctwtmp*   fwtmp*      prtacct*    startup*
$
```

Con questi strumenti la contabilizzazione può essere attivata e disattivata; gli strumenti consentono di effettuare un'analisi dettagliata dell'uso del sistema in termini di tempo, utilizzo di disco, di elaborazione e del tempo di collegamento. Solo un gestore di sistema esperto può utilizzare completamente questi strumenti, che d'altronde non vengono impiegati molto in pic-

coli sistemi UNIX su personal. Potete disabilitare i riferimenti in **crontab** se gli strumenti sono presenti ma non volete impiegarli.

ADDEBITO DELL'USO DEL SISTEMA CON IL PACCHETTO sar

I sistemi completamente configurati prevedono il pacchetto **sar** (system activity reporting), che fornisce un'analisi completa di tutta l'attività svolta nel sistema UNIX durante un certo intervallo di tempo. Il pacchetto **sar** include strumenti di registrazione di dati storici dell'utilizzo della CPU, del disco e dei buffer; di contatori dell'attività di disco e delle periferiche tty; di stime sull'accesso ai file e altre misure interne. L'interpretazione di questi dati è un compito da veri esperti. Gli strumenti **sar**, se presenti, si trovano nel directory **/usr/lib/sa** e i dati che producono in **/var/adm/sa**. Esaminate questi directory di tanto in tanto per assicurarvi che i file contenenti informazioni non crescano troppo, impegnando eccessivo spazio su disco. Gli strumenti **sar** vengono di solito eseguiti tramite **cron** e i file **crontab** possono includere uno o più comandi relativi a **sar**. Di solito viene utilizzato **sys crontab**, ove le linee che eseguono i comandi **sadc**, **sa1** e **sa2** sono relative agli strumenti **sar**. Potete disattivare anche la registrazione storica delle attività eseguite da **sar** commentando le corrispondenti linee nei file di **crontab**.

Il pacchetto **sar** include un altro programma di valutazione dei tempi simile a **time**, denominato **timex**, che fornisce una migliore informazione diagnostica di **time**, specialmente per quanto riguarda le procedure di shell che generano processi figli. Tenete presente che il pacchetto **sar** deve essere comunque presente nella macchina perché **timex** possa fornire questa uscita dettagliata. Il comando **timex** in assenza di opzioni produce lo stesso tempo reale, utente e sys che si ottiene con la funzione **time**; sono previste le opzioni **-p** per ottenere i dati del tempo di elaborazione, **-o** per i dati di utilizzazione di blocchi su disco, **-s** per i dati dell'attività totale del sistema, riguardanti il comando che viene eseguito sotto il controllo di **timex**; per esempio:

```
$ timex -o cat /etc/profile >/dev/null

real        0.93
user        0.00
sys         0.26

CHARS TRNSFD = 145216
BLOCKS READ  = 76

$
```

Le altre opzioni generano in uscita risultati ugualmente dettagliati.

Capitolo 21

Inizializzazione e arresto del sistema

In un sistema UNIX sono normalmente sempre in esecuzione diversi processi, per cui è molto dannoso spegnere direttamente la macchina alla fine del lavoro. Il sistema UNIX dispone di strumenti espressamente progettati per realizzare una sequenza predefinita di azioni che consente di avvertire tutti gli utenti prima dello spegnimento. Si tratta della cosiddetta procedura di chiusura (*shutdown*), che occorre seguire strettamente per garantire il corretto funzionamento del sistema quando la macchina viene riaccesa. La procedura di accensione è ugualmente complessa, ma sono ugualmente disponibili gli strumenti per inizializzare il sistema correttamente. In questo capitolo esamineremo i passi che il sistema segue durante le procedure di accensione e di spegnimento della macchina e accenneremo brevemente ad alcuni dei possibili stati di sistema, soffermandoci sui più comuni problemi che si possono presentare nella procedura di inizializzazione.

21.1 L'ambiente UNIX in attività

Durante il regolare funzionamento di un sistema UNIX, esistono in macchina molti processi attivi contemporaneamente. Naturalmente i processi di sistema sono sempre in esecuzione e un gestore di sistema alla console dispone di uno shell e di alcuni programmi associati con la sessione. Inoltre, saranno in esecuzione programmi di proprietà degli altri utenti che si sono collegati alla macchina da terminali remoti, così come possono essere in

esecuzione attività come il trasferimento di posta elettronica in background, i trasferimenti di dati **uucp**, oppure operazioni di stampa. Inoltre, come abbiamo accennato nel capitolo precedente, la presenza di numerosi buffer comporta che non sempre il contenuto *logico* dei vostri file corrisponda effettivamente al contenuto *reale* delle memorie di massa. Per esempio, quando scrivete un file mediante un editor, quel file probabilmente verrà realmente scritto sul disco secondi o minuti dopo che siete ritornati in shell e avete preso a eseguire altri comandi.

Tutte queste considerazioni, e diverse altre che vedremo tra breve, implicano la necessità di svolgere correttamente le operazioni di spegnimento della macchina utilizzando, quando è possibile, gli strumenti predisposti a questo scopo.

Naturalmente c'è sempre la possibilità di incidenti: può verificarsi un'interruzione nella distribuzione elettrica che spegne brutalmente la macchina nel corso della sua attività di regime. Le versioni di sistema UNIX più moderne e le macchine con hardware più complesso possono prevedere mezzi per ovviare alle improvvise mancanze di tensione, oltre a controllare direttamente lo spegnimento fisico del sistema. Potete comunque evitare questi problemi riducendo la possibilità di spegnimento accidentale della macchina.

21.2 Chiusura del sistema

Una corretta procedura di interruzione delle attività consente di avvertire gli utenti di uscire dal sistema prima che la macchina venga spenta, disattiva con cura tutti i processi non essenziali, aggiorna i vari file system e file storici, sincronizza il contenuto del disco con i buffer interni e infine disattiva tutti i restanti processi. Alcuni sistemi possono automaticamente parcheggiare le testine del disco come parte della procedura di spegnimento; altri sistemi provvedono anche a togliere fisicamente la tensione alla macchina con un interruttore comandato da software.

IL COMANDO shutdown

Per chiudere il sistema potete utilizzare diversi strumenti; in ogni caso, l'uso di uno qualsiasi di questi è sempre preferibile al togliere direttamente tensione alla macchina. Lo strumento più sicuro, ma anche il più lento, è il comando **shutdown**, che si trova nel directory **/sbin/shutdown**; poiché è una procedura shell potete esaminarlo per una migliore comprensione delle azioni svolte. Come ogni strumento per l'attivazione e disattivazione della macchina, **shutdown** è riservato al supervisore e può essere lanciato solo dalla console di sistema e dal directory root (*/*). Il comando **shutdown** visualizza un messaggio di errore e non esegue l'attività se non sono rispettate le condizioni precedenti.

Le procedure di spegnimento della macchina e i messaggi visualizzati durante la fase di disattivazione variano sensibilmente tra le diverse versioni di sistema UNIX. Molti sistemi mettono a disposizione del superuser un menu nelle procedure di gestione del sistema.

Il comando **shutdown** era originariamente interattivo, e il superuser controllava le azioni da prendere durante la procedura di cessazione dell'attività. Il comando può ancora essere utilizzato interattivamente, ma le recenti versioni del sistema UNIX prevedono l'opzione **-y**, che istruisce il comando **shutdown** a prendere direttamente tutte le decisioni necessarie:

```
# shutdown -y
```

Questa forma è molto più semplice da utilizzare rispetto a quella che non prevede l'opzione **-y**. Prima di lanciare questo comando, è buona norma di cortesia verificare che gli altri utenti non abbiano in corso qualche attività critica; utilizzate i comandi **who** e **ps -af** per determinare l'attività in corso sul sistema. Inoltre, dovete verificare che nessun lavoro sia in corso di stampa e che nessun trasferimento di dati **uucp** sia in esecuzione, dal momento che tutte queste attività ripartono dall'inizio, dopo l'inizializzazione, se sono state interrotte a seguito della chiusura del sistema.

Quando viene lanciato, **shutdown** avverte tutti gli utenti che, essendo la macchina in fase di spegnimento, devono uscire dal sistema prima che cessi l'attività. La Figura 21.1 mostra alcuni messaggi che il sistema visualizza

```
# cd /
# shutdown -y

Shutdown started. Mon jun 11 20:29:49 MDT 1991

Broadcast Message from root (console) on my_sys Mon Jun 11 20:29:49 MDT 1991
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.

Changing to init state 0 - please wait.
#
INIT: New run level: 0
The system is coming down. Please wait.
System services are now being stopped.
Print services stopped
xntad: received signal 15

The system is down.
Reboot the system now.
```

Figura 21.1 Tipici messaggi alla console durante la sequenza di disattivazione.

alla console durante la fase di disattivazione; una parte è diretta anche a tutti gli utenti collegati e una parte è limitata alla console di sistema. I messaggi di avvertimento che iniziano con "Broadcast Message ..." sono inviati a tutti gli utenti che hanno il collegamento in corso; vengono inviati direttamente ai terminali di tutti gli utenti dal comando **/usr/sbin/wall** (write to all) immediatamente dopo l'inizio del processo di chiusura dell'attività. Il comando **shutdown** rimane quindi inattivo per sessanta secondi prima di continuare le operazioni di chiusura. Gli utenti devono rispondere subito al primo messaggio chiudendo ogni file che avevano aperto, terminando la loro sessione e scollegandosi.

Tutti gli utenti (a eccezione del login id **root** sulla console di sistema) devono immediatamente intraprendere le azioni previste.

Dopo aver visualizzato questi messaggi, il comando **shutdown** disattiva tutti i processi attivi, aggiorna correttamente il disco e infine arresta il sistema operativo. Arrivati a questo punto, è finalmente possibile togliere la tensione alla macchina oppure iniziare una operazione di inizializzazione. Attendete sempre che appaia il messaggio "Reboot the system now", o un messaggio equivalente, prima di togliere realmente la tensione o inizializzare il sistema, in modo da avere la certezza che il processo di arresto dell'attività del sistema si è completato con successo.

Il comando **shutdown** per default attende 60 secondi dopo il messaggio di avvertimento prima di iniziare la sequenza effettiva di arresto del sistema; è possibile modificare l'intervallo di tempo specificando l'opzione **–g** (grace), seguita dal numero di secondi di attesa che volete interporre prima dell'inizio della fase di disattivazione. Questo comando

```
# shutdown –y –g300
```

stabilisce un'attesa di 5 minuti dopo il messaggio di avvertimento. Potete prolungare il periodo per permettere agli utenti di completare l'attività prima che abbia inizio la fase di disattivazione, o diminuirlo per ridurre i tempi di disattivazione quando questo non comporta nessun rischio. In casi estremi potete selezionare **–g10** per ridurre al minimo l'intervallo di attesa; viene accettato anche **–g0**, tuttavia un valore nullo non garantisce un arresto corretto di tutti i processi. Intervalli molto ridotti devono essere usati solo se siete certi che non vi sono attività in corso nella macchina, perché occorre sempre garantire agli utenti la possibilità di concludere le loro sessioni prima della chiusura del sistema. Se ci sono molti terminali attivi il messaggio di avvertimento può non essere visualizzato su tutti i terminali prima che il sistema si arresti.

21.3 La sequenza di inizializzazione

Quando viene messa sotto tensione, la macchina segue una complessa procedura di avvio. Questa sequenza d'inizializzazione può richiedere diversi minuti, dipendendo dall'hardware e dal software installato sulla macchina, e non vi è modo di renderla più rapida. Il processo d'inizializzazione include diverse verifiche di coerenza e spesso cerca di mettere riparo alle anomalie individuate, specialmente quelle relative ai file del disco rigido. La maggior parte delle macchine UNIX prevede procedure interne che abbreviano queste verifiche di errori se la precedente operazione di **shutdown** è stata completata correttamente. Così, la sequenza di caricamento che segue un'improvvisa messa fuori tensione per un qualunque accidente, è molto probabilmente più complessa e completa di un'operazione d'inizializzazione che ha luogo dopo una normale operazione di chiusura. In ogni caso, la sequenza di inizializzazione consente spesso di risolvere i problemi del sistema; per questo motivo, la vostra prima soluzione per malfunzionamenti della macchina deve essere quella di effettuare l'operazione d'inizializzazione.

La Figura 21.2 mostra una sequenza d'inizializzazione come viene visualizzata sulla console di sistema. L'uscita in console varia in relazione al tipo di CPU utilizzata, alla versione di sistema UNIX installata, all'hardware e al software ausiliario presente sulla macchina. L'esempio della figura è preso da una tipica macchina SVR4.

Nella figura non compaiono i messaggi prodotti dal controllo iniziale hardware che la maggior parte dei piccoli calcolatori esegue prima dell'inizializzazione. Questo controllo consiste di solito in una verifica della memoria e comprende una visualizzazionee dell'hardware installato; è familiare alla maggior parte di utenti di PC. Questo controllo è a cura della ROM del sistema hardware e non dipende dal sistema operativo impiegato; alcune versioni software in ROM possono richiedere una password prima di lanciare l'inizializzazione.

Nella figura manca anche la prima parte della procedura d'inizializzazione, eseguita da un *ROM loader* che ha il compito di caricare dal disco la prima parte del sistema operativo. In realtà un software in ROM (detto anche bootstrap firmware) carica un altro programma di caricamento (bootstrap software) che a sua volta ha il compito di caricare il sistema UNIX. Questo programma di caricamento software ausiliario è memorizzato sul disco di sistema, per cui deve essere caricato da hardware e dalla ROM permanente. Come regola, tutti i sistemi operativi, inclusi MS-DOS, OS/2 e il sistema UNIX, mantengono i loro programmi di caricamento nella stessa posizione di disco all'interno della partizione a loro riservata, per cui il programma di caricamento in ROM può sempre rintracciare il bootstrap software. Questo consente di cambiare il sistema operativo cambiando semplicemente la partizione di avvio di default da cui il bootstrap in ROM carica il bootstrap software.

```
Booting the UNIX System...

total real memory   = 8388608
total available mem = 7122944

AT&T UNIX System V/386 Release 4.0 Version 2.0

Copyright (c) 1984,1986,1987,1988,1989,1990 AT&T
Copyright (c) 1987,1988 Microsoft Corp.
All Rights Reserved

Wangtek PC-36/EV-811 cartridge tape controller was found at address 00000300H.
PC586 v2.7 Copyright(c) 1987,1988,1989 Intel Corp., All Rights Reserved
PC586 board 0 was found, Ethernet Address: 00:00:1c:00:02:f0
Node: my_sys
The system is coming up. Please wait.
System V Streams TCP Release 1.0
(c) 1983,1984,1985,1986,1987,1988,1989,1990 AT&T
(c) 1986,1987,1988,1989,1990 Sun Microsystems
(c) 1987,1988,1989,1990 Lachman Associates, Inc. (LAI)
   All Rights Reserved
Print services started
The system is ready.

Welcome to the AT&T 386 UNIX System
System name: my_sys

Console Login:
```

Figura 21.2 Tipica visualizzazione sulla console di una semplice sequenza di inizializzazione.

Dopo che il programma di caricamento software è stato messo in memoria, la ROM gli trasferisce il controllo, ed esso entra in esecuzione. La macchina è a questo punto obbligata a caricare UNIX perché il programma di caricamento software può solo trattare il suo sistema operativo; quando inizia la sua esecuzione, visualizza il messaggio:

```
Booting the UNIX System...
```

e poi carica il kernel di sistema operativo, che è normalmente **/stand/unix**. Premendo un tasto mentre appare il messaggio "Booting..." il programma di caricamento consente di inserire il pathname di un altro kernel da caricare, in questo caso il kernel alternativo deve risiedere nel directory **/stand**; in SVR4 non è possibile introdurre un pathname completo se non sotto **/stand**. Generalmente non si intraprende nessuna azione durante questa fa-

se dell'inizializzazione. Dopo un breve intervallo, il programma di caricamento inizia a caricare il kernel standard.

Il file **/stand/unix** si trova sul disco rigido; contiene il kernel, la parte di sistema UNIX che risiede in memoria reale permanentemente. Il programma di caricamento software legge il kernel dal disco, lo carica nella memoria della macchina e poi gli cede il controllo, in modo che il sistema UNIX cominci a inizializzare se stesso.

Nel corso della sequenza d'inizializzazione può essere visualizzata la dimensione della memoria reale installata nel sistema, determinata dal kernel stesso. Precedenti versioni spesso richiedevano che l'informazione della dimensione di memoria disponibile venisse compilata alla configurazione del sistema, mentre SVR4 può configurarsi autonomamente in base alla memoria effettivamente installata. Se la memoria installata nella macchina viene estesa, la visualizzazione della relativa informazione viene modificata e il sistema UNIX si adatta automaticamente. Se il valore di *total real memory* differisce dalla quantità di memoria fisica che vi risulta installata nella vostra macchina, è sicuramente presente un malfunzionamento hardware che deve essere risolto perché possiate utilizzare la macchina. Il messaggio *total available memory* visualizza la quantità di memoria reale che il sistema rende disponibile ai normali processi, dopo che il sistema UNIX ha occupato quanto gli occorre. In SVR4 la differenza tra questi due valori numerici rappresenta la memoria reale utilizzata dal kernel. Inoltre, il sistema UNIX alloca quantità significative di memoria per i suoi buffer, che operano in una maniera simile a un disco di RAM. Alcune versioni di SVR4 visualizzano il numero di buffer allocati durante la sequenza d'inizializzazione; l'informazione può apparire su video con una forma simile:

```
buffers = 900K
```

Il numero di buffer allocati durante la fase d'inizializzazione costituisce un parametro di configurazione di SVR4 che potete cambiare, se necessario, ma di solito il valore di default soddisfa la maggior parte delle esigenze.

I programmi di utente in esecuzione possono impiegare la memoria disponibile, detratti i buffer. I più moderni sistemi UNIX possono trasferire (*swap* o *page*) segmenti di memoria dalla RAM sul disco quando il sistema ha bisogno di maggiore memoria. In altre parole, possono essere in esecuzione molti più programmi di quanti potreste aspettarvi in base al valore fornito dal messaggio "total available mem". Come al solito, maggiore è la memoria reale disponibile, più efficiente risulta il sistema UNIX, perché la frequenza delle operazioni di trasferimento su disco è limitata.

Successivamente, il sistema UNIX inizializza se stesso e ogni dispositivo hardware collegato. La prima serie di notizie di copyright, indicate in Figura 21.2 proviene da questa inizializzazione dei dispositivi; ulteriori o diverse notizie possono apparire su altre versioni che hanno installato differente software e hardware.

Nel corso del processo d'inizializzazione il kernel esamina i *device driver* associati alle schede aggiuntive installate e fornisce messaggi del tipo "Wangtek..." e "PC586...". Di solito appare un messaggio simile per ogni dispositivo installato.

Quando l'inizializzazione è conclusa, il sistema UNIX è attivo e in esecuzione, anche se devono essere completati ulteriori passi prima che il sistema sia pronto per accettare il collegamento degli utenti. Le linee che compaiono in Figura 21.2 comprese tra la linea "Node:" e l'apparizione del prompt di login, provengono da procedure shell d'inizializzazione che potete facilmente esaminare. Le vedremo tra poco dopo una breve digressione.

In un sistema che funziona correttamente non occorre eseguire nessuna azione tra la fase di accensione e l'apparizione conclusiva del prompt di login. Comunque, se il sistema cade prima della fase d'inizializzazione, appare un altro messaggio:

```
There may be a system dump memory image in the dump device.
Do you want to save it? (y/n)
```

Il contenuto della memoria (memory dump o core image) serve per il debugging del kernel; potete premere **n** per ignorare l'operazione.

21.4 Stati di init

La procedura di inizializzazione di sistema UNIX prevede la possibilità di scegliere lo "stato" in cui lasciare il sistema a inizializzazione conclusa. In altre parole, il sistema può assumere diversi modi di operabilità, detti stati di **init**, da **/sbin/init**, che è il programma responsabile di mantenere il sistema funzionante correttamente. Questi stati sono molto diversi fra loro e il sistema può essere solo in uno di essi in un qualunque momento. Lo stato in cui si trova la macchina viene determinato dalle procedure di inizializzazione e di arresto del sistema. Lo stato più comune è il *modo multiutente*, che supporta la presenza di più utenti contemporaneamente. Un altro stato che ha avuto una notevole importanza storica, ma è ancora utilizzato talvolta nei piccoli sistemi moderni, è detto *modo monoutente*; si tratta di una versione multiprocesso del sistema UNIX, che supporta più processi, ma ammette un solo utente. Quando la macchina lavora in modo monoutente è attiva solo la console di sistema. Oggi il modo monoutente è utilizzato raramente, tuttavia può essere conveniente per utilizzare la macchina impedendo altre attività, come l'accesso in rete o i login remoti. Lo *UNIX User's Manual* raccomanda che le funzioni di inizializzazione della data e dell'ora di sistema siano eseguite solo in modo monoutente, ma nei moderni sistemi UNIX è possibile effettuarle in modo multiutente.

Oltre ai modi multiutente e monoutente esistono altri stati di init; tutti

Tabella 21.1 Stati di **init** per i sistemi UNIX SVR4.

Stato	Funzione
0	Spegnimento della macchina
1	Modo monoutente
2	Modo multiutente
3	Multiutenza con funzionalità della rete
4	Non impiegato
5	Arresto di ROM (o arresto e reinizializzazione)
6	Arresto e reinizializzazione
s	Modo monoutente
S	Modo monoutente con console remota

gli stati possibili vengono identificati con particolari codici numerici, elencati in Tabella 21.1.

Lo stato di init **0** consente di disattivare la macchina e arrestare il sistema UNIX. Il modo monoutente è conosciuto come stato di init **1**, oppure anche come stato **s** o **S**, a seconda che sia attiva la console di sistema oppure un terminale remoto. Il modo multiutente è noto come stato di init **2**; lo stato **3** viene utilizzato nei sistemi SVR4 quando sono attive le funzioni di rete. Una parte delle funzioni di supporto della rete può essere presente anche nello stato 2, ma per utilizzare completamente le funzioni di rete occorre essere in stato 3. Lo stato **4** non viene quasi mai utilizzato, gli stati **5** e **6** su alcune macchine significano rispettivamente "arresto di ROM" e "arresto e reinizializzazione".

Nella maggior parte delle macchine basate su CPU 80x86 lo stato 4 non viene utilizzato, mentre ambedue gli stati 5 e 6 causano la reinizializzazione.

MODIFICA DELLO STATO DI init

Il comando **shutdown** per default mette la macchina nello stato 0, preparando lo spegnimento del sistema; tuttavia con l'argomento **-i** potete esplicitamente scegliere lo stato init fra quelli possibili. Per esempio, il comando:

```
# shutdown -y -g45 -i0
```

disattiva il sistema, mentre

```
# shutdown -y -i6
```

reinizializza la macchina. Molte versioni del comando **shutdown** supportano solo gli stati **init** 0, s, 5, 6.

Normalmente il sistema si trova nello stato 2 a meno che non utilizziate le funzioni di rete, nel qual caso si trova nello stato 3. Il comando **telinit** con-

sente di cambiare dall'uno all'altro di questi stati; per esempio, il comando seguente permette di passare all'uso della rete dallo stato 2:

```
# telinit 3
```

Potete successivamente ritornare nello stato 2 con il comando:

```
# telinit 2
```

21.5 Il file /etc/inittab

Al completamento dell'inizializzazione interna, il sistema lancia il processo demon **/sbin/init** (initialization) che assume il controllo della sequenza d'inizializzazione.

Il processo **init** rimane attivo per tutto il tempo in cui il sistema è in esecuzione, e svolge diverse importanti funzioni, ma la più importante è di assicurare che tutti i processi di sistema siano in esecuzione quando occorrono. Per esempio, quando entrate nel sistema da console, il programma **vtgetty** viene rimpiazzato dal vostro shell e il processo **vtgetty** cessa di esistere; quando uscite dal sistema lo shell cessa di esistere e lascia una porta di terminale non attiva, che non è utilizzabile.

Il processo **init** riconosce quando lo shell termina e rigenera (*respawn*) il programma **vtgetty** per la console, visualizzando un nuovo prompt **Console Login:**. In realtà, il processo **init** viene informato da un segnale di sistema quando lo shell termina e intraprende allora la corrispondente azione. In effetti, **init** ha questo compito anche per molti altri processi di sistema e assicura che siano in esecuzione i processi fondamentali; se **init** stesso dovesse terminare, nessun processo è in grado di rigenerarlo, per cui il sistema gradualmente decadrebbe fino al completo malfunzionamento. Fortunatamente, risulta molto difficile disattivare **init**.

Il processo di inizializzazione **init** riceve istruzioni dal file **/etc/inittab**, che controlla tutti gli stati di init e la riattivazione dei processi quando vengono disattivati. Il file **inittab** è un database tipico di sistema UNIX, le cui linee si compongono di diversi campi separati dal carattere : (due punti). La Figura 21.3 indica un tipico file **inittab** di un piccolo sistema SVR4. Il contenuto di **inittab** differisce notevolmente in relazione al software installato nella macchina, al numero di terminali remoti che possono essere collegati e al tipo di versione di sistema UNIX in uso. Quando il processo **init** inizia, legge ogni linea del file in sequenza e intraprende un'azione dipendente dal contenuto della linea.

Il primo campo di ogni linea è un identificatore che distingue in modo univoco la linea e il secondo campo definisce gli stati di init per cui la linea

```
$ cat /etc/inittab
cc::sysinit:/sbin/chkconsole >/dev/sysmsg 2>&1
ap::sysinit:/sbin/autopush -f /etc/ap/chan.ap
ak::sysinit:/sbin/wsinit 1>/etc/wsinit.err 2>&1
ck::sysinit:/sbin/setclk </dev/console >/dev/sysmsg 2>&1
bchk::sysinit:/sbin/bcheckrc </dev/console >/dev/sysmsg 2>&1
is:2:initdefault:
r0:0:wait:/sbin/rc0    off 1> /dev/sysmsg 2>&1 </dev/console
r1:1:wait:/sbin/rc1    1> /dev/sysmsg 2>&1 </dev/console
r2:23:wait:/sbin/rc2   1> /dev/sysmsg 2>&1 </dev/console
r3:3:wait:/sbin/rc3    1> /dev/sysmsg 2>&1 </dev/console
r5:5:wait:/sbin/rc0 reboot 1> /dev/sysmsg 2>&1 </dev/console
r6:6:wait:/sbin/rc6 reboot 1> /dev/sysmsg 2>&1 </dev/console
sd:0:wait:/sbin/uadmin 2 0 > /dev/sysmsg 2>&1 </dev/console
fw:5:wait:/sbin/uadmin 2 2 > /dev/sysmsg 2>&1 </dev/console
rb:6:wait:/sbin/uadmin 2 1 > /dev/sysmsg 2>&1 </dev/console
li:23:wait:/usr/bin/ln /dev/systty /dev/syscon >/dev/null 2>&1
sc:234:respawn:/usr/lib/saf/sac -t 300
co:12345:respawn:/sbin/vtgetty console console
$
```

Figura 21.3 Tipico file **/etc/inittab** di un piccolo sistema SVR4.

è attiva. Questo campo può indicare più stati, come per esempio 23, per cui la linea risulta attiva negli stati di init 2 e 3. Se questo campo è vuoto, la linea è attiva in tutti gli stati di init, come nel caso delle prime due linee in Figura 21.3.

L'ultimo campo della linea contiene una linea comando che **init** esegue quando la macchina si trova negli stati specificati nel secondo campo. Si tratta di una normale linea di comando di shell che supporta la ridirezione dell'ingresso o/e dell'uscita, come si vede in parecchie linee in Figura 21.3. Poiché **init** crea un processo di shell per eseguire il comando, sono ammesse sia procedure di shell sia programmi eseguibili; se occorre, viene effettuata l'espansione della linea di comando.

AZIONI DI init

Il terzo campo descrive l'azione che **init** deve intraprendere quando si trova in uno degli stati indicati nel secondo campo. Diverse sono le possibilità. Il comando **off** elimina, se esiste, il comando indicato. Il comando **once** fa eseguire il programma quando si attiva lo stato specificato, senza aspettare che l'esecuzione venga completata; il processo **init** continua nel suo lavoro, indipendentemente dal fatto che il comando termini o continui l'esecuzione. Il comando **wait** induce **init** a eseguire il programma quando si attivano gli stati specificati e ad attendere la conclusione del processo prima di conti-

nuare; in questo modo, è possibile fare eseguire da **init** linee di comando secondo un ordine predefinito, perché l'operatore **wait** impone che il comando specificato su una linea concluda l'esecuzione prima che venga eseguita la linea successiva. I comandi specificati con i termini **boot** e **bootwait** vengono eseguiti solo quando **init** legge il file **inittab** nella fase di inizializzazione e non in altri momenti. Questi termini differiscono nel fatto che **bootwait** induce **init** ad aspettare che il comando sia completato, al contrario di **boot**. Il comando **initdefault** ha un particolare significato, non include una linea di comando e induce **init** ad attivare lo stato specificato quando viene lanciato la prima volta. Le linee che contengono **sysinit** vengono eseguite dal sistema al suo inizio e vengono completate prima che appaia il prompt **Console Login:**. Infine, il comando **respawn** induce **init** a lanciare il processo quando sono attivi gli stati indicati e a far ripartire il programma tutte le volte che **init** rileva che il programma non è più in esecuzione. Il secondo campo può contenere molti altri comandi, che vengono però raramente utilizzati nella maggior parte delle macchine.

AZIONI ESEGUITE ALL'INIZIALIZZAZIONE

Sulla base di queste informazioni, potete leggere il file **inittab** per vedere come si svolgono il processo di caricamento e le normali attività del sistema UNIX. Poiché il processo **init** legge il file sequenzialmente, è possibile seguire dettagliatamente la fase di inizializzazione. Nell'esempio di Figura 21.3 le prime linee (**chkconsole**, **autopush**, **wsinit**) iniziano la sequenza d'inizializzazione determinando un corretto ambiente del driver di dispositivo per la console; questi comandi vengono usati nel sistema *streams* per caricare nuovi moduli a seconda del tipo di monitor video in uso. Il comando **setclk** carica l'orologio data-e-ora del sistema UNIX in base all'orologio hardware. La verifica del file system viene effettuata dalla procedura **/sbin/bcheckrc** (boot check run control; in altri contesti può essere specificato **rc**); questa procedura viene impiegata in tutti gli stati di init e viene eseguita solo al momento dell'inizializzazione di sistema; produce la linea "Node" della Figura 21.2. La procedura **bchkrc**: ha il compito di verificare la coerenza della struttura del file system prima che il sistema operativo ne faccia uso; la procedura comunque non ha effetto se un esame preliminare consente di determinare che il file system è sicuramente integro, una volta che il sistema è stato disattivato correttamente. Al contrario, se il sistema fosse stato disattivato a causa di una caduta di tensione o di qualche errore interno, viene eseguito il comando **fsck** (file system check) che consente di ripristinare l'integrità del file system. Vedremo tra breve questo comando.

La successiva linea **initdefault** causa l'inizializzazione della macchina nello stato 2; da qui fino alla fine della sequenza di inizializzazione **init** eseguirà solo linee che contengono 2 nel secondo campo. Se la linea **initdefault** non è presente, **init** richiede di specificare uno stato da console, poiché que-

sta non è una soluzione molto pratica, **initdefault** è quasi sempre presente. Potete configurare l'inizializzazione della macchina nello stato 3 cambiando il 2 nella linea **initdefault** in un 3.

PROCEDURE rc

Le successive linee specificano procedure da eseguire quando sono richiesti stati specifici; essendo tutti file di comandi, ne potete esaminare il contenuto. Generalmente viene eseguito **/sbin/rc2** (run control for state 2) poiché risulta molto frequente attivare lo stato 2; anche in questo caso il processo **init** attende la conclusione della procedura prima di continuare. La procedura **/sbin/rc2** contiene gran parte del codice di inizializzazione che consente di creare il normale ambiente operativo. Il contenuto di questa procedura differisce in base alla versione impiegata, ma nei sistemi SVR4 la sua funzione principale è di esaminare il directory **/etc/rc2.d**, qui rappresentato:

```
$ ls -F /etc/rc2.d
K20nfs          K89xdaemon*     S11uname        S70uucp*  S88smtpd*
K30fumounts     S01MOUNTFSYS*   S15mkdtab*      S75cron*  S89xdaemon*
K40rumounts     S02PRESERVE     S20sysetup*     S75rpc    xS21perf*
K50rfs          S05RMTMPFILES*  S69inet         S801p*
$
```

I vostri file probabilmente differiscono da quelli indicati in questo esempio. La procedura **/sbin/rc2** esegue le procedure che si trovano in quel directory secondo un ordine ben preciso: tutti i file i cui nomi iniziano con la lettera **K** vengono eseguiti per primi, nella sequenza di ordinamento dei loro nomi; poi vengono eseguiti tutti i file i cui nomi iniziano con la lettera **S**, sempre nella sequenza di ordinamento dei nomi. Di solito i nomi con K sono usati per sopprimere (kill) attività, mentre i nomi con S vengono usati per lanciare (start) processi. Seguendo questi criteri i progettisti software possono inserire nuove funzioni che sono attive durante i cambiamenti dello stato di inizializzazione, semplicemente aggiungendo nuovi file al directory **/etc/rc2.d**. Potete esaminare questo directory per conoscere in dettaglio il processo di inizializzazione della macchina. Nell'esempio precedente le procedure **K** scaricano qualunque configurazione di rete esistente, in quanto è possibile che la macchina sia entrata nello stato 3; le procedure **K20nfs** e **K50rfs** svolgono questi compiti. Queste parti non sono presenti in **rc2.d** se le funzioni di rete non sono installate sulla vostra macchina.

Il demon **xntad**, un *name server* usato da X Window System, viene prima ucciso (**K89xdaemon**), in seguito, verso la fine della sequenza (**S89xdaemon**) viene riavviato per poter eseguire **X** in rete.

Le procedure **S**, eseguite successivamente, ricaricano i file system locali (**S01MOUNTFSYS**), puliscono i file nei directory temporanei di sistema (**S02PRESERVE** e **S05RMTMPFILES**), puliscono i directory **uucp**

(**S70uucp**), inizializzano le parti della rete locale attiva sotto lo stato 2 (**S69inet, S75rpc, S88smtpd**), e fanno partire l'esecuzione del programma **cron** (**S75cron**). Poiché vengono eseguiti solo i file con nomi che iniziano con le lettere **K** o **S**, **/etc/rc2** ignora il file **xS21perf**.

Questo stesso schema viene utilizzato quando **init** attiva gli stati 0, 1, 3 e 6. Le procedure **/sbin/rc0**, **/sbin/rc1**, **/sbin/rc3** o **/sbin/rc6** vengono eseguite, se richieste, per scandire rispettivamente i directory **/etc/rc0.d**, **/etc/rc1.d**, **/etc/rc3.d** o **/etc/rc6.d**. Seguendo la sequenza di queste procedure e il contenuto dei directory, potete seguire le azioni che il sistema UNIX effettua quando cambia stato.

Dopo che le procedure **rc** hanno completato tutte le attività nei loro rispettivi directory, l'ambiente di esecuzione della macchina è cambiato. Questo è ciò che avviene in pratica nel passaggio tra gli stati 1, 2 e 3; tuttavia, se l'utente desidera reinizializzare la macchina (stati 0, 5, 6) deve essere richiesta un'effettiva inizializzazione. Questa è la funzione delle linee **uadmin** nella Figura 21.3; il comando **uadmin** comprende diversi argomenti che forzano la reinizializzazione della macchina.

SERVICE ACCESS FACILITY

L'accesso alla macchina da dispositivi remoti, inclusi i terminali e modem come pure connessioni LAN, in SVR4 è supportato dalla nuova funzione SAF, *Service Access Facility*. SAF rimpiazza i processi **getty** e **listen** delle precedenti relèase di sistema UNIX, ed è strutturato in diversi livelli. Al livello più alto si trova il processo **sac** (service access controller), che gestisce diversi servizi, inclusa la sorveglianza delle porte di terminali (una volta dominio di **getty**) e l'accesso alla rete, per **uucp** e altri servizi non relativi alla gestione di login. Il programma **sac**, lanciato tramite **inittab**, consulta una tabella di servizi attivi, o abilitati, e lancia processi demon per ciascun particolare servizio, per sorvegliare le porte nei loro rispettivi domini. Generalmente **sac** lancia **inetd** (il demon Ethernet di base), un nuovo **listen** (network listener) e **ttymon** (terminal port monitor). A loro volta questi servizi lanciano processi specifici (handler), come **login**, quando riscontrano dell'attività da parte di uno dei dispositivi sotto il loro controllo. Questo schema risulta molto più generale ed efficiente della precedente architettura **getty**, dato che un unico processo può gestire diverse porte. Il punto di entrata di questo intero sistema è la singola linea **sac** in **inittab**; potete seguire questa catena di eventi esaminando i processi e i pid padre nell'output di **ps −ef**.

SAF ha spostato il controllo della gestione dei dispositivi dal processo **init** a servizi specifici; in questo modo possono essere risolte in maniera logica e coerente le esigenze specifiche di singoli specifici servizi.

Un caso speciale è quello della console di sistema, che deve essere attiva in tutti gli stati **init**. Poiché **sac** è attivo solo negli stati 2 e 3, il "port monitor" di console deve essere un processo separato controllato direttamente da **init**; **vtgetty** alla fine di **inittab** svolge questo compito e produce il

prompt **Console Login:**. Quando un utente si collega, **vtgetty** scompare e il suo posto viene preso dallo shell di utente; quando l'utente si scollega, scompare lo shell, e **init** rigenera una nuova istanza di **vtgetty**, che si mette in attesa di un nuovo login.

MODIFICHE AL FILE inittab

Nei sistemi SVR4 si presentano poche necessità di apportare modifiche in **inittab**, poiché le operazioni di gestione delle porte sono sotto il controllo di SAF.

In release precedenti **inittab** conteneva singole linee **getty** per ciascuna porta di terminale, linee simili a quella di **vtgetty** in Figura 21.3 . Se nel vostro sistema sono presenti queste linee, le potete disabilitare cambiando in **off** l'azione **respawn**, e viceversa riabilitarle ripristinando **respawn** in luogo di **off**; potete anche modificare il campo velocità di una linea **getty**. Queste modifiche possono essere apportate solo dal superuser che può modificare con un editor il file **inittab** secondo le necessità. Tenete presente che in alcune versioni del sistema le modifiche a **inittab** non permangono quando si installano nuovi pacchetti hardware/software, pertanto, ogni qual volta modificate la configurazione del sistema verificate la presenza delle vostre modifiche.

Il programma **init** legge il file **inittab** una sola volta, quando parte; se apportate delle modifiche al file, queste non hanno effetto fino a quando non informate **init** che il file è stato modificato. Utilizzate a questo scopo il programma **telinit** (tell init):

```
# telinit q
```

L'argomento **q** di **telinit** induce **init** a rileggere il file **inittab** e a intraprendere le azioni indicate dalle modifiche apportate al file dopo che **init** lo ha letto la prima volta. Lo stato **init** non viene modificato.

21.6 Approfondimenti

Le procedure d'inizializzazione e disattivazione per un sistema UNIX possono risultare molto complicate. Di solito quando si verifica qualche anomalia anche leggera, il sistema nel suo complesso non funziona correttamente. Piuttosto che verificare manualmente la procedura d'inizializzazione e la sequenza di eventi necessari per cambiare gli stati di **init**, risulta più semplice ricaricare il software di sistema dai dischi originali. Se avete salvato correttamente i vostri dati, non correte il rischio di perdere informazioni importanti, quando è necessaria un'operazione di reinizializzazione. Fortunatamente, la sequenza di caricamento risulta abbastanza stabile, per cui raramente si verificano malfunzionamenti, a meno che non abbiate modificato il file **inittab** o le procedure eseguite da **rc0**, **rc1**, **rc2** e **rc3**.

PROCEDURA ABBREVIATA DI DISATTIVAZIONE IN SVR4

Le complesse azioni eseguite a inizializzazione del sistema sono indispensabili per la gestione del sistema, e anche se la sequenza è relativamente lunga raramente si sente l'esigenza di abbreviarla. Al contrario, una più breve procedura di arresto può essere talvolta desiderabile. Spesso non vi sono utenti nel sistema, e voi potete comprendere abbastanza i processi del sistema da capire che nessuno può essere malamente influenzato da un improvviso arresto. Se sapete che la macchina si trova in uno stato relativamente inattivo, potete aggirare il comando **shutdown** e disattivare la macchina molto rapidamente. In questo caso, occorrono tre azioni fondamentali. Primo, i buffer del sistema UNIX devono essere scaricati su disco; secondo, occorre smontare ogni altro file system che non sia il file system root; terzo, deve essere presente il segnale di correttezza del disco, in modo che non vengano richieste verifiche e ripristini alla successiva inizializzazione. Questa procedura abbreviata è disponibile solo per i sistemi SVR4 e SVR3, ma non per le versioni meno recenti. Naturalmente la procedura è riservata al superuser.

La prima operazione viene trattata dal comando **sync**, che aggiorna il disco rigido della macchina in modo che sia sincronizzato col contenuto dei buffer. Questo comando viene normalmente eseguito due o tre volte di seguito:

```
# sync
# sync
#
```

La seconda operazione viene effettuata con i comandi **umount** o **umountall**, per smontare tutti i file system che avete montato in aggiunta al file system root, e rimuovere così i file system ausiliari.

La terza azione è di pertinenza del comando **uadmin**, che appare in **inittab**, ma può essere anche eseguito direttamente dal superuser. Il comando accetta due argomenti che specificano le azioni da intraprendere: il primo argomento 2 causa l'arresto del sistema; il secondo, con 1 o 2 causa l'immediata reinizializzazione della macchina con un reset hardware, con 0 (zero) toglie tensione alla macchina. Per esempio:

```
# uadmin 2 0
Reboot the system now.
```

Quando appare il messaggio "Reboot...", potete spegnere la macchina: questa operazione richiede solo due o tre secondi. Gli argomenti di **uadmin** non rappresentano stati di init, ma un codice speciale utilizzato solo da **uadmin**. Questa procedura abbreviata di disattivazione deve essere impiegata con molta attenzione, solo quando siete certi che l'attività del sistema sia estre-

mamente limitata, cioè nel caso in cui non sia collegato nessun utente e non sia in corso nessuna stampa e nessun trasferimento remoto di dati.

IL COMANDO fsck

Alla procedura d'inizializzazione sono generalmente associate alcune importanti azioni di verifica sul file system. Poiché il corretto funzionamento del sistema UNIX dipende dalla correttezza del file system, è previsto uno speciale strumento per la verifica e il recupero di situazioni non congruenti. Si tratta del comando **/sbin/fsck** (file system check), che solo il superuser può utilizzare. Le funzioni di questo comando sono molteplici e il suo impiego è complesso. In fase d'inizializzazione, la procedura **/sbin/bcheckrc** verifica se il flag di validità del file system è stato scritto durante la sequenza di disattivazione. Il comando **shutdown** scrive il flag correttamente, ma questo non accade nel caso di disattivazioni improvvise. Quando il flag esiste, la procedura di caricamento assume che il file system sia corretto, anche se potrebbe non esserlo; il comando **fsck** non viene eseguito e la sequenza d'inizializzazione risulta sensibilmente più rapida. Se il flag non ha il valore corretto, la procedura **bcheckrc** assume che il file system può essere stato danneggiato e automaticamente esegue **fsck**. Quando all'inizializzazione viene eseguito **fsck** sul video della console appaiono ulteriori informazioni.

Il comando **fsck** può anche essere eseguito alla console dal superuser, specificando come argomento il nome del file system da controllare (come in Figura 21.4), tuttavia, in mani inesperte può causare gravi danni al file system. Se occorre effettuare una verifica del file system è preferibile reinizializzare il sistema, a meno che non conosciate molto a fondo l'uso di **fsck**. Se utilizzate manualmente **fsck** è consigliabile comunque impiegarlo su un file system separato, non montato, piuttosto che su un file system montato e attivo. Potete usare **fsck** sia su dischi rigidi che su dischetti; dovete specificare con l'opzione **-F** il tipo di file system, di solito **s5** o **ufs**.

Il comando **fsck** effettua il controllo sul file system in cinque fasi distinte:

1. Verifica le tabelle interne delle dimensioni dei file con le dimensioni effettive dei file su disco.
2. Verifica la congruenza dei pathname per i directory e per i file.
3. Verifica la corretta connessione fra i file e i directory cui appartengono.
4. Verifica la coerenza dei contatori di link dei file e dei rispettivi nomi per garantire i riferimenti corretti.
5. Verifica che tutti i blocchi di disco non utilizzati siano correttamente compresi nell'elenco dei blocchi liberi del file system.

Quando riscontra degli errori, il comando **fsck**, o visualizza un messaggio di errore, o richiede al superuser di disporre del file in qualche modo, can-

```
# fsck -F ufs /dev/rdsk/0s1

** /dev/rdsk/0s1
** Last Mounted on /
** Phase 1 - Check Blocks and Sizes
INCORRECT BLOCK COUNT I=76 (1 should be 0)
CORRECT? y

INCORRECT BLOCK COUNT I=68199 (1 should be 0)
CORRECT? y

** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
UNREF FILE I=68199 OWNER=uucp MODE=0
SIZE=0 MTIME=Jun 22 08:22 1991
CLEAR? y

** Phase 5 - Check Cyl groups
BLK(S) MISSING IN BIT MAPS
SALVAGE? y

10244 files, 160869 used, 134117 free (4661 frags, 16182 blocks,
1.6% fragmentation)
/dev/rdsk/0s1 FILE SYSTEM STATE SET TO OKAY

*** FILE SYSTEM WAS MODIFIED ***
#
```

Figura 21.4 Tipica uscita su console di **fsck**.

cellandolo o ricollegandolo nel file system. Le risposte **y** alle domande che appaiono in Figura 21.4 sono esempi di risposte manuali a queste richieste. Quando **fsck** viene eseguito nella procedura d'inizializzazione, sceglie direttamente le migliori azioni da compiere in ogni caso, senza richiedere il vostro intervento. Generalmente, in fase d'inizializzazione la procedura **fsck** viene eseguita su ogni file system che era attivo al momento della caduta del sistema.

Quando **fsck** rintraccia un file o una parte di file che non ha un collegamento corretto nel file system, lo collega automaticamente al directory **lost+found**. Questo directory è fondamentale per il corretto funzionamento di **fsck**: deve infatti esistere in ogni directory radice di ogni file system che è stato montato. Nel caso in cui la macchina possieda due file system, saranno presenti i directory **/lost+found** e **/usr/lost+found**. Il directory **lost+found** deve essere presente anche nei dischetti che contengono un file system.

Una volta conclusa l'operazione d'inizializzazione, il superuser può esa-

minare questi file o parti di file per verificare se contengono informazioni significative. Un file viene collocato in questo directory quando **fsck** rileva che il file contiene un errore, perciò, dovete esaminate con attenzione i file che si trovano nei directory **lost+found** perché provengono da altri directory del file system. Qualche volta si tratta di parti di cui si sono perduti i collegamenti, con nessun contenuto utile, ma talvolta potrebbero essere file system critici che venivano modificati dal sistema quando la macchina si è spenta. I directory **lost+found** possono crescere sensibilmente, per cui occorre esaminarli di quando in quando, e cancellare i file sconosciuti; tuttavia, trattate i file **lost+found** con molta attenzione perché risulta sempre difficile stabilire il loro contenuto e la locazione del file system da cui provengono.

Capitolo 22

Sicurezza

Poiché UNIX è un sistema multiutente, consente agli utenti di collegarsi in diversi modi e mette a disposizione parecchi strumenti per la comunicazione tra utenti e macchine diverse. Comunque, oggi è ormai abituale che anche persone non autorizzate tentino di entrare nei sistemi, spinte da motivi diversi, che vanno dalla semplice dimostrazione di abilità, al danneggiamento doloso, fino al furto di dati e programmi. In pratica, i numerosi strumenti che il sistema UNIX mette a disposizione per semplificare la comunicazione tra utenti si rivelano un'arma a doppio taglio; dovrete equilibrare la facilità di accesso per "amici" con la prevenzione degli accessi da parte di "nemici".

Il sistema UNIX è stato originariamente sviluppato per soddisfare le esigenze di piccoli gruppi di utenti, che condividevano tutte le risorse della macchina. Non esistevano rigide limitazioni all'accesso ai file e ai comandi di altri utenti, o anche ai dati più riservati utilizzati per gestire il sistema

UNIX stesso; in questo modo, qualunque utente poteva cancellare o modificare i file o anche disattivare definitivamente il sistema.

Nel corso degli anni si è andata imponendo una filosofia diretta a garantire una maggiore sicurezza, e oggi possiamo considerare la release SVR4 adeguatamente protetta. Un esperto gestore di sistema può controllare totalmente l'accesso al sistema, per cui il sistema UNIX, come la maggior parte dei sistemi operativi, dà ampie garanzie sotto l'aspetto della sicurezza. Alcune versioni di SVR4 sono state certificate da parte dell'U.S. Department of Defense con i livelli di sicurezza B2 e B3. Comunque, i problemi sulla sicurezza sono complessi a causa dell'esistenza di una grande varietà di sottosistemi, ciascuno dei quali deve essere adeguatamente protetto per raggiungere la sicurezza ottimale nell'intero sistema.

Questo capitolo esamina il problema della sicurezza degli elaboratori e tratta alcuni strumenti e comandi strettamente legati alla sicurezza del sistema. Mentre acquisterete sempre maggiore confidenza con UNIX, la sicurezza del sistema diventerà un aspetto sempre più importante. Potete prendere misure per evitare accessi al sistema non autorizzati, ma un complesso sistema operativo ha una naturale tendenza nel tempo verso una diminuzione della sicurezza; in altre parole, dovete essere costantemente attenti a individuare punti deboli nella sicurezza e pronti a intervenire per eliminarli.

Se la sicurezza è un problema primario per il vostro sistema, potete richiedere il software AT&T UNIX System V/MLS (multilevel security), che implementa i livelli di sicurezza B2 o B3.

22.1 Una politica di sicurezza

All'interno di un sistema UNIX, o di una rete di sistemi, il supervisore o lo stesso gruppo degli utenti, deve stabilire una politica di sicurezza coerente con cui definire le regole di assegnamento di nuovi identificatori di utente, il tipo di protezione con password richiesta all'interno del sistema, quali diritti di accesso assegnare automaticamente in fase di creazione dei file e che tipo di collegamenti ammettere con le reti LAN e con il mondo esterno. La politica deve essere resa nota ai nuovi utenti e occorre eseguire regolari operazioni di pulizia del file system per garantire la compatibilità con la politica adottata. Se il sistema è relativamente isolato dal mondo esterno e ha un piccolo gruppo di utenti che dimostrano di avere gli stessi interessi, allora il problema della sicurezza passa in secondo piano. Al contrario, se il sistema è di grandi dimensioni, supporta diversi gruppi di utenti, è di pubblico utilizzo, oppure contiene dati molto riservati (o privati), allora occorre una politica della sicurezza molto più restrittiva. Ogni utente deve assumersi la responsabilità di adeguarsi ai criteri di sicurezza, ma un supervisore responsabile può eseguire regolarmente delle verifiche e informare gli utenti dei risultati dei controlli effettuati.

Ai fini della sicurezza è anche molto importante conoscere bene il comportamento del sistema; se il gestore del sistema e i singoli utilizzatori fanno frequente uso di **ps**, **who**, **ls** e altri comandi che forniscono informazioni sul sistema, acquisteranno familiarità con le normali azioni della macchina. In questo modo, gli stati della macchina potranno essere tenuti costantemente sotto controllo, e deviazioni dalla normalità verranno prontamente notate e segnalate al gestore del sistema, per un adeguato intervento.

I problemi della sicurezza del sistema si dividono in quattro categorie: la prima categoria riguarda la protezione dei file e dei dati privati dagli altri utenti; la seconda categoria interessa i file chiave del sistema operativo che devono risultare protetti dal danneggiamento, intenzionale o accidentale; la terza categoria comprende i problemi relativi alla sicurezza fisica della macchina; la quarta si riferisce alla protezione contro espliciti attacchi di "hackers" intenzionati a penetrare o disturbare il sistema. Esamineremo in dettaglio ognuna di queste categorie nei prossimi paragrafi.

22.2 Protezione dei dati da altri utenti

Quando utilizzate una macchina assieme ad altri utenti, dovete stabilire quali e quanti dati condividere con gli altri utenti. Come abbiamo visto nel Capitolo 4, i file nel file system possiedono tre livelli di diritti di accesso: quelli che si riferiscono ai singoli utenti, quelli relativi al gruppo a cui l'utente appartiene e, infine, i diritti di accesso per tutti gli altri utenti della macchina. Di solito, in una piccola macchina dove gli utenti hanno in comune molti interessi, il superuser definisce un unico gruppo a cui gli utenti appartengono. In un ambiente di questo tipo, gli utenti possono condividere alcuni file a livello di gruppo e anche proteggere i file che desiderano mantenere a uso privato. In sistemi di maggiore dimensione, dove esistono diverse comunità di utenti con interessi distinti, si formano invece parecchi gruppi.

Come abbiamo descritto nel Capitolo 4, il comando **ls -l** visualizza i diritti di accesso di un file o di un directory:

```
$ ls -l /etc/inittab
-r--r--r--    1 root     sys      526 Apr 10 19:49 /etc/inittab
$
```

Il file possiede tre insiemi di diritti di accesso per ognuno dei tre diversi livelli di sicurezza: ossia, accesso in lettura, in scrittura e in esecuzione per il proprietario, per il gruppo a cui l'utente appartiene e per tutti gli altri utenti. Ogni file è di proprietà di un identificatore di login ed è anche di proprietà di un gruppo. Nel precedente esempio, il proprietario del file è **root**, l'identificatore di gruppo è **sys** e il file risulta leggibile da parte di tutti, ma non è accessibile in scrittura o in esecuzione da parte di nessuno.

Quando create un nuovo file, ne siete il proprietario, e al file viene assegnato l'identificatore del gruppo a cui appartenete. Se siete proprietario di un file potete utilizzare il comando **chown** per cedere la proprietà del file ad altri utenti, oppure utilizzare il comando **chgrp** per cambiare l'identificatore di gruppo del file. Generalmente non è possibile riacquisire la proprietà del file una volta che l'avete ceduta ad altri, a meno che il file non sia accessibile in lettura per cui vi sia possibile farne una copia di vostra proprietà. In ogni caso, solo il superuser può cambiare i diritti di accesso di ogni file che si trova nel sistema.

DIRITTI DI ACCESSO ASSEGNATI AI NUOVI FILE

Una volta creato un file, dovreste verificare i diritti di accesso con il comando **ls -l**, per assicurarsi che corrispondano a quelli richiesti. Dovete stabilire se sia accettabile che ogni utente che appartiene al vostro gruppo abbia accesso ai dati, o se ogni altro utente debba avere il permesso di leggere o scrivere il file; in altre parole, dovete determinare i diritti di accesso dei vostri nuovi file in modo da soddisfare precisi requisiti.

Il sistema UNIX assegna al creatore del file la proprietà del file, e assegna il file al gruppo a cui il creatore appartiene. Questi principi non possono essere cambiati, potete invece inizializzare una variabile di sistema associata con il vostro identificatore di login, che determina i diritti di accesso di un file senza il vostro intervento esplicito. Questa variabile di sistema è denominata **umask** (user file-creation mask) ed è accessibile con il comando **umask**. Potete determinare il valore corrente di **umask** lanciando il corrispondente comando senza argomenti:

```
$ umask
000
$
```

Il risultato consiste di tre cifre ottali che si riferiscono ai diritti di accesso del proprietario, del gruppo e degli altri utenti, da sinistra a destra. Questo numero viene denominato *mask* (maschera) in quanto ogni cifra viene sottratta al diritto di accesso che a livello di sistema viene assegnato ai nuovi file. Normalmente questo diritto di accesso a livello di sistema è **-rw-rw-rw-**, ma può differire in alcuni sistemi e programmi. Poiché la variabile **umask** dell'utente viene sottratta a questo valore di default, non potete attivare con **umask** i diritti di accesso che sono normalmente disattivati, ma potete disattivare i diritti di accesso che sono normalmente attivati. Ovviamente, se siete proprietari di un file, potete utilizzare il comando **chmod** per attivare esplicitamente i diritti di accesso.

Ogni cifra ottale nella variabile **umask** contiene tre bit binari che annullano i diritti di accesso: la cifra 1 annulla il diritto di accesso all'esecuzione,

la cifra 2 annulla il diritto di accesso in scrittura e la cifra 4 annulla il diritto di accesso in lettura; la cifra 0 conferma il valore di default. La variabile **umask** dell'esempio precedente (000) significa che non viene cambiato nessuno dei valori di default, mentre, se la variabile **umask** avesse valore 022, allora il gruppo di utenti e gli altri utenti non potrebbero accedere in scrittura ai file creati. Per esempio:

```
$ umask
000
$ > def.perm
$ ls -l def.perm
-rw-rw-rw-          1 giorgio     utenti     0 May 10 14:57 def.perm
$ umask 022
$ umask
022
$ > no.write
$ ls -l no.write
-rw-r--r--          1 giorgio     utenti     0 May 10 14:58 no.write
$ umask 777
$ > no.perm
$ ls -l no.perm
----------          1 giorgio     utenti     0 May 10 14:58 no.perm
$
```

La variabile **umask** di default contiene 000, e i file vengono creati con i diritti di accesso di default. Quando modificate il valore a 022, gli altri utenti non possono accedere in scrittura ai file da voi creati. Un valore di 777 annulla i diritti di accesso per tutti gli utenti, per cui nessuno può accedere all'ultimo file creato!

Per modificare la vostra variabile **umask** potete usare il comando **umask** con il codice ottale come argomento. Questa modifica non permane dopo che siete usciti dal sistema; se volete modificare permanentemente la variabile **umask** dovete includere il comando **umask** nel file privato **.profile**. In questo modo non dovrete intervenire per assegnare i diritti di accesso a ogni file che create.

22.3 Come crittografare un file

Potete proteggere maggiormente i dati che necessitano di trattamenti speciali crittografando i file. Gran parte dei sistemi UNIX negli Stati Uniti fornisce strumenti per crittografare i file in accordo con la password che specificate; solo introducendo di nuovo la password valida, è possibile accedere correttamente a questi file. La crittografia dei file non è disponibile nelle versioni di sistema UNIX vendute al di fuori degli Stati Uniti.

Gli editor come **ed**, **vi** ed **emacs** consentono di creare e aggiornare file

crittografati. In pratica, potete richiedere al vostro editor di decifrare il file in fase di caricamento e di crittografarlo di nuovo quando lo scarica su disco. L'opzione **-x** indica che il file in input è crittografato:

```
$ vi -x file.critt
Key:
```

L'editor chiede di introdurre la password o chiave di crittografia, con le stesse modalità utilizzate per la password quando vi collegate al sistema; come la password, la chiave non appare su video. Quando cifrate un file qualsiasi password è accettabile; naturalmente dovrete usare la stessa password per decifrare il file. Se inserite la password correttamente, il file viene decifrato e reso disponibile per l'editor; quando riscrivete il file su disco durante o dopo la sessione di editor, il file viene crittografato di nuovo. Potete utilizzare questa procedura per creare un nuovo file o aggiornare un file esistente.

Le versioni del sistema UNIX vendute negli Stati Uniti forniscono anche un programma filtro che esegue le operazioni di crittografia e decrittografia. Il comando **crypt** legge lo standard input e scrive sullo standard output. Se l'ingresso è un semplice file di testo in chiaro, l'uscita viene crittografata; se il file in ingresso è crittografato, allora l'uscita viene decrittografata. Come accade per le procedure di editor, **crypt** chiede di specificare una password:

```
$ cat testo | crypt
Enter key:
```

Come al solito, la password non viene visualizzata.

Sfortunatamente, l'algoritmo con cui i file vengono crittografati in ambiente di sistema UNIX è troppo ben conosciuto ed esistono programmi che sono in grado di decrittografare il contenuto dei file crittografati. Per questi motivi, non è conveniente fare eccessivo affidamento sulla cifratura dei file, specialmente se siete in un ambiente dove si possono temere intrusioni ostili. Appositi programmi di decifrazione analizzano le frequenze dei caratteri contenuti nei file crittografati e le confrontano con le frequenze dei caratteri nei testi in lingua inglese. Per contrastare l'azione di questi programmi, potete cambiare le frequenze dei caratteri nel file di testo in chiaro, prima di crittografarlo, con un altro filtro come **pack** o **compress**; per esempio:

```
$ compress testo
$ crypt < testo.Z > out.file
```

Il file compresso non può essere analizzato da nessun programma di decifrazione conosciuto. Naturalmente, quando decrittografate il file dovete ricordarvi di disimpaccarlo:

```
$ crypt < out.file > testo.Z
$ uncompress testo.Z
$
```

Dovete impaccare il file prima di crittografare e disimpaccarlo dopo averlo decrittografato.

Naturalmente, la più sicura procedura di protezione dei file consiste nel salvataggio del file su dischetto o su nastro, cancellazione del file dalla macchina e conservazione del supporto magnetico sotto la vostra personale cura.

22.4 Identificatori e password di login

Il cuore dello schema di protezione del sistema UNIX consiste nell'identificatore di login e nella password dei singoli utenti. Se si riesce a prevenire l'accesso al sistema ai potenziali malintenzionati si evitano eventuali danni, ma la protezione offerta dalla password si dimostra così limitata in molte macchine che anche un intruso inesperto può entrare nello shell. Ogni utente ha la responsabilità di custodire gelosamente le password e di cambiarle regolarmente.

Molti identificatori di utente su piccoli sistemi non prevedono affatto password oppure le password sono talmente simili all'identificatore di login da risultare inefficaci in termini di sicurezza. Sfortunatamente, la maggior parte degli utenti non intende impiegare per le password nomi difficili da ricordare, come sarebbe necessario per renderle effettivamente sicure, e quindi è probabile che le password col passare del tempo vengano scoperte. Ogni utente dovrebbe essere spinto a possedere una password, e la password dovrebbe avere un limitato periodo di validità, per obbligare l'utente a cambiarla con regolarità. Dal momento che la password viene memorizzata in forma crittografata, neanche il gestore di sistema può conoscerla; fortunatamente, la tecnica di decifrazione sopra accennata, basata sui calcoli della frequenza delle lettere, non può essere applicata su un testo così breve come una password.

Lo strumento che consente di cambiare la password è il comando **passwd**. Come abbiamo detto nel Capitolo 2, questo comando richiede di indicare la password corrente e poi di specificare due volte la nuova password prima che la modifica abbia effetto.

Nella maggior parte dei sistemi UNIX occorre seguire precise regole che definiscono la validità della password, e anche se queste regole non sono imposte dal software di sistema, forniscono valide indicazioni per creare la password. Una buona password deve avere una lunghezza non inferiore a sei caratteri, di cui almeno uno (preferibilmente due) è un carattere numerico o altro carattere non alfabetico; è consigliabile mescolare caratteri minuscoli e maiuscoli e adottare una parola inusuale o una sequenza di carat-

teri imprevedibile. Alcuni esempi di password non accettabili per la loro banalità sono: il vostro identificatore di login, il vostro nome, il nome di vostro figlio, il numero della vostra stanza, il numero telefonico, il vostro segno zodiacale, il vostro indirizzo e simili.

22.5 Storia di login

Molte delle versioni di SVR4 indicano l'ultimo momento in cui il vostro identificatore di login è stato utilizzato. Queste informazioni appaiono su video quando entrate nel sistema:

```
login: giorgio
Password:

Login last used: Wed Oct 24 15:11:02 1991

$
```

Questo messaggio ha lo scopo di mettervi sull'avviso se altri utenti utilizzano il vostro login id. Se l'indicazione non coincide con il momento del vostro ultimo collegamento, allora qualcuno utilizza indebitamente il vostro identificatore e dovete immediatamente modificare la vostra password.

Questa caratteristica è gestita dal programma **login** in fase di verifica della password, mantenendo nel vostro home directory il file di lunghezza zero **.lastlogin**. La data e l'ora relative all'ultimo collegamento con il sistema coincidono con la data di modifica di questo file. Il file **.lastlogin** è di proprietà del sistema, non del singolo utente e i diritti di accesso ne rendono difficile ogni sua eventuale modifica, come potete vedere:

```
$ ls -l $HOME/.lastlogin
-r---------    1 root     sys        0 Oct 28 15:11 .lastlogin
```

Quella che abbiamo descritto non risulta una misura di protezione di grande portata, ma consente di mettervi sull'avviso se altri utenti conoscono il vostro identificatore di login.

22.6 Il superuser

I normali utenti possono utilizzare solamente i file e i dati privati, e quelli del gruppo a cui appartengono. Tuttavia, tutte le macchine UNIX prevedono l'identificatore di login **root** che consente di accedere in lettura, scrittura ed esecuzione a tutti i file e i directory. Questo utente viene denominato **superuser** (super permissions). Inoltre, potete anche utilizzare il comando

su (superuser) per acquisire lo stato di superuser senza uscire dal sistema e collegarvi di nuovo come **root**:

```
$ /sbin/su
Password:
#
```

Il Capitolo 12 tratta dettagliatamente il superuser.

22.7 Il file delle password

L'informazione critica che controlla i login di utente è mantenuta in un semplice database denominato **/etc/passwd**. Questo file è accessibile in lettura da parte di tutti gli utenti, ma non è accessibile in scrittura:

```
$ ls -l /etc/passwd
-r--r--r--    1 root     root      526 Apr 10 19:49 /etc/passwd
$
```

Questi diritti di accesso debbono essere rigorosamente mantenuti invariati, perché se tutti possono accedere al file in scrittura, allora la sicurezza del sistema può essere violata con estrema facilità.

A ogni utente corrisponde una linea nel file delle password; sono presenti anche diversi identificatori di login standard a livello di sistema necessari per il funzionamento corretto del sistema. La Figura 22.1 mostra il contenuto di un tipico file **passwd** per un sistema SVR4. Ogni linea del file si compone di più campi, delimitati dal carattere : (due punti). Il primo campo della linea contiene l'identificatore di login dell'utente il secondo campo contiene **x** come un segnaposto per la password. Il terzo campo contiene la rappresentazione numerica dell'identificatore di utente e il quarto campo contiene la rappresentazione numerica dell'identificatore di gruppo. Questi due campi, uniti ai diritti di accesso dei file, stabiliscono chi può accedere ai file nel sistema. Il quinto campo indica una nota di commento che di solito contiene il nome e l'indirizzo dell'utente. Il campo successivo contiene il directory preferenziale (home) dell'utente, infine, l'ultimo campo memorizza il pathname completo dello shell di login dell'utente. Se l'ultimo campo è vuoto, il pathname di default dello shell di login è **/sbin/sh**.

Nelle versioni precedenti del sistema UNIX, il secondo campo conteneva realmente la password crittografata di ciascun utente; mentre le versioni SVR3 hanno introdotto nel sistema un secondo file che contiene la password crittografata e altre informazioni attinenti. Si tratta del file **/etc/shadow**, che deve essere accessibile in lettura solo da parte dell'utente **root**:

```
$ ls -l /etc/shadow
-r---------    1 root     root      187 Apr 10 19:49 /etc/shadow
$
```

```
$ cat /etc/passwd
root:x:0:1:0000-Admin(0000):/:
daemon:x:1:1:0000-Admin(0000):/:
bin:x:2:2:0000-Admin(0000):/usr/bin:
sys:x:3:3:0000-Admin(0000):/:
adm:x:4:4:0000-Admin(0000):/var/adm:
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp:
lp:x:7:8:0000-LP(0000):/home/lp:/sbin/sh
nuucp:x:10:10:0000-uucp(0000):/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
sync:x:67:1:0000-Admin(0000):/:/usr/bin/sync
install:x:101:1:Initial Login:/home/install:
sysadm:x:0:0:general system administration:/usr/admin:/usr/sbin/sysadm
vmsys:x:102:100:FACE executables:/home/vmsys:/sbin/sh
oasys:x:103:1:Object Architecture Files:/home/oasys:/sbin/sh
leo:x:104:1:Leo:/home/leo:/usr/bin/ksh
pat:x:105:1:Pat:/home/pat:
giorgio:x:106:1:Giorgio:/home/giorgio:/usr/bin/ksh
$
```

Figura 22.1 Un tipico file **/etc/passwd**.

Il file **/etc/shadow** contiene l'identificatore di login di utente, la password crittografata, un codice numerico che descrive quando la password è stata cambiata e il numero, minimo e massimo, di giorni richiesti tra due modifiche della password:

```
# cat /etc/shadow
root:k2kLQBmtvd3Xw:7337::::::
daemon:NONE:7337::::::
bin:NONE:7337::::::
sys:NONE:7337::::::
adm:NONE:7337::::::
uucp:NONE:7337::::::
lp:NONE:7337::::::
nuucp::7388::::::
listen:np:7337::::::
sync:NONE:7337::::::
install:q20ahn0ya9Dwg:7337::::::
sysadm:SYfb818vwtwgA:7388::::::
vmsys:*LK*:::::::
oasys:*LK*:::::::
leo:lftgKW03qFGwe:7388:1:1000:900:::
pat:wouOQWqw34fgeu:7388:1:1000:900:::
giorgio:qUWmdk4ROSqwe:7394:1:1000:900:::
#
```

Ogni linea del file **/etc/shadow** corrisponde a una linea nel file **/etc/passwd**. Quando **/etc/shadow** è presente, il campo che specifica la password in

/etc/passwd viene sostituito da un carattere **x**, altrimenti il secondo campo in **/etc/passwd** corrisponde al secondo campo indicato nel precedente esempio di **/etc/shadow**.

Il file **/etc/shadow** viene creato dal comando **pwconv**, che legge **/etc/passwd** per ricavare le necessarie informazioni. Ogni volta che cambiate manualmente **/etc/passwd**, dovete immediatamente eseguire **pwconv** per essere certi che i cambiamenti vengano riportati anche nel file **/etc/shadow**. Dal momento che i file **passwd** e **shadow** devono essere accessibili solo in lettura, nessuno tranne il superuser è in grado di modificarli.

UN ESEMPIO DI FILE /etc/passwd

La Figura 22.1 mostra il contenuto di un tipico file **/etc/passwd** di una macchina SVR4. Il sistema nella configurazione di default include diversi identificatori di login, ancora prima che vengano inclusi identificatori di utenti. Questi login id di sistema sono indispensabili per il corretto funzionamento del sistema; non possono essere cancellati o modificati senza causare danno al sistema. In nessun login che non sia relativo a un utente reale devono essere modificati il login id, l'identificatore di utente o di gruppo, l'home directory e neanche lo shell di default.

Il login **root** identifica il supervisore, ha un identificatore di utente zero e il suo home directory è il directory radice / (barra). Il login **root** deve sempre avere una password; la mancanza di questa protezione per questo login rappresenta la più grave violazione di sicurezza in ambiente UNIX, dal momento che l'utente root può accedere a qualsiasi risorsa del sistema. Gli altri pochi identificatori di utente sono utilizzati per i restanti strumenti di gestione. In sistemi di grandi dimensioni, questi identificatori sono utilizzati dalle persone che hanno la responsabilità della gestione dei diversi sottosistemi. Normalmente su una piccola macchina con un unico gestore di sistema, potete disabilitare questi identificatori di login scrivendo la stringa NONE o qualche altra stringa di testo in chiaro nel campo password del file **/etc/shadow**. Dal momento che la password contenuta in **/etc/shadow** viene decrittografata, una stringa di testo in chiaro genera una stringa senza senso che gli utenti non sono in grado di scoprire quando cercano di collegarsi.

Il superuser svolgerà anche le funzioni degli identificatori di login disabilitati, cioè degli identificatori che nel file **/etc/shadow** del precedente esempio hanno password NONE o *LK*.

Poiché il software addizionale aggiunge spesso anche login id di gestione, queste funzioni di login possono differire in relazione alla caratterizzazionee dell'hardware della macchina.

Il login **nuucp** viene utilizzato da macchine remote quando si collegano per trasferire messaggi; lo shell per questo login è **/usr/lib/uucp/uucico**, cioè il programma che rende possibile il trasferimento di dati per il sottosistema **uucp**. Dal momento che non si tratta di un normale shell, e dal momento

che il programma **uucico** garantisce la massima sicurezza, non occorre specificare nessuna password per il login **nuucp**. L'identificatore di login **uucp** riguarda invece il gestore del sottosistema **uucp** e deve essere disabilitato.

Alla fine della Figura 22.1 sono elencati tre utenti individuali (leo, pat e giorgio) che nel terzo campo del file **/etc/shadow** hanno un proprio numero di identificazione, pur appartenendo allo stesso gruppo 1, come indicato nel quarto campo. Questi utenti devono avere tutti la password in **/etc/shadow**; si può notare che pat utilizza lo shell di default, mentre giorgio e leo hanno uno shell differente quando entrano nel sistema.

In conclusione, ripetiamo che tutti gli identificatori di utente contenuti nel file **/etc/passwd** devono possedere una password a eccezione di **nuucp**; si tratterà di password crittografate, nel caso di utenti reali e del superuser, oppure di password di testo in chiaro per tutte le altre sessioni.

ELIMINAZIONE E INSERIMENTO DI NUOVI UTENTI

Generalmente aggiungere o rimuovere utenti dal sistema è facile, comunque queste azioni sono riservate al superuser. Per l'esecuzione di queste operazioni sono previsti gli appositi strumenti **useradd**, **userdel** e **usermod**; l'uso di questi mezzi è preferibile all'esecuzione di modifiche manuali, o all'utilizzazione del comando obsoleto **passmgmt**. Se modificate manualmente file di controllo degli utenti, come **/etc/passwd**, dovete avere cura di conservare la proprietà e i diritti originali dei file.

Come già detto, ciascun utente ha un identificatore di login, un identificatore numerico di utente e di gruppo, un home directory, uno shell di default e una password. Quando create un nuovo utente dovete specificare i valori di tutti questi elementi; il comando **useradd**, dedicato all'immissione di nuovi utenti nel sistema, prevede valori di default per molti di questi dati, ma per altri dovete necessariamente provvedere voi a fornire i valori. L'opzione **-D** (display) di **useradd** consente di conoscere i valori di default, per esempio:

```
# useradd -D
group=other,1 basedir=/home skel=/etc/skel
shell=/sbin/sh inactive=0 expire=
#
```

Questo esempio mostra i normali default assegnati ai nuovi utenti in assenza di scelte specifiche. I nuovi utenti apparterranno al gruppo **other**, con identificatore di gruppo (gid) 1; i loro directory preferenziali (home directory) saranno sotto il directory **/home**; il loro shell di default sarà lo shell standard. Le voci *inactive* ed *expire* sono relative ai parametri di durata della validità della password, trattati più oltre in questo capitolo.

I valori di default sono contenuti nel file **/etc/skel**; possono essere modifi-

cati mediante l'opzione **-D** di **useradd** e un argomento in linea di comando per il valore di ciascun default. In piccoli sistemi questa procedura non è conveniente.

Per aggiungere un nuovo utente al sistema specificate l'identificatore di login dell'utente nella linea di comando di **useradd**:

```
# useradd giorgio
#
```

Questo comando crea gli elementi dell'utente nei file **/etc/passwd** ed **/etc/shadow**, usando i valori di default, e crea anche l'home directory per l'utente.

Se non ci sono problemi, non ci sono messaggi. Se l'identificatore di login esiste già, **useradd** lo segnala, come nell'esempio:

```
# useradd giorgio
UX: useradd: ERROR: giorgio is already in use. Choose another.
#
```

Ogni identificatore di login deve essere univoco, mentre l'home directory, l'identificatore numerico di utente e altre informazioni possono essere condivisi da vari utenti, anche se questo non è di solito un buon criterio. La contabilizzazione dell'utilizzo del sistema risulta notevolmente facilitata se ogni utente ha un suo proprio ambiente. L'unica eccezione è l'identificatore di gruppo, che per definizione è comune a diversi utenti.

Quando create un utente dovreste possibilmente includere un commento contenente il nome dell'utilizzatore, la locazione e ogni altra informazione utile; questo commento viene inserito nell'elemento di **/etc/passwd** dell'utente e in un grande sistema può essere utile per rintracciare gli utilizzatori. Per includere il commento dovete specificare nella linea di comando di **useradd** l'opzione **-c** seguita dal commento protetto fra virgolette, come nell'esempio:

```
# useradd -c "Giorgio, 555-1234, stanza 320" giorgio
#
```

Ulteriori argomenti nella linea di comando di **useradd** consentono di rimpiazzare i valori di default; con **-u** potete specificare un identificatore di utente, con **-g** un identificatore di gruppo, con **-d** un home directory, con **-s** uno shell, con ancora altre opzioni potete modificare altri parametri relativi al login. Con l'opzione **-e** (expire) potete stabilire una data di scadenza di un login; in questo modo potete creare un login temporaneo non più valido oltre una determinata data. Per esempio:

```
# useradd -e "12/31/91" -c "Login temporaneo per progetto ALFA" giorgio
#
```

Questa opzione accetta diversi altri formati per la data.

22.8 Scadenza della password

Quando create un identificatore di login usando **useradd**, l'elemento creato in **/etc/shadow** ha una password bloccata, in modo che l'utente non è ancora abilitato ad accedere al sistema. Perché l'utente sia abilitato ad accedere al sistema dovete abilitare il login definendo alcuni parametri della durata di validità della password.

Il software di gestione delle password consente (ma prossimamente imporrà!) al gestore del sistema di stabilire una procedura di durata di validità della password che obbliga tutti gli utilizzatori a sostituire periodicamente le loro password. Quando viene determinata una password, viene avviato un temporizzatore, e quando l'intervallo stabilito è trascorso, la password viene scartata e l'utente obbligato a determinarne una nuova. Quando crea un nuovo utente, il superuser abilita la durata di validità della password usando il comando **passwd**; la procedura inoltre annulla lo stato di bloccaggio della password assegnata al nuovo utente. Queste possibilità sono limitate al superuser, e sono supportate da varie opzioni nella linea di comando di **passwd**.

Per stabilire il numero massimo di giorni di validità di una password senza essere cambiata, usate l'opzione **-x** seguita da un numero di giorni, come nell'esempio:

```
# passwd -x 120 giorgio
#
```

Per stabilire il numero minimo di giorni che una password deve essere attiva prima di poter essere cambiata, usate l'opzione **-n**, come nell'esempio:

```
# passwd -n 7 giorgio
#
```

Queste opzioni ritornano senza messaggi se non ci sono errori. Dovete stabilire l'intervallo massimo prima dell'intervallo minimo, oppure stabilirli con ambedue le opzioni nella stessa linea di comando.

Il comando **passwd** con l'opzione **-s** (status) fornisce un rapporto dello stato della password di un utente, per esempio:

```
# passwd -s giorgio
giorgio LK 00/00/00 7 120
#
```

Come potete notare, dopo l'identificatore dell'utente viene indicato lo stato: LK significa bloccato (locked), PS significa correttamente protetto con password, NP significa assenza di password; successivamente viene indicata la data dell'ultima sostituzione di password (00/00/00 se l'utente è nuovo); e infine gli intervalli minimo e massimo sopra descritti.

Dopo avere determinato i parametri di validità della password potete rimuovere lo stato di bloccaggio, con l'opzione **-d** (delete), come segue:

```
# passwd -d giorgio
#
```

L'opzione **-d** cancella anche la password di un normale utente; in questo caso deve essere usata con molto ritegno, perché un login senza password rappresenta un rischio per la sicurezza.

Nessuno è in grado di creare una nuova password per un altro utente, cosicché quando viene creato un nuovo utente la procedura migliore che il gestore del sistema può seguire è quella di entrare immediatamente nel sistema usando il nuovo identificatore di login e stabilire manualmente una password iniziale, quindi rientrare di nuovo come **root** ed eseguire il comando:

```
# passwd -f giorgio
#
```

che con l'opzione **-f** (force) fa scadere immediatamente la password appena creata. In questo modo, quando l'utente entra in login per la prima volta, userà la password iniziale, ma verrà obbligato a cambiarla immediatamente, come nell'esempio:

```
login: giorgio
Password:
Your password has expired. Choose a new one
Old password:
#
```

Questo messaggio appare anche agli utenti abituali quando è trascorso il tempo massimo ammesso per la loro password.

In generale, la scadenza delle password fornisce un aiuto al mantenimento della sicurezza, perché le password degli utenti col tempo diventano note pubblicamente, e l'obbligo di cambiarle regolarmente può ridurre questo rischio. L'obbligo per gli utenti di cambiare password due o tre volte in un anno è di solito sufficiente; tuttavia, una debolezza di questo sistema è l'abitudine di alternare tra due password preferite che sono una banale permutazione l'una dell'altra. Questa abitudine deve essere scoraggiata, per i motivi finora discussi.

22.9 Cancellazione di un utente

Quando un utente cessa, per qualunque motivo, di utilizzare il sistema, il gestore dovrebbe prendere immediatamente l'iniziativa di cancellare l'accesso dell'utente al sistema. Questo può essere ottenuto in due modi, a seconda

delle esigenze. In un caso, potete bloccare la password dell'utente, senza tuttavia cancellare né l'home directory dell'utente né l'identificatore di login.

Questa soluzione impedisce all'utente di utilizzare il sistema, ma successivamente potete sbloccare il login senza perdita di dati. Potete utilizzare il comando **passwd** con l'opzione **-l** (lock):

```
# passwd -l giorgio
#
```

In alternativa, potete cancellare definitivamente l'utente dal sistema, usando il comando **userdel** (user delete), come nell'esempio:

```
# userdel giorgio
#
```

Questo comando non restituisce messaggi se non ci sono errori, comunque non può essere usato se l'utente ha una sessione di login in corso. Il comando **userdel** non cancella l'home directory dell'utente, dimodoché, se occorre, potete conservare i dati che vi sono contenuti. Per cancellare l'utente e l'home directory e tutti i subdirectory relativi, potete usare l'opzione **-r** (remove), come nell'esempio:

```
# userdel -r giorgio
#
```

Con il comando **usermod** è possibile modificare i parametri associati a un utente esistente, evitando di cancellare e quindi ridefinire l'utente.

Dovete aver cura di aggiornare costantemente il sistema quando qualche utente cessa di usarlo. Spesso, soprattutto nei sistemi di grande dimensione con centinaia di utenti, quando un utente lascia il sistema, per una qualsiasi ragione, il gestore non ne disabilita immediatamente l'identificatore di login. Una situazione di questo tipo comporta molti rischi per la sicurezza, dal momento che gli ex utenti potrebbero avere dei motivi di rancore e potrebbero vendicarsi causando seri danni al sistema. Dovreste quindi esaminare periodicamente con particolare cura il file **passwd** e cancellare gli utenti che non sono più attivi da lungo tempo.

22.10 Creazione di gruppi

Quando aggiungete un nuovo utente a un identificatore di gruppo già esistente, o al gruppo di default, non occorre eseguire nessuna ulteriore attività dopo avere usato i comandi **useradd** e **passwd**. Al contrario, se intendete definire un nuovo gruppo per l'utente, o se volete far apparire l'utente in più di un gruppo, dovete aggiungere quel gruppo al file **/etc/group**. Potete

usare i comandi **addgrp** (add to group) e **delgrp** (delete from group) che consentono di creare nuovi gruppi. Esaminiamo il file **/etc/group** per conoscere i gruppi in uso, come nell'esempio:

```
$ cat /etc/group
root::0:root
other::1:leo,pat,giorgio
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
mail::6:root
tty::7:root,tty
lp::8:root,lp
daemon::12:root,daemon
uucp::5:root,uucp
vm::100:
$
```

Questo file contiene una linea per ogni gruppo definito nel sistema con i campi separati dal carattere due punti. Il primo campo della linea contiene il nome del gruppo e il terzo campo l'identificatore numerico del gruppo, che compaiono nei dati prodotti dal comando **ls -l**. Il secondo campo contiene una password crittografata, che l'utente deve specificare per cambiare i gruppi con il comando **newgrp** (new group), se gli utenti sono abilitati a usare questo comando per cambiare gruppo. Generalmente agli utenti dovrebbe essere impedito di cambiare il gruppo.

L'ultimo campo contiene l'elenco degli identificatori degli utenti che appartengono a quel gruppo, separati da una virgola. Un utente piò appartenere a più gruppi e in questo modo avere accesso ai file che appartengono a diversi gruppi. Per ridistribuire gli utenti fra i vari gruppi, dovete modificare manualmente con editor il file **/etc/group**, limitando le modifiche all'ultimo campo. A tutti gli utenti deve essere concesso l'accesso in lettura al file **/etc/group**, ma nessuno deve avere l'accesso in esecuzione o in scrittura.

22.11 Lo shell rsh

Lo shell standard consente agli utenti di spostarsi nel file system, di eseguire un elevato numero di comandi, di aggiornare la variabile PATH, ecc. Comunque, in quasi tutti i sistemi UNIX è disponibile uno shell ausiliario denominato **rsh** (restricted shell), che concide con la versione eseguibile del normale shell, ma esiste nel sistema sotto due diversi nomi e si comporta diversamente a seconda del nome con cui è stata invocata. In SVR4 lo shell **rsh** è di solito collocato nel directory **/usr/lib**; notate che non si tratta dello shell remoto usato con le reti locali, solitamente collocato in **/usr/sbin/rsh**. Il programma **rsh** ha minori possibilità rispetto allo shell normale e può es-

sere assegnato agli utenti, come **/usr/lib/rsh**, al posto di quello standard, mediante il comando seguente:

```
# useradd -s /usr/lib/rsh giorgio
#
```

Dopo questo comando l'utente utilizza lo shell ridotto invece dello shell standard.

Il programma **rsh** differisce dal normale shell nelle seguenti caratteristiche:

1. L'utente non può utilizzare il comando **cd** per cambiare directory, ma è limitato al solo home directory.
2. L'utente non può cambiare il contenuto della variabile PATH, per cui, nella variabile PATH sono validi solo i comandi impostati dal gestore di sistema.
3. L'utente non può specificare i comandi o i file utilizzando un pathname completo, per cui può accedere solo ai file che si trovano nell'home directory e nei subdirectory.
4. L'utente non può ridirigere l'output tramite gli operatori > o >>.

Queste limitazioni vengono imposte una volta eseguito il file utente **.profile**. Il gestore di sistema inizializza un ambiente ridotto nel file utente **.profile**, compresa la variabile PATH che referenzia un ristretto directory **bin**; quindi assegna la proprietà del file **.profile** a **root**, e dichiara il file accessibile in lettura da tutti gli utenti, ma non accessibile in scrittura o esecuzione da nessuno. In questo modo, gli utenti con uno shell limitato sono confinati all'home directory e ai comandi consentiti nella variabile PATH.

Lo shell **rsh** impedisce agli utenti meno esperti di allontanarsi troppo dall'ambiente a loro destinato ed entrare nei file di sistema; generalmente, questo shell viene assegnato a utenti che hanno interessi limitati per entrare nel sistema, come utilizzatori di solo word processor. Di solito non conviene limitare la libertà degli utenti esperti con **rsh**, dal momento che interferirebbe inutilmente con il loro lavoro. Inoltre, il programma **rsh** non è completamente sicuro e un utente esperto può facilmente trovare un modo per passare in uno shell normale. Non è possibile considerare il programma **rsh** un valido strumento di sicurezza contro intrusioni dolose, tuttavia si dimostra utile per impedire agli utenti inesperti di danneggiare inavvertitamente se stessi o il sistema.

22.12 Protezione del sistema UNIX e dei file

Anche gli utenti esperti possono danneggiare il sistema UNIX se hanno troppa libertà di accesso a file chiave, come per esempio i dati di **/etc/passwd** o di **uucp**. Occorre evitare che la sicurezza del sistema si riduca

in conseguenza dei normali cambiamenti avvenuti durante la vita del sistema. La soluzione è semplice da spiegare, ma difficile da realizzare: mantenete semplicemente tutti i diritti di accesso e di proprietà dei file come erano stati stabiliti in fase di installazione del sistema. Generalmente, il caricamento iniziale del sistema UNIX inizializza tutti i file e i directory con diritti di accesso corretti e sicuri. La sola eccezione può derivare dagli identificatori di login di sistema in **/etc/passwd**, che di solito non prevedono password per default. Comunque, con il tempo i diritti di accesso divengono gradualmente sempre meno restrittivi, proprio perché nessuno dedica un'attenzione particolare alla sicurezza. Insomma, i file diventano più accessibili, le password scompaiono, le sessioni di utente non attive si accumulano, ecc.

Per fronteggiare una situazione di questo tipo, una buona politica consiste nel ricaricare regolarmente la macchina con la versione originale del software di sistema, anche se questo comporta del lavoro aggiuntivo per gli utenti e per il gestore di sistema, che deve ricaratterizzare ogni volta il sistema. Se, comunque, sospettate violazioni della sicurezza, applicare questa procedura una o due volte all'anno si dimostra l'unico modo per garantire la protezione del sistema. Ricaricando il sistema potete anche assicurarvi che il file delle password sia aggiornato, e inoltre ottimizzare lo spazio su disco eliminando comandi e file che non vengono utilizzati; comunque, una buona procedura di salvataggio di dati rende l'operazione di ricaricamento del sistema molto più semplice.

Un secondo possibile approccio consiste nell'eseguire regolari revisioni della macchina e del file system, possibilmente tramite una procedura automatizzata di comandi che verifica la sicurezza dei diritti di proprietà e di accesso dei file. Nessuno strumento di questo tipo è ufficialmente disponibile in SVR4, ma diversi esempi si sono diffusi in rete; infatti, alcuni "vaccini" prodotti lavorano in questa maniera per riscontrare varie forme possibili di alterazione dei file. Strumenti simili sono necessariamente dipendenti dall'installazione specifica, ma un esperto gestore di sistema è in grado di crearsi la procedura adatta.

22.13 Sicurezza fisica

Abbiamo finora focalizzato la nostra attenzione sugli identificatori di login di utente e sul modo con cui garantire la sicurezza per i semplici utenti. Esistono, tuttavia, molti più rischi potenziali per la sicurezza di quelli causati dagli utenti autorizzati a utilizzare il sistema; dovete anche considerare il caso di persone che possono accedere fisicamente alla macchina in vostra assenza.

In ogni caso, la principale forma di sicurezza per un sistema UNIX è la sicurezza fisica. Se riuscite a impedire l'accesso alla macchina installandola in una stanza chiusa a chiave e impedendo ogni possibile collegamento

esterno con modem o reti locali, potete star tranquilli che la sicurezza non verrà violata. Un sistema veramente sicuro non possiede collegamenti esterni a eccezione dei terminali in connessione diretta installati in stanze chiuse a chiave. Se tutti gli identificatori di login sono correttamente protetti con password, non si possono verificare accessi non autorizzati.

Comunque, tenete presente che molti sistemi UNIX sono accessibili reinizializzando da un dischetto o da un nastro "master". Molte macchine recenti prevedono una protezione con password in ROM per controllare le inizializzazioni, ma neanche questo sistema è completamente inviolabile. Un intruso in grado di arrivare alla macchina può inizializzarla con una propria versione del sistema operativo e godere di tutti i privilegi di accesso di un supervisore. In conclusione, ai fini della sicurezza del sistema dovete prima di tutto garantire che la macchina stessa non sia fisicamente accessibile a persone non autorizzate.

Una seconda forma, più contorta, di attacco alla sicurezza fisica può provenire dal software e dalle applicazioni che installate. Qualunque disco o pacchetto applicativo che caricate sulla macchina può contenere delle trappole per la sicurezza, aggiunte da chi lo ha sviluppato o da una persona che ha usato quel software prima di voi. In una situazione tipica il software viene caricato correttamente, ma alcune parti di esso prefigurano una violazione di sicurezza, che diventa attiva una volta che viene eseguita quella parte di programma. Il programma stesso può causare un danno alla macchina, oppure eseguire qualche modifica al sistema per consentire a un utente malintenzionato di intromettersi. Questi problemi di solito non si verificano con pacchetti disponibili sul mercato, ma il software privo di certificazione deve essere considerato con sospetto, soprattutto se si tratta di software gratuito disponibile su base non professionale o di copie non ufficiali distribuite manualmente. Generalmente non è possibile scoprire questi problemi prima che il danno sia arrecato, per cui occorre essere estremamente prudenti prima di aggiungere software sconosciuto nella macchina. Nel caso peggiore, i gestori di sistema consci dei problemi della sicurezza possono richiedere l'accesso al codice sorgente delle applicazioni, esaminarlo con attenzione in un ambiente sicuro e poi compilarlo direttamente per la corrispondente macchina. Questa procedura generalmente non risulta applicabile, ma consente qualche volta di riconoscere un comportamento sospetto in chi fornisce il software.

22.14 Reti locali

Gli ambienti in cui molte macchine sono collegate in rete locale (LAN o local area network) costituiscono rischi non indifferenti per la sicurezza dei sistemi, perché se un utente malintenzionato può entrare in una delle macchine collegate in rete, è in grado di accedere anche a tutte le altre macchine.

Risulta quindi estremamente importante per la sicurezza delle macchine collegate in rete che tutti gli utenti di tutte le macchine garantiscano rigorosamente la sicurezza delle password.

Dal momento che gli utenti di LAN lavorano generalmente su un unico progetto, molte LAN forniscono strumenti che consentono di condividere con facilità i file e i dati tra le macchine, come NFS o **rlogin**. Naturalmente questo è rischioso, ma può essere accettato se le singole macchine sono dotate di mezzi di protezione. Qualche volta è richiesta una password ulteriore prima di consentire l'accesso alla rete. Questo può rallentare il ritmo delle abituali attività degli utenti, ma può garantire una protezione aggiuntiva quando la sicurezza costituisce il maggiore problema.

Molte LAN prevedono i "domini", costituiti da gruppi di macchine che condividono facilmente file o login remoti; le macchine al di fuori di un determinato dominio possono essere escluse dall'accesso a quel dominio. Con questa architettura di domini, la violazione della sicurezza di una macchina nella LAN non compromette la sicurezza dell'intera LAN; pertanto, i domini possono alleggerire notevolmente l'impegno per la sicurezza. Dovrete comunque concordare con il gestore della vostra LAN una adeguata politica di sicurezza.

22.15 La sicurezza di uucp

I sottosistemi di comunicazione di dati **uucp** costituiscono un potenziale rischio per la sicurezza, dal momento che questi strumenti sono progettati per consentire l'accesso remoto. Quando è attivo, **uucp** instaura il collegamento con un'altra macchina ed esegue i comandi in modalità remota su quella macchina, leggendo e scrivendo i file come richiesto. Se la sicurezza è ridotta al minimo, allora un attacco, o anche un errore involontario nell'utilizzo di **uucp**, può causare seri danni alla macchina. In ogni modo, il recente pacchetto software **uucp** è sicuro e affidabile, a condizione che si rispettino le precauzioni di sicurezza. Come in molti altri aspetti nel sistema UNIX, la sicurezza del sistema **uucp** degrada naturalmente con il tempo, per cui occorre dedicare molta attenzione per tenere sotto controllo gli strumenti di **uucp**.

Il maggiore elemento di sicurezza associato con **uucp** è costituito dai diritti di accesso ai file di **uucp**. Il contenuto del directory **/usr/lib/uucp** deve mantenere i diritti di accesso che aveva quando avete inizialmente caricato il software sul sistema. In pratica, dovrete elencare questi file con **ls -la** subito dopo aver installato il software nella macchina e in seguito verificare con cura la correttezza dei diritti di accesso ogni volta che apportate delle modifiche nei directory **/usr/lib/uucp** oppure **/etc/uucp**.

Il sistema **uucp** è progettato in modo da svolgere tutta la sua attività nella macchina in una gerarchia pubblica di directory; questa area pubblica deve

essere obbligatoriamente presente se prevedete una qualsiasi attività con **uucp** nella macchina. Inoltre, deve esistere un elenco di comandi che possono essere eseguiti da una macchina remota mediante il comando **uux**. Generalmente la locazione del directory pubblico è **/var/spool/uucppublic** (noto anche come **/usr/spool/uucppublic**) e l'elenco dei comandi disponibili in modalità remota comprende solo il comando **rmail**. Non potete ricevere posta da macchine remote se non permettete alle altre macchine di eseguire il vostro comando **rmail**.

IL FILE DEI DIRITTI DI ACCESSO DI uucp

Potete personalizzarvi i diritti di accesso che si riferiscono al sistema **uucp**. I dati di controllo sono contenuti in un file, che viene letto dai programmi **uucp** quando vengono eseguiti; se una certa richiesta non è ammessa dai dati di controllo, la richiesta viene rifiutata e il collegamento **uucp** interrotto. Vengono accolte solo le richieste contenute in un elenco di richieste ammesse. Il file **/etc/uucp/Permissions** (noto anche come**/usr/lib/uucp/Permissions**) contiene le informazioni sui diritti di accesso; solo il superuser e il login **nuucp** devono essere in grado di leggerlo.

I diritti di accesso possono variare da un'ampia libertà di accesso a una rigorosa limitazione. Di solito i diritti di default stabiliti quando la macchina viene installata, sono molto restrittivi. Un esempio di **Permissions** estremamente permissivo è di questo tipo:

```
# cat /etc/uucp/Permissions

# This entry is wide-open....
LOGNAME = uucp:nuucp REQUEST = yes SENDFILES = yes READ = / WRITE = /
MACHINE = OTHER COMMANDS = ALL REQUEST = yes READ = / WRITE = /
#
```

Il formato è un normale file ASCII; le linee che iniziano con il carattere # (cancelletto) vengono interpretate come commenti, mentre le altre linee contengono i dati. La struttura base si compone di linee contenenti la coppia *parola-chiave=valore*, dove le parole chiave definiscono le caratteristiche controllate e i valori stabiliscono i diritti di accesso permessi. Nel caso di più valori associati a una parola chiave, le diverse voci devono essere separate con un : (due punti), come mostrato nell'esempio precedente nel riferimento **LOGNAME=uucp:nuucp**. Ogni linea può controllare uno specifico insieme di diritti di accesso, determinando le modalità con cui la macchina chiama altre macchine o, viceversa, le modalità di trattamento delle altre macchine quando chiamano la vostra. Spesso occorre definire diverse linee di entrambi i tipi per adattare il sistema **uucp** alle vostre esigenze.

I DIRITTI DI ACCESSO DI DEFAULT

I diritti di accesso di default di **uucp** sono relativamente restrittivi, per cui è garantita una buona sicurezza se utilizzate semplicemente la linea:

```
LOGNAME = nuucp
```

quale unico contenuto del file **Permissions**. La parola chiave LOGNAME indica gli identificatori di login in grado di richiedere i servizi di **uucp**. Questo vale non per gli utenti che operano localmente sulla macchina, ma si riferisce alle macchine remote che possono chiamare la vostra macchina richiedendo i servizi di **uucp**.

La semplice linea LOGIN precedente permette al solo identificatore di login **nuucp** di accedere in modalità remota al sistema **uucp**. Ricordatevi che lo shell di default per questo login è **/usr/lib/uucp/uucico**, un programma per le comunicazioni di dati del tutto sicuro. Con questo semplice file **Permissions**, il sistema **uucp** è limitato al trasferimento di file con il directory di dominio pubblico **/usr/spool/uucppublic**, e le macchine remote possono utilizzare solo il comando **rmail**.

PERSONALIZZAZIONE DEL FILE Permissions

Potete aggiungere altre linee al file **Permissions** per adeguare la sicurezza di **uucp** alle vostre esigenze. Due differenti tipi di linee sono contenuti in un file **Permissions** più complesso: quelle che modificano le azioni del sistema nel caso di chiamate in arrivo e quelle che modificano le azioni in seguito ai collegamenti con le macchine che vengono chiamate. La linea "LOGNAME=..." interessa tutte le chiamate in ingresso e le linee "MACHINE..." interessano le macchine che vengono chiamate. Il file **Permissions** può contenere solo una linea del tipo "LOGNAME....", mentre non esistono limiti al numero di linee "MACHINE...". Quando modificate le linee del file **Permissions**, aggiungete altre coppie *parola-chiave=valore* sulla stessa linea. Se la linea supera la lunghezza ammessa, potete terminare la linea con il carattere \ (barra inversa) prima del ritorno a capo, e proseguire sulla linea o linee successive.

CONTROLLO DELLE CHIAMATE IN INGRESSO CON LOGNAME

Quando un'altra macchina chiama la vostra, potete consentire il trasferimento dei lavori che sono in coda, oppure potete far richiamare l'altra macchina dalla vostra prima di trasferire i lavori. Questa seconda procedura è maggiormente sicura, poiché la macchina chiamante può mentire sul suo **uname** e ricevere file non a essa destinati. Questa situazione costituisce un inganno al sistema **uucp** e l'unico modo certo per evitarlo è di far chiamare alla vostra

macchina l'altra macchina ogni volta che ha lavori in coda da trasferire a un'altra macchina. Aggiungete la parola chiave SENDFILES= alla linea "LOGNAME=..." per stabilire se i lavori debbono o no essere inviati quando la vostra macchina viene chiamata. Per esempio, la linea:

```
LOGNAME = nuucp SENDFILES = yes
```

consente di eseguire il trasferimento, mentre

```
LOGNAME = nuucp SENDFILES = no
```

inibisce il trasferimento in ogni caso. La linea **SENDFILES = call**, di default, significa che i file sono inviati a un'altra macchina solo quando la vostra macchina la chiama, non quando è l'altra macchina a chiamare.

Analogamente, potete aggiungere la parola chiave REQUEST= per stabilire se la vostra macchina consente o no a una macchina remota di richiedere file dal vostro sistema. Per esempio:

```
LOGNAME = nuucp SENDFILES = yes REQUEST = yes
```

permette a una macchina remota di richiedere vostri file, indipendentemente dalla loro presenza o no nella coda di file da trasferire. Questa caratteristica può essere molto pericolosa e di solito deve essere definita con **REQUEST = no**.

Potete anche specificare i directory che le macchine remote possono utilizzare per leggere o scrivere i file, aggiungendo le parole chiave READ= e WRITE= alla linea "LOGNAME=...". I directory così indicati definiscono una sottogerarchia, per cui anche ogni directory che si trova sotto il directory specificato può essere letto o scritto. Se utilizzate **READ = /** e **WRITE = /**, rendete possibile la lettura e la scrittura da tutti i directory della macchina, soluzione decisamente sconsigliabile. Di default sono predisposte le dichiarazioni **READ = /var/spool/uucppublic** e **WRITE = /var/spool/uucppublic**, che referenziano il directory pubblico raccomandato.

Quando una macchina vi chiama, potete predisporre il sistema **uucp** per rifiutare la chiamata e far richiamare immediatamente la macchina chiamante dalla vostra macchina. Aggiungete **CALLBACK = yes** alla linea "LOGNAME=..." per attuare questo comportamento di richiamata, che altrimenti non viene attivato automaticamente poiché il default è **CALLBACK = no**.

CONTROLLO DELLE CHIAMATE IN USCITA CON MACHINE

I diritti di accesso relativi alle chiamate in uscita possono essere modificati per una particolare macchina o un insieme di macchine specificando "MACHINE=" all'inizio della linea. Tutte le modifiche apportate rispetto

ai valori di default sono associate alle macchine indicate esplicitamente nella linea "MACHINE=". Se volete che tutte le macchine chiamanti abbiano gli stessi diritti di accesso, potete specificare la speciale stringa OTHER dopo MACHINE= per referenziare tutte le macchine, oppure potete indicare solo alcune macchine. Le parole chiave REQUEST=, SENDFILES=, READ= e WRITE= possono apparire nella linea MACHINE=, così come nella linea LOGNAME=.

Inoltre, potete cambiare la lista di comandi che le altre macchine possono eseguire sulla vostra macchina sia quando le chiamate, sia quando lanciano richieste **uux** sulla vostra macchina. Per cambiare i comandi, aggiungete alla linea MACHINE= la parola chiave COMMANDS=, seguita dalla lista dei comandi separati dal carattere due punti. Per esempio:

```
MACHINE=OTHER COMMANDS=rmail:news:lp
```

In questo esempio, le richieste **uux** in arrivo da macchine remote possono utilizzare i comandi **rmail**, **news** e **lp**. Nell'esempio il comando **rmail** è sempre richiesto, ma la presenza di **lp** consente alle macchine remote di inviare i lavori alla vostra stampante attraverso il sistema **uucp**. Tutti i comandi indicati nella lista COMMANDS= devono essere collocati nei directory **/sbin**, **/usr/sbin** o **/usr/bin**. Potete utilizzare COMMANDS=ALL per permettere alla macchina remota di lanciare tutti i comandi. L'uso di COMMANDS=ALL è sconsigliabile a meno che non vi possiate fidare completamente della macchina remota, comunque non deve essere assolutamente impiegato insieme a MACHINE=OTHER.

IL COMANDO uucheck

La personalizzazione del file **Permissions** può essere verificata con il comando **/usr/lib/uucp/uucheck**, che controlla la correttezza del sottosistema **uucp**, con speciale attenzione ai dati di **Permissions**. Il comando **uucheck** è riservato al superuser, dal momento che i diritti di accesso di **uucp** devono essere tenuti segreti. Utilizzate

```
# /usr/lib/uucp/uucheck -v
```

per avere informazioni molto precise sulla configurazione dei diritti di accesso di **uucp**. Il comando **uucheck** è uno strumento molto utile che può essere determinante per la comprensione dei diritti di accesso di **uucp**. La Figura 22.2 mostra i risultati prodotti dal comando **uucheck** analizzando la caratterizzazione del file **Permissions** "wide-open" già mostrato. L'analisi è realizzata in due fasi. La prima comprende la verifica della correttezza dei file e directory **uucp**; se non vengono riscontrati problemi, non produce alcun risultato in uscita. La seconda fase invece elenca i dati del file **Permissions** in modo dettagliato, in una forma facile da comprendere.

```
# /usr/lib/uucp/uucheck -v
*** uucheck: Check Required Files and Directories
*** uucheck: Directories Check Complete

*** uucheck: Check /usr/lib/uucp/Permissions file
** LOGNAME PHASE (when they call us)

When a system logs in as: (uucp) (nuucp)
        We DO allow them to request files.
        We WILL send files queued for them on this call.
        They can send files to
             /
        Sent files will be created in /var/spool/uucp
        before they are copied to the target directory.
        They can request files from
             /
        Myname for the conversation will be my_sys.
        PUBDIR for the conversation will be /var/spool/uucppublic.

** MACHINE PHASE (when we call or execute their uux requests)

When we call system(s): (OTHER)
        We DO allow them to request files.
        They can send files to
             /
        Sent files will be created in /var/spool/uucp
        before they are copied to the target directory.
        They can request files from
             /
        Myname for the conversation will be my_sys.
        PUBDIR for the conversation will be /var/spool/uucppublic.

Machine(s): (OTHER)
CAN execute the following commands:
commands (ALL), fullname (ALL)
*** uucheck: /etc/uucp/Permissions Check Complete

#
```

Figura 22.2 Esempio di risultati prodotti dal comando **uucheck.**

MACCHINE REMOTE SCONOSCIUTE E OPERAZIONE DI POLLING

Se una chiamata in ingresso proviene da una macchina che il sistema non riconosce perché non è compresa nei suoi file **Systems**, la chiamata può essere accettata, oppure il sistema **uucp** può eseguire un apposito file di comandi che tiene una storia delle chiamate pervenute da macchine ignote o

che provvede a qualche simile precauzione ai fini della sicurezza. Quste possibilità non sono supportate dal file **Permissions**, ma sono gestite in modo leggermente differente. Se il file **/usr/lib/uucp/remote.unknown** è presente ed è eseguibile, la chiamata che proviene da una macchina remota viene terminata e viene lanciata la procedura **remote.unknown**. Se il file invece non è presente oppure non è eseguibile, la chiamata viene accettata nella categoria **MACHINE = OTHER**. Generalmente, la funzione **remote.unknown** è disattivata per poter ricevere posta da macchine che possono conoscere la vostra macchina, ma che voi non conoscete. Comunque, quando si sospetta la possibilità di violazioni di sicurezza attraverso il sistema **uucp**, la funzione può essere abilitata rendendo **remote.unknown** eseguibile. In SVR4 per motivi di sicurezza **remote.unknown** è un file programma binario, ma voi potete rimpiazzarlo con una procedura comandi se volete personalizzarne le azioni.

Inoltre, è possibile configurare il sottosistema **uucp** per interrogare (poll) un altro sistema. Questo significa che una macchina può chiamare un altro sistema e richiedere le applicazioni a essa destinate. Questa caratteristica viene spesso utilizzata quando una macchina non può chiamare la vostra, mentre voi potete chiamarla. Per esempio, una macchina potrebbe supportare solo le chiamate in ingresso.

Quando una macchina ne chiama un'altra, la macchina chiamata può trasferire le applicazioni in uscita che sono in attesa destinate alla macchina chiamante. Questa funzione generalmente è disabilitata dalla dichiarazione di default **SENDFILES = no** nel file **Permissions** della macchina ricevente. Con **SENDFILES = no**, la macchina che intende trasferire dati deve chiamare la macchina ricevente e la ricevente non può raccogliere i messaggi quando chiama la macchina che trasmette. Definite **SENDFILES = yes** sulla macchina se volete che possa trasferire le applicazioni in uscita quando viene chiamata.

22.16 Approfondimenti

I problemi di sicurezza che abbiamo trattato precedentemente hanno fondamentale importanza in ogni sistema UNIX di uso comune ben gestito. Le misure adottate nel sistema UNIX per garantire la sicurezza risultano abbastanza efficaci se vengono gestite con cura e attenzione, ma non possono realisticamente impedire a un esperto malintenzionato di accedere al sistema. Esistono molti esperti di UNIX e una parte di essi ha come unico obiettivo quello di entrare senza autorizzazione nei sistemi UNIX.

ATTACCHI AL SISTEMA

Generalmente un attacco al sistema proviene dagli "hacker" che intendono dimostrare la loro abilità a vostre spese. Di solito un intruso di questo tipo irrompe nella macchina, curiosa un poco nel file system, si annoia rapida-

mente ed esce senza arrecare danno; in qualche caso, l'obiettivo è di impossessarsi di alcuni dati o, semplicemente, di danneggiare il sistema.

Esaminiamo uno scenario di un attacco al sistema. Un malintenzionato riesce a ottenere il numero telefonico del vostro modem (o LAN) utilizzando informazioni di altri hacker oppure componendo casualmente numeri di telefono. Quindi tenta ripetutamente di collegarsi alla macchina, fino a quando scopre un identificatore di login non protetto, che gli consente di accedere alla macchina. A questo punto, è in grado di esaminare, con assoluta tranquillità, il file di password e il resto del sistema, fino a quando non trova il modo per passare all'identificatore di login **root**. Spesso vengono cambiati i file di sistema, adattati i diritti di accesso e alterati alcuni comandi di sistema per rendere in seguito più facile l'ingresso nel sistema. I collegamenti in rete vengono poi esplorati per trovare macchine vicine a cui estendere l'attacco; quando la vostra macchina è stata abbastanza esplorata, viene spesso apportato qualche danneggiamento per puro divertimento. Infine, l'informazione di accesso viene passata a un altro malintenzionato che ripete lo stesso processo di disturbo nel sistema.

Da questo scenario potete trarre alcune conclusioni relative alla sicurezza del sistema da intrusioni indebite. Primo, la protezione con password è fondamentale per impedire a malintenzionati di accedere al sistema; secondo, una volta che questa situazione si è verificata, scoprire e isolare gli intrusi può risultare estremamente difficile; terzo, se il sistema è stato violato una prima volta, è molto probabile che venga attaccato nuovamente e che l'attacco si estenda anche alle macchine vicine; quarto, se rispondete con azioni inadeguate alle intrusioni, chi si è introdotto nel vostro sistema impara presto le misure di protezione adottate e continua a sconfiggerle.

COMPORTAMENTO DIFENSIVO

Quasi sempre la prima reazione a un sospetto di attacco è il rifiuto di quello che accade. Vi chiedete solo perché sono cambiati i diritti di accesso al file o perché il file di password è stato improvvisamente modificato; spesso non ricordate esattamente le condizioni del sistema. Naturalmente, un intruso malintenzionato si avvantaggia di questa situazione confusa per insinuarsi maggiormente nel sistema.

Quando siete sicuri che il sistema sia stato attaccato, generalmente trascurate la pericolosità dell'attacco. Anche quando sono attivi login sconosciuti, spesso cercate di contattare gli utenti con **mail** o **write** per chiedere educatamente perché utilizzano la vostra macchina; questo avvertimento è quanto un intruso si aspetta. Col prossimo passo, potete annunciare agli utenti collegati (di solito con **news**) che la macchina è stata attaccata, ma questo comporta di avvertire nuovamente gli intrusi che sono stati scoperti. Successivamente potete cercare di ripristinare le misure di sicurezza, cambiando per esempio le password con password altrettanto banali di quelle

che, con ogni probabilità, avevano permesso all'intruso di entrare nel sistema. Infine, dopo il verificarsi di ripetuti attacchi, potete reagire in maniera eccessiva, disattivando totalmente tutti i collegamenti con la macchina.

Questo comportamento difensivo viene sfruttato adeguatamente dagli intrusi più esperti. Infatti, avete ripetutamente avvisato l'intruso che siete a conoscenza che la sicurezza del sistema è stata violata e avete preso contromisure con tale lentezza che l'intruso ha avuto tutto il tempo per prevenire le vostre mosse. Gli intrusori possono arrivare a installare nel software un *cavallo di Troia* per catturare le password quando gli utenti accedono al sistema, modificare permessi di accesso ai file e così via. D'altronde, un'eventuale troppo drastica azione intrapresa contro l'intruso può ritorcersi contro di voi e intralciare l'utilizzazione regolare della macchina.

Abbiamo descritto all'inizio di questo capitolo il comportamento migliore per fronteggiare un attacco. Se scoprite un attacco, non dovete affatto cambiare il sistema mentre pianificate la difesa. È particolarmente imprudente emettere un annuncio con **news** per avvertire tutti gli utenti collegati delle modifiche che intendete apportare al sistema. La prima azione da prendere che possa risultare evidente è il cambiamento improvviso e completo del sistema: cambiate i numeri di telefono per i modem, ricaricate il software di sistema dall'inizio, cambiate tutte le password e gli identificatori di login e ristabilite i collegamenti **uucp** con le altre macchine. Se le misure di protezione vengono adottate radicalmente e silenziosamente, avete risolto il problema. In nessun caso dovete far apparire nella macchina alcuna informazione che possa rivelare che la macchina è stata attaccata, tantomeno descrizioni dei cambiamenti che avete intenzione di apportare.

COME SCOPRIRE UN ATTACCO AL SISTEMA

Sfortunatamente, la scoperta di un attacco può risultare difficile, dal momento che la maggior parte degli intrusi è più esperta della maggior parte degli utilizzatori attaccati. I punti chiave per verificare la presenza di intrusi sono i seguenti: modifiche ai diritti di accesso dei file associati alla sicurezza trattati in questo capitolo, oppure modifiche ai file dati di **cron**, a **/etc/inittab** ed **etc/profile**, e ai file di controllo di Service Access Facility. Ogni modifica al contenuto di questi file è sospetta, specialmente se riguarda **/etc/passwd**, **/etc/uucp/Permissions**, o i file del sistema **uucp** ed **/etc/profile**. I file di storia dei collegamenti con **uucp** possono fornire informazioni sull'attuazione di insoliti collegamenti **uucp** con altre macchine sconosciute. Anche utenti non abituali della macchina che hanno utilizzato il sistema nelle ore notturne sono un motivo per essere sospettosi. Il file **/var/adm/sulog** registra tutti gli utilizzi del comando **su**, che viene spesso usato dagli intrusi.

Gli intrusi più esperti di solito cercano di non lasciare tracce. Un intruso esperto può modificare il file **sulog** per rimuovere le tracce, ma i diritti di accesso e la data di modifica dei file possono rivelare il fatto che sono stati cambiati.

Il gestore di sistema responsabile della sicurezza può realizzare un file di comandi che ricerca nel file system le modifiche ai diritti di accesso dei file e alle dimensioni di programmi eseguibili. Uno strumento di questo tipo risulta specifico per ogni versione di UNIX e del relativo hardware, ma può avere come parte comune il processo di ricerca di eventuali attacchi al sistema. Se lo strumento di difesa si trova su dischetto invece che sul file system principale, difficilmente può essere neutralizzato dagli intrusi malintenzionati.

CAVALLI DI TROIA, VIRUS, VERMI

La più tortuosa e sgradevole forma di attacco si verifica quando gli intrusi modificano la macchina sostituendo alcune parti chiave del software con copie alterate che consentono facilmente l'intrusione, dette "cavallo di Troia". Questo di solito capita solo quando avete avvisato gli utenti, e quindi anche gli intrusi, che la sicurezza della macchina è stata violata e può risultare molto difficile da scoprire.

Potete utilizzare le vostre procedure periodiche di salvataggio per copiare il programma corrotto sul vostro archivio; in fase di ricostituzione del sistema, la tendenza naturale è di ripristinare la copia salvata piuttosto che ricaricare il sistema ex-novo. Questo procedimento spesso ripristina proprio il programma corrotto, per cui l'azione intrapresa risulta utile per l'esperienza dell'intruso.

Quando sospettate la presenza di intrusi, le vostre copie di salvataggio possono non essere più affidabili, per cui dovete utilizzarle con molta cautela per un ripristino. In pratica, è opportuno salvare solo i vostri dati d'utente e non il software di sistema; quando ricaricate il sistema, dovete ricorrere alla versione originale del software di sistema che avete acquistato insieme alla macchina.

I programmi "virus" e "vermi" (worm) agiscono con gli stessi principi degli hacker, ma possono infettare e diffondersi in altre macchine senza l'azione diretta di una persona. Un virus è un programma che in qualche modo si inserisce in una parte esistente di software, mentre il verme è un programma completo che si inserisce nelle attività e sopravvive grazie alla natura multiprocesso del sistema. Di solito, l'esecuzione di un programma infettato, o il montaggio di un disco infettato, attiva un virus o un verme, che in seguito infetta la macchina ospite. L'infezione può causare un danno immediato, ma più spesso il virus attende un certo periodo di tempo, durante il quale infetta ancora altri dischetti (oppure la LAN); quindi, a un certo evento scatenante, viene risvegliato e distrugge il sistema ospite.

Queste infezioni sono estremamente difficili da scoprire e riparare; la difesa migliore consiste nel prevenirle. Siate estremamente accorti nelle connessioni con altre macchine e con il software che installate.

SYSTEM V/MLS

Il sistema *Multilevel Security* è una versione di SVR4 che rientra nelle Security Class B2 e B3 dell'U.S Government. Le specifiche di queste classi prevedono elenchi di accesso file per file, verifiche a livello singolo utente e gruppo prima di permettere l'accesso, livelli di sicurezza gerarchici, tipo "secret" e "top secret". Il livello B2 esige inoltre un modello di sicurezza formale che può essere verificato al momento del progetto e durante l'implementazione. Questi sistemi sono assolutamente sicuri, ma impongono un significativo decremento nelle prestazioni del sistema. Se avete necessità di questi livelli di sicurezza consultate il vostro fornitore del sistema UNIX.

Capitolo 23

X Window System

Una delle maggiori innovazioni degli ultimi anni è stata l'incorporazione di X Window System come parte dello standard industriale del sistema UNIX. Il sistema X Window System, chiamato anche semplicemente X, fornisce un ambiente di lavoro su intero schermo, che consente la visualizzazione simultanea sullo schermo di varie finestre. Le nuove applicazioni possono essere sviluppate in modo da trarre pieno vantaggio dall'ambiente a finestre, che include la grafica ad alta definizione, il supporto del colore e l'uso del mouse o altri strumenti simili. Le applicazioni sviluppate in precedenza possono continuare a funzionare in speciali finestre in emulazione di terminale, che operano come i terminali ASCII vecchia maniera.

X fornisce un ambiente standard di sviluppo e di uso, praticamente in tutte le versioni di sistema UNIX, garantendo un'eccellente portabilità delle applicazioni; questo, a sua volta, rende più facile e meno costoso lo sviluppo delle applicazioni. Sono stati creati per SVR4 diversi sistemi GUI (Graphical User Interface) basati su X; molti produttori di sistemi UNIX ne adottano uno per le loro release. I sistemi più diffusi sono il sistema OPEN LOOK, supportato da AT&T e Sun Microsystems e il sistema Motif supportato da Open Software Foundation; comunque, esistono diversi altri GUI. I sistemi Open Desktop e Looking Glass sono commercializzati dai rivenditori di sistemi UNIX/386 SCO e Interactive Systems, rispettivamente. Sono disponi-

bili anche molte altre opzioni; l'ambiente di utente utilizzato da NeXT e pochi altri sistemi è la principale eccezione all'impiego di X nel mondo UNIX. D'altra parte, X si sta diffondendo così decisamente che ne vengono proposte versioni per quasi tutti i computer e sistemi operativi, anche al di fuori del sistema UNIX. Il supporto di X è attualmente disponibile su Macintosh, su grosse macchine MS-DOS e su sistemi OS/2.

Il sistema X richiede macchine con elevate caratteristiche, trattandosi di un ambiente software molto esteso e complesso, che necessita di hardware con caratteristiche sofisticate. Per trarre vantaggio delle possibilità di X è necessario un video con display grafico ad alta risoluzione (a colori, o monocromo, ma di livello EGA, preferibilmente VGA, o superiore) e un mouse, o un altro dispositivo di puntamento. Se usate un terminale (invece della console di sistema), dovete disporre di una connessione ad alta velocità con la macchina host; una connessione con modem 9600 bps non è sufficiente. Questi sono i prerequisiti minimi per utilizzare X; tuttavia, per ottenere prestazioni adeguate a un utilizzo effettivo, la vostra macchina deve avere un'alta velocità di CPU, molto spazio su disco, molta memoria reale RAM; tutto in misura superiore a quelle che sarebbero le vostre normali necessità. Se svolgete intenso lavoro grafico, il supporto hardware della virgola mobile (*floating-point*) è senz'altro utile, anche se la maggior parte delle applicazioni X (come **xterm**) non impiega molto l'aritmetica in virgola mobile. Consultate il vostro fornitore di sistema UNIX o X.

23.1 Concetti base di X

X Window System è basato sul concetto di *display* logico. Un display in genere è un normale dispositivo grafico come un CRT, ma include una tastiera e uno strumento di puntamento, di solito il mouse; deve disporre anche di capacità di elaborazione, pertanto un semplice terminale non si può considerare un display. Un display può avere più di un CRT; in altri termini, una tastiera e un mouse possono essere condivisi tra diversi CRT. CRT multipli nello stesso display vengono detti *screen* (schermi). Nella maggior parte dei casi un display dispone di un solo screen. Comunemente, la console di sistema, con la tastiera, il mouse e la CPU, costituisce un display con uno screen.

Ogni applicazione dispone di una sua propria finestra (o più di una, se ne ha necessità) sullo schermo; l'uscita dell'applicazione avviene solo in quella finestra. Un'applicazione può creare, modificare, o cancellare, le proprie finestre, ma di regola non può influenzare le finestre di proprietà di altre applicazioni. L'utente può selezionare la finestra su cui inviare l'input da tastiera o da mouse. L'uscita delle diverse applicazioni può apparire simultaneamente nelle rispettive finestre, ma l'ingresso può essere diretto a una sola finestra alla volta.

IL SERVER X

La parte principale del pacchetto X è costituita dal software dedicato alla gestione del display; questo software è chiamato *display server*, o semplicemente *server*. Il compito del server è quello di interpretare i tasti della tastiera, i click del mouse e visualizzare le immagini sul dispositivo display. Il server non ha compiti specifici per le applicazioni; la maggior parte delle caratteristiche richieste dalle specifiche applicazioni viene trattata da processi separati. Il server deve essere in funzione durante tutta la vostra sessione X, poiché le applicazioni comunicano con esso per visualizzare l'uscita o leggere l'entrata; può comunicare simultaneamente con applicazioni multiple e inviare al "cliente" opportuno il dato di ogni tasto, in modo che possiate eseguire senza difficoltà diverse applicazioni allo stesso tempo.

APPLICAZIONI CLIENTE

Il lavoro specifico di un'applicazione viene eseguito da programmi cliente che comunicano col server tutte le volte che ne hanno necessità; in questo modo, l'ambiente di X viene suddiviso in due mondi distinti: uno orientato al display e l'altro orientato all'applicazione. La conseguenza più importante di questa suddivisione è quella che le applicazioni non conoscono l'hardware che esegue le operazioni di I/O e diventano perciò portabili su altre macchine. Il server verrà sviluppato una volta per tutte dal produttore del sistema e le applicazioni clienti potranno essere portate facilmente da un elaboratore all'altro. Un'altra conseguenza importante è che il server e il cliente possono funzionare su macchine separate e comunicare via LAN.

X Window System è orientato al funzionamento in rete; questo significa che voi potete far funzionare il vostro server X su una "piccola" macchina locale, mentre i clienti funzioneranno su una macchina host remota, più veloce (e più costosa), riducendo comunque il costo totale della vostra installazione di elaboratori.

GESTORI DELLE FINESTRE

Nell'architettura X né il server né il cliente conoscono l'aspetto e il comportamento del sistema di finestre. I bordi delle finestre, i meccanismi per la creazione e lo spostamento delle finestre, i menu sullo schermo e molte altre caratteristiche generali, sono realizzati da un terzo componente dell'architettura di X. Questi servizi vengono svolti da un tipo speciale di applicazione cliente, il gestore di finestra. Quando un cliente vuole creare una finestra, il cliente e il server consultano il gestore di finestra per determinare come deve essere visualizzata la finestra. Poiché il gestore di finestra è un cliente, voi potete agevolmente cambiare l'aspetto e il comportamento della

vostra installazione X, cambiando il gestore di finestra che state usando. I singoli utenti possono lavorare in differenti ambienti X usando semplicemente differenti gestori di finestra.

Nei sistemi SVR4 vengono offerti tre principali gestori di finestra; ciascuno di essi fornisce servizi molto differenti, come pure differenti aspetti e comportamenti; voi potrete scegliere secondo le vostre preferenze. Il gestore di finestra OPEN LOOK **olwm** è lo standard ufficiale di SVR4; implementa un'interfaccia utente raccomandata da AT&T e Sun Microsystems, che hanno sviluppato insieme lo standard. Il gestore di finestra Motif **mwm** fornisce un aspetto e un comportamento molto simili a quelli di OS/2 Presentation Manager; è stato sviluppato dal consorzio OSF come alternativa di OPEN LOOK, per portare in X l'aspetto e il comportamento di OS/2 e Microsoft Windows. Se conoscete già OS/2 o Windows, potete dare la preferenza a Motif. Ambedue questi gestori forniscono un aspetto e un comportamento fissi, che seguono differenti standard pubblicati; tuttavia, sono configurabili i colori dello schermo e alcune altre caratteristiche. La terza alternativa, *Tom's Window Manager* (**twm**), è così chiamata dal suo creatore, Tom LaStrange, ma spesso viene chiamata anche *Tab Window Manager*. Si tratta di uno strumento totalmente configurabile, che consente agli utenti ampia libertà nella creazione dell'ambiente da loro preferito. Ovviamente, il configurare **twm** è un'operazione molto più difficile che accettare uno standard predefinito; i progettisti sia di **olwm** che di **mwm** hanno svolto numerose prove presso gli utenti per definire un aspetto e un comportamento ottimali, di facile apprendimento e uso. Una quarta alternativa, *Generic Window Manager* (**gwm**), è un nuovo gestore di finestra ancora più configurabile di **twm**, ma non altrettanto diffuso. È possibile configurare **gwm** in modo da emulare sia **mwm** che **olwm**; tuttavia, **gwm** impiega un linguaggio di controllo tipo LISP, abbastanza difficile da conoscere a fondo. Gestori di finestra meno recenti, come **awm** e **uwm**, sono profondamente obsoleti, e in futuro non verranno certamente mantenuti.

In pratica, alcune parti dell'interfaccia utente possono essere incluse nelle applicazioni stesse, invece che nel gestore di finestra. Può capitarvi di acquistare applicazioni commerciali che sono state progettate per essere usate solo con **mwm**, oppure solo con **olwm**; speciali *X toolkit* assistono lo sviluppatore nel rispettare gli standard di interfaccia utente. Quando acquistate applicazioni X in commercio, assicuratevi di stare acquistando la versione adatta per il vostro gestore di finestra che preferite.

Il resto di questo capitolo presuppone un ambiente OPEN LOOK, tuttavia, molti comandi OPEN LOOK hanno equivalenti comandi Motif e anche equivalenti generici in X. Se state usando un'altra implementazione di X, consultate la vostra documentazione per determinare i corrispondenti nomi dei comandi scelti dal produttore della vostra versione.

23.2 Avviamento di X

Di solito, X viene avviato dallo shell di login, ma potete anche predisporre il vostro **.profile** per avviare X automaticamente al login. Nel sistema SVR4 il comando **olinit** (OPEN LOOK initialize) avvia una sessione X; il comando **xinit** (X initialize) è il suo equivalente generico, che molti sviluppatori del sistema includono in una procedura chiamata **xstart**. Tutti questi comandi svolgono le stesse funzioni fondamentali: lanciano il server X, stabiliscono l'ambiente per ospitare X ed eseguono una procedura shell di avvio dal vostro home directory. In questa procedura di comandi potete includere qualsiasi comando o dati di avvio che riteniate opportuni; in questo capitolo troverete in seguito altre informazioni su come configurare l'avvio di X.

Prima del lancio di OPEN LOOK per la prima volta in un nuovo identificatore di login, il superuser deve stabilire un ambiente X iniziale col comando **oladduser**, come nell'esempio:

```
# oladduser loginid
```

loginid è il nome dell'utente per il quale deve essere configurato l'ambiente OPEN LOOK. Il comando **oladduser** installa diversi file nell'home directory dell'utente; dopo questo, potete lanciare X dallo shell, semplicemente con

```
$ olinit
```

Potete includere il comando di lancio nel vostro **.profile** per entrare immediatamente in X quando entrate in login; tuttavia, se usate anche terminali connessi tramite chiamata telefonica, configurate il vostro **.profile** per escludere X.

Dopo l'avvio di X, vedrete un display con finestre multiple, generalmente simile a quello rappresentato nella Figura 23.1. Potete muovere il puntatore a freccia sullo schermo, movendo il mouse.

Lo sfondo chiaro è chiamato *Workspace* (area di lavoro), *desktop* (scrivania), *root window* (finestra radice). Le singole applicazioni avranno le loro proprie finestre all'interno di questo sfondo chiaro. In basso a destra dello schermo appaiono due applicazioni, un orologio (il cliente **xclock**), e uno strumento di notifica della posta (**xbiff**). La grande finestra in alto è un emulatore di terminale (il cliente **xterm**), che consente di eseguire normali processi shell e quindi qualsiasi applicazione UNIX in vecchio stile. La grande finestra in basso a sinistra è il gestore di file OPEN LOOK (*File Manager*), uno strumento per esaminare file e directory, che ha mutuato molte idee dai sistemi Macintosh e Presentation Manager. Potete avere contemporaneamente sullo schermo quante finestre vi occorrono, tuttavia, se ne avete molte, le prestazioni complessive risulteranno ridotte.

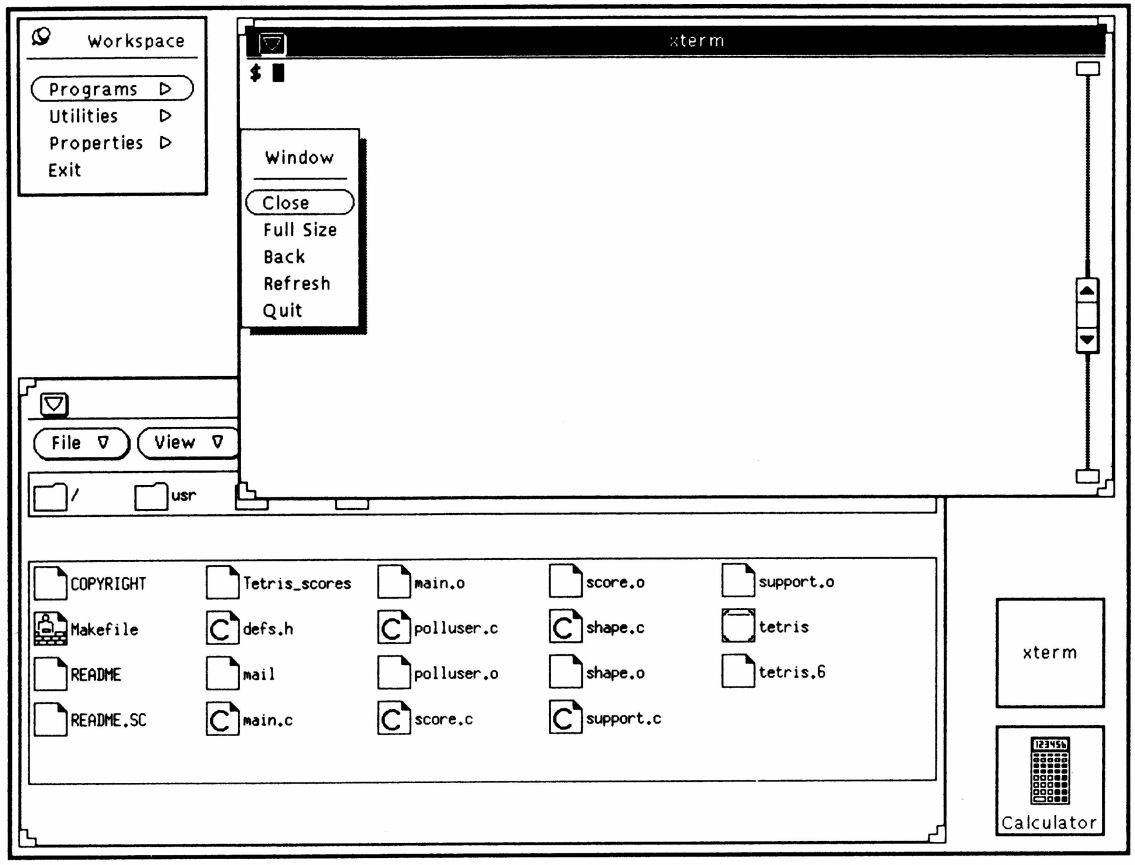


Figura 23.1 Il menu della finestra OPEN LOOK.

Per concludere la sessione X e tornare in shell con le predisposizioni di default, muovete il puntatore del mouse nella finestra radice, quindi premete e trattenete il bottone destro del mouse. Apparirà il menu *Workspace*, mostrato nella parte superiore sinistra della Figura 23.1; muovete il mouse fino a portare il puntatore sulla voce *Exit*, quindi rilasciate il bottone. Vi può venire richiesta una conferma della conclusione della sessione, con un altro click di mouse, quindi la sessione X terminerà; verrete riportati nel vostro shell di login, oppure verrete direttamente disconnessi dal login.

23.3 La variabile di ambiente DISPLAY

Poiché X è orientato al funzionamento in rete, non esiste motivo di presumere che i clienti X visualizzino le loro finestre su un determinato display nella rete; in altri termini, un cliente può essere in funzione in una macchina e la finestra relativa apparire su un'altra macchina. Per questo motivo, occorre informare X di quale macchina (e display) state usando, assegnando ed esportando la variabile di ambiente *DISPLAY*, che contiene il nome del vostro server X, nella macchina host. A questo scopo, usate questi comandi:

```
$ DISPLAY = hostname:0.0
$ export DISPLAY
```

hostname è il nome della macchina nella quale è in funzione il server X; molto spesso, quando il server è in funzione nella stessa macchina delle applicazioni clienti, questo sarà **uname** della vostra macchina. La prima cifra dopo i due punti indica il numero del display per quella macchina server (di solito zero) e la cifra dopo il punto indica il numero di schermo nel display. Nella maggior parte dei casi saranno ambedue zero. Potete anche usare

```
$ DISPLAY = hostname:0
```

In questo caso, il numero di schermo assume il valore di default zero. Se la variabile di ambiente non è predisposta, X tenterà di aprire la console del sistema che state usando, assumendo questo valore di DISPLAY:

```
$ DISPLAY = UNIX:0.0
```

Se X non riesce ad aprire questo dispositivo, riceverete un messaggio d'errore, come nell'esempio:

```
$ olinit
olinit: Can't open display
$
```

Quando vedete questo messaggio dovete controllare che la vostra variabile DISPLAY sia predisposta correttamente.

Spesso, i messaggi di errore come questo non appaiono direttamente alla console, ma vengono inviati nel file **$HOME/.oliniterr**; se **olinit** non funziona come vi aspettavate, ma non vedete messaggi di errore, leggete il file **.oliniterr**.

23.4 Gestione delle finestre

Quando X è in funzione, voi potete creare, spostare, o cancellare le finestre quando occorre. Queste procedure differiscono tra i diversi gestori di finestra, essendo in stretta dipendenza dei diversi metodi usati per gestire le finestre. Nel capitolo si presuppone che stiate usando **olwm**.

X Window System include uno *screensaver* (salva-video) automatico, che cancella lo schermo dopo dieci minuti senza attività alla tastiera o al mouse; comunque, non viene mai interrotta né perduta l'uscita sullo schermo. Per riattivare lo schermo, muovete leggermente il mouse.

DESTINAZIONE DELL'INPUT

L'uscita destinata a una finestra appare tutte le volte che l'applicazione la scrive; questo è uno schema di output non-bloccante. Al contrario, l'entrata da tastiera o da mouse deve essere diretta dall'utente a una determinata finestra; in entrata può essere selezionata solo una finestra alla volta. Se nessuna finestra è selezionata in entrata, o se è selezionata la finestra radice, l'input viene scartato dal server X. La finestra selezionata avrà una barra del titolo in alta intensità luminosa, oppure un bordo di colore differente dalle finestre non selezionate.

Per segnalare a X quale finestra volete che diventi la destinazione dell'input, potete scegliere tra due sistemi. Il primo, chiamato *click-to-type* richiede che posizioniate il puntatore del mouse all'interno della finestra e che diate un click col bottone sinistro del mouse. Questo seleziona quella finestra come destinazione dell'input; di solito, il bordo della finestra viene evidenziato con un colore contrastante per segnalarne la selezione. Potete ora spostare il mouse ovunque sullo schermo, la finestra selezionata continuerà a ricevere l'input. Questo è il metodo di default in OPEN LOOK.

Il secondo metodo, detto *focus-follows-mouse*, non è disponibile in tutte le release OPEN LOOK. Usando questo metodo, dovete posizionare il puntatore del mouse all'interno della finestra che volete diventi la destinazione dell'input; quando spostate il puntatore in un'altra finestra, la destinazione dell'input cambia. Per scegliere questo metodo, all'interno dei vostri file **.olinitrc** o **.xinitrc** dovete eseguire **olwm** con l'opzione **-f** (focus) in linea di comando, come descritto nel seguito del capitolo.

Se avete disponibili ambedue i metodi, provateli entrambi e scegliete quello che preferite.

CONTROLLO DELLA POSIZIONE DELLA FINESTRA

Le molteplici finestre sullo schermo si possono sovrapporre, e voi potete cambiare la dimensione e la posizione delle finestre secondo le vostre necessità. Potete anche iconizzare una finestra, ovvero renderla molto piccola, e collocarla in una posizione ove non dia fastidio, senza con questo arrestare l'applicazione. Queste operazioni differiscono a seconda del gestore di finestra che state usando.

Con **olwm** per iconizzare una finestra dovete muovere il puntatore del mouse nel piccolo box nell'angolo a sinistra in alto del bordo della finestra, quindi premere il bottone sinistro del mouse; apparirà il menu *Window*, mostrato nella Figura 23.1. Trattenendo abbassato il bottone, spostate il mouse sull'opzione Close, quindi rilasciate il bottone. La finestra reale scomparirà e verrà sostituita da una rappresentazione in piccolo della finestra; questa icona di solito apparirà nella parte sinistra in basso dello schermo. Se iconizzate una finestra **xterm**, l'icona conterrà un piccolo disegno, oppure la scritta *xterm*.

Per ripristinare un'icona alla dimensione piena, dovete dare due click del bottone sinistro del mouse, mentre il puntatore si trova sull'icona; oppure, scegliere Restore dal menu Window associato a quella icona.

Oltre all'iconizzazione di una finestra, il menu Window consente di espandere una finestra all'intera dimensione dello schermo, mettere una finestra in fondo al gruppo di finestre sovrapposte, rivisualizzare una finestra e il suo contenuto e infine uscire dall'applicazione. Potete invocare il menu Window con un click del bottone destro del mouse in qualunque posizione nel bordo della finestra.

Per spostare una finestra nello schermo, posizionate il puntatore sul bordo della finestra (uno qualsiasi, ma il bordo superiore risulta più comodo); quindi, premete il bottone sinistro del mouse e tenetelo abbassato. Muovendo il mouse vedrete una rappresentazione schematica della finestra muoversi nello schermo; posizionate lo schema dove volete la finestra, e rilasciate il bottone del mouse. La finestra comparirà nella nuova collocazione.

Il click del bottone sinistro del mouse sul bordo della finestra mette la finestra in cima al gruppo delle finestre sovrapposte, esponendone il contenuto per le vostre necessità.

SELEZIONE DI FINESTRE MULTIPLE

Nella terminologia OPEN LOOK, la pressione del bottone sinistro del mouse nel bordo della finestra compie un'operazione di selezione; seleziona una

finestra e deseleziona tutte le altre. Inoltre, porta la finestra selezionata in cima al gruppo delle finestre sovrapposte. La pressione del pulsante mediano nel bordo della finestra compie un'azione di aggiustamento; alterna lo stato selezionato della finestra, senza portarla in cima o deselezionare altre finestre.

Usando questi due bottoni, potete selezionare molteplici finestre. Selezionate la prima finestra col click del bottone sinistro nel bordo della finestra, quindi selezionate una seconda finestra col click del bottone mediano nel bordo di questa finestra. Adesso avete selezionato due finestre; se premete e tenete abbassato il bottone sinistro nel bordo di una qualsiasi delle due finestre, potete spostarle ambedue.

Come vedrete più avanti in questo capitolo, anche altre funzioni associate col menu Workspace interessano tutte le finestre selezionate. Potete deselezionare una finestra col click del bottone mediano del mouse nel bordo della finestra; potete deselezionate tutte le finestre meno una, col click del bottone sinistro del mouse nel bordo di questa finestra.

RIDIMENSIONAMENTO DI UNA FINESTRA

Le piccole aree a forma di **L** negli angoli dei bordi delle finestre sono usate per ridimensionare una finestra. Se posizionate il puntatore in una di queste aree d'angolo, e premete e tenete abbassato il pulsante sinistro del mouse, potete allungare o comprimere la finestra, fino a che non rilasciate il pulsante. Apparirà uno schema della finestra per mostrarne la nuova dimensione e forma. Notate che l'angolo opposto rimane immobile, cosicché non potete ridimensionare e spostare una finestra in una sola operazione. Non potete neppure ridimensionare un'icona.

IL MENU WORKSPACE

Quando spostate il puntatore nell'area di sfondo al di fuori di ogni finestra e premete il pulsante destro del mouse, apparirà il menu *Workspace* o *Function*, mostrato in alto a sinistra nella Figura 23.1. Questo menu vi dà la possibilità di lanciare nuove applicazioni, uscire totalmente dall'ambiente X, cambiare le vostre predisposizioni ed eseguire altre funzioni.

Il menu Workspace e molti altri menu OPEN LOOK, possono essere "appesi a uno spillo" nello schermo, in modo che non scompaiano dopo che avete fatto una selezione. Tenete abbassato il pulsante destro del mouse e spostate il puntatore sul disegno di uno spillo nell'angolo a sinistra in alto del menu; lo spillo si muoverà e il menu verrà appuntato sullo schermo. Per staccare un menu dallo spillo date un click sullo spillo col bottone sinistro del mouse.

La funzione Exit conclude **olwm** e quindi l'intera sessione X, riportandovi nel vostro shell di login.

Le voci nel menu Workspace possono contenere sottomenu; questi sono menu aggiuntivi associati alle voci del menu principale. Se fate scorrere il puntatore verso destra, vicino alla punta della freccia di una delle voci del menu principale, apparirà il sottomenu sopra lo spazio del menu Workspace. A questo punto potete selezionare voci dal sottomenu prima di rilasciare il bottone del mouse.

La voce di menu Programs contiene un elenco delle applicazioni che potete eseguire direttamente, senza la necessità di usare lo shell **xterm** per lanciarle; si tratta di solito di programmi clienti X che svolgono servizi utili. Quando lanciate una di queste applicazioni mediante un click sulla relativa voce del menu, di solito apparirà sullo schermo la finestra corrispondente e potrete usare l'applicazione. Per terminare l'applicazione dovrete usare la procedura prevista dalla stessa applicazione.

Potete configurare parte del contenuto dei sottomenu Programs, e talvolta dell'intero menu Workspace, modificando, con editor, il file esistente nel vostro home directory, fra i due file **.olprograms** oppure **.openwin-menu**; comunque, di solito il menu Properties include una voce per cambiare l'elenco dei programmi.

Il sottomenu Utilities fornisce diversi utili servizi; in Tabella 23.1 ne viene dato un tipico elenco. Se presente, la voce *Lock Screen* lancia il cliente X **xlock**, che cancella lo schermo e visualizza un disegno matematico, fino a che non introducete la vostra password; questo vi consente di mantenere in login il vostro terminale, ma protetto contro l'uso di estranei. Tenete però conto che l'applicazione **xlock** consuma notevoli risorse di CPU, perciò il suo uso è sconsigliato se avete in macchina lavori di lunga durata. L'applicazione Print Screen contiene diversi sottomenu che consentono di selezionare una parte del display per la stampa. La voce Network Administration consente di cambiare l'elenco delle macchine remote (host) che possono creare finestre sul vostro display. Se avete una LAN e volete eseguire clienti X su altre macchine, ma ottenere le loro finestre sul vostro display, dovete elencare i nomi degli host relativi sotto questa voce; questo menu esegue l'applicazione **xhost**.

La voce di menu Properties, che compare in diversi menu OPEN LOOK, è usata per cambiare la configurazione, ovvero le proprietà, di singoli menu, o dell'intero display. Queste funzioni di solito fanno parte dell'appli-

Tabella 23.1 Tipico sottomenu Utilities di OPEN LOOK.

Nome	Descrizione
File Manager	Invoca il File Manager di OPEN LOOK
Network Administration	Modifica l'accesso host al vostro display X
Refresh	Rivisualizza l'intero schermo
Print Screen	Stampa il contenuto dello schermo
Lock Screen	Lancia **xlock**

cazione **olwsm** (Workspace Manager), ma possono essere incluse in un programma separato **xproperties** o **xset**. Da questo menu potete selezionare la forma di caratteri preferita, il colore del display, la posizione delle icone ed altri elementi variabili; poiché le proprietà disponibili differiscono enormemente tra le diverse versioni, dovrete verificare le possibilità mediante delle prove.

BARRE DI SCORRIMENTO

Alcune finestre possono avere delle barre di scorrimento orizzontali o verticali; queste barre consentono di fare scorrere il contenuto della finestra all'interno dei bordi della finestra stessa. La Figura 23.1 mostra una grande finestra **xterm** con una barra di scorrimento OPEN LOOK verticale. La barra include diverse zone che attivano differenti funzioni quando le toccate col puntatore del mouse. In basso e in alto trovate gli ancoraggi del "cavo", che connette un "ascensore" centrale; nell'ascensore, una freccia verso l'alto e una freccia verso il basso, racchiudono una piccola area centrale di trasporto. Per spostare il contenuto della finestra dentro la sua regione di scorrimento, posizionate il puntatore nell'area di trasporto e premete il bottone sinistro del mouse; potete adesso trasportare l'ascensore su o giù (oppure a destra o a sinistra) per far scorrere il contenuto della finestra. Potete scorrere la finestra di una linea mediante un click sulle frecce, verso l'alto o verso il basso, oppure potete spostarvi direttamente all'inizio o alla fine della regione di scorrimento, mediante un click sull'ancoraggio in alto o in basso del cavo. Per spostare la finestra di un'intera schermata, date un click nella zona del cavo. Per utilizzare queste tecniche, l'applicazione deve supportare il formato OPEN LOOK delle barre di scorrimento. Alcune applicazioni possono usare un formato differente dalla Figura 23.1, ma la maggior parte delle applicazioni OPEN LOOK impiega una sequenza simile nelle manipolazioni col mouse.

La dimensione della regione di scorrimento è determinata dall'applicazione stessa, che ricorda una propria "storia"; alcune applicazioni possono ricordare una regione molto estesa, ma, in ogni caso, potete solo scorrere all'interno dell'area limitata che l'applicazione può memorizzare.

Alcune applicazioni OPEN LOOK consentono di suddividere una finestra in pannelli separati, ciascuno controllato da proprie barre di scorrimento. Di solito è possibile suddividere una finestra sia in orizzontale che in verticale. Consultate la vostra documentazione OPEN LOOK per conoscere le applicazioni che consentono questa funzione.

IL SISTEMA HELP

OPEN LOOK e molti altri ambienti X includono un sistema di aiuto basato su finestre, che quando occorre fornisce informazioni e notizie. Di solito il

sistema di aiuto è associato al tasto funzione F1; comunque, questo può essere cambiato tramite il menu Properties, oppure tramite il database delle risorse X, trattato più avanti in questo capitolo. Per richiedere Help, spostate il puntatore del mouse su una finestra o un altro oggetto sullo schermo, quindi premete F1; il risultato assomiglierà a quello mostrato nella Figura 23.2. Il sistema Help fornisce informazioni su molti oggetti; se le informazioni di aiuto non sono disponibili, il sistema rimane silente, oppure risponde con un messaggio "No help available". Di solito la finestra di aiuto quando appare viene appuntata sullo schermo; potete staccarla e abbandonarla tramite un click sullo spillo. Anche i menu transitori possono disporre di informazioni di aiuto, ma dovrete probabilmente attaccarli sullo schermo, perché il gestore di Workspace possa riconoscere la finestra e trovare il relativo testo di aiuto.

23.5 Il File Manager

Il *File Manager* fornisce all'utente un accesso al sistema UNIX tramite un'interfaccia grafica. Un esempio della visualizzazione del File Manager appare a sinistra in basso nella Figura 23.1; si tratta del File Manager OPEN LOOK, ma molti pacchetti X includono un'applicazione simile, come menzionato brevemente nel Capitolo 4. Potete lanciare il File Manager dal sottomenu Utilities del menu Workspace.

Il File Manager vi consente di muovervi all'interno del file system, di selezionare un sottoinsieme di file in un directory per un'operazione di spostamento o cancellazione e di aprire subdirectory. Se date un ckick col bottone sinistro del mouse su un oggetto, voi selezionate quell'oggetto per un'azione, a sua volta selezionata dai menu nella parte superiore sinistra della finestra File Manager. Col bottone mediano potete selezionare molteplici oggetti per una stessa azione.

Con un doppio click del bottone sinistro del mouse su un oggetto, voi causate l'apertura dell'oggetto; se si tratta di un directory (folder, o cartelletta), il contenuto dell'oggetto rimpiazzerà il contenuto della finestra grande, e il nome del directory apparirà nell'elenco nel mezzo della finestra di File Manager. Altri oggetti provocano azioni differenti. Un file di testo viene trattato in una propria finestra dall'editor scelto in base alla vostra variabile di ambiente EDITOR; anche un programma eseguibile viene eseguito in una propria finestra. Quando un'applicazione termina, o viene conclusa con la voce Quit nel menu Window, la sua finestra scompare.

OPERAZIONI DI TASPORTO

Una operazione di trasporto (*drag-and-drop*) consente di copiare oggetti da una locazione a un'altra. Per esempio: su un file o un directory date un click

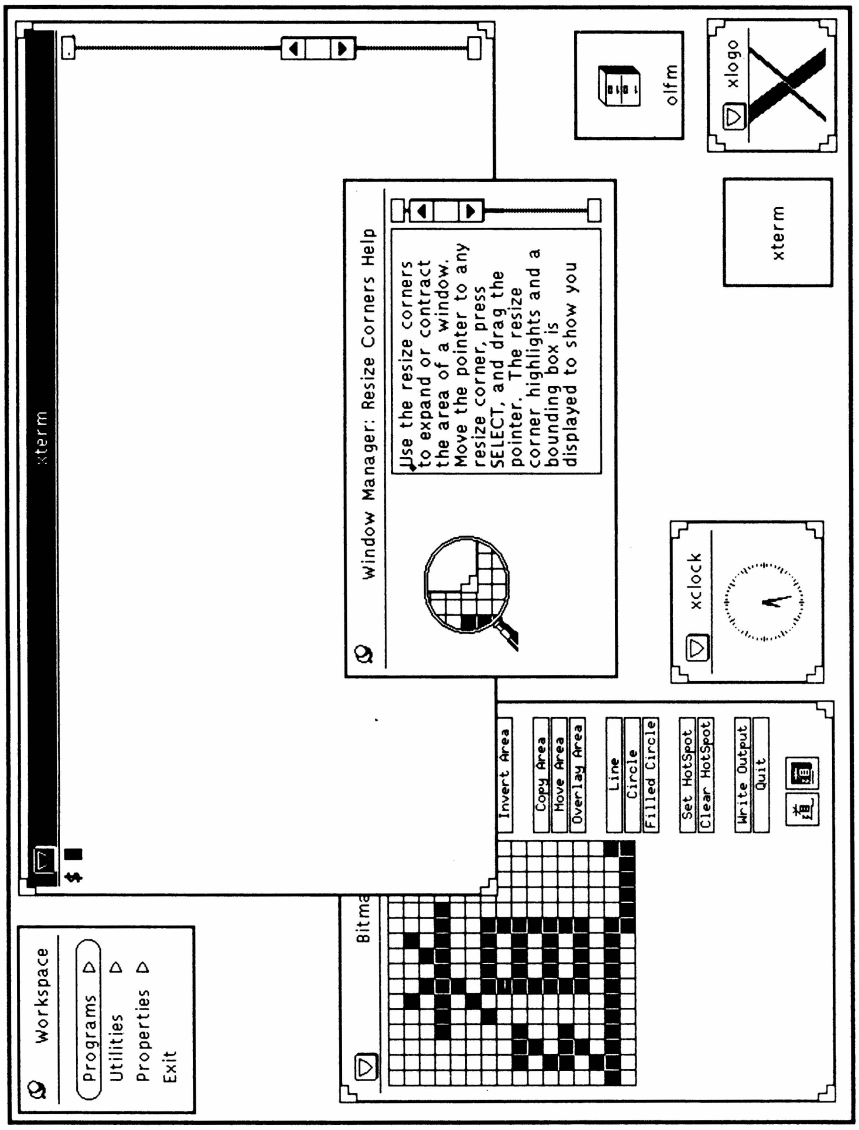


Figura 23.2 Il menu di aiuto di OPEN LOOK.

col bottone sinistro del mouse, quindi spostate il puntatore del mouse trattenendo abbassato il bottone; il mouse trasporterà quell'oggetto nella nuova cartelletta, e quando rilasciate il bottone l'oggetto sarà copiato nella nuova locazione. Potete fare copie di file singoli e di interi directory; invece, la funzione multipla del bottone centrale in molte release non ha effetto. Se assieme al bottone select tenete abbassato anche il tasto delle maiuscole, l'oggetto viene spostato anziché copiato.

Se trasportate un directory in Workspace (sfondo) e quindi rilasciate il bottone, creerete una nuova istanza del File Manager; questa nuova finestra conterrà il directory che avete trasportato. A questo punto, se trasportate un oggetto fra le finestre File Manager, l'oggetto verrà copiato o spostato. Potete ottenere sul video un numero qualsiasi di finestre File Manager distinte.

Anche certe applicazioni consentono di ricevere oggetti con questa tecnica nella loro finestra, ma non tutte, in quanto l'operazione deve essere prevista dalla singola applicazione. Se l'operazione è prevista, l'applicazione userà l'oggetto ricevuto come un proprio argomento; per esempio, un'applicazione Wastebasket cancellerà un file o un directory che vi avete trasportato. Per determinare quali applicazioni supportano questa operazione consultate la documentazione di ciascuna applicazione.

CORRISPONDENZA DI NOMI CON MODELLI

La linea Directory in alto a destra nella finestra File Manager vi permette di introdurre da tastiera un pathname, quindi di dare un click sulla voce di menu Match per entrare in quel directory. Nel directory selezionato potete selezionare un sottoinsieme di file, introducendo nel campo Pattern un modello di nome di file con metacaratteri (i familiari operatori * e ?), prima di dare il click su Match.

Con i menu File, View ed Edit sono disponibili funzioni di sistema ancora più complesse. Il File Manager è un'interfaccia realmente semplice col file system UNIX.

23.6 Applicazione cliente xterm

La più importante applicazione cliente è il programma **xterm** (X terminal); fornisce una finestra di emulazione di terminale, in cui una normale shell è in attesa dei vostri comandi. In una finestra **xterm** potete eseguire qualsiasi comando UNIX, o procedura shell; potete anche lanciare altri clienti X, quando vi occorrono. Di solito, quando lanciate l'ambiente X, viene lanciato almeno un **xterm**; potete creare finestre **xterm** addizionali, sia tramite il menu Workspace, che tramite shell (che lavora in un'altra finestra **xterm**!); quindi potete lanciare applicazioni in queste finestre.

Per lanciare una finestra **xterm** da shell, usate

```
$ xterm &
```

Dopo un'attesa, la nuova finestra comparirà nella posizione e nelle dimensioni di default. Nella finestra sarà in funzione lo shell specificato nella vostra variabile di ambiente SHELL. Notate che le applicazioni lanciate in questo modo da shell, sono in genere eseguite in background (con **&**), perché le applicazioni X vengono di solito terminate da voci di menu, anziché direttamente con CTRL-D. Potete terminare una finestra **xterm** uscendo dalla sua shell, sia con CTRL-D, che con il comando **exit**, oppure anche selezionando Exit nel relativo menu.

Quando viene creata una finestra **xterm**, il server X predispone una variabile TERM per quella finestra nell'ambiente UNIX, in modo che le normali applicazioni a tutto video, come **vi**, possano funzionare correttamente. Il cliente **xterm** emula i terminali DEC VT102 e Tektronix 4014; spesso emula anche il modo nativo della console di sistema, ma non perfettamente.

È preferibile assegnare il valore

```
TERM = xterm
```

anziché la console standard o VT102.

Potete spostare o iconizzare la finestra senza interferire con alcuna applicazione in corso nella finestra, ma se cancellate la finestra anche ogni applicazione in corso in essa verrà terminata, a meno che non abbiate specificato **nohup** al lancio dell'applicazione. Potete anche ridimensionare la finestra **xterm**, ma le applicazioni in corso di esecuzione non riceveranno notifica del cambiamento delle dimensioni della finestra. Invece, se, terminata un'applicazione, ridimensionate la sua finestra **xterm**, quindi rilanciate l'applicazione, questa si adatterà al nuovo formato della finestra.

MENU xterm

Come molti clienti X, **xterm** ha un proprio insieme di menu per la gestione delle proprie funzioni; questi menu hanno aspetti differenti nelle diverse release. In alcune versioni, potete richiamare un menu premendo il bottone destro del mouse mentre il puntatore è all'interno della finestra di **xterm**; in questo caso avrete un solo menu. In altre versioni, dovete tenere abbassato il tasto CTRL e premere uno dei bottoni del mouse; in questo caso avrete tre menu **xterm**, uno per ogni bottone del mouse. SVR4.0 supporta il primo caso, perciò rende disponibile di solito un solo menu, che comunque contiene la maggior parte delle voci della versione con tre menu. Questo menu, mostrato nella Figura 23.3, consente di inviare segnali ai processi in corso nella finestra, di rinfrescare (rivisualizzare) il contenuto del video, di predisporre particolari proprietà e di svolgere molte altre funzioni.

Potete selezionare il display grafico Tektronix 4014 con la voce Show Tek del menu. Entrando nel modo 4014 compariranno dei sottomenu; comunque, questa funzione è necessaria solo nel caso che abbiate software specifico per il terminale 4014. Di solito la finestra 4014 appare in aggiunta alla finestra normale, perciò potete annullare la finestra 4014 senza annullare anche la finestra **xterm**.

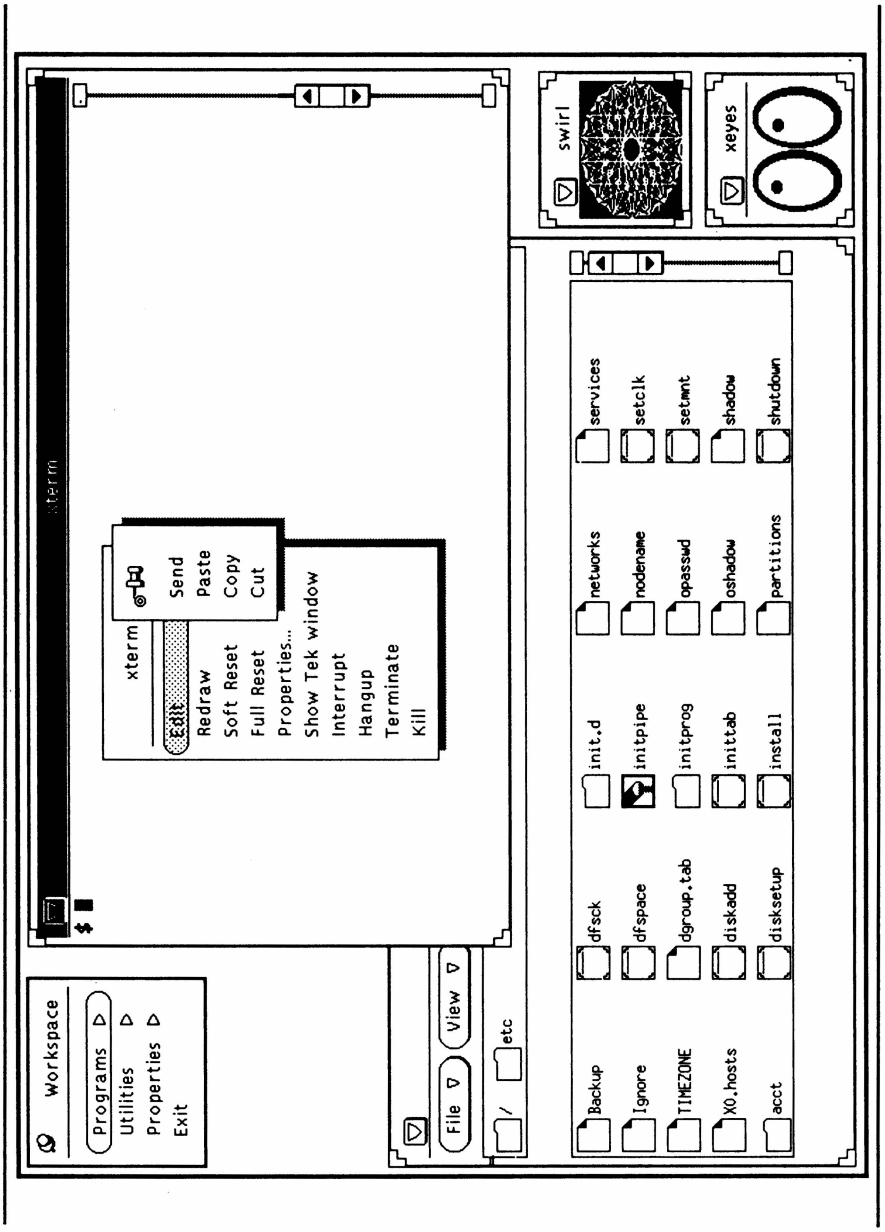


Figura 23.3 Il menu **xterm** di OPEN LOOK SVR4.0.

SPOSTA E COPIA CON xterm

L'applicazione **xterm** consente di selezionare (*select* o *cut*) un blocco di testo in una finestra e applicarlo (*paste*), spostandolo o copiandolo, in un'altra finestra, oppure in un'altra posizione nella stessa finestra. Questa è la funzione del sottomenu Edit mostrato nella Figura 23.3. Anche altri clienti X supportano questa operazione; utilizzando questi clienti potrete spostare o copiare tra applicazioni diverse.

Il meccanismo delle funzioni di spostamento (cut-and-paste) e di copia (copy-and-paste) differisce grandemente fra le diverse implementazioni di X e persino nelle differenti versioni delle specifiche OPEN LOOK; in pratica, per queste funzioni dovrete adottare il meccanismo previsto nel vostro sistema.

Se nella finestra **xterm** sono abilitate le barre di scorrimento, potete scorrere su una parte precedente del testo, selezionarla e quindi reintrodurla in un editor oppure nella linea di comando **xterm**; il blocco di testo selezionato viene evidenziato con un colore diverso da quello del testo circostante.

Selezionate l'inizio del blocco di testo premendo il bottone sinistro del mouse, scorrete fino alla fine del testo desiderato dove rilasciate il bottone; muovete il puntatore nella finestra in cui volete applicare il materiale selezionato e premete il bottone destro del mouse per richiamare il menu; mantenendo abbassato il bottone destro, prendete il menu Edit e scegliete Send per copiare il testo selezionato nella posizione del cursore della finestra di destinazione. In alternativa, potete segliere Copy o Cut; la funzione Copy, copia il testo selezionato in una memoria interna, detta *clipboard*; la funzione Cut, copia il testo, ma lo cancella dalla sua collocazione originale. Tutte e tre le opzioni (Send, Copy e Cut) copiano il testo in clipboard; potete usare ripetutamente Paste o Send per recuperare il contenuto di clipboard e collocarlo nella finestra corrente.

Esiste solo un'unica memoria interna clipboard, perciò ogni operazione Copy, Cut o Send scarterà il testo precedente e conserverà l'ultimo testo selezionato. Potete tuttavia "aggiustare" la selezione del testo già evidenziato sullo schermo, aggiungendo del nuovo testo: premendo il bottone centrale del mouse e scorrendo sopra un altro blocco di testo, questo testo verrà aggiunto al testo già contenuto in clipboard. Potete in qualunque momento selezionare o deselezionare una regione di testo mediante il bottone sinistro del mouse.

OPZIONI NELLA LINEA DI COMANDO xterm

Lanciando **xterm** dalla linea di comando (o anche da un'altra finestra **xterm**), potete includere molte opzioni nella linea di comando. Il cliente **xterm** supporta le opzioni della linea di comando di cliente descritte nel paragrafo successivo, più alcune opzioni addizionali specifiche di **xterm**. Per maggiori informazioni consultate la man page **xterm(1)**.

L'opzione più importante consente di scegliere un comando da eseguire nella finestra **xterm**, in luogo dello shell. A questo scopo, usate l'opzione **–e** (execute), seguita dalla linea del comando da eseguire. L'opzione **–e** e il suo comando associato devono comparire come ultimo argomento, perché **xterm** utilizza la fine della linea di comando per determinare la fine del comando da eseguire. Per esempio, volendo lanciare allo stesso tempo **xterm** e l'editor **vi** al suo interno, potete usare questo comando:

```
$ xterm –e vi nomefile &
```

Quando viene usata con l'opzione **–e**, l'applicazione **xterm** si conclude alla conclusione dell'applicazione associata, perciò la finestra creata nell'esempio precedente scomparirà quando voi uscirete dall'editor. Questa funzionalità è utile spesso in procedure di comandi, o in ambienti in rete, dove con l'opzione **–e** potete usare il comando **rlogin** per lanciare una sessione in una macchina remota.

23.7 Argomenti generici in linea di comando di X

Il cliente **xterm**, come molti altri clienti X, supporta molte opzioni di linea di comando che consentono di controllare l'apparenza delle loro finestre. Potete specificare la geometria della finestra (dimensioni e posizione della finestra sullo schermo), i colori della finestra, il tipo di carattere da usare per il testo nella finestra, l'etichetta nel bordo della finestra e nell'icona e molti altri attributi della finestra. Potete usare questi argomenti di linea di comando non solo con **xterm**, ma con la maggior parte dei clienti. Queste opzioni sono in particolare molto utili per personalizzare il sottomenu Programs nel menu Workspace.

GEOMETRIA DELLA FINESTRA

Per selezionare le dimensioni e la posizione della finestra, usate l'argomento **–geometry**, seguito da una specifica nella forma

```
COLSxROWS+XLOC+YLOC
```

dove *COLS* e *ROWS* sono le dimensioni della finestra **xterm**, espresse in caratteri e *XLOC* e *YLOC* danno la posizione iniziale della finestra, misurata in pixel, a partire dall'angolo in alto a sinistra dello schermo. Per esempio, una finestra di 24 righe per 80 colonne, posizionata in orizzontale 100 pixel, e in verticale 200 pixel, dall'angolo in alto a sinistra, viene specificata con

```
$ xterm –geometry 80x24+100+200 &
```

Potete omettere una di queste due classi di informazioni e accettare i relativi valori di default; per esempio, per creare una finestra 24×80 nella posizione di default, potete usare

```
$ xterm -geometry 80x24 &
```

e per creare una finestra delle dimensioni di default, nella posizione +200+300, potete usare

```
$ xterm -geometry +200+300 &
```

Tutte le precedenti specificazioni di locazione creano uno spostamento della finestra a partire dall'angolo in alto a sinistra dello schermo; in alternativa, potete definire spostamenti a partire da un qualunque angolo dello schermo. Nelle specifiche di locazione, il **+** (più) assume il lato sinistro come punto zero orizzontale e il lato superiore come punto zero verticale; il **—** (meno) assume il lato destro come zero orizzontale e il lato inferiore come zero verticale. Per esempio:

```
$ xterm -geometry -200+300 &
```

collocherà la finestra nello schermo a 300 pixel dal lato superiore e a 200 pixel dal lato destro. Sono ammesse tutte le combinazioni possibili.

Tenete presente che, diversamente da **xterm**, molti clienti X richiedono la specificazione della finestra in pixel, anziché in caratteri; se vedete apparire finestre in strane dimensioni, controllate la coerenza di questi parametri.

TIPI DI CARATTERI

Anche il tipo di carattere, o font, usato all'interno della finestra del cliente può essere predisposto nella linea di comando. Usate l'opzione **-font** seguita dal nome del font da usare; per esempio:

```
$ xterm -font 8x13bold &
```

In questo esempio, *8x13bold* è un nome di font noto a X. Molte implementazioni di X supportano numerosi font, in alcuni casi fino a 600; per facilitare la scelta di un font, il cliente **xlsfonts** (X list fonts) elenca tutti i nomi di font conosciuti in una implementazione di X, come nell'esempio:

```
$ xlsfonts | tail -15
-adobe-courier-bold-o-normal--14-100-100-100-m-90-iso8859-1
-adobe-courier-medium-o-normal--14-140-75-75-m-90-iso8859-1
-adobe-symbol-medium-r-normal--12-120-75-75-p-74-adobe-fontspecific
-bitstream-charter-bold-i-normal--19-180-75-75-p-117-iso8859-1
-bitstream-charter-medium-i-normal--19-180-75-75-p-103-iso8859-1
```

```
6x10
8x13
8x13bold
9x15
cursor
fixed
pc12x20
pc6x10
vrb-25
btbold
$
```

Come potete vedere, i nomi dei font sono espressi in forme differenti; i lunghi nomi nelle prime cinque righe dell'esempio contengono la codifica delle dimensioni del font, densità dei punti e altre informazioni; i nomi più brevi sottostanti di solito sono sinonimi dei nomi più lunghi, perciò si possono ottenere caratteri con lo stesso aspetto, pur con nomi di font differenti.

Quando volete usare un nome di font lungo, potete usare una sintassi con metacaratteri; per esempio:

```
*-courier-bold-o-normal--*-100-*
```

è un nome accettabile per il primo font nell'elenco precedente. La man page **X(1)** di solito fornisce informazioni sulle regole per i nomi di font e sul modo come il server X risolve i metacaratteri nei nomi.

Per esaminare un particolare font, potete usare **xfd** (X font display), come nell'esempio:

```
$ xfd -fn nomefont &
```

Dopo l'opzione **-fn** indicate il nome del font, come appare nell'elenco ottenuto con **xlsfonts**. L'applicazione mostra una griglia di tutti i caratteri nel font specificato. Potete, volendo, spendere un poco di tempo a provare diversi font, prima di decidere quale preferire nelle vostre finestre **xterm**.

I font stessi risiedono in uno o più directory, in cui **xlsfonts** e le altre applicazioni ricercano il font specificato; questi directory sono indicati nella variabile di ambiente *XWINFONTPATH*:

```
$ echo $XWINFONTPATH
/usr/X/lib/fonts/Xol,/usr/X/lib/fonts/misc,/usr/X/lib/fonts/100dpi
$
```

Si tratta di un elenco di directory che contengono font, separati da virgole. Potete cambiare i font disponibili, cambiando semplicemente questa variabile di ambiente; purtroppo, molte applicazioni si aspettano di trovare determinati font e, in caso contrario, alcune potrebbero non funzionare. Inol-

tre, il server X ricerca nei directory suddetti il primo nome di font che corrisponde al nome specificato; perciò, usando nomi che includono metacaratteri, potreste ottenere risultati inattesi.

COLORI

Con **xterm** e con la maggior parte degli altri clienti X potete usare le opzioni **–fg** (foreground) per scegliere il colore dei caratteri e l'opzione **–bg** (background) per scegliere il colore dello sfondo. Ambedue queste opzioni richiedono come argomento il normale nome di un colore, come nell'esempio:

```
$ xterm –fg white –bg blue &
```

che crea una schermata in bianco su sfondo blu. Usate il cliente **xcolors** per ottenere l'elenco dei nomi di tutti i colori disponibili col vostro display; se il vostro sistema non dispone del cliente **xcolors**, potete ottenere l'elenco dei colori con il menu Properties di Workspace. Se avete un display monocromo, l'unica scelta possibile è la seguente:

```
$ xterm –fg white –bg black &
$ xterm –fg black –bg white &
```

In sostituzione dei due comandi precedenti potete usare il comando

```
$ xterm –rv &
```

L'opzione **–rv** (reverse video) alterna i colori dei caratteri e dello sfondo.

TITOLI E NOMI

In molte finestre potete specificare l'etichetta nel bordo superiore con l'opzione **–title**, **–tl**, oppure **–T**, che richiede come argomento una stringa di caratteri protetta; potete indicare anche l'etichetta dell'icona della finestra, con l'opzione **–n** o **–in** (icon name); per esempio:

```
$ xterm –tl "Finestra principale" –in "Shell P." &
```

In genere, il nome dell'icona del cliente è il titolo della finestra, ma alcune applicazioni usano il nome dell'applicazione come nome dell'icona.

ICONE ED HELP

L'argomento **–iconic** provoca il lancio del cliente come icona, anziché come una normale finestra. In aggiunta, molti clienti X supportano l'argomento

-help, che visualizza un messaggio di aiuto, con l'elenco degli argomenti ammessi dal cliente in linea di comando. Dopo la stampa dell'elenco, il cliente termina l'esecuzione.

23.8 Clienti X

Per X sono disponibili numerosi programmi cliente, molti dei quali sono di pubblica disponibilità, di solito in forma sorgente, e ne compaiono regolarmente di nuovi. Molti sono passatempi e giochi, programmi di utilità relativi a X stesso come **xlsfonts**, applicazioni come editor multimediali, strumenti di grafica e fogli elettronici. La Tabella 23.2 elenca la maggior parte dei più utili programmi cliente; l'intero elenco non sarà probabilmente disponibile nella vostra macchina, perciò verificate quali applicazioni avete disponibili nel directory **bin** di X, di solito **/usr/X/bin** oppure **/usr/bin/X11**. Qui di seguito ne viene descritto un piccolo numero.

IL CLIENTE xhost

Il cliente **xhost** è uno strumento di sicurezza utile se state usando X in una LAN. Il cliente **xhost**, oppure **olam**, controlla le altre macchine nella rete abilitate a visualizzare finestre sul vostro display. In effetti, voi potete eseguire il server X su una macchina della rete e il cliente X su un'altra macchina; se ciò è abilitato da **xhost**, il cliente è in grado di visualizzare le proprie finestre sul vostro schermo.

Per conoscere quali altre macchine possono creare finestre sulla vostra macchina, usate **xhost** senza argomenti nella linea di comando, come nell'esempio:

```
$ xhost
access control enabled (only the following hosts are allowed)
localhost
yoursys
$
```

Potete abilitare l'accesso di tutte le altre macchine, con

```
$ xhost +
all hosts being allowed (access control disabled)
$
```

oppure disabilitare l'accesso per tutte le macchine con

```
$ xhost -
all hosts being restricted (access control enabled)
$
```

Tabella 23.2 Comuni applicazioni di clienti X.

Nome	Tipo	Descrizione
olwm	Window Manager	OPEN LOOK
mwm		Motif
twm		wm ufficiale X11R4 di MIT
gwm		wm generico tipo LISP
xauth	Utility	Gestisce lista di autorizzazione di accesso utenti alla macchina
xev		Visualizza il contenuto di ogni evento X
xfd		Visualizza i font
xhost		Gestisce la lista di host che possono creare finestre sul vostro display
xlock		Blocca lo schermo; password per sbloccarlo
xlsfonts		Lista i nomi di tutti i font disponibili
xmodmap		Carica nell'ambiente una mappa X di tastiera
xperfmon		Visualizza un grafico del carico di CPU e di rete
xpr		Prepara una finestra per la stampa
xrdb		Carica nell'ambiente un database di risorse X
xrefresh		Rivisualizza l'intero schermo
xset		Predispone le preferenze dell'utente (properties)
xuwd		Visualizza una finestra scaricata da **xwd**
xwd		Scarica in un file una mappa di bit di una finestra
olam	OPEN LOOK	Controllo dell'accesso alla rete
olfm		Gestione dei file
olpixmap		Editor di mappe di pixel (a colori)
olprintscreen		Scarica in stampa la finestra
olwsm		Gestore di Workspace
bitmap	Applicazione	Editor di bitmap
hexcalc		Calcolatore in aritmetica esadecimale
tgif		Strumento per disegno
xbiff		Visualizza finestra di "mailbox"; segnala l'arrivo della posta
xcal		Calendario da scrivania
xcalendar		Calendario da scrivania
xcalc		Calcolatore matematico, per l'uso col mouse
xclipboard		Memoria temporanea per le operazioni di copia e sposta
xedit		Semplice editor di testo, per l'uso col mouse
xmag		Visualizza bit significativi di una sezione del display
xman		Visualizza la man page
xmail		Legge e crea messaggi della posta
xmessage		Visualizza una finestra contenente il testo di un messaggio

Tabella 23.2 Comuni applicazioni di clienti X (*continua*)

Nome	Tipo	Descrizione
xmh		Legge e crea messaggi della posta
xpic		Strumento per disegno
xps		Prova di file PostScript
xrn		Legge e crea messaggi di notizie
xsetroot		Cambia i parametri di visualizzazione dello sfondo
xtroff		Prova di ditroff
aquarium	Giochi	Visualizza nello sfondo un pesce in movimento
ico		Visualizza una palla che rimbalza
kaleid		Visualizza in una finestra un caleidoscopio che cambia in continuazione
xclock		Visualizza un orologio
xdemo		Visualizza diverse dimostrazioni di X
xeyes		Visualizza due occhi che si muovono seguendo il puntatore
xgranite		Visualizza nello sfondo un disegno di granito
xloadimage		Visualizza un'immagine in una finestra o nello sfondo
xlogo		Visualizza in una finestra il logo X Window System
xphoon		Visualizza nello sfondo un'immagine delle fasi della luna

Per abilitare o disabilitare singole macchine, dovete includerle o escluderle da una lista di accesso, specificandone il nome nella linea di comando **xhost**, dopo il carattere + o −, come in questo esempio:

```
$ xhost + yoursys
yoursys being added to access control list
$
```

Quando il server X entra in esecuzione, consulta il file **etc/X0.hosts** nella macchina locale; questo file contiene un elenco di host, uno per linea, cui viene concesso di default l'accesso. Per esempio:

```
$ cat /etc/X0.hosts
yoursys
othersys
$
```

Lo zero nel nome del file significa display zero nella macchina locale e corrisponde allo zero nel nome in DISPLAY, come in

```
DISPLAY = hostname:0
```

Se sulla macchina avete più di un display, potete creare i file addizionali necessari **/etc/X?.hosts**.

I CLIENTI xset E xsetroot

Il cliente **xsetroot** ha lo scopo di rendere più piacevole l'aspetto del display. Vi consente di stabilire le proprietà dello sfondo di Workspace o root, cioè dello spazio inutilizzato dietro le finestre sullo schermo. Accetta le usuali opzioni X di linea di comando per predisporre i colori dei caratteri e dello sfondo e così via; in aggiunta, può accettare dopo l'argomento **-bitmap** il nome di un file con una mappa di bit (bitmap) per scegliere un disegno da visualizzare nello sfondo. Provate, per esempio, questo comando:

```
$ xsetroot -rv -bitmap /usr/X/lib/bitmaps/nights -fg blue -bg red
```

Sono disponibili molti disegni di dominio pubblico; di solito sono inclusi in **/usr/X/lib/bitmaps** oppure in **/usr/include/X11/bitmaps**. Se una mappa risulta più piccola dell'area dello schermo, **xsetroot** crea sullo schermo un effetto di mattonelle, ciascuna riproducente una copia dello stesso disegno.

Potete anche predisporre uno sfondo continuo, con

```
$ xsetroot -solid colore
```

rimpiazzando *colore* col nome del colore desiderato.

L'applicazione **xset** è simile a **xsetroot**, ma predispone solo caratteristiche generiche, come percorsi di directory per i font, parametri di mouse e altri analoghi. Il comando

```
$ xet -q
```

con q minuscola (query), visualizza le preferenze supportate da **xset**, in corso di uso.

Sia **xsetroot** che **xset** vengono richiamati dal menu Properties di Workspace mentre scegliete le caratteristiche da voi preferite, ma l'uso diretto dei comandi consente di scegliere maggiori possibilità e opzioni.

IL CLIENTE xcalc

Il cliente **xcalc** è un calcolatore orientato all'uso del mouse, che opera come una calcolatrice tascabile. Di solito consente di scegliere fra la normale notazione algebrica e la notazione polacca inversa. La versione **hexcalc** consente di eseguire calcoli nella numerazione in base esadecimale.

LETTURA DELLA POSTA E xbiff

Il cliente **xbiff** (chiamato così perché l'implementatore aveva un cane che aggrediva i postini!) visualizza una tipica cassetta postale americana in una piccola finestra. Quando viene ricevuta della nuova posta, la bandierina sulla cassetta viene alzata, il video suona e spesso la finestra viene messa in reverse, o video inverso. Dopo che avete letto la vostra posta, **xbiff** abbassa automaticamente la bandierina; potete ripristinare l'icona anche con un click sulla bandierina alzata.

Il cliente **xbiff** non dispone di uno strumento per leggere direttamente la posta; per questo scopo, dovete entrare in una finestra **xterm** e lanciare il vostro normale programma **mail**. Solo alcune release X includono strumenti per leggera la posta, ma tutte includono **xbiff**.

Di solito, è possibile cambiare la mappa di bit che visualizza l'icona **xbiff**, ma le procedure variano fra le diverse versioni del programma; spesso potete definire una risorsa X che punta alla mappa di bit desiderata. Consultate la man page per i dettagli.

23.9 Approfondimenti

Il sistema X Window mette a disposizione un ambiente ricco di possibilità e ampiamente configurabile. Invece di usare le predisposizioni di default di SVR4 descritte finora, potete controllare direttamente l'intero ambiente X e personalizzarlo a vostro piacimento; inoltre, non abbiamo neanche citato numerose caratteristiche e clienti disponibili.

Nella maggior parte di sistemi X, potete usare direttamente X Window System, utilizzando **xinit** in luogo di **olinit**; questo, è vero, elimina l'ambiente OPEN LOOK, ma vi consente il controllo totale sugli equivalenti strumenti progettati per un X Window System generico. Molti programmi e procedure di OPEN LOOK e di Motif sono stati adattati da questi strumenti generici, ma le procedure generiche originali possono ancora essere usate.

LINEA DI COMANDO DI xinit

Il comando **xinit** consente di usare in linea di comando opzioni che personalizzano il comando stesso, ma consente anche di includere opzioni che vengono passate al server X (in genere chiamato **X** sia in *User's Manual* che nella lista di **ps**). Il comando **xinit** può essere abbastanza complesso; la forma base è la seguente:

```
$ xinit [[cliente] opzioni] [ – – [server] [display] opzioni ]
```

Anche il comando **olinit** ammette una forma completa simile, ma viene usata raramente.

Le opzioni per **xinit** stesso vanno inserite nella linea di comando per prime, seguite da – –, e quindi dalle opzioni specifiche per il server. Il primo gruppo di opzioni, se presente, di solito include un nome di cliente; se il nome di cliente è omesso, viene assunto **xterm** come cliente di default. Questa parte della linea di comando consente di specificare anche i colori dello schermo e altre proprietà per il cliente. Il secondo gruppo di opzioni, per il server, consente di selezionare un display, scegliere un server diverso, e simili. Per i dettagli sulla configurazione del server consultate la man page **X(1)**. Come esempio, considerate questi due comandi, che sono all'incirca equivalenti:

```
$ olinit
$ xinit -geometry 80x24- -:0 -vdctype vga800x600
```

Questa forma diretta di linea di comando non è di solito necessaria, a meno che non vogliate lanciare il server su una macchina, da un'altra macchina della rete.

PROCEDURA COMANDI D'AVVIO DI xinit

Quando **xinit** è in esecuzione, lancia il server X per il vostro display, quindi esegue il file **$HOME/.xinitrc** (init run control); la versione corrispondente per OPEN LOOK di questo file è **$HOME/.olinitrc**. Ambedue questi file sono procedure shell che possono contenere qualunque comando a vostra scelta; vengono di solito usati per lanciare il cliente X da voi prescelto, per lanciare comandi che personalizzano la sessione X e per lanciare il gestore di finestra prescelto; come mostrato nell'esempio:

```
$ cat $HOME/.olinitrc
olwm &
olfm &
xclock - geometry 40x40-0+0 &
xbiff -rv -geometry 40x40-2-2 &
xmodmap .Xmodmap
xterm -geometry 80x24+10+0 &
$
```

Questo è un esempio di un semplice file **.olinitrc**; più avanti nel capitolo ne troverete uno più complesso.

Se il file non esiste, **olinit** o **xinit** predisporranno un semplice ambiente di default; questo dovrebbe di solito risultare in un ambiente X funzionante, ma in pratica probabilmente non utilizzabile, mancando del gestore di finestra. Il comando **olinit**, da parte sua, di solito lancia per default il cliente **olwsm**, e quindi, in questo caso, potrete avere disponibile il menu Workspace.

TERMINE DELLA SESSIONE X

Quando con la voce di menu Exit terminate **olwsm**, termina l'intera sessione X. Il comportamento di **xinit** è diverso, in quanto opera lanciando il server e quindi eseguendo la procedura shell in **.xinitrc**; quando questa procedura termina, **xinit** presume che abbiate finito di usare la sessione X, perciò, termina il server e vi riporta nello shell di login. Questo significa che il programma **xinit** (o il programma che rimpiazza **xinit** tramite **exec** nel file **xinitrc**) deve essere in esecuzione per tutto il tempo della vostra sessione X, e che la procedura shell **.xinitrc** non deve finire fino a che non avete finito di usare la sessione X. Questi prerequisiti di solito vengono raggiunti lanciando in background (cioè, con l'operatore **&**) tutti i comandi in **.xinitrc**, eccetto l'ultimo. Potrete organizzare la procedura in modo che l'ultimo comando sia un comando che potete terminare, per concludere la vostra sessione X; di solito sarà il vostro gestore di finestra, poiché fornisce menu e comandi a uso dell'intera sessione. Per esempio, la Figura 23.4 mostra un file **.xinitrc** abbastanza complesso, che distingue tra diverse macchine, lancia clienti diversi a seconda della macchina e infine lancia per ultimo il gestore di finestra. Inoltre, questo **.xinitrc** configura la sessione utente con i programmi **xrdb** e **xmodmap**, due programmi di utilità descritti più avanti in questo capitolo.

Il gestore di finestra viene lanciato per ultimo con **exec**, in modo che la procedura **.xinitrc** non procede oltre, ma è il solo comando che non viene lanciato in background; in questo modo, quando selezionate **exit** dal menu del gestore di finestra, termina l'intera sessione X.

IL DATABASE DELLE RISORSE

X Window System mantiene un database interno di risorse, che può essere consultato dalle applicazioni per predisporre i valori dei propri parametri utilizzati. In questo database delle risorse, di solito vengono tenuti i valori della geometria della finestra, i font, i colori e altre informazioni relative a X, ma possono essere inclusi anche informazioni o dati particolari per l'applicazione. I menu Properties usano questo database per cambiare i font di default, i colori e tutte le altre proprietà della vostra sessione (o di singole finestre); i cambiamenti da voi effettuati tramite i menu Properties vengono introdotti nel database.

Le informazioni raccolte in questo database di risorse consentono di predisporre le vostre caratteristiche preferite evitando di includere lunghe liste di opzioni nelle linee di comando da voi usate. Di solito, le risorse possibili sono molto più numerose di quelle che compaiono nei menu Properties; infatti, le risorse possono essere specificate a livello di finestra, se un'applicazione supporta questo livello di dettaglio, perciò alcune applicazioni multifinestra ammettono diversi colori o font per ciascuna finestra.

```
#!/bin/sh
# Un esempio di .xinitrc

HOST=`echo $DISPLAY | cut -d: -f1`

# Commentate queste linee se non le usate
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
#sysresources=/usr/X/lib/xinit/.Xresources
#sysmodmap=/usr/X/lib/xinit/.Xmodmap

# Aggiunte in default e keymap
if [ -f "$userresources" ]; then
    xrdb -merge $userresources
fi
if [ -f "$sysmodmap" ]; then
    xmodmap $sysmodmap
fi
if [ -f "$sysresources" ]; then
    xrdb -merge $sysresources
fi
if [ -f "$usermodmap" ]; then
    xmodmap $usermodmap
fi

# Scegliete il vostro wm!
# WMCMD="exec twm -f $HOME/-twmrc"
WMCMD="exec olwm -f"
# WMCMD="exec gwm -f mwm"
# WMCMD="exec mwm -f"
export WMCMD

# predisposizione speciale a seconda della macchina
if [ $HOST = "my_sys" -o $HOST = "unix" -o $HOST = "localhost" ]
then
  xphoon
  xperfmon -geometry 200x300-4-4 user system input output collision &
  xclock -geometry 50x80-1+1 &
  xbiff -geometry -1+90 >/dev/null &

  xterm -title $HOST -geometry 80x30+120-4 -fg white -bg blue \
  -font -adobe-courier-bold-r-normal--18-180-75-75-m \
  -110-iso8859-1 &

  xterm -geometry 80x30+0+0 -name login -fg white -bg blue \
  -C -font -adobe-courier-bold-r-normal--18-180-75-75-m\
  -110-iso8859-1 &
```

Figura 23.4 Un file **.xinitric** complesso.

```
xterm -title yoursys -geometry 80x12+1-4 -fg white -bg blue \
-font -adobe-courier-bold-r-normal-- -18-180-75-75-m\
-110-iso8859-1 \
-e rlogin yoursys &

$WMCMD

else    # display monocromo -- non predispone i colori
xclock -geometry 50x50-1+1 &
xperfmon -geometry 200x300-4-4 user system input output collision &

xterm -geometry 80x30+0+0 -name login -C \
-font -adobe-courier-bold-r-normal-- -18-180-75-75-m\
-110-iso8859-1 &

$WMCMD
fi
```

Figura 23.4 Un file **.xinitrc** complesso (*continua*).

Le informazioni delle risorse sono di solito tenute in un normale file di testo nel vostro home directory, e questo file viene letto da X al momento di entrare in esecuzione; dopo di ciò, le informazioni sono disponibili a ogni applicazione che le richiede, senza la necessità di rileggere il file. In realtà, possono venire consultati diversi file in successione, in modo da predisporre alcuni default di sistema e quindi integrare i vostri propri default.

Per primo, viene letto il file **.Xdefaults** nel vostro home directory; quindi, viene esaminata la variabile di ambiente *XENVIRONMENT* e se contiene il pathname di un altro file di risorse, viene letto anche questo file. Se la variabile XENVIRONMENT non esiste, viene letto il file col nome **$HOME/.Xdefaults-***nome-host*, in cui *nome-host* è il nome della macchina su cui è in esecuzione il cliente (non necessariamente il server). Le informazioni dei file letti per ultimi rimpiazzano quelle dei file che sono stati letti in precedenza.

Il contenuto di tutti questi file **.Xdefaults** segue un unico formato: un elenco di linee con coppie nome-valore. Queste coppie sono specifiche per ciascuna applicazione, ma nel nome possono comprendere dei metacaratteri che consentono di effettuare delle scelte di risorse sia a livello globale che a livello specifico. Spesso i nomi includono combinazioni finora non usuali di caratteri minuscoli e maiuscoli; dovrete rispettare esattamente queste combinazioni e quelle di metacaratteri. Come principio, i nomi che iniziano con una lettera maiuscola definiscono classi di risorse, mentre i nomi in caratteri minuscoli definiscono singole istanze delle risorse di quel cliente. Questa distinzione risulta utile se volete predisporre il valore di una risorsa una prima volta e quindi rimpiazzarlo per una determinata finestra.

Parte di un semplice file **.Xdefaults** viene mostrata in questo esempio:

```
$ tail -21 $HOME/.Xdefaults

! predispone un percorso di ricerca per file bitmap
*bitmapFilePath:        /home/giorgio/bitmaps:/usr/X/lib/bitmaps

*selectBtn:     <Button1>
*adjustBtn:     <Button2>
*menuBtn:       <Button3>
*beep:      always
*borderColor:       red
*iconGravity:       south
xterm*background:       blue
xterm*boldFont:8x13bold
xterm*cursorColor:      white
xterm*font:     8x13bold
xterm*foreground:       white
xterm*geometry:80x24+0+40

xbiff*fullPixmap:               xbiff.xbm
xbiff*fullPixmapMask:           xbiff__mask.xbm
xbiff*emptyPixmap:              xbiff__empty.xbm
xbiff*emptyPixmapMask:          xbiff__empty__mask.xbm
```

Ogni elemento del file è contenuto in una linea separata; le linee di spazi sono ignorate; i commenti seguono l'operatore ! e si estendono fino alla fine della linea. Le linee delle risorse iniziano con un nome, che termina col carattere :; segue un carattere di spaziatura (nell'esempio un tab); quindi, il rimanente della linea contiene il valore della risorsa relativa. Notate che gli spazi successivi fino alla fine della linea non vengono ignorati dall'applicazione, ma sono considerati come facente parte della risorsa; per questo motivo, dovete concludere ogni linea con RETURN immediatamente alla fine della risorsa. Potete continuare un valore di una risorsa sulla successiva linea del file, terminando la linea intermedia col carattere \, a protezione del carattere di RETURN.

I nomi sono composti secondo un criterio gerarchico, in base al quale la prima parte contiene il nome del cliente, seguito dal nome della finestra e infine dal nome della risorsa. In questo modo, è possibile individuare le risorse indipendentemente per ciascun cliente e anche per ciascuna finestra di un singolo cliente. Potete rimpiazzare ogni componente, eccetto la risorsa, con un * (asterisco), per significare qualunque valore per quel componente; per esempio, la linea:

```
*bitmapFilePath:        /home/giorgio/bitmaps:/usr/X/lib/bitmaps
```

userà la risorsa specificata, per tutte le applicazioni; mentre la linea:

```
xterm*Foreground:       white
```

impiegherà il colore bianco per gli sfondi di tutte le finestre **xterm**. Di solito, non è facile determinare i singoli nomi delle finestre, perciò, molto spesso i componenti centrali del nome conterranno *, a meno che il file di default non sia stato predisposto dall'implementatore dell'applicazione. Nelle man page dei clienti potrete talvolta trovare un elenco delle risorse di uso corrente; nei casi più semplici potrete costruire tale lista per tentativi coi menu Properties.

IL COMANDO xrdb

Oltre a predisporre la lettura del file **.Xdefaults** all'avvio di X, potete caricare file addizionali, oppure cambiare il file dei default mentre X è in esecuzione. A questo scopo, dovete istruire X di aggiungere o rileggere un file; usate il comando **xrdb** (X resource database), con un nome di file, per rimpiazzare il contenuto corrente del database delle risorse con il contenuto del file specificato. In luogo di un file potete utilizzare anche lo standard input. Con l'opzione **–merge**, **xrdb** aggiunge il contenuto del file al database corrente, invece di rimpiazzarlo:

```
$ xrdb –merge .Xresources
```

Potete esaminare il contenuto corrente del database con l'opzione **–query**, come nell'esempio:

```
$ xrdb –query
```

I clienti X differiscono enormemente tra di loro nell'uso delle risorse e nei nomi previsti; eccetto per poche risorse standard, come la geometria della finestra, i colori e i font. Consultate le man page di ciascun cliente per conoscere se e come utilizza le risorse.

Oltre a quanto descritto fin qui, molte applicazioni prevedono un loro file privato di risorse, che specifica valori di default a livello di sistema, delle risorse supportate dall'applicazione. Questi file di risorse sono collocati nel directory **/usr/X/lib/app-defaults**, oppure **/usr/lib/X11/app-defaults**. Se volete cambiare il comportamento dell'applicazione per ciascun utente potete cambiare questi default a livello di sistema; in ogni caso, questi default di applicazione possono suggerirvi molte idee per l'uso delle risorse in un vostro proprio file **.Xdefaults**.

LA MAPPA DI TASTIERA E IL COMANDO xmodmap

Un'altra importante area di configurabilità nel sistema X Window è costituita dalla mappa della tastiera. Il significato di ogni tasto della tastiera e dei click del mouse è definito a livello hardware da codici numerici di tasto

(*keycode*), che vengono convertiti dal server X in generici simboli di tasto (*keysym*); a loro volta, questi simboli vengono tradotti in caratteri o stringhe da trasmettere all'applicazione cliente. Potete quindi modificare le tabelle di traduzione, per riorganizzare la vostra tastiera. È stato pubblicato un insieme di tabelle di traduzione in *Dvorak Keyboard*, ma più probabilmente a voi sarà sufficiente scambiare pochi tasti della vostra tastiera; oppure modificare i tasti CTRL e ALT; oppure usare un tasto funzione altrimenti inutilizzato, per trasmettere una particolare sequenza di caratteri, per rendervi più semplice l'impiego del sistema. Il comando **xmodmap** (modify keymap), accetta come argomento un nome di file, oppure legge il suo standard input e utilizza le informazioni contenute per riorganizzare la mappa della tastiera.

L'uso in questo file di un linguaggio di comandi consente il controllo completo della tastiera. Il file è un normale file di testo, con un comando per linea; le linee di spazi e quelle che iniziano con ! sono ignorate. Ogni linea inizia con un comando; i comandi più utili sono: **keycode**, che assegna un simbolo keysym a un numero keycode; **keysym**, che assegna un nuovo simbolo keysym a un simbolo già esistente; **remove**, che cancella dalla mappa un simbolo keysym; **add**, che aggiunge alla mappa un nuovo simbolo. Dopo il comando, viene il keysym, o keycode, oggetto del comando, seguito da = e dal nuovo nome. Per esempio:

```
$ cat $HOME/.Xmodmap
! scambia ESC e accento grave
! nota: questi keycode dipendono dalla macchina; verificate
! sul vostro sistema con "xev" o "xèvent" i keycode usati
keycode 9 = quoteleft
keycode 49 = Escape

! scambia le funzioni di tasto CTRL sinistro e blocco maiuscole
remove Lock = Caps_Lock
remove Control = Control_L
keysym Control_L = Caps_Lock
keysym Caps_Lock = Control_L
add Lock = Caps_Lock
add Control = Control_L
$
```

Sul vostro sistema i codici numerici di particolari tasti possono essere differenti; nel sistema di questo esempio, 9 è il keycode per il tasto ESC e 49 è il keycode per il tasto ` (accento grave), situato a sinistra in alto nella tastiera. Il secondo gruppo di comandi inverte il significato del tasto CTRL di sinistra, col tasto di blocco maiuscole. Prima, le precedenti definizioni vengono tolte dalla mappa; quindi, vengono assegnati i nuovi simboli; infine, vengono aggiunti alla tabella i nuovi valori.

I codici numerici possono essere esaminati col comando **xev**, oppure **xe-**

vents, che visualizza il contenuto di tutti gli eventi X destinati a una finestra. Potete lanciare **xev** e premere dei tasti: otterrete l'indicazione di quale keycode viene assegnato a ciascun tasto e del corrispondente nome di keysym associato. Con un piccolo lavoro di indagine, potrete determinare come la tastiera è mappata nel vostro sistema e se necessario modificarla. Alcune informazioni possono essere trovate anche nella man page **kbd(7)**.

Poiché **xmodmap** modifica il server X e può solo abbinare singoli tasti con singoli nomi simbolici, la tastiera viene riorganizzata per tutti i clienti.

È disponibile anche una procedura che usa il database delle risorse per cambiare il significato dei tasti per una particolare applicazione, se l'applicazione supporta queste trascodifiche. Questa procedura consente anche di definire stringhe da abbinare a un tasto, come in questo esempio:

```
$ cat .Xresources
xterm*VT100.translations: #override <Key>F4: string ("troff -mm") \n \
    <Key>F5: string ("rlogin yoursys") \n
$
```

Questa complessa definizione rimpiazza il significato dei tasti funzione F4 e F5, in maniera che quando viene premuto uno di questi tasti, **xterm** vede la stringa corrispondente specificata. Con questo metodo potete riassegnare un qualsiasi tasto. Abbiate cura di copiare esattamente l'esempio qui sopra, ponendo i caratteri **\n** (barra inversa n) alla fine di ciascuna transcodifica definita, e aggiungendo un altro \ a protezione del reale ritorno a capo, come in questa definizione di risorse. I simboli F4 e F5 nell'esempio sono i nomi simbolici usati da **modmap**; per definire Shift F4 usate

```
Shift <Key>F4
```

e per definire CTRL-F4 usate

```
Ctrl <Key>F4
```

Alcuni utenti X hanno definito un intero proprio ambiente di tastiera mediante queste transcodifiche; tuttavia, le risorse di transcodifica possono non funzionare correttamente in tutte le versioni di SVR4; verificate per sicurezza il vostro sistema.

PROGRAMMAZIONE DEL MENU WORKSPACE olwm

Potete configurare il menu di molti gestori di finestra, in modo che contenga i comandi che usate abitualmente. Le procedure da seguire variano a seconda dei gestori di finestra, ma nella maggior parte dei casi viene usato un file che contiene la configurazione; quando il gestore di finestra entra

in esecuzione, legge quel file e costruisce il menu. Il gestore di finestra OPEN LOOK usa il file **.olprograms** oppure **openwin-menu**, nel vostro home directory, per configurare rispettivamente il menu Programs e il menu Workspace; tuttavia, il vostro sistema potrebbe prevedere solo uno di questi file. In genere, ogni linea del file specifica una voce del menu. Se il vostro sistema prevede il file **.openwin-menu**, potete anche definire i sottomenu a richiesta.

Il file **.olprograms** contiene un elenco di coppie nome-valore separate da :; la parte a sinistra appare nel menu Programs, la parte a destra viene eseguita alla selezione della voce di menu specificata. Il primo elemento del file contiene il default. Per esempio:

```
$ cat $HOME/.olprograms
Xterm...:        exec xterm -fn 8x13bold -geometry 80x24+20+30
Tetris: exec xterm -geometry 80x24 -e /usr/src/tetris/tetris
Lock Screen:  exec xlock
Calculator:      exec xcalc
$
```

Il gestore di Workspace crea una finestra, quindi esegue l'applicazione specificata nella parte destra; quando l'applicazione termina, la finestra viene distrutta.

USO DI X CON DOS E CONSOLE VIRTUALI

X Window System è bene integrato con gli altri strumenti orientati allo schermo in SVR4; se avete una macchina che supporta queste possibilità, potete eseguire una sessione X sotto un terminale virtuale sulla console. Non è possibile eseguire molteplici sessioni X sotto terminali virtuali multipli, ma è possibile usare una sessione X e sessioni di console contemporanee. Come al solito, potete eseguire sessioni DOS sull'intero schermo, in altre console virtuali. Potete usare la normale sequenza *hot key* (di solito ALT-SYSREQ-*tasto funzione*), per scambiare le sessioni, e potete ritornare alla console originale con ALT-SYSREQ-H (home). Questi procedimenti funzionano anche se non avete lanciato l'applicazione **vtlmgr**. Se **vtlmgr** è in esecuzione, potete anche passare ad altri teminali virtuali che contengono sessioni DOS o altre attività.

Oltre a tutto questo, potete lanciare sessioni DOS all'interno di una o più finestre **xterm**, e potete lanciare Microsoft Windows all'interno della sessione DOS dentro una sessione X. Possono essere necessarie alcune prove per determinare come configurare la finestra **xterm** nella dimensione e forma corrette per una sessione DOS.

Naturalmente, queste molteplici sessioni richiedono cospicue quantità di memoria reale, di spazio su disco e di tempo di CPU; può darsi che le presta-

zioni del sistema risultino inadeguate a svolgere così tante operazioni contemporaneamente.

AUTORIZZAZIONE ALL'USO DI X

In molte versioni SVR4 di X, gli utenti non sono ammessi a utilizzare X se non sono stati preventivamente autorizzati dal gestore del sistema. In parte questo è dovuto a motivi di sicurezza, ma principalmente al fatto che **.profile**, PATH e altre parti dell'ambiente degli utenti devono essere modificati per creare un ambiente X funzionante.

Non tutti gli strumenti per l'utente gestore del sistema prevedono una opzione per la gestione di X. Alcune release prevedono i programmi **oladduser** (add a user), **olremuser** (remove a user) e **olsetvar** (set OPEN LOOK environment variables); questi programmi di solito risiedono nel directory **/usr/X/admin**. Se nessuna di queste possibilità è disponibile, il gestore del sistema dovrà configurare manualmente l'ambiente X di ogni utente.

Tutte queste procedure agiscono modificando i **.profile** degli utenti, oppure installando il file **.olinitrc**, o altri file descritti in questo capitolo. Potete seguire queste modifiche esaminando il vostro **.profile**, lanciando **oladduser**, e quindi esaminando ancora il file. Spesso al vostro **.profile** viene aggiunta la linea

```
. $HOME/.olsetup
```

per predisporre le variabili di ambiente corrette al momento del vostro login in macchina. Il file **.olsetup** di solito è abbastanza semplice, ma contiene le variabili di ambiente critiche, come mostra l'esempio:

```
$ cat .olsetup
OLINVOKE=no export OLINVOKE #!@ Non cambiate questa linea !@
DISPLAY=unix:0 export DISPLAY #!@ Non cambiate questa linea !@
XNETACCESS=on #!@ Non cambiate questa linea !@
PATH=$PATH:/usr/X/bin/ exports PATH #!@ Non cambiate questa linea !@
XWINFONTPATH="/usr/X/lib/fonts/Xol,/usr/X/lib/fonts/misc,\
/usr/X/lib/fonts/75dpi,/usr/X/lib/fonts/100dpi" export XWINFONTPATH
if [ "$OLINVOKE" = "yes" -a -r /usr/X/bin/olinit -a -x \
 /usr/X/bin/olinit -a -r /usr/X/bin/olwsm -a -x /usr/X/bin/olwsm
then
    /usr/X/bin/olinit /usr/X/bin/olwsm -- -xnetaccess $XNETACCESS
fi
$
```

Questo file predispone le variabili di ambiente corrette, quindi esamina la variabile OLINVOKE per determinare se lanciare o no l'ambiente X.

La procedura **oladduser**, inoltre, copia versioni di default di **.olprograms**,

.olinitrc e **.Xdefaults**, nel vostro home directory; se usate **xinit** invece di **olinit**, potrete operare manualmente queste aggiunte al vostro file **.profile**.

LA GESTIONE DELLA SESSIONE E I TERMINALI X

Il terminale X è un prodotto relativamente nuovo; si tratta di terminali con schermi grafici di alta qualità e memoria di ampia dimensione, ma privi di CPU di uso generale e di dischi; di solito contengono un'implementazione in ROM del server X; la connessione delle applicazioni clienti al terminale avviene tramite LAN. I terminali X sono ottimizzati per l'esecuzione del server, e possono perciò risultare meno costosi di equivalenti microcomputer aventi la capacità di eseguire sia il server che il cliente.

Per contrasto, il sistema UNIX deve essere predisposto in maniera speciale al supporto dei terminali X, in quanto questi entrano direttamente nel sistema X, non potendo passare attraverso la usuale procedura di login. In questa speciale configurazione del sistema UNIX, uno speciale processo X, denominato **xdm** (X display manager), oppure **xsm** (X session manager), rimpiazza Service Access Facility nella gestione dei login da terminali X. Il processo **xdm** (o il suo equivalente), è un demon che deve essere in esecuzione nella macchina usata dal terminale X. I terminali X possono essere configurati per scegliere l'elaboratore host desiderato.

In esercizio, **xdm** ascolta i messaggi di connessione da terminali X; quando il terminale X viene messo sotto tensione o in reset, invia un segnale al gestore di display, che visualizza l'equivalente dei messaggi di richiesta **login:** e **Password:**. Se il login viene accettato, **xdm** esegue il file **.olinitrc** o **.xinitrc** dell'utente, quindi termina il suo compito. Quando l'utente conclude la sessione, **xdm** rileva l'evento e ridispone l'ambiente per un successivo login.

Il gestore di display può differire tra le diverse release di X e anche per i diversi produttori di terminali X, perciò dovrete seguire le procedure particolari del vostro sistema.

Capitolo 24

Le reti

Il supporto per le reti locali (LAN, *local area networks*), è molto migliorato in SVR4, rispetto alle precedenti release del sistema operativo UNIX. In aggiunta al supporto a basso livello del driver in kernel, è disponibile software semplice e di facile uso per connettere le due principali LAN disponibili in ambiente UNIX, Ethernet e StarLan.

Ethernet venne in origine sviluppata per le versioni BSD del sistema UNIX, ed è diventata in seguito uno standard industriale per la connessione di numerosi tipi di macchine con sistemi operativi differenti. La rete mondiale Internet, una rete su lunghe distanze o WAN, *wide area network*, può connettersi facilmente alle reti Ethernet. Il supporto completo per Ethernet e i servizi relativi è fornito solo dalla Release 4 di System V. Questo significa che SVR4 adesso può partecipare pienamente in reti di macchine che usano il protocollo di comunicazione TCP/IP, indipendentemente dal fatto che le macchine nella rete siano sistemi System V, oppure no.

La rete alternativa, StarLan, fu sviluppata da AT&T utilizzando hardware e protocolli di comunicazione propri; il supporto StarLan venne introdotto in System V nella Release 3 ed è ancora ben presente in Release 4. Altre implementazioni di rete, come il token-ring e alcune reti di PC, non sono altrettanto comuni sotto il sistema UNIX, anche se ne esistono numerose alternative.

Queste reti hanno tutte una velocità relativamente alta, almeno 1 MB e spesso 10 e 20 MB al secondo; questo consente l'impiego di strumenti software che richiedono risposte veloci e alte prestazioni, come nel caso della possibilità di montaggio di file system su macchine remote in modo che appaiano cone se fossero su di un disco locale. I montaggi remoti possono avere grande valore per evitare l'esistenza di copie ridondanti di file e directory in una rete di macchine, semplificando la gestione dei file ed evitando il rischio di avere copie di file non aggiornate. Inoltre, l'uso appropriato di montaggi remoti può consentire il risparmio di notevoli quantità di spazi su disco, considerando la rete nel suo insieme. In pratica, i montaggi remoti hanno consentito la realizzazione di macchine senza dischi, introdotte per la prima volta in ambiente BSD, dove l'intero file system è dislocato su una macchina remota. Strumenti addizionali consentono di effettuare login su macchine remote con un livello di prestazioni che supera ampiamente le capacità degli emulatori di terminali come **cu**. Anche X Window System e altri ambienti con finestre gestiti in rete traggono vantaggio dall'alta velocità delle LAN.

Tre classi generali di software utilizzano l'architettura di queste due LAN. Il *Network File System* (NFS) è principalmente orientato all'ambiente Ethernet, il *Remote File Sharing* (RFS) è orientato principalmente verso l'ambiente StarLan; tuttavia, con SVR4 è possibile usare una qualsiasi delle due architetture LAN con uno qualunque dei due sistemi software, anche se questo viene fatto raramente. Una terza classe, i comandi in *remote access* (accesso remoto), sono trasparenti al tipo di rete e funzionano senza distinzioni ugualmente bene con un qualsiasi tipo di rete.

Per l'uso dei comandi e delle idee discusse in questo capitolo è necessario che la vostra macchina sia connessa a una delle due LAN citate; se non disponete di nessuno di questi due tipi di LAN, o se la vostra è una macchina isolata (standalone), questo capitolo non è di utilità per voi. Se volete esplorare le possibilità, dovrete consultare il vostro fornitore richiedendo informazioni sulla possibilità di collegarvi in LAN e usare macchine SVR4 tramite la rete.

24.1 Accesso a macchine remote

Se la vostra rete è già funzionante, potete usare diversi comandi che consentono facilmente di eseguire un login su un'altra macchina nella vostra LAN. Questi strumenti vi permettono di eseguire comandi sulla macchina remota; sono simili nei concetti ai vari strumenti **uucp**, ma sono progettati per LAN ad alta velocità, anziché per lente linee telefoniche. Per usare questi strumenti di accesso remoto dovere avere la disponibilità di un login valido sulla macchina remota che volete usare.

IL COMANDO rlogin

Il comando **rlogin** (remote login) consente di lanciare una normale sessione login su una macchina remota. Richiede come argomento *hostname* o *uname* della macchina remota; per esempio:

```
sis_locale$ rlogin sis_remoto
sis_remoto$
```

Presupponendo che siate autorizzato a usare la macchina *sis_remoto*, dopo una breve attesa vedrete un prompt della macchina remota: da questo momento potrete eseguire comandi su quel sistema. Quando avete concluso, per terminare la sessione remota potete usare CTRL-D oppure **exit**; per esempio:

```
sis_remoto$ exit
Connection closed.
sis_locale$
```

Ritornerete allora nella macchina locale e potrete proseguire su questa la vostra sessione. Per chiudere una sessione remota potete usare anche la sequenza ~. (tilde punto).

Il comando **rlogin**, di default, stabilisce sulla macchina remota un ambiente simile a quello che vedreste eseguendo il login direttamente su quella macchina; anche sulla macchina remota avrete assegnato il vostro home directory. Il valore **TERM** e poche altre variabili di ambiente vengono passate dalla macchina locale, in modo che le applicazioni a pieno schermo funzionino come al solito.

Se sulla macchina remota non avete un identificatore di login, **rlogin** vi richiede una password; potete ignorare la richiesta con un RETURN e **rlogin** vi richiederà un login; potete allora introdurre un altro identificatore di login, se ne avete la disponibilità. In alternativa, **rlogin** prevede l'opzione **-l** (login), che accetta un altro identificatore di login come argomento; il comando tenta questo login e vi richiede una password, come nell'esempio:

```
sis_locale$ rlogin -l gian sis_remoto
Password:
```

Quando è attiva la vostra sessione remota, tutti i vostri comandi vengono inviati alla macchina remota, mentre la vostra sessione locale attende fino a che non uscite dal sistema remoto. Se state usando uno shell con controllo di attività, potete sospendere temporaneamente la sessione remota introducendo un carattere ~ (tilde), seguito dal carattere di sospensione CTRL-Z; questo arresta la sessione **rlogin** fino a che voi non la riavviate con il controllo di attività. Se introducete CTRL-Z non preceduto da tilde, **rlogin** trasmette il carattere CTRL-Z alla macchina remota, dove lo shell tenterà di sospendere un'attività.

IL COMANDO rcp

Potete copiare file tra due macchine col comando **rcp** (remote copy). Questo comando si usa in modo del tutto simile al normale comando **cp**, con la differenza che gli argomenti prendono la forma

```
macchina:nomepath
```

In mancanza della parte *macchina:*, viene assunta come default la macchina locale; può essere omesso il *nomepath* di destinazione, se volete usare lo stesso nome del file di origine. Per esempio, per copiare il file **/tmp/doc.file** dalla macchina locale a una macchina remota, potete usare

```
sis_locale$ rcp /tmp/doc.file sis_remoto:
```

Se usate l'opzione **–p** il comando **rcp** conserva nella copia i permessi di accesso e le date del file originale. Con l'opzione **–r** (recursive) **rcp** copia l'intero sottoalbero di directory sottostante il path di origine specificato; in questo caso la destinazione deve essere un nome di directory.

Il comando **rcp** consente di copiare file dalla macchina locale a una macchina remota, da una macchina remota alla macchina locale, oppure fra due macchine remote; è sufficiente indicare dove necessario la parte *macchina:* del nome file. Tuttavia, dovete avere accesso a un login in ambedue le macchine, e tutti i file da copiare devono potere essere letti da voi. Inoltre, nella vostra rete potrebbero essere previste restrizioni sulla copia di file in rete; se incontrate dei problemi dovrete interpellare il gestore della vostra rete.

IL COMANDO rsh

Il comando **rsh** (remote shell) si connette a una macchina remota ed esegue un comando specificato nella sua linea di comando. Il comando copia il suo standard input in quello del comando remoto e copia lo standard output e standard error del comando remoto in standard output e standard error locali. Il comando richiede come primo argomento il nome del sistema host remoto, e di seguito la linea di comando da eseguire sulla macchina remota; per esempio:

```
sis_locale$ rsh sis_remoto cat /tmp/doc.file
```

visualizza il file **/tmp/doc.file** della macchina remota **sis_remoto**. Potete utilizzare ridirezioni e pipeline per combinare comandi locali e comandi remoti, come in questo esempio:

```
sis_locale$ cat /tmp/doc.file ¦ rsh sis_remoto wc –l >/tmp/size
```

Questo permette di trasferire l'esecuzione di programmi di lunga durata nelle macchine meno impegnate della rete.

Se la parte comando viene omessa e usate solo

```
sis_locale$ rsh sis_remoto
```

rsh si comporta come **rlogin** e crea uno shell nella macchina remota.

Se nella linea di comando proteggete gli operatori di ridirezione, questi verranno interpretati dalla macchina remota invece che dalla macchina locale; per esempio:

```
sis_locale$ cat /tmp/doc.file | rsh sis_remoto wc -l ">" /tmp/size
```

creerà il file **/tmp/size** su **sis_remoto** anziché sulla macchina locale.

Come nel caso degli altri comandi di accesso remoto, per usare **rsh** dovete avere un login valido sulla macchina remota e disporre dei diritti d'accesso necessari per eseguire i comandi.

Non dovete confondere lo *shell remoto* con lo *shell ridotto*, pur avendo ambedue lo stesso nome **rsh**; lo shell ridotto è **/usr/lib/rsh** oppure **/sbin/rsh**, mentre lo shell remoto di solito è **/usr/bin/rsh**. Fate comunque attenzione a non usare il comando sbagliato; per evitare confusione potete usare il pathname completo oppure creare un alias. In alcuni sistemi il comando shell remoto è già stato rinominato, di solito **nsh** o **remsh**; verificate per sicurezza la vostra macchina.

Non potete rinominare voi stessi lo shell remoto, perché potrebbe essere rinominato come la macchina host da chiamare. Così, se provate

```
# ln rsh remsh
# remsh hostname
remsh: unknown host
#
```

lo shell remoto tenterebbe di connettere la macchina host *remsh*, cosa che certo non era nelle vostre intenzioni. D'altra parte, questa è una possibilità utile, perché permette di abbinare **rsh** a un nome di host e di usare quindi comandi come

```
# sis_remoto cat /tmp/file
```

per eseguire comandi sul sistema **sis_remoto**.

24.2 Comandi informativi per le reti

Sono disponibili numerosi comandi che forniscono informazioni sulla rete e i suoi utenti, consentono di conoscere dove un utente ha una sessione di login in corso e di rilevare l'impegno di una macchina nella rete.

IL COMANDO rusers

Il comando **rusers** (remote users) fornisce l'elenco degli utenti in corso di collegamento su ciascuna macchina nella rete, come nell'esempio:

```
sis_locale$ rusers
sis_locale     giorgio
sis_remoto     jim pat root
sis_locale$
```

Questa forma di linea di comando di solito elenca tutti gli utenti attivi di tutte le macchine nella rete, però alcune versioni di SVR4 richiedono la specificazione di un nome di host nella linea di comando. Nell'elenco non compaiono le macchine che non hanno utenti in corso di collegamento; con l'opzione **-a** (all) è possibile ottenere l'elenco di tutte le macchine nella rete, anche se inattive.

Per ottenere l'elenco di una sola determinata macchina, specificate come argomento il nome di host, come nell'esempio:

```
sis_locale$ rusers sis_remoto
sis_remoto     jim pat root
sis_locale$
```

Questa opzione è utile se la vostra rete comprende numerosi sistemi.

L'opzione **-l** (long) consente di ottenere un elenco dettagliato, comprendente la porta di terminale usata dagli utenti, l'ora del loro collegamento e il tempo di utilizzazione di CPU. Per esempio:

```
sis_locale$ rusers -l sis_locale
giorgio     sis_locale:console     Apr 28 10:45        35
sis_locale$
```

L'ultima colonna indica il tempo in minuti da quando l'utente ha introdotto un comando.

Un comando simile chiamato **rwho** funziona solo se in una macchina è stato installato lo speciale demon **rwhod**; comunque, è sempre preferibile usare il comando **rusers -l**.

IL COMANDO finger

Per ottenere informazioni della rete viene maggiormente usato il comando **finger**; questo comando fornisce notizie su ogni utente, indicando per ciascuno la porta di terminale usata e l'ora di quando ha effettuato il login con la macchina. Di default, **finger** elenca gli utenti della macchina locale, come in questo esempio:

```
sis_locale$ finger
Login       Name              TTY       Idle     When  Where
root        0000-Admin(0000)  console     58  Sat 10:45
giorgio     Giorgio           pts011          Sat 11:44  sis_remoto
sis_locale$
```

La colonna **Idle** indica il tempo in minuti da quando l'utente ha introdotto l'ultimo comando; se l'utente è entrato con **rlogin** da un host remoto, la macchina viene indicata nella colonna **Where**.

Se a **finger** viene fornito come argomento un identificatore di login (o una lista di identificatori di login), si ottiene un rapporto più dettagliato per quell'utente; per esempio:

```
sis_locale$ finger giorgio
Login name: giorgio                     In real life: giorgio
Directory: /home/giorgio                Shell: /usr/bin/ksh
On since Apr 28 11:44:50 on pts011        from sis_remoto
3 minutes 50 seconds Idle Time
New mail received Mon Apr 16 15:32:46 1991;
  unread since Sat Apr 28 11:44:19 1991
No Plan
sis_locale$
```

Le linee "New mail..." informano dell'ultima volta che l'utente ha ricevuto e letto la posta; "Plan" si riferisce al contenuto del file **$HOME/.plan** dell'utente, se quel file esiste. Se voi o il vostro gruppo utilizzate frequentemente **finger**, potete lasciare nel file **.plan** la vostra agenda o altre informazioni a uso degli altri utenti; se il file non esiste **finger** inserisce il messaggio "No Plan". Viene visualizzato anche il file **$HOME/.project**, se esiste.

Per ottenere informazioni sugli utenti di macchine remote, potete usare questi formati:

```
$ finger utente\@macchina
$ finger \@macchina
```

Abbiate cura di proteggere il carattere @ con \, altrimenti in SVR4 la linea di comando verrà rifiutata. La prima forma fornisce le informazioni su *utente* di *macchina*; la seconda informa su tutti gli utenti di *macchina*.

Il comando **finger** prevede diverse varianti di visualizzazione dell'uscita, selezionate da opzioni nella linea di comando. Potete usare **-l** per ottenere un formato esteso, **-s** per un formato contenuto, **-q** per un formato ancora più conciso, **-h** per sopprimere l'uscita delle notizie contenute in **.project**, **-p** per sopprimere le notizie di **.plan**. Infine, l'opzione **-f** sopprime l'uscita di intestazioni, per facilitare l'utilizzo di **finger** in procedure di comandi.

Come molti altri programmi informativi, **finger** richiede che uno speciale processo demon sia in funzione sulla macchina destinazione, altrimenti il comando non può funzionare.

IL COMANDO ping

Il comando **ping** tenta l'invio di un messaggio a una macchina remota; se la macchina è in funzione e in ascolto dell'attività nella rete, risponderà al messaggio. Con questo è possibile verificare se una macchina è attiva nella rete, come nell'esempio:

```
sis_locale$ ping sis_remoto
sis_remoto is alive
sis_locale$
```

Con l'opzione **-s** (statistics) il comando **ping** invia una sequenza di messaggi e fornisce un rapporto sul trasferimento dati tra le due macchine, come nell'esempio:

```
sis_locale$ ping sis_remoto
PING sis_remoto: 56 data bytes
64 bytes from sis_remoto (130.0.17.32): icmp_seq = 0. time = 10. ms
64 bytes from sis_remoto (130.0.17.32): icmp_seq = 1. time = 180. ms
64 bytes from sis_remoto (130.0.17.32): icmp_seq = 2. time = 0. ms
64 bytes from sis_remoto (130.0.17.32): icmp_seq = 3. time = 0. ms
64 bytes from sis_remoto (130.0.17.32): icmp_seq = 4. time = 10. ms
64 bytes from sis_remoto (130.0.17.32): icmp_seq = 5. time = 0. ms


----sis_remoto PING statistics----
6 packets transmitted, 6 packets received, 0% packet loss
round-trip (ms) min/avg/max = 0/33/180
```

Con l'opzione **-s** il comando **ping** trasmette pacchetti fino a che non premete DEL; allora, fornisce le informazioni statistiche ed esce. Il procedimento è molto utile per verificare, in casi dubbi, il corretto funzionamento della rete.

24.3 Accesso a file remoti

Il sistema UNIX SVR4 fornisce strumenti che consentono di utilizzare file fisicamente allocati su supporti appartenenti a macchine remote. Usando questi strumenti, potete montare nel vostro file system directory dislocati su macchine remote e potete usare queste risorse come se fossero risorse della vostra macchina. In pratica, l'uso di questi strumenti non differisce

dal montare un dischetto o un disco rigido locale, eccetto che, nel caso di file montati in remoto, il sistema ottiene i dati attraverso la LAN.

Questi comandi di accesso remoto differiscono internamente a seconda se voi state usando il Network File System (NFS) oppure il Remote File Sharing (RFS), ma gli strumenti a livello utente provvedono a ridurre o eliminare il più possibile le differenze.

La macchina locale può usare risorse di una macchina remota solo se la macchina remota permette questo accesso; il gestore di sistema di una macchina deve esplicitamente condividere (*share*) gli alberi di directory con le altre macchine della rete, perché sia ammesso l'accesso remoto ai file. Una volta che un directory è dichiarato condiviso, le altre macchine possono montare questo directory, con l'usuale comando **/sbin/mount**.

Tutti gli strumenti di questo tipo agiscono su un intero directory; potete montare un directory e tutti i suoi subdirectory, ma non potete montare un singolo file. Un directory può essere montato in un ben definito punto dell'albero di directory locale, detto punto di *mount*; questo punto di montaggio consiste in un directory vuoto in una posizione qualsiasi del file system. Dovete avere un punto di mount separato per ogni directory montato simultaneamente, ma potete riusare un punto di mount, se eseguite **unmount** prima di montare un altro directory nello stesso punto. Il sistema vi permette anche di montare un directory sopra un altro directory non vuoto, ma questo è sconsigliabile. Se usate di frequente montaggi remoti è opportuno che vi creiate dei punti di montaggio con directory vuoti dedicati. Una volta che un directory remoto è stato montato, lo potete usare praticamente come un normale directory locale.

24.4 Determinazione delle risorse montabili

Dopo che un'altra macchina ha dichiarato condivisi alcuni directory (come descritto in seguito in questo capitolo), quando vi occorrono, potete montare questi directory sulla vostra macchina. Potete con facilità determinare le risorse condivise disponibili nella rete usando **dfshares** (distributed file shares); senza argomenti, questo comando fornisce l'elenco di tutte le risorse disponibili alla vostra macchina tramite la rete, come nell'esempio:

```
sis__locale$ dfshares
RESOURCE                        SERVER      ACCESS    TRANSPORT
  sis__locale:/export           sis__locale rw        tcp
  sis__remoto:/usr/src/access   sis__remoto ro        tcp
sis__locale$
```

Potete selezionare una macchina specifica, indicando come argomento di **dfshares** il nome di host; questo è di solito opportuno con reti molto estese.

Tenete presente che alcune versioni di SVR4 richiedono che la vostra macchina sia in stato di init 3 (descritto in seguito in questo capitolo), perché il comando **dfshares** possa funzionare.

24.5 Montaggio di risorse da macchine remote

Potete montare un directory remoto nel punto di mount locale, se conoscete il pathname della risorsa remota. Solo il superuser della macchina locale può eseguire montaggi remoti. Abbiate cura di usare come punto di mount un directory vuoto; se necessario, create un nuovo directory, come nell'esempio:

```
# mkdir /usr/src/remote
# mount -F nfs sis__remoto:/usr/src/access /usr/src/remote
```

Indicate il nome del sistema host remoto, seguito da : e dal pathname remoto, senza spazi interposti; quindi indicate il punto di mount locale. Se l'operazione ha successo, il directory remoto è visibile nel vostro file system locale.

Per montare una risorsa remota dovete specificare il tipo di file system usato per la risorsa da montare; a questo scopo dovete usare l'opzione **-F**. Per le reti, sono disponibili i tipi di file system **nfs** e **rfs**. Alcune release di SVR4 possono consentire di omettere l'opzione **-F**, ma l'abitudine di usarla sempre può evitare degli errori. Il valore specificato per l'opzione **-F** deve coincidere con il tipo di rete in uso sia sulla macchina locale che sulla macchina remota.

La risorsa remota verrà montata, per questa utilizzazione, con i diritti di accesso predisposti al momento della definizione di condivisione della risorsa (rw, lettura-scrittura; ro, solo lettura). Potete richiedere diritti di accesso ridotti, con l'opzione **-o**, come nell'esempio:

```
# mount -F nfs -o ro sis__remoto:/usr/src /usr/src/remote
```

Non potete richiedere diritti di accesso più ampi di quelli specificati nella condivisione della risorsa.

MONTAGGIO DI RISORSE MULTIPLE

Potete montare una serie di directory con il comando **mountall**, oppure includere una serie di comandi **mount** in una procedura di comandi da eseguire in blocco. Il comando **mountall**, di default, preleva le sue direttive dal file **/etc/vfstab** (virtual file system table), mostrato in questo esempio:

```
# tail -2 /etc/vfstab
/dev/dsk/f1                    /dev/rdsk/f1    /install          s5   -  no  -
sis__remoto:/usr/src/access    -               /usr/src/remote   nfs  -  yes ro
#
```

Per ciascun montaggio è presente una linea, ciascuna linea comprende diversi campi, separati tra loro da spazi. I campi contengono: la collocazione del file system (dispositivo blocco, dispositivo raw); il punto locale di mount; il tipo di file system (**nfs, rfs, s5, ufs**); l'indicazione se per montare la risorsa è richiesta o no una password; **yes** o **no** a seconda se la risorsa deve essere montata automaticamente, o no; in ultimo, le opzioni per il montaggio.

Potete creare un vostro file per descrivere i montaggi che volete eseguire, con lo stesso formato del file **/etc/vfstab**; quindi, potete fornire il nome del file come argomento al comando **mountall**.

ELENCO DELLE RISORSE MONTATE

Potete ottenere l'elenco di tutte le risorse montate, col comando **mount** privo di argomenti; otterrete l'elenco di tutti i file system montati, sia locali che remoti. Questo uso del comando **mount** non è limitato solo al superuser.

SMONTAGGIO DI RISORSE REMOTE

Una risorsa montata rimane disponibile fino a che non viene smontata, o fino alla chiusura del sistema. Per rilasciare la risorsa potete usare il comando **umount**, con argomento il pathname locale, come nell'esempio:

```
# umount /usr/src/remote
```

Potete smontare con un'unica operazione tutte le risorse montate, usando il comando **umountall**. Questo comando deve essere comunque usato a ragion veduta, perché oltre a smontare tutte le risorse remote, può smontare anche risorse locali, come i dischetti.

24.6 Montaggi automatici

Il sesto campo delle linee del comando **mountall** nel file **/etc/vfstab** determina il montaggio di risorse al momento dell'inizializzazione del sistema; non trova significato quando il comando viene eseguito dalla linea di comando. Nell'inizializzazione, invece, **yes** o **no** nel sesto campo viene usato per stabilire se la risorsa deve essere montata automaticamente, o no. Il file **vfstab** controlla i montaggi sia locali che remoti. Esaminate il vostro file corrente **/etc/vfstab** per conoscere i montaggi al momento dell'inizializzazione.

Inoltre, potete configurare NFS (ma non RFS) per automontare un directory remoto quando un utente richiede di accedervi; in altre parole, il directory non viene montato fino a che non è necessario, ma a quel momento viene montato senza intervento del superuser. Il comando **automount** è un demon che sorveglia le richieste di open per file compresi in un elenco di directory predefinito; quando rileva un tentativo del genere, **automount** consulta un database per determinare come montare quel file, quindi esegue il mount. Questa caratteristica può essere molto utile, perché spesso la macchina locale può essere influenzata negativamente, se una risorsa montata da NFS si trova su una macchina non funzionante, o se un segmento della LAN non è operante. Per contro, rallenta l'accesso ai file dell'intero sistema, anche per directory che non rientrano nell'elenco di **automount**. Consultate la man page **automount(1)** per maggiori informazioni sull'uso di questo servizio.

24.7 Clienti, server e stati init

In SVR4, alcuni aspetti dell'accesso remoto ai file sono intesi per *clienti*, o macchine che montano risorse da altre macchine e altri aspetti sono intesi per *server*, o macchine che forniscono ai clienti i file da montare. Una macchina può essere cliente, o può essere cliente e server insieme, mentre non esiste la condizione solo di server. I comandi **rlogin**, **finger** e altri già descritti in questo capitolo, sono destinati sia a clienti che a server.

In genere, i server sono macchine veloci, con grandi dischi, mentre i clienti sono di solito stazioni di lavoro personali; tuttavia, questa distinzione non è concettualmente importante. Potete considerare ogni macchina un server quando condivide i propri file, ma potete riportarla allo stato di cliente quando volete solo montare file da altre macchine. In questo senso, gli strumenti di gestione della rete nel sistema UNIX implementano uno schema LAN *peer-to-peer* (pari a pari).

In pratica, tuttavia, esiste una ben definita differenza nelle prestazioni fra le macchine usate come clienti e quelle usate come server; queste ultime richiedono molti processi aggiuntivi per "ascoltare" le richieste di montaggio provenienti da altre macchine della rete ed eseguire le operazioni necessarie; un cliente, al contrario, non deve ascoltare, deve solo fare delle richieste quando vuole un servizio. In altri termini, le prestazioni in velocità delle macchine clienti risultano migliori di quelle dei server.

In SVR4, la distinzione fra cliente e server è implementata con lo *stato init* in cui la macchina si trova in un dato momento (per ulteriori informazioni sullo stato init consultate il Capitolo 21). Lo stato init 2 è usato per macchine standalone e macchine solo clienti; lo stato 3 è riservato ai server. Nello stato init 2 è possibile montare risorse da un'altra macchina, ma per condividere le risorse una macchina deve essere in stato init 3.

Il comando **who** – **r** (run level) informa sullo stato init corrente della macchina, come nell'esempio:

```
# who -r
    .           run-level 2   Mar   6 05:07     2     0     S
#
```

Il comando indica anche la data e l'ora dell'inizializzazione della macchina, assieme ad altre informazioni.

Per passare nello stato init 3, usate

```
# telinit 3
```

Quando siete pronti a ritornare in modo cliente, usate

```
# telinit 2
```

Se state usando il sistema RFS, per passare nello stato init 3 dovete usare il comando **rfstart** e usare il comando **rfstop** per ritornare in stato init 2.

Se mentre richiedete di montare una risorsa remota da un'altra macchina ricevete un messaggio d'errore simile a questo:

```
mount: sis_remoto:/export server not responding:
RPC: Program not registered
```

verificate che la macchina remota sia sicuramente in funzione e nello stato init 3.

24.8 Condivisione di risorse con altre macchine

Per concedere ad altre macchine l'accesso ai vostri directory, la vostra macchina deve essere nello stato init 3; inoltre, dovete dichiarare la condivisione delle risorse. Questa è una precauzione per la sicurezza, che fornisce controlli aggiuntivi oltre ai normali diritti di accesso ai file.

Il comando **share**, riservato al superuser, abilita altre macchine a montare vostri directory; il comando prevede diverse opzioni nella linea di comando per specificare il tipo di accesso remoto ammesso. Per il sottoalbero di un directory potete specificare l'accesso in sola-lettura (read-only), o in lettura-scrittura (read-write); usate l'opzione – **o** (only), seguita da **rw** (read-write), oppure **ro** (read-only). Dopo l'opzione – **o** specificate il directory che volete condividere, seguito da un nome per la risorsa; questo nome serve di riferimento quando volete revocare la dichiarazione di condivisione della risorsa. Se volete, potete aggiungere un testo descrittivo della risorsa, protetto, con l'opzione – **d**. Ecco un esempio:

```
# share -o ro -d "Normal Exports" /export MIEIFILE
```

Questo comando dichiara condivisibile l'albero di directory **/export**, con il nome **MIEIFILE**; comunque, le macchine remote non sono abilitate a scrivere in nessuna locazione dell'albero.

È anche possibile specificare queste restrizioni per ogni singola macchina abilitata a montare il directory; in altri termini, potete prevedere diversi livelli di accesso per le diverse macchine nella rete. Secondo l'esempio precedente, **share** abilita tutte le macchine nella rete allo stesso livello di accesso richiesto (*ro*). Per assegnare il livello di accesso a una particolare macchina, occorre aggiungere la forma *=macchina* dopo l'argomento dell'opzione **-o**; per esempio:

```
# share -o ro, rw=sis__remoto /export MIEIFILE
```

Questo limita tutte le macchine all'accesso in sola-lettura, eccettuata la macchina **sis__remoto**, che è abilitata in lettura-scrittura. Potete specificare un numero qualsiasi di macchine; i diversi elementi devono essere separati da una virgola, ma non devono contenere spazi, altrimenti la linea di comando viene male interpretata.

ELENCO DELLE RISORSE CONDIVISE

Il comando **share** senza argomenti elenca i directory della vostra macchina condivisi, come nell'esempio:

```
#sis__locale$ share
MIEIFILE        /export          rw                    "Normal Exports"
SRC1            /usr/src/access  ro,rw=sis__remoto "My Source Tree"
sis__locale$
```

La prima colonna indica il nome della risorsa, assegnato alla dichiarazione della condivisione; la seconda colonna il directory condiviso; la terza i permessi d'accesso; l'ultima contiene la descrizione, se è stata specificata. Questa utilizzazione di **share** è consentita a tutti gli utenti e non solamente al superuser.

CONDIVISIONE DI RISORSE MULTIPLE

Potete dichiarare la condivisione di un gruppo di risorse, con il comando **shareall**, come nell'esempio:

```
# shareall cmdfile
#
```

Il file *cmdfile* contiene una serie di linee di comando **share**, come descritto precedentemente.

Se nella linea di comando **shareall** non viene specificato il nome di un file, viene usato il file di default **/etc/dfs/dfstab** (distributed file sharing table); di solito il file **dfstab** contiene commenti con istruzioni sul modo di compilare la tabella.

I directory condivisi con i comandi **share** e **shareall** non permangono condivisi, se la macchina abbandona lo stato init 3 e vi rientra, a meno che non prevediate a questo scopo linee di comando in **dfstab**. Il file **dfstab** viene trattato tutte le volte che la macchina entra nello stato init 3, perciò potete predisporre il vostro ambiente in condivisione una volta per tutte e verrà assegnato tutte le volte che la macchina viene usata come server. Comunque, abbiate cura di includere in **dfstab** solo i directory che volete sempre condivisi; per condivisioni occasionali utilizzate un diverso vostro file.

REVOCA DELLA CONDIVISIONE DI RISORSE

Per rimuovere un directory dall'elenco delle risorse condivise, usate il comando **unshare**, che richiede come argomento il pathname della risorsa; per esempio:

```
# unshare /export
```

Il comando non fornisce alcun messaggio se l'operazione viene eseguita con successo. Alcune versioni di **unshare** possono accettare il nome della risorsa in luogo del pathname, per identificare la risorsa da rimuovere.

Se un utente su una macchina remota ha eseguito un **cd** nel directory condiviso, o sta utilizzando un file, **unshare** ha ugualmente effetto, ma l'utente riceverà un messaggio d'errore al suo prossimo accesso alla risorsa non più condivisa. Per esempio:

```
sis_remoto$ ls /mnt
ls: /mnt: Stale NFS file handle
```

Poiché questo può risultare spiacevole ai vostri utenti remoti, dovete assicurarvi che le risorse siano inutilizzate prima di revocarne la condivisione.

Il comando **unshareall** revoca la condivisione con tutti gli altri utenti di tutti i vostri directory condivisi al momento.

ELENCO DELLE RISORSE CONDIVISE IN USO

Il comando **dfmounts** (distributed file mounts) fornisce un elenco delle altre macchine che hanno montato le vostre risorse condivise. Questo consente di conoscere le utilizzazioni da parte di altri utenti, prima di revocare la

condivisione con **unshare**. Il comando **dfmounts** senza argomenti elenca gli utilizzatori di tutte le vostre risorse condivise, come nell'esempio:

```
# dfmounts
RESOURCE    SERVER PATHNAME    CLIENTS
MIEIFILE    sis__locale /export    sis__remoto
#
```

Il formato in uscita può differire a seconda che usiate NFS oppure RFS, ma in ambedue i casi include il nome della risorsa e le macchine clienti che hanno montato la risorsa.

CONSIDERAZIONI SULLA SICUREZZA DELLE RISORSE CONDIVISE

Di solito, una politica di rete stabilisce il livello di condivisione consentito; questa politica è determinata dalle necessità degli utenti e da considerazioni di sicurezza. In un ambiente omogeneo dove ognuno ha necessità di accedere ai dati degli altri utenti, molti directory, o i directory superiori nella gerarchia del file system, saranno probabilmente condivisi; in un ambiente che deve essere più sicuro, dove i dati pubblici sono contenuti in un ridotto albero di directory, solo un piccolo numero di directory dovrà essere condiviso. In SVR4 il directory **/export** è riservato per la condivisione; questo directory può essere vuoto in una configurazione di default, ma voi potete includervi il materiale che risulta opportuno condividere.

Tenete ben presente che, una volta che una macchina condivide le proprie risorse, molte macchine nella rete possono accedere a quelle risorse; è consigliabile revocare la condivisione non appena le risorse non sono più necessarie nella rete.

24.9 Approfondimenti

Come potete immaginare, la gestione della rete è un argomento complesso e difficile; la descrizione precedente comprende solo i comandi e le procedure di uso più generale. Con l'incorporazione di NFS e RFS, questi argomenti hanno accresciuto ancora la loro complessità. Per diventare realmente un esperto di reti avrete bisogno di molta esperienza; le parti seguenti sono solo un'introduzione ad alcuni degli argomenti interessati. La procedura per la configurazione di una rete non viene descritta in questo testo; si presuppone che la rete sia già installata e funzionante, e si descrivono solo le procedure per aggiungere una macchina a una rete esistente.

IL COMANDO rdate

Il comando **rdate** (remote date) permette di inizializzare la data e l'ora locale da una macchina remota della vostra LAN; questa sorgente principale della data è denominata *date server*. Se fate uso di un qualsiasi strumento che compara le date di modifica dei file su diverse macchine, è opportuno avere una unica gestione centralizzata della data e dell'ora. Il comando **rdate** richiede come argomento il nome di host, legge la data e l'ora da quella macchina e li carica nella macchina locale; per esempio:

```
# rdate sis_remoto
Sat Apr 28 11:59:03 1991
#
```

Se ha successo, **rdate** visualizza la data e l'ora aggiornate. L'uso di **rdate** è riservato al superuser; spesso viene introdotto in una procedura di comandi eseguita all'inizializzazione, per garantire che la macchina venga sincronizzata immediatamente appena messa in funzione.

IL COMANDO telnet

Il comando **telnet** è simile nelle sue funzioni al comando **rlogin**, ma può essere usato anche in situazioni in cui **rlogin** è inadeguato. Per esempio, **rlogin** può essere usato solo fra sistemi UNIX, mentre **telnet** può comunicare tra sistemi operativi differenti tramite una LAN; inoltre, **telnet** usa un canale di comunicazione diverso, quindi può essere usato quando **rlogin** non funziona per un qualunque motivo. Alcuni terminali X usano **telnet** per stabilire il contatto iniziale con la macchina cliente X che inizia il primo **xterm** per una sessione. Il comando ha anche molti altri impieghi; tuttavia, poiché fornisce un minor numero di servizi rispetto a **rlogin**, l'uso di **telnet** dovrebbe essere riservato a situazioni speciali.

Il comando **telnet** richiede come argomento un nome di host o un indirizzo Internet, e si connette al servizio di **login** di quella macchina remota, come nell'esempio:

```
$ telnet sis_remoto
Trying 127.12.0.30 ...
Connected to sis_remoto
Escape character is '^]'.

sis_remoto login:
```

Dovete effettuare un normale login alla macchina remota; tutti i caratteri da voi introdotti vengono passati per il trattamento alla macchina remota e voi potete eseguire sulla macchina remota i comandi normalmente.

Tenete conto del fatto che, al contrario di **rlogin**, il comando **telnet** non

passa il vostro ambiente corrente all'elaboratore host remoto al momento della connessione; dovete predisporre esplicitamente il vostro ambiente nel vostro **.profile** nella macchina remota. Il comando **telnet** può emulare internamente un terminale VT102, perciò dovrete compiere alcune prove con diversi valori della variabile **TERM**, fino a ottenere i migliori risultati. Di solito, dovrete assegnare a **TERM** i valori "VT102", "VT100", oppure "ansi".

Il comando **telnet** include un modo comandi che fornisce numerose opzioni; potete passare dal modo terminale al modo comandi con il carattere escape di **telnet**, che di default è CTRL-] (control parentesi quadra chiusa). Per esempio:

```
$ CTRL-]
telnet>
```

Il nuovo prompt proviene dal modo comandi locale di **telnet**; a questo punto, potete introdurre uno dei numerosi comandi **telnet**. Usate **close** per concludere la sessione con l'host remoto, rimanendo in modo comandi **telnet**; usate **quit** per concludere sia la sessione remota che il comando **telnet**, ritornando in shell locale, come nell'esempio:

```
telnet> quit
Connection closed.
$
```

Dovreste uscire dall'elaboratore remoto prima di usare il comando **quit**, ma talvolta il logout termina il comando **telnet** e vi riporta in shell locale.

Il comando **?** fornisce un breve elenco di aiuto con tutti i comandi disponibili. Un solo RETURN in modo comandi vi riporta in shell dell'elaboratore host remoto.

PROCESSI DEMON DI RETE

Molti dei servizi descritti richiedono che nella macchina server siano in funzione processi demon; questi demon sono di solito lanciati quando la macchina entra in stato init 3. Potete esaminare le differenze dell'ambiente dei processi fra lo stato init 2 e lo stato init 3 della vostra macchina, mediante il raffronto dei processi elencati da **ps –ef** nelle due situazioni. Oltre a questi demon fondamentali, altri demon addizionali, non lanciati di default, vengono richiesti da alcuni dei comandi di accesso remoto. Se comandi come **rusers** o **finger** non funzionano, questo probabilmente dipende dalla mancanza di speciali demon dedicati; consultatevi col gestore della rete per predisporre il lancio di questi demon che risultino mancanti nella vostra rete.

INSERIMENTO DI UNA NUOVA MACCHINA NELLA RETE

Per includere una nuova macchina in una rete LAN esistente, dovete configurare correttamente gli strumenti di rete. Le procedure differiscono fra le reti; tuttavia dovete sempre completarle subito dopo avere inserito fisicamente il sistema nella rete.

Se usate solo uno dei due tipi di rete NFS o RFS, dovete informare gli strumenti di rete di quale tipo viene fatto uso; l'informazione è contenuta nel file **/etc/dfs/fstypes** (file system types). All'installazione del sistema, di solito questo file contiene ambedue i tipi di rete, come nell'esempio:

```
# cat /etc/dfs/fstypes
nfs     Network File System Utilities: 2.0
rfs     RFS Utilities: 2.0
#
```

Potete escludere la rete inutilizzata inserendo un # all'inizio della riga del tipo di rete che non usate (la riga diventa un commento). In mancanza di questa operazione, il sistema presuppone che utilizziate ambedue i tipi di rete; molti comandi DFS ne risulterebbero confusi.

PREDISPOSIZIONE DI NFS

Dopo avere connesso fisicamente la vostra rete, dovete configurare il sistema NFS; l'operazione è relativamente facile, ma deve essere eseguita correttamente. Prima di iniziare, dovete possedere tre informazioni per la vostra macchina: il suo indirizzo Internet; il suo indirizzo fisico; il suo nome di host. L'indirizzo Internet è un numero in quattro parti che individua la macchina con un indirizzo logico che deve essere unico in tutto il mondo; viene assegnato dal gestore della rete in accordo con lo schema gerarchico di numerazione internazionale. L'indirizzo fisico di hardware è stabilito dal controller hardware Ethernet ed è anch'esso unico nel mondo; viene di solito visualizzato all'inizializzazione della macchina.

ELENCO DEGLI HOST

Tutti i comandi di rete descritti utilizzano nomi logici di macchine, che sono i nomi di host delle macchine nella rete. La rete, tuttavia, per la comunicazione tra le macchine utilizza internamente uno schema di indirizzamento numerico; questo è l'indirizzo Internet. Il file **/etc/inet/hosts** (collegato a **/etc/hosts**), contiene la mappatura tra queste due forme di indirizzamento, e i comandi di rete devono consultare questo file per connettersi alla macchina remota. Il file ha questo aspetto:

```
# cat /etc/inet/hosts
# If the yellow pages is running, this file is only used when booting
#
# Internet host table
0.0.0.0         anyhost
127.0.0.1       localhost
130.0.17.35     sis_locale
130.0.21.37     sis_remoto
#
```

Il file **/etc/inet/hosts** contiene l'esatto indirizzo Internet e il nome di host di ogni macchina nella rete; ogni sistema deve essere descritto da una linea come la seguente:

```
130.0.17.35  sis_locale
```

Nell'esempio precedente, le linee

```
0.0.0.0      anyhost
127.0.0.1    localhost
```

sono riferimenti speciali. Il riferimento **anyhost** fornisce un indirizzo generico per qualunque macchina, o meglio, per macchine che non conoscono il proprio indirizzo; questo indirizzo spesso viene usato per messaggi *broadcast*, diretti a tutte le macchine. Il riferimento **localhost** è un altro termine generico che si riferisce a comunicazioni in *loopback* fra applicazioni sulla macchina locale. Il commento all'inizio del file **/etc/inet/hosts** ha lo scopo di ricordare che questo file non viene usato quando è in esercizio il sistema *Yellow Pages*; questo sistema verrà descritto alla fine di questo capitolo.

PROCEDURE DI PREDISPOSIZIONE DI NFS

Molte schede Ethernet di recente diffusione forniscono automaticamente il loro indirizzo fisico al kernel UNIX quando la macchina viene inizializzata; in caso contrario, all'inizializzazione deve essere consultato il file **/etc/bootptab** per determinare l'indirizzo logico Internet a partire dall'indirizzo fisico. Verificate la presenza di questo file nel vostro sistema; nel caso sia presente, includetevi l'indirizzo Internet e l'indirizzo fisico della vostra macchina.

Successivamente, aggiungete i vostri nomi di host e indirizzo Internet nel file **/etc/inet/hosts**. Se la vostra macchina è una stazione di lavoro senza dischi, oppure se inizializza da un'altra macchina anziché dal proprio disco, avrete necessità di aggiornare anche il file **/etc/bootparams**, in modo che la vostra macchina possa conoscere a quale *boot server* richiedere i servizi. Se la vostra rete è già funzionante, queste operazioni dovrebbero essere suffi-

cienti. Ricordate, comunque, che queste procedure non devono essere usate, se nella vostra rete è in funzione il servizio Yellow Pages. Dopo una reinizializzazione, la vostra macchina dovrebbe essere inserita nella LAN.

Potete verificare la configurazione con

```
# ping localhost
```

Questo comando funziona, se l'hardware di rete nella vostra macchina funziona correttamente. Il comando

```
# ping nome-host
```

in cui *nome-host* è il nome della vostra macchina, funziona se l'indirizzamento nella rete è corretto. Per verificare se la vostra macchina può comunicare con altre macchine tramite la rete, usate come *nome-host* il nome di un'altra macchina nella vostra rete.

PREDISPOSIZIONE DI UNA MACCHINA IN RFS

Le procedure di predisposizione differiscono fra NFS e RFS; accertatevi di usare la procedura corretta per la vostra rete.

Per usare RFS, dovete configurare il dominio, e l'apparenza della vostra macchina nel dominio oppure potete inserirvi in un dominio esistente. Un dominio è una suddivisione separata di una rete più grande, controllata da demon che sono in funzione in un *name server* del dominio. Se create un nuovo dominio, la vostra macchina può essere il name server primario o secondario, oppure una macchina normale che può condividere e montare risorse. Potete circoscrivere la vostra macchina in uno specifico dominio della rete, oppure potete proteggere l'accesso alle risorse mediante password. Prima di configurare un nuovo dominio, o installare per la prima volta RFS nella vostra macchina, consultate la documentazione RFS e il gestore della vostra rete, perché RFS e i domini implicano molte procedure opzionali e sofisticati criteri di sicurezza.

DOMINI E NAME SERVER IN RFS

Per dominio si intende un gruppo di macchine che acconsentono a comunicare tra loro utilizzando il sistema di rete RFS, ma possono escludere altre macchine che non sono membri del dominio. Anche le reti NFS possono prevedere i domini, generalmente controllati dal gestore della rete. In questo testo tratteremo solo i domini RFS.

Una grande LAN che collega molte macchine RFS può avere diversi domini separati che non entrano in conflitto tra loro. Una macchina può apparte-

nere a più domini, ma di solito le macchine di un dominio non accedono ai file su macchine di un altro dominio.

Ogni dominio ha esattamente un unico *primary name server*; questa macchina è responsabile della gestione del dominio nel suo complesso, e di solito appartiene al gestore della rete o del dominio. Inoltre, è possibile definire uno o più *secondary name server*, che non vengono abitualmente utilizzati per la gestione del dominio, ma possono eseguire le funzioni di routine di *name server* se la macchina primaria è fuori servizio per qualche motivo. Il primary name server esegue speciali processi demon che hanno il compito di controllare il dominio e le sue attività. Altre macchine possono unirsi al dominio per condividere o montare file, senza essere name server.

DEFINIZIONE DI UN DOMINIO

Se intendete definire un nuovo dominio, innanzitutto dovete predisporre il primary name server. Se vi unite a un dominio esistente, dovete chiedere al gestore del dominio di includere la vostra macchina nel database del dominio, se la vostra macchina deve condividere risorse, oppure se nel dominio sono in vigore procedure di verifica del cliente. Dopo di questo, potete iniziare l'attività nella rete RFS e montare risorse remote quando necessario.

Per attivare un nuovo dominio, sono necessari più passi, da eseguire sulla macchina dedicata come primary name server del nuovo dominio. Innanzitutto, definite il nome del dominio con il comando

```
# dname -D dominio
```

dove *dominio* è l'identificatore univoco del dominio. Il nome di un dominio deve avere una lunghezza non superiore a 14 caratteri e deve essere unico nella vostra rete.

Dovete poi assegnare una rete al dominio, dal momento che una macchina può essere collegata a più di una rete, come nell'esempio:

```
# dname -N starlan
```

In questo esempio, **starlan** indica il file speciale contenuto in **/dev** che definisce il collegamento con la rete che intendete utilizzare; del tutto casualmente, coincide col nome della rete. Se usate RFS con un Ethernet dovete indicare invece **tcp**.

Il comando **dname** può anche riportare il nome del dominio e il tipo di rete su una macchina. Utilizzate

```
# dname -a
```

per vedere queste informazioni, una volta creato il dominio.

DEFINIZIONE DI NAME SERVER

Successivamente occorre creare un file che contiene una tabella di name server per il dominio. Un dominio deve avere solo un primary name server, ma può avere diversi secondary name server. Il primary name server esegue i processi demon di gestione del dominio ed effettua le verifiche di sicurezza; i secondary name server prendono in carico questi compiti quando il primary name server non è in funzione. Poiché i compiti del name server aggiungono ulteriore carico alla macchina, di solito vengono specificati pochi secondary name server su un dominio.

Sulla macchina designata come primary name server, utilizzate un editor di testo per creare o aggiornare il file **/etc/rfs/starlan/rfmaster** (remote file master), che contiene la lista principale di name server per il dominio. Quando RFS è in esecuzione, questo elenco apparirà su tutte le altre macchine server nel dominio. Nel terzo componente del pathname di questo file (nell'esempio **starlan**) dovete usare lo stesso nome di rete che avete usato nell'assegnare la rete al dominio (in questo caso, era appunto **starlan**).

Il file **rfmaster** contiene la lista dei name server, specifica se la macchina è un primary o un secondary name server e indica l'indirizzo di rete di ciascun server:

```
sis__locale$ cat /etc/rfs/starlan/rfmaster
# qui di seguito sono elencati i primary e secondary name server
locale p locale.giorgio
locale s locale.sis__locale
# seguono gli indirizzi di dominio e di rete
locale.giorgio a giorgio.serve
locale.sis__locale a sis__locale.serve
sis__locale $
```

Ogni linea si compone di tre campi separati da un carattere spazio. Alcune versioni consentono di inserire solo un carattere spazio o un carattere di tabulazione; un numero maggiore di caratteri spazio può causare un'errata interpretazione del file.

Nel file **rfmaster** si trovano due tipi di linee: quelle che descrivono i name server e quelle che specificano gli indirizzi di rete dei name server. Il primo campo indica il dominio o il nome della macchina, il secondo è un codice di azione e il terzo è un indirizzo di rete o di dominio. I name server vengono elencati per primi. Nel precedente esempio, il dominio è chiamato **locale** e il primary (**p**) name server è la macchina **giorgio** all'indirizzo di dominio **locale.giorgio**. La macchina **sis__locale** è il secondary (**s**) name server e il suo indirizzo di rete è **locale.sis__locale**. È possibile avere solo una linea **p**, ma diverse linee **s**.

Gli indirizzi di rete vengono poi messi in corrispondenza con gli indirizzi di dominio (**a**) nelle linee successive; solo i name server debbono essere

menzionati nelle linee **a**. Il primo campo indica l'indirizzo del dominio e il terzo campo è l'indirizzo di rete. Gli indirizzi di rete nella forma *uname.serve* sono tipici delle reti StarLan. Nel file **rfmaster** possono essere inserite linee di commento, che iniziano con il carattere # (segno di cancelletto). Verificate che **rfmaster** sia esatto, scrivetelo e uscite dall'editor. Verificate che **root** abbia la proprietà del file e che i diritti di accesso siano 0644.

AGGIUNGERE LE MACCHINE AL DOMINIO

A questo punto, potete aggiungere le macchine al dominio, tramite il comando **rfadmin** con l'opzione **-a** (add):

```
# rfadmin -a dominio.nome-host
Enter password for nome-host:
```

dove *dominio* è il nome del dominio e *nome-host* è il nome della macchina che intendete includere nel dominio. Il comando **rfadmin** richiede di specificare una password, che dovrà essere introdotta dal gestore sulla macchina *nome-host* quando attiverà il sistema della rete su quella macchina. Potete premete solo il tasto di ritorno a capo per non imporre la verifica di password, ma quando aggiungete nuove macchine accertatevi di rispettare la politica di sicurezza della vostra rete.

Per rimuovere una macchina dal dominio usate **rfadmin** con l'opzione **-r** (remove), come nell'esempio:

```
# rfadmim -r dominio.nome-host
```

Per visualizzare il name server del dominio, utilizzate il comando

```
# rfadmin
```

senza argomenti. Il name server corrente per il dominio deve comunque essere la vostra macchina, altrimenti non potete eseguire nessuno dei compiti di gestione del dominio.

Adesso, tramite il comando **rfstart**, potete attivare il sistema RFS, prima sul primary name server e poi sulle macchine secondary name server. A questo punto, il vostro dominio è attivo e in esecuzione, per cui può essere utilizzato da altre macchine. Per avere a disposizione le risorse remote, usate i comandi **share** e **mount**.

Potete determinare in qualunque momento lo stato corrente di RFS con l'opzione **-q** (query) di **rfadmin**, come nell'esempio:

```
# rfadmin -q
RFS is running
#
```

In presenza di qualsiasi malfunzionamento, verificate sempre che RFS sia running.

CAMBIARE UN DOMINIO ESISTENTE

Potete passare le responsabilità di un primary name server a un secondary name server con il seguente comando:

```
# rfadmin -p
```

Questa operazione è necessaria per esempio prima di spegnere la macchina primary per manutenzione, e va intesa come una modifica temporanea nei name server. Non appena il primary rientra in funzione, lanciate

```
# rfadmin -p
```

sul corrente name server per restituire le responsabilità al primary. Per cambiare permanentemente il primary name server, occorre riscrivere il file **rfmaster**. Assicuratevi di fermare con **rfstop** il sistema RFS su tutti i primary e secondary name server del dominio, prima di cambiare i name server. Quindi, modificate il file **rfmaster** col nuovo primary, attivate RFS con il comando **rfstart** per il nuovo primary, infine, fate partire le macchine secondary.

Quando i name server funzionano correttamente, la macchina primary distribuisce regolarmente i file chiave di gestione ai secondary name server. Se il primary name server improvvisamente cessa di funzionare, le responsabilità passano automaticamente al primo secondary name server presente nel file **rfmaster**; il malfunzionamento non influenza alcuna delle risorse montate nel dominio, a meno che non risiedano sulla macchina fuori uso. Comunque, le macchine secondary non possono eseguire permanentemente i compiti di name server, per cui se il primary non ritorna presto in linea, dovete passare la responsabilità del primary a un'altra macchina, di solito una dei primi secondary. Controllate attentamente l'eventuale presenza di errori nel file **rfmaster** e nelle risorse condivisibili, perché i secondary non possiedono sempre file di gestione perfettamente correnti.

ALTRE CARATTERISTICHE DI RFS

Il sistema RFS consente di mettere in corrispondenza gli utenti su macchine remote con gli identificatori di utente e gli identificatori di gruppo sulle macchine server. In questo modo, viene attuato un controllo completo dei diritti di accesso concessi agli utenti remoti sulle vostre risorse. La gestione di queste corrispondenze può risultare complessa, per cui prima di farne uso dovete consultare la documentazione del sistema RFS.

Infine, esistono diversi strumenti per misurare l'utilizzo di RFS e affinarne le prestazioni secondo le vostre esigenze. Il comando **fusage** (file usage) consente di analizzare come i clienti impiegano le vostre risorse; il comando **sar** (system activity reporting) determina quanto tempo di CPU occorre per accedere ai file in modalità remota. Potete cambiare i server e le loro responsabilità per minimizzare il tempo di CPU nell'intero dominio; potete modificare i valori dei parametri RFS per limitare il massimo numero di utilizzatori delle vostre risorse, o il massimo numero di processi associati con RFS, e porre altre limitazioni. Si tratta comunque di funzioni complesse che devono essere usate con attenzione.

USO CONTEMPORANEO DI NFS E RFS SU UNA MACCHINA

Normalmente, per la vostra rete sceglierete fra NFS o RFS, o un altro protocollo di comunicazione compatibile. La decisione dipende dalla necessità per la vostra macchina di inserirsi in una rete già esistente che usa uno di questi sistemi. Comunque, SVR4 consente di usare differenti protocolli su una stessa rete, anche se il caso in pratica è abbastanza raro. Per realizzare questa possibilità, dovete configurare tutte le reti correttamente, come descritto; quindi, potrete selettivamente condividere e montare risorse da una o dall'altra. Potete ottenere informazioni sulle risorse disponibili in ciascun sistema di rete e potete avviare e fermare una o l'altra indipendentemente; comunque, sia NFS che RFS richiedono che la macchina sia nello stato init per condividere le risorse.

Tutti i comandi **share, mount, unshare, umount, shareall, unshareall, umountall** e **dfshares** accettano nella linea di comando l'argomento **-F**, che specifica quale sistema intendete usare con quel comando, come nell'esempio:

```
# share -F nfs -o ro /usr/src
```

Questo argomento addizionale della linea di comando è obbligatorio solo quando state usando contemporaneamente NFS e RFS; mentre, eccetto che con il comando **mount**, non è obbligatorio quando usate solo uno dei due sistemi.

COMANDI RFS DI SVR3 SOTTO SVR4

Oltre alle nuove procedure **share** e **mount**, SVR4 supporta tuttora i comandi usati per gestire una rete RFS in SVR3. Poiché i nuovi comandi sono di uso molto più semplice e supportano sia NFS che RFS, è preferibile usarli tutte le volte che è possibile; ma, se volete gestire diversi domini RFS nella vostra rete, potrete avere necessità dei vecchi comandi per ottenere un controllo individuale sui domini e sui loro membri.

Per ottenere l'elenco delle risorse RFS disponibili, potete usare il comando **nsquery**, come nell'esempio:

```
sis__locale$ nsquery
RESOURCE   ACCESS      SERVER              TRANSPORT   DESCRIPTION
DATA       read/write  locale.sis__locale  starlan     Real database
DATA__BKUP read-only   locale.sis__locale  starlan     Backup database
SCRATCH    read/write  big.sis__locale     starlan     For temp moves
GIORGIO    read/write  big.giorgio         starlan     giorgio public directory
BILL       read-only   locale.sis__remoto  starlan     bill public directory
sis__locale$
```

Nell'esempio, sono presenti tre macchine in due differenti domini; i domini sono chiamati *locale* e *big*, le macchine sono **sis__locale**, **sis__remoto** e **giorgio**. Potete limitare l'uscita di **nsquery** alle risorse in un sottoinsieme dell'intero elenco, indicando come argomento la macchina o il dominio, come negli esempi:

```
sis__locale$ nsquery locale.sis__locale
sis__locale$ nsquery sis__locale
sis__locale$ nsquery big.
```

Nel riferimento a un dominio è obbligatorio il punto.

Il comando **/usr/sbin/fuser** (file system users) riporta gli utenti correnti delle vostre risorse RFS condivise; il comando accetta come argomento un directory e fornisce l'identificatore di processo dei processi che stanno usando la risorsa. Con l'opzione **-u** (users) ottenete anche l'indicazione degli identificatori di login dei proprietari dei processi, come nell'esempio:

```
# fuser -u /export
/export:      121c(giorgio)   123p(jim)   135c(jim)
```

In questo esempio, la risorsa **/export** viene usata dai tre processi con PID 121, 123 e 135. L'identificatore di utente viene indicato tra parentesi accanto al numero del processo; la lettera **c** (current directory) segnala gli utenti che usano la risorsa come loro directory corrente, la lettera **p** (parent) segnala la risorsa come directory padre del processo, la lettera **r** (root) segnala la risorsa come directory root del processo. Per le risorse remote, l'elenco include ogni file all'interno della risorsa montata. Il comando **fuser** può riportare anche l'uso di risorse locali, come i file system montati; questo può essere utile per identificare gli utilizzatori di dischetti o altri supporti rimovibili.

In un caso di emergenza, potete utilizzare il comando **fumount** (forced unmount) per forzare lo smontaggio di una risorsa, anche se in corso di utilizzazione da parte di utenti. Poiché questa procedura può causare seri inconvenienti agli utenti remoti, deve essere usata in casi del tutto particolari.

Il comando **fumount** richiede come argomento il nome di una risorsa, e permette di predisporre un ritardo opzionale con l'opzione **-w** (wait), seguita da un numero che esprime i secondi di ritardo, come nell'esempio:

```
# fumount -w 30 DATA
```

Il comando **fumount** lancia un processo **rfuadmin** (remote file user administration) su ciascuna macchina cliente che sta usando la risorsa; questo processo ha il compito di smontare la risorsa da quella macchina. Se è stata usata l'opzione **-w** il processo **rfuadmin** avvisa dello smontaggio tutti gli utenti clienti col comando **wall**, quindi attende il tempo di ritardo, prima di avviare localmente lo smontaggio della risorsa con **fumount**.

I processi che accedevano alla risorsa verranno cancellati, quindi, se siete un utente cliente e ricevete un messaggio del tipo

```
DATA is being removed from the system in 30 seconds.
```

dovete immediatamente abbandonare l'uso della risorsa, terminando i programmi che la usano, o usando **cd** per uscire dal directory della risorsa. Questa procedura viene eseguita automaticamente quando l'elaboratore host viene spento.

I comandi **adv** (advertise) e **unadv** (unadvertise) funzionano come **share** e **unshare**, ma accettano come argomenti il nome della risorsa e un pathname; per esempio:

```
# adv DATA /usr/giorgio/my.data
```

Il nome di risorsa viene assegnato alla risorsa condivisa, specificata col pathname del directory locale da condividere.

I comandi **adv** e **unadv** prevedono nella linea di comando anche altri indicatori e argomenti, in particolare uno che consente di scegliere per la risorsa un determinato dominio. Consultate le man page opportune per maggiori dettagli.

Sono previsti molti altri comandi di gestione RFS, ma la maggior parte è destinata alla gestione di reti con rigide restrizioni di sicurezza.

PREDISPOSIZIONE DI MAIL SU UNA RETE

La posta elettronica basata su LAN di solito non usa il servizio di transport **uucp** descritto nei Capitoli 14 e 15; invece, vengono spesso usati un protocollo e demon speciali. Il comando **smtp** (simple mail transport protocol) viene usato internamente per trasferire la posta mediante LAN, mentre il demon **smtpd** è in ascolto per la posta in arrivo. Nessuno di questi programmi è accessibile agli utenti. Il normale programma **mail** può, invece, essere

configurato per ricercare sia nei file dati di **uucp** che nelle tabelle degli host di rete **/etc/inet/hosts**, e usare la prima via di inoltro che trova disponibile nella macchina.

Nei sistemi SVR4, il file di controllo per questo processo di ricerca è **/etc/mail/mailsurr** (mail surrogate); quando **mail** è in funzione, legge questo file per "riscrivere" un indirizzo ed eseguire l'opportuno agente transport di mail, per quella parte di mail. Il file **mailsurr** contiene un elenco di complesse regole che assomigliano a espressioni regolari; ogni indirizzo di mail è accoppiato con una di queste regole e il comando associato viene eseguito. L'ordine delle regole nel file **mailsurr** è significativo, perché il primo accoppiamento che contiene un comando di agente transport termina la ricerca.

In molte release standard di sistema SVR4, le linee di **mailsurr** che supportano il transport di mail **uucp** sono abilitate, mentre sono disabilitate le linee per **smtp**. Per esempio:

```
# tail -13 /etc/mail/mailsurr

# For remote mail via uucp and smtp. Uucp is first because
# it is more universal and handles binary mail properly.
#
'.+'      '!([^!]+)!(.+)'      '< /usr/bin/uux - \1!rmail (\2)'
#'.+'     '!([^!]+)!(.+)'      '< /usr/lib/mail/surrcmd/smtpqer %R \1 \2'

# If none of the above work, ship remote mail to a smarter host.
# Make certain that SMARTERHOST= is defined in /etc/mail/mailcnfg.
# If there is no smarter host, then routed mail fails here.
#
'.+'      '!(.+)'      'Translate R=!%X!\1'
#
```

Le linee che iniziano con # sono commenti. Nell'esempio precedente, il transport **uucp** è abilitato, mentre **smtp** è disabilitato. Potete abilitare la posta LAN cancellando il # all'inizio della linea con **/usr/lib/mail/surrcmd...**, come in questo esempio:

```
# tail -13 /etc/mail/mailsurr

# For remote mail via uucp and smtp. Uucp is first because
# it is more universal and handles binary mail properly.
#
'.+'      '!([^!]+)!(.+)'      '< /usr/bin/uux - \1!rmail (\2)'
'.+'      '!([^!]+)!(.+)'      '< /usr/lib/mail/surrcmd/smtpqer %R \1 \2'

# If none of the above work, ship remote mail to a smarter host.
# Make certain that SMARTERHOST= is defined in /etc/mail/mailcnfg.
# If there is no smarter host, then routed mail fails here.
#
'.+'      '!(.+)'      'Translate R=!%X!\1'
#
```

Notate che in quest'esempio le regole **uucp** vengono esaminate prima delle regole **smtp**; potete invertire l'ordine di queste due linee per cambiare il comportamento.

L'ultima linea dell'esempio consente al sistema **mail** di trasferire il messaggio a un gateway host nella LAN, se non riesce a risolvere l'indirizzo. Anche questa linea non deve essere un commento, se la vostra LAN prevede una tale macchina gateway.

Inoltre, potete aggiungere o creare linee nel file **/etc/mail/mailcnfg**, per descrivere il vostro dominio e il nome di host del gateway, come nell'esempio:

```
# cat /etc/mail/mailcnfg
DOMAIN = .mydomain.com
SMARTERHOST = yoursys
#
```

Chiedete al gestore della vostra rete i nomi del dominio e della macchina gateway.

Se avete il pacchetto di compatibilità BSD, potete configurare **mail** per usare la versione **sendmail** in luogo della versione **mailsurr**. Dovrete rimpiazzare i programmi **/usr/bin/mail** e **/usr/bin/rmail** con i loro equivalenti BSD, rispettivamente **/usr/ucblib/binmail** e **/usr/ucblib/binrmail**. Muovete il programma **sendmail.cf** in **/etc** e rimpiazzate anche **/etc/mail/mailsurr** con **/usr/ucblib/mailsurr**. Per integrare questo processo nel vostro normale uso di **mail** occorreranno alcuni tentativi; come al solito, probabilmente il gestore della vostra rete disporrà di maggiori informazioni riguardo all'adattamento del sistema **sendmail**.

APPLICAZIONI COMMERCIALI CON RETI

Molte applicazioni commerciali sulle reti, come i database, sono basate sui concetti implementati nelle utility UNIX descritte in questo capitolo. Molte di queste applicazioni fanno uso di un processo locale, che invia tramite la rete una richiesta di servizio a un singolo processo server centralizzato, su una macchina server. Questo processo server di solito è un demon nella macchina server, che "ascolta" le richieste in arrivo da parte dei clienti e le soddisfa quando si presentano. Spesso, il processo locale cliente gestisce le interfacce utente, la connessione e le procedure di comunicazione, mentre il server svolge il lavoro effettivo dell'applicazione. Se il processo server centralizzato, o la rete nel suo insieme, non funzionano correttamente, il processo locale non può funzionare.

Molte applicazioni commerciali prevedono un demon server di licenza, centralizzato, che viene interessato prima di lanciare un'applicazione locale. Questo server di licenza verifica quante istanze dell'applicazione sono

ammesse a funzionare contemporaneamente nell'intera rete; se quel numero non è stato superato, il nuovo utente viene ammesso all'uso dell'applicazione, viene incrementato il contatore e l'applicazione viene lanciata come richiesto, di solito sulla macchina locale. Quando l'applicazione viene conclusa, viene di nuovo interessato il server di licenza e il contatore degli utenti viene decrementato. Il costo dell'applicazione può dipendere, o dal numero di istanze dell'applicazione contemporaneamente in uso, o da un conteggio cumulativo del numero di utilizzazioni dell'applicazione.

CONDIVISIONE DI DISPOSITIVI PERIFERICI

Il sistema RFS, a differenza del sistema NFS, consente la condivisione di dispositivi periferici e il montaggio. Un dispositivo consiste in un file speciale nel directory **/dev**, che mette in connessione con un dispositivo hardware, come una stampante o un dischetto. Potete montare il directory **/dev** di una macchina remota in un punto di mount nel file system locale e usare i dispositivi remoti connessi a quella macchina, come se fossero dispositivi locali. Fate attenzione a non montare un directory **/dev** remoto sotto il vostro directory **/dev** locale, perché questa operazione può danneggiare il vostro sistema UNIX. Dopo il montaggio, potete normalmente aprire, leggere e scrivere sul dispositivo remoto come su un dispositivo locale; tuttavia, in SVR4 non tutti i dispositivi supportano l'accesso remoto, pertanto, dovrete verificare come ciascun dispositivo risponde sotto RFS.

CONCETTI DELL'IMPLEMENTAZIONE DELLE RETI

Il supporto delle reti è basato su tre caratteristiche recentemente aggiunte al System V: *Virtual File System* (VFS), *stream* e *socket*. Il VFS fornisce all'interno del kernel una caratteristica generale, che consente di stabilire un'equivalenza fra due qualsiasi file system diversi. In altri termini, potete considerare un file system MS-DOS o NFS come un tipo di file system e usare questo tipo di file system estraneo come usereste un normale file system; sono disponibili tutti i comandi e la ridirezione. La complessità richiesta da questo processo è nascosta all'interno del kernel e non è visibile agli utenti. A sua volta, VFS si basa su *stream*, una caratteristica generale di device driver, che consente la configurazione di canali di comunicazione a *run-time*. I *socket* sono interfacce astratte di comunicazione, basate sulle caratteristiche di stream; consentono alle applicazioni di comunicare, nascondendo tuttavia gli aspetti di comunicazioni alle stesse applicazioni.

Queste possibilità di gestione della rete a basso livello nel sistema UNIX consentono lo sviluppo di *protocol stack*, che implementano la comunicazione effettiva tramite la rete. Il più diffuso è TCP/IP (Transmission Control Protocol/Internet Protocol), usato dal sistema NFS; un altro è il protocol

stack ISO (International Standard Organization) per Open System Interconnect (OSI), usato da RFS. Lo scopo di questi diversi livelli di software è quello di separare le operazioni logicamente distinte in parti isolate che possono essere mantenute e aggiornate in modo indipendente. Lo scopo finale è quello di rendere invisibile all'utente la complessità del sistema.

Potete esaminare l'uso del sistema dei protocolli e dei socket con il comando **netstat**; il comando prevede molte opzioni e può fornire informazioni in uscita molto complesse. Per maggiori informazioni consultate la man page **netstat(1M)**.

L'interfaccia principale ai livelli inferiori dei sistemi NFS e TCP/IP è il comando **ifconfig** (interface configuration); questo comando consente di aggiungere o cancellare protocolli, di cambiare la maschera della macchina (*net mask*), di cambiare l'indirizzo di broadcast e altri parametri. Il gestore della vostra rete vi potrà spiegare come **ifconfig** viene impiegato nella vostra rete.

IL SISTEMA YELLOW PAGES

Il sistema *Yellow Pages* è un database server centralizzato, che può gestire identificatori login di utente, nomi di host, indirizzamenti, indirizzi di posta elettronica e numerosi altri tipi di dati gestionali, in un sistema UNIX. Il sistema **yp** è stato sviluppato da Sun Microsystems e viene adesso incluso in molte release di SVR4.

La decisione se usare o no **yp** è fatta dal gestore della rete; viene utilizzato molto spesso in LAN Ethernet che usano NFS. Il sistema consente al gestore della rete di mantenere una singola copia centralizzata dei dati chiave necessari per tenere in funzione un sistema UNIX; quando una macchina, o un'applicazione, necessita di questi dati, consulta le pagine gialle centrali, anziché i file locali. Questa ridirezione è trasparente alle applicazioni, che operano nella stessa maniera, sia che il sistema **yp** sia in funzione, sia che non lo sia. Il sistema Yellow Pages supporta molte caratteristiche, compresa la creazione di domini simili ai domini RFS.

Generalmente, i seguenti file locali vengono rimpiazzati dai loro equivalenti **yp**:

```
/etc/passwd
/etc/shadow
/etc/hosts          (etc/inet/hosts)
/etc/bootparams
/etc/ethers
/etc/group
/etc/networks       (etc/inet/networks)
/etc/protocols      (/etc/inet/protocols)
/etc/services       (/etc/inet/services)
/etc/netgroup
/etc/hosts.equiv
/.rhosts
```

In altri termini, la versione locale di questi file non viene usata quando **yp** è operativo, quindi, per modificare il vostro ambiente non potete modificare le versioni locali dei file; invece, voi (o il gestore della rete) dovete modificare il database centralizzato. In alcune installazioni di **yp** possono essere rimpiazzati anche altri file.

Il comando del sistema **yp** più utile per gli utenti è il comando **ypcat** (yellow pages **cat**), che visualizza il contenuto di uno o più database del sistema. Il comando

```
sis__locale$ ypcat hosts
```

elenca i nomi di host presenti nella rete e

```
sis__locale$ ypcat aliases
```

elenca gli alias di mail degli utenti della rete.

Potete elencare i database disponibili, detti anche *maps*, con l'opzione **-x**, come nell'esempio:

```
sis__locale$ ypcat -x
Use "passwd" for map "passwd.byname"
Use "group" for map "group.byname"
Use "networks" for map "networks.byaddr"
Use "hosts" for map "hosts.byaddr"
Use "protocols" for map "protocols.bynumber"
Use "services" for map "services.byname"
Use "aliases" for map "mail.aliases"
Use "ethers" for map "ethers.byname"
sis__locale$
```

Per inizializzare e lanciare i servizi **yp** per una rete vengono usati i comandi **ypinit** e **ypbind**; il comando **ypbind** è un demon che deve essere in funzione in ciascuna macchina nel dominio di **yp**.

Quando la rete è fuori servizio per un motivo qualsiasi, l'accesso al sistema **yp** è bloccato, e molte applicazioni che usano i dati non saranno disponibili (per esempio, alcuni sistemi di posta, reinizializzazione della macchina e anche login remoti). Per maggiori informazioni, se la vostra rete utilizza **yp**, consultate il gestore della vostra rete.

Capitolo 25

Configurazione del sistema

Quando acquistate un nuovo calcolatore e un pacchetto software UNIX, il primo passo è quello di caricare il software di sistema sul disco rigido della macchina e, durante questa fase, di prendere decisioni per scegliere la configurazione da attribuire alla macchina. Le versioni recenti di sistema UNIX semplificano questo processo di caricamento, perché il sistema suggerisce dei valori standard per i parametri di configurazione che l'utente può accettare, o rifiutare. Si tratta infatti di valori di compromesso, che si adattano alla maggioranza delle situazioni, ma voi potete cambiarli per scegliere una configurazione di sistema che risponda maggiormente alle vostre esigenze.

In questo capitolo analizzeremo i problemi connessi alla caratterizzazione di un sistema UNIX, per quanto riguarda sia la configurazione hardware, sia la configurazione software; parleremo soprattutto dei sistemi SVR4 installati su potenti microcalcolatori, ma gli argomenti trattati si dimostrano di grande utilità anche per le versioni UNIX supportate da altri elaboratori.

25.1 Hardware e sistema UNIX

Il sistema UNIX è stato originariamente sviluppato per i grandi minicomputer, e solo recentemente è stato portato sui microcomputer, perciò per il suo funzionamento richiede notevoli risorse hardware. Per questo motivo, l'implementazione del sistema UNIX sulle macchine più piccole non ha riscosso un grande successo.

Solo la costante diminuzione dei prezzi di hardware ha reso il sistema UNIX veramente applicabile su personal computer. La release SVR4 è un grande e complesso sistema operativo che richiede ampie risorse hardware; come prerequisito richiede una macchina con word di 32 bit.

L'80386 E ALTRI MICROPROCESSORI

Il microprocessore Intel 80386 è la CPU ideale per un sistema UNIX personale. Infatti, possiede la pura velocità di CPU necessaria a un sofisticato sistema operativo e include la presenza di un efficiente supporto per la gestione della memoria virtuale e altre caratteristiche che consentono di garantire ottime prestazioni. Anche macchine basate sul processore Motorola 68020 e microprocessori a 32 bit simili possono costituire la base di ottimi sistemi UNIX. Altre CPU con più alte prestazioni, come Intel 80486, Motorola 68030 e 68040 e molte CPU RISC possono essere usate per eccellenti sistemi multiutente. Comunque, il sistema personale col miglior rapporto costo-prestazioni è sempre la macchina della classe PC AT 80386. Macchine basate su 80386/SX possono essere utilizzate col sistema UNIX, ma di solito la velocità del sistema ne risente negativamente; se ne avete la possibilità preferite un 80386/DX con memoria cache e una velocità di CPU di almeno 20 MHz.

Sistemi di maggiori dimensioni che supportano molti utenti, o che funzionano principalmente come server, possono richiedere sistemi più veloci, con costi adeguatamente più alti.

Naturalmente, ogni particolare macchina richiede un software adatto, per cui alcune versioni del sistema UNIX destinate a macchine 80386 non possono funzionare con il 68020 o altre CPU e viceversa. Al contrario, le versioni di UNIX destinate al microprocessore 80386 possono operare anche su macchine 80486 (e viceversa) e alcuni software del sistema UNIX 80286 possono funzionare su macchine 80386 e 80486, ma le prestazioni risultano inferiori a quelle che si ottengono con l'implementazione nativa. Molte versioni del sistema SVR4 riconoscono automaticamente la differenza di architettura fra il bus ISA (industry standard architectur) e il più recente EISA (extended ISA); al contrario, MCA (micro-channel architecture) o altre configurazioni di bus possono richiedere una versione speciale del sistema SVR4. Anche molti sistemi RISC richiedono una loro propria versione di SVR4.

La scelta della macchina è complicata spesso dalla necessità di un particolare software applicativo, come pacchetti CAD, o sistemi di desktop publishing, che possono essere supportati solo da limitate piattaforme hardware. In conclusione, prima di acquistare una macchina, dovete assicurarvi che esista una versione adatta di tutto il software che voi intendete usare.

CONSIDERAZIONI SULLA MEMORIA RAM

Il sistema UNIX richiede che la macchina disponga di una quantità significativa di memoria reale, o RAM. La versione SVR4 richiede almeno 4 MB di RAM, ma è consigliabile una memoria di maggiori dimensioni, in quanto l'effettivo potenziale del sistema non appare evidente fino a quando la memoria reale non raggiunge almeno 6-8 MB. Per i sistemi multiutente o file-server molto impegnati, è indispensabile ancora più memoria, e per un sistema particolarmente sfruttato non è insolito disporre di 16 MB e oltre di memoria reale. Ovviamente, memorie di queste dimensioni non sono necessarie per sistemi personal monoutente ma, in ogni caso, la versione SVR4 richiede almeno 4 MB per funzionare.

In fase d'inizializzazione, il sistema UNIX può determinare la quantità di memoria installata sulla macchina, memoria che poi utilizzerà completamente. Accertatevi che gli interruttori (switch) siano nella posizione corretta, in modo che i diagnostici all'accensione della macchina possano determinare tutta la memoria effettivamente presente.

La memoria di UNIX non è strutturata a banchi con switch, o espansa (expanded), come quella utilizzata dal sistema MS-DOS per espandersi oltre il limite massimo di 640 kB; il sistema UNIX mantiene una gestione lineare della memoria e quindi ogni switch o scheda di espansione di memoria deve essere posizionata nella modalità non a pagine (nonpaged, extended). Se gli switch sono posizionati per allocare memoria nella modalità a pagine, il sistema UNIX non utilizza quella parte di memoria, che viene quindi sprecata. Spesso i rivenditori di hardware possono predisporre correttamente la memoria, quando acquistate la macchina, purché li informiate che la macchina utilizzerà il sistema UNIX.

CONSIDERAZIONI SUI DISCHI

Generalmente occorre almeno un drive a dischetto per caricare inizialmente il software di sistema e il dischetto è comunque utile per salvare i dati e trasferire i file facilmente da una macchina a un'altra. Esistono più formati di dischetto e tra questi i dischetti da $5^{1/4}$ pollici e quelli da $3^{1/2}$ pollici con custodia in plastica. Molte versioni di SVR4 vengono fornite in ambedue questi formati.

Non è facile copiare i dischetti di sistema UNIX da un formato a un altro, perché la procedura che inizializza il sistema UNIX da dischetto è vincolata a un particolare formato di dischetto. Per tutti i sistemi UNIX questa procedura di caricamento da dischetto è indispensabile almeno una volta, quando cioè viene installato il software di sistema.

Dovete generalmente configurare la macchina in modo che il tipo di drive a dischetto che usate per l'inizializzazione di sistema sia il drive 0 (detto drive A in PC/AT).

Il sistema UNIX richiede che la macchina disponga di almeno un disco rigido per funzionare correttamente. Infatti, il sistema UNIX viene caricato su disco rigido, con cui interagisce frequentemente nelle sue successive operazioni, mentre i dischetti vengono utilizzati principalmente per il caricamento iniziale di software e per il salvataggio dei dati. Il sistema UNIX può leggere e scrivere da dischetto seguendo le stesse modalità che adotta per il disco rigido, ma l'accesso ai dischetti risulta sensibilmente più lento rispetto all'accesso su disco rigido.

Molte release SVR4 supportano svariati tipi di dischi rigidi (almeno SCSI, ESDI, ST506, IDE, IDI), ma di solito non sono supportati tipi diversi sulla stessa macchina. Abbiate cura di scegliere il tipo di disco adatto al controller del disco e verificate che il vostro sistema UNIX supporti quel tipo di disco. È possibile installare più dischi dello stesso tipo, talvolta nel numero massimo di due.

Maggiori sono le dimensioni del disco rigido installato sulla macchina, maggiori sono i vantaggi per il sistema. L'esperienza dimostra che l'impiego dello spazio su disco tende a raggiungere la dimensione fisica del disco, per cui acquistare subito un disco rigido di maggiore dimensione risulta più economico che essere costretti a sostituire in seguito il disco rigido. Il sistema SVR4 richiede almeno 60 MB di memoria sul disco rigido per suo proprio uso (e si tratta della dimensione minima necessaria), ma è preferibile disporre di un disco di 80 MB o anche di dimensione maggiore. Con X Window System e pacchetto di sviluppo, la richiesta arriva a 140 MB e dovete inoltre aggiungere a questo lo spazio richiesto da un possibile sistema indipendente MS-DOS che condivida il disco. In macchine che svolgono molte attività sono comuni dischi di 200 MB e oltre. Il sistema UNIX può gestire tutto lo spazio disponibile su disco, senza limitazioni artificiali.

Inoltre, il sistema UNIX gestisce dischi multipli e versioni del sistema per grandi minicomputer possono avere tre o quattro dischi, ciascuno con capacità da 400 a 600 MB. Tuttavia, i piccoli personal computer possono essere limitati a una sola scheda di controller del disco, capace a sua volta di controllare solo due dischi. Consultate anche per questo il vostro rivenditore hardware per ottenere informazioni complete sulla macchina che vi interessa.

Due grandi dischi rigidi sono generalmente sufficienti per soddisfare la maggior parte delle esigenze, inoltre potete acquistare il secondo disco rigi-

do quando il primo disco si sta avviando a riempirsi e cablarlo all'unità di governo (controller) di disco rigido esistente, senza dover ricaricare il primo disco.

Le operazioni su disco rigido vengono controllate da un apposito software (device driver) che dipende dal controller di cui dispone la macchina; se il controller non è in grado di accedere a un certo tipo di disco, la scelta di dotare la macchina di un nuovo controller può rivelarsi inutile se la scheda non è accompagnata anche dal relativo driver. Il software di controllo del disco che funziona in ambiente MS-DOS, o in altri sistemi operativi, non può essere installato sul sistema UNIX. Prima di acquistare un disco rigido, consultate quindi il rivenditore di fiducia per accertare se il disco può essere utilizzato con il vostro controller.

Se avete intenzione di collegare il vostro sistema SVR4 a una rete locale, potete avere la possibilità di usare una macchina senza dischi; le macchine senza dischi ottengono tutti i loro dati da un disco remoto di un'altra macchina connessa alla rete locale, anziché da un disco rigido connesso alla macchina. Una macchina senza dischi viene inizializzata attraverso la rete, e tutti i suoi file e comandi risiedono su un disco remoto. Chiedete al vostro gestore di rete maggiori informazioni sulle macchine senza dischi; tuttavia tenete ben presente che una macchina con un disco locale raggiungerà sempre prestazioni altamente superiori a quelle di una macchina senza dischi.

LA CONSOLE E IL MONITOR

Tutti i sistemi UNIX devono avere una console di sistema. Sui sistemi di maggiori dimensioni si tratta generalmente di un terminale stampante di basso costo, collegato permanentemente alla macchina e che non viene frequentemente utilizzato. Su macchine più piccole tipo personal, invece, la console esistente svolge, oltre i compiti di console di sistema, anche quelli di terminale dell'utente. La console di sistema viene utilizzata per visualizzare i messaggi in fase d'inizializzazione e i messaggi di errore del sistema. Inoltre, SVR4 limita l'impiego dell'identificatore di login **root** alla console di sistema.

Per controllare la console si utilizza un device driver, per cui non potete probabilmente cambiare il monitor senza installare un nuovo driver di video. Per esempio, le macchine UNIX basate sul microprocessore 80386 di solito possiedono i driver per supportare gli standard VGA ed EGA. Inoltre, alcune release prevedono speciali driver grafici per monitor ad alta risoluzione. Ancora una volta, la migliore soluzione è sempre quella di consultare il fornitore di hardware per essere certi che una macchina sia compatibile con la versione del sistema UNIX che utilizzate.

PORTE E TERMINALI

I terminali remoti utilizzati per macchine UNIX sono generalmente periferiche seriali ASCII, funzionanti in modo carattere, che possono collegarsi a una porta seriale **tty**, di solito denominata COM1 o COM2 sulle macchine tipo PC/AT. Queste porte di comunicazione asincrone non possono supportare terminali grafici senza il supporto di un particolare driver. È possibile collegare alle macchine UNIX diversi tipi di terminali ASCII, se il database **terminfo** contiene la descrizione dei terminali; è anche possibile collegare modem esterni alle porte seriali, per cui gli utenti remoti possono stabilire un colloquio telefonico con il sistema UNIX; in nessuno di questi casi occorre altro software o particolari driver.

La maggior parte dei microcalcolatori dispone di una o due porte seriali; in ogni caso, la vostra macchina deve avere almeno una porta seriale, per sfruttare appieno le migliori caratteristiche del sistema UNIX.

La presenza di solo una o due porte di comunicazione asincrona può impedirvi di collegare la macchina a qualche rete locale che utilizza una porta seriale della macchina, perché per esempio una porta è occupata dalla stampante seriale e un'altra dal mouse. Dovete analizzare attentamente come prevedete di utilizzare le porte seriali prima di investire in una macchina o in una particolare versione del sistema UNIX. Gli utenti del sistema UNIX scoprono con l'esperienza che le macchine non dispongono mai di un numero sufficiente di porte seriali per consentire libertà di espansione. Potrete dotare il sistema di schede multiporta in commercio, contenente ciascuna quattro od otto porte, ma consultatevi col fornitore prima dell'acquisto per essere certi che il sistema che intendete acquistare supporti una particolare scheda multiporta.

Con SVR4 potete utilizzare anche X Window System da un terminale remoto, se si tratta di un terminale sufficientemente intelligente; questi terminali adatti per X hanno una connessione ad alta velocità col sistema, solitamente attraverso una LAN, in modo che non impegnano una porta seriale. I terminali per X comprendono un mouse, un display grafico ad alta risoluzione e un server software per X.

LIMITAZIONI DELLA LICENZA SVR4

Le versioni di SVR4 per microcalcolatori vengono vendute sotto due forme, a seconda che gli utenti del sistema UNIX siano due o più di due; alcuni rivenditori propongono anche versioni per 16 utenti. In altre parole, il software di sistema presenta delle limitazioni nella versione a due utenti, per cui non più di due utenti possono collegarsi al sistema contemporaneamente, anche se la macchina dispone di più di due porte. Un utente può essere sulla console e l'altro su una porta seriale oppure i due utenti possono utilizzare contemporaneamente la macchina tramite porte seriali, se la conso-

le non viene utilizzata. Per supportare due collegamenti remoti contemporanei, occorrono due porte seriali. La versione a due utenti è meno costosa e risulta di solito adeguata per piccoli sistemi UNIX su PC.

La versione ristretta a due utenti non limita il numero di porte disponibili per le stampanti, le reti locali, ecc., ma limita il numero di sessioni di utente che possono essere attive contemporaneamente.

La versione completa di SVR4 non pone limitazioni al numero di utenti che possono collegarsi contemporaneamente alla macchina, fino al numero fisico di porte di cui dispone la macchina.

STAMPANTI

A causa della carenza potenziale di porte seriali, la maggior parte dei microcalcolatori che utilizzano il sistema UNIX adopera stampanti parallele piuttosto che stampanti seriali. Generalmente i personal dispongono di una sola porta parallela e una seconda porta parallela può essere aggiunta tramite una scheda di espansione (LPT1 e LPT2 rispettivamente). Queste porte parallele non possono essere utilizzate per i terminali, per cui generalmente ne occorre solo una, a meno che la macchina non disponga di più di una stampante. Il sistema **lp** può supportare diversi tipi di stampanti, sia parallele che seriali, ma dovete, anche in questo caso, avere conferma che il sistema UNIX e l'hardware della macchina siano in grado di supportare i dispositivi di stampa che intendete utilizzare. Le comuni stampanti a matrice di punti o a caratteri vengono di solito supportate senza difficoltà, ma le stampanti laser sofisticate o altre periferiche meno comuni richiedono maggiori conoscenze da parte dell'utente per farle funzionare correttamente.

Le stampanti che accettano il linguaggio di descrizione pagina PostScript, e anche le diffuse stampanti HP LaserJet, sono supportate in SVR4 e usano di solito porte seriali. Se avete una di queste stampanti e dovete connetterla a una porta seriale assicuratevi di disporre di porte seriali in numero sufficiente per le vostre necessità.

RETI

Grazie ai vantaggi economici conseguenti alla condivisione di stampanti e di altre periferiche, quasi tutti i moderni sistemi UNIX sono collegati a reti locali. Molte reti utilizzano schede proprie e particolari driver, per cui non utilizzano porte parallele o seriali, che vengono così lasciate libere per altri impieghi. Inoltre, alcune reti consentono di girare le chiamate telefoniche in arrivo a uno speciale modem della rete stessa a una particolare macchina della rete. Questa soluzione può risultare notevolmente meno costosa rispetto a quella di dotare ogni macchina di porte di comunicazione e di modem. La rete locale dominante nel campo dei sistemi UNIX è la rete Ethernet supportata dalla maggior parte dei rivenditori di sistemi UNIX.

Una rete consente anche di definire come *file server* una macchina che dispone di memorie di massa di capacità maggiore, fra tutte quelle della rete, consentendo ulteriori riduzioni dei costi. Questi file server vengono adibiti al compito di memorizzare grandi database o di eseguire il salvataggio di dati dalle singole macchine, compito che sta alla base del principio delle macchine senza dischi.

NASTRI MAGNETICI

Quando un sistema UNIX diventa molto grande, e anche la capacità del disco rigido aumenta, l'uso del dischetto come mezzo di caricamento iniziale del sistema, e per il salvataggio dei file d'utente, diventa troppo difficile. In conseguenza, SVR4 viene spesso distribuito su nastro magnetico e il sistema viene caricato da quel nastro; dopo che il sistema è stato caricato, il dispositivo a nastro può essere utilizzato come periferica ad alta capacità per i salvataggi. Poiché i prezzi dei dispositivi a nastro continuano a diminuire, il nastro diventerà sempre più un'alternativa attraente dei dischetti. Tuttavia, in alcuni casi per caricare il sistema dal nastro occorre comunque inizializzare la macchina da dischetto e in questo caso il dispositivo a nastro non può rimpiazzare del tutto il dispositivo a dischetto.

Sono disponibili diversi formati di nastro e schede di controller del nastro; come al solito, dovete verificare che il vostro rivenditore di sistema UNIX supporti un determinato tipo di unità a nastro prima di acquistarla. I formati più comuni sono il formato QIC (quarter inch cartridge) e il formato ad alta densità DAT (8 mm); questi formati hanno delle sottocategorie: i più comuni sono QIC-24 (60 MB) e QIC-150 (150 MB). Inoltre, se prevedete di usare i nastri per trasferire dati tra macchine diverse, dovete anche verificare che tutte le macchine leggano e scrivano lo stesso formato.

25.2 Una configurazione minima di sistema

In base a quanto abbiamo detto, è possibile configurare un sistema UNIX su un personal indipendente con una sola porta seriale, un modem e una stampante parallela. La porta seriale e il modem consentono di effettuare e ricevere le chiamate telefoniche, in momenti diversi. Un utente remoto può chiamare mentre un altro utente è collegato alla console e contemporaneamente può essere attiva l'operazione di stampa sulla porta parallela. Tenete presente che sono richiesti almeno 4 MB di memoria e almeno 60 MB di spazio su disco rigido. In seguito potete aggiungere altre porte o schede di rete se avete accertato che la macchina e il sistema UNIX sono in grado di supportarle.

Se volete usare X Window System avrete necessità anche di un mouse e un minimo di 80 MB di spazio su disco.

25.3 Installazione fisica del sistema

Collegare fisicamente insieme le parti del sistema dipende talmente dalla macchina e dalle periferiche associate che qui possiamo appena accennare alcuni concetti. È indispensabile appoggiarsi all'installatore di hardware per assicurarsi che i cavi per i collegamenti con le periferiche siano corretti. Esistono grandi diversità nei cavi per collegare le periferiche e anche i cavi RS-232 e le periferiche "standard" possono differire in molti aspetti.

Gli esperti generalmente predispongono la macchina con la minima configurazione hardware, caricano il software e verificano che tutto funzioni al meglio, prima di collegare altre periferiche, come stampanti o terminali. Questo approccio graduale consente di familiarizzare con la macchina prima di affrontare i problemi che derivano dal collegamento di nuovi tipi di cavi o dal sottosistema **lp**. Vi consigliamo quindi di cominciare con un sistema ridotto che funzioni e poi aggiungere e mettere in funzione gradualmente altri componenti, fino a quando non raggiungete la configurazione finale.

Un aspetto da considerare è il "sesso" dei connettori sulla macchina e sui dispositivi periferici; dovete acquistare cavi compatibili con i connettori della macchina e delle periferiche, per evitare di dovere impiegare numerosi adattatori "maschio-maschio" o "femmina-femmina" per potere effettuare i collegamenti.

Inoltre, i cavi destinati al collegamento di un modem esterno a una porta seriale non sono adatti al collegamento diretto di un terminale locale, perché il modem e il terminale usano "pin" diversi per segnalare che sono in attività. Per collegare un terminale a una porta destinata a un modem può essere necessario uno speciale cavo *null modem*; un cavo di questo tipo è di solito necessario per collegare una stampante PostScript a una porta seriale. Conviene richiedere direttamente al vostro fornitore di hardware i cavi necessari per i collegamenti, per evitare fastidiosi problemi.

Molte macchine prevedono che alcuni parametri di configurazione, come la memoria RAM totale, numero e tipo di dischi, adattatori video, siano memorizzati in memorie non volatili supportate da batterie tampone. Il sistema UNIX normalmente non fa grande uso di queste informazioni, ma la macchina può non inizializzarsi correttamente se questi parametri non sono esatti; se aggiungete componenti hardware assicuratevi di aggiornare anche questa memoria.

Infine, non trascurate i fattori umani legati alla vostra posizione di lavoro al calcolatore. Posizionate il video, il drive, la tastiera, la sedia e il piano di lavoro nella posizione per voi più comoda. Questo sembra ovvio, ma è sorprendente come molte stazioni di lavoro risultino poco confortevoli. Se siete indolenziti, doloranti o avete mal di testa dopo aver utilizzato la macchina, probabilmente questo non dipende dalla difficoltà del sistema UNIX, ma solo dal fatto che la vostra sedia non ha l'altezza corretta rispetto alla tastiera.

25.4 Verifica della configurazione iniziale

Prima di caricare il software sul disco rigido, dovete verificare che la macchina funzioni correttamente. La maggior parte delle macchine viene consegnata con un *disco diagnostico* che, inserito nel drive a dischetto all'accensione, consente di effettuare una completa verifica dell'hardware. Altre macchine dispongono di sofisticati diagnostici in memorie non volatili supportate da batterie tampone, con una procedura che consente di avviare il sottosistema diagnostico.

Se la macchina entra in funzione con un prompt diagnostico su console potete essere sicuri che sia almeno in parte funzionante; tuttavia è buona norma eseguire, se non tutte, almeno una buona parte delle prove disponibili nella procedura diagnostica prima di procedere oltre. Prima di caricare ogni altro software dovreste essere convinti che la macchina funzioni almeno fino a questo livello.

Generalmente il disco diagnostico inizializza la macchina e visualizza un menu che consente di scegliere tra diverse operazioni di prova, incluse la formattazione di dischetti e la verifica delle superfici del disco rigido. Spesso a questo livello è possibile configurare la macchina, per quanto il sistema UNIX dopo aver iniziato la procedura d'inizializzazione non faccia uso del BIOS presente in molte macchine PC/AT. La Figura 25.1 illustra un esempio di menu dopo che il dischetto diagnostico ha inizializzato la macchina. Se il disco diagnostico non inizializza, oppure non potete entrare nel diagnostico in ROM, probabilmente il sistema è guasto in una maniera che non può essere ripristinata a livello software, oppure è danneggiato il disco diagnostico stesso.

Nel menu diagnostico della Figura 25.1, le opzioni 1 e 2 permettono di formattare e copiare i dischetti; potete utilizzarle per copiare i dischetti di *floppy boot disk* (il primo e il secondo disco del System Installation Set) prima di caricare il sistema UNIX per la prima volta. Questa operazione è opportuna per motivi di sicurezza perché i dischetti di boot possono venire fa-

```
SELECT AN OPTION

0 - SYSTEM CHECKOUT
1 - FORMAT DISKETTE
2 - COPY DISKETTE
3 - PREPARE SYSTEM FOR MOVING
4 - SETUP
9 - END DIAGNOSTICS

SELECT THE ACTION DESIRED
?
```

Figura 25.1 Menu di un dischetto diagnostico.

cilmente rovinati con un impiego maldestro. I dischetti d'inizializzazione in SVR4 possono comunque essere protetti contro la scrittura, mentre in SVR3 e sistemi precedenti questi dischetti devono essere accessibili anche in scrittura.

L'opzione 4 – Setup – consente di impostare la data e l'ora di sistema, ma si tratta di una funzione inutile nei sistemi UNIX, che dispongono di strumenti migliori per svolgere questi compiti, sia nelle procedure di configurazione, sia nel corso del funzionamento.

L'opzione 3 – Prepare System for Moving – viene utilizzata per parcheggiare le testine del disco rigido, per evitare che il disco rigido si danneggi quando il sistema deve essere trasferito. Selezionate sempre questa opzione ogni volta che intendete trasportare la macchina, piuttosto che correre il rischio di danneggiare il disco.

L'opzione 0 – System Checkout – consente di esaminare l'hardware installato e le posizioni degli switch utilizzati per abilitare la RAM, i dischi e altro hardware. Selezionate questa opzione per verificare quale cognizione ha il sistema della configurazione hardware a disposizione. Generalmente, le incompatibilità sono dovute a impostazioni scorrette degli switch nella scheda principale della macchina, oppure nella ROM se la vostra macchina la possiede. Se avete cambiato la configurazione hardware della macchina, anche solo aggiungendo altra RAM, dovete probabilmente aggiornare le posizioni degli switch sulla scheda di sistema per riflettere la nuova configurazione dell'hardware. Alcune macchine non usano più switch fisici per configurare l'hardware, ma possiedono una ROM controllata da software che può essere modificata con l'opzione "Setup" del menu del disco diagnostico. Seguite in ogni caso le istruzioni del fabbricante o del venditore quando dovete cambiare le posizioni degli switch.

Può capitare qualche volta che parte dell'hardware installato non sia funzionante. Se la procedura di controllo lanciata dall'opzione "System Checkout" termina regolarmente, allora la configurazione hardware principale funziona correttamente, ma questo non garantisce che la macchina sia funzionante in ogni sua parte e non verifica assolutamente l'integrità del software. Generalmente, eventuali anomalie che si verificano nella fase "System Checkout" richiedono qualche riparazione alla macchina, a meno che non si tratti solamente di cambiare le posizioni degli interruttori sulla scheda di sistema o su un'altra scheda.

Una parte molto importante della procedura diagnostica consiste nella verifica di regolarità della superficie del disco rigido (hard disk surface test). Questa procedura verifica il corretto funzionamento di tutte le parti del disco, scrivendo dati su ogni blocco di disco e poi rileggendolo. Se una di queste operazioni origina un errore significa che esiste un blocco difettoso sul disco; l'indirizzo su disco di quest'area difettosa viene incluso nella tabella delle tracce difettose presenti sul disco (bad track table). Un piccolo numero di tracce difettose è comunque normale in un disco nuovo; questi

blocchi difettosi non verranno utilizzati quando la macchina è in funzione. La verifica di regolarità della superficie del disco può durare circa un'ora, ma è raccomandabile ogni volta che caricate il software di sistema, perché la presenza non segnalata di blocchi difettosi può portare il sistema UNIX a seri malfunzionamenti. Potete evitare molti problemi, se curate di mantenere aggiornata la tabella dei blocchi difettosi presenti su disco, eseguendo la verifica di regolarità della superficie del disco ogni volta che ricaricate il software di sistema. Tenete comunque ben presente che la verifica delle superfici distrugge il contenuto di tutte le partizioni, quindi non deve essere usata dopo che il sistema è stato già caricato sul disco.

Quando avete finito con le operazioni del disco diagnostico, selezionate l'opzione 9, oppure spegnete la macchina. Rimuovete il disco diagnostico; poi reinizializzate la macchina dal sistema UNIX, se è già installato, oppure dal dischetto inizializzante (boot disk), se state appena iniziando l'installazione del software.

25.5 Partizioni del disco rigido

Una volta che l'hardware è configurato e collegato correttamente, dovete caricare il sistema UNIX sul disco rigido della macchina. La procedura di caricamento del software di sistema si compone di diverse fasi e richiede delle decisioni che dipendono dalla configurazione del disco rigido.

Un disco rigido può essere suddiviso in diverse parti separate, o partizioni, che diversi sistemi operativi (come UNIX e MS-DOS) possono utilizzare. Generalmente le diverse partizioni non sono "visibili" tra loro, cioè, quando un sistema operativo utilizza una parte del disco, non può accedere alle altre partizioni. Il comando **fdisk** consente di cambiare la partizione attiva, in modo che dopo un'operazione d'inizializzazione viene utilizzata la partizione selezionata ed è possibile caricare un altro sistema operativo. Le partizioni vengono prestabilite in fase di configurazione della macchina e non possono essere cambiate senza cancellare il disco rigido. Di conseguenza, dovete decidere in quante partizioni suddividere il disco rigido prima di caricare il sistema UNIX.

La maggior parte delle versioni SVR4 per macchine 80×86 supporta partizioni multiple, in modo che una partizione può essere utilizzata per il sistema UNIX e un'altra per il sistema operativo MS-DOS oppure OS/2. Quindi, se vi interessa utilizzare un altro sistema operativo oltre a UNIX occorre definire una seconda partizione. Lo spazio su disco che viene impiegato per un'altra partizione non è disponibile per il sistema UNIX, per cui dovete accertarvi che sia disponibile abbastanza spazio sul disco rigido per definire altre partizioni. Generalmente per una partizione MS-DOS vengono riservati da 5 a 10 MB del disco; di più, se avete grandi esigenze di memoria in ambiente MS-DOS, o se utilizzate OS/2. Se non utilizzate un sistema operativo diverso da UNIX, non occorre definire una seconda partizione.

Riservare una seconda partizione dedicata a MS-DOS o a OS/2 non è necessario per *Merge*, che consente di lanciare una sessione MS-DOS in ambiente UNIX. In sostanza, con il software *Merge* i sistemi UNIX e MS-DOS condividono lo stesso file system, per cui non è richiesta un'ulteriore partizione. Dovete riservare una partizione separata solo se avete in previsione di eseguire MS-DOS od OS/2 come sistema indipendente.

Se avete più di un disco rigido, la partizione DOS eventualmente esistente, e la partizione di base del sistema UNIX, devono trovarsi ambedue sul primo disco rigido.

25.6 Aree di swap e dump

Un'altra considerazione che interessa la configurazione del disco rigido riguarda l'area di swap. Quando i processi riempiono completamente la RAM disponibile, il sistema UNIX automaticamente scarica alcuni processi inattivi su disco, allo scopo di liberare memoria a vantaggio dei processi attivi. Questa operazione di scaricamento viene effettuata senza alcun intervento da parte dell'utente; lo spazio su disco utilizzato per lo swap non fa parte del file system, tuttavia si trova nella partizione utilizzata dal sistema UNIX, ma consiste in un'area di disco separata da un punto di vista logico. La dimensione ottimale dell'area di swap dipende dalla quantità di memoria installata, oltre che dal grado di utilizzo della macchina; maggiore è la disponibilità di RAM, minore risulta l'area di swap necessaria; se la macchina è intensamente impegnata ed esistono molti processi attivi contemporaneamente, occorre un'area di swap maggiore. L'area di swap non è disponibile per i file, anche se non viene impiegata in queste operazioni di scaricamento di processi. Dal momento che il sistema UNIX cade brutalmente quando l'area di swap si riempie, con un messaggio di errore di *panic*, conviene sempre eccedere nelle dimensioni dell'area di scambio, anziché sbagliare dedicando poco spazio. La maggior parte dei sistemi SVR4 assegna di default all'area di swap una dimensione del doppio della memoria reale RAM, il che risulta un buon compromesso per soddisfare molte esigenze. Se avete un server sovraccaricato, con una RAM insufficiente, è utile avere un'area di swap più grande, ma si tratta di una situazione inconsueta. Se la macchina è una stazione di lavoro personale con pochi processi simultanei, e disponete di una memoria RAM eccezionalmente ampia, ma avete carenza di spazio su disco, allora potete rimediare riducendo l'area di swap. SVR4 consente anche di riservare un'area di *dump*, usata per memorizzare una copia del sistema operativo in caso di caduta (crash). Alla pari dell'area di swap, anche questo spazio viene preso dallo spazio disponibile per i file nella partizione di sistema UNIX; al contrario dello spazio per swap, lo spazio per dump non è comunque indispensabile al funzionamento del sistema. A meno che non siate coinvolti in lavoro di sviluppo del kernel o dei device driver, lo spazio per dump non è necessario.

25.7 File system

È possibile suddividere lo spazio disponibile per i file in più file system separati. In realtà il concetto di file system multipli interessa praticamente solo le macchine che dispongono di due o più dischi rigidi; generalmente, è presente un solo file system su ogni disco, ciò consente al sistema UNIX di gestire efficientemente l'accesso ai diversi dischi fisici. Su macchine che possiedono un solo disco, conviene definire un ridotto numero di file system; i sistemi SVR4 che hanno un solo disco rigido funzionano bene con solo due file system: uno per l'inizializzazione della macchina (**/stand**) e l'altro per il resto del sistema (il file system /).

Con un limitato numero di file system risulta più semplice gestire lo spazio disponibile su disco. Un file o directory non può essere suddiviso tra due file system, ma deve essere totalmente contenuto entro un unico file system; avendo troppi file system è irritante terminare lo spazio in uno di essi e avere una quantità di spazio inutilizzato negli altri. Gestire tanti piccoli file system è anche fonte di errori.

Esistono comunque alcuni vantaggi legati alla creazione di file system separati: un'eventuale totale alterazione di uno di essi non influenza gli altri e se un file system si riempie totalmente non ruba spazio agli altri. Se volete garantirvi maggiore isolamento di utenti o di file e trascurate l'utilizzazione globale dello spazio su disco, potete avere convenienza nell'utilizzare diversi file system. I collegamenti simbolici vi consentono di far sì che un file fisicamente presente in un file system sembri appartenere anche a una diversa gerarchia di directory.

La fase di configurazione del sistema può di default creare diversi file system su un unico disco rigido, specialmente se di grandi dimensioni; in questo caso, dovrete intervenire con opportune azioni nel processo di configurazione, se vi interessa definire un numero minore di file system. Se avete dischi rigidi multipli è preferibile creare un unico file system su ogni disco; questo è il minimo possibile, perché un file system non può estendersi su diversi dischi.

TIPI DI FILE SYSTEM

In SVR4 potete selezionare anche il tipo di file system da installare. Sono supportati due tipi: il tipo **s5** è lo standard file system System V, ereditato da SVR2 e SVR3; l'altro tipo **ufs** è l'adattamento dal sistema BSD. Alcune applicazioni software richiedono l'uno o l'altro tipo; fatevi indicare dal venditore dell'applicazione il tipo richiesto con sicurezza. Se la scelta è possibile, il tipo **ufs** è preferibile; fornisce prestazioni di velocità migliori e consente nomi di file di lunghezza fino a 256 caratteri, eliminando il limite di 14 caratteri che ancora esiste nel tipo **s5**.

25.8 Pacchetti software SVR4

La release SVR4 viene spesso distribuita suddivisa in diversi pacchetti. Se conoscete l'uso previsto della macchina, potete selezionare un sottoinsieme dei pacchetti disponibili e caricare solo quelli indispensabili. La Tabella 25.1 mostra i principali pacchetti indipendenti in SVR4, tuttavia alcune versioni possono combinare i pacchetti elencati in gruppi più grandi. In genere, se selezionate un pacchetto della Tabella 25.1 dovrete installare tutte le parti che lo compongono, altrimenti il sistema potrebbe non funzionare. D'altra parte, se installate un ridotto numero di pacchetti impegnate un minore spazio su disco e avete meno processi demon inutili residenti nel sistema. Un sistema inizializza più rapidamente e fornisce migliori prestazioni di velocità se ha un ridotto numero di pacchetti installati.

Nell'ordine di installazione dei pacchetti occorre rispettare delle dipendenze predefinite; seguite con cura la documentazione del vostro sistema per evitare un ordine di installazione scorretto. Una sequenza di caricamento scorretta, nel caso migliore causerà un abort del processo con una segnalazione d'errore; nel caso peggiore un improvviso malfunzionamento del sistema durante l'uso.

25.9 Installazione del software di sistema

Con queste premesse, potete configurare la macchina secondo le vostre esigenze quando caricate il sistema UNIX da dischetto. Esistono alcune differenze tra le procedure di caricamento delle versioni di sistema UNIX, in SVR4 e anche in altre versioni, per cui non tutto quello che indicheremo negli esempi che seguono troverà un esatto riscontro nel vostro sistema.

Innanzitutto inizializzate la macchina utilizzando il primo dischetto del *Foundation Set*. La procedura d'installazione richiede le informazioni di configurazione del disco, formatta il disco rigido correttamente e poi carica un kernel da dischetto. A questo punto dovete estrarre il dischetto e reinizializzare con il disco rigido. Il processo d'installazione continuerà richiedendo altri parametri della configurazione hardware della macchina, quindi caricherà la parte restante del sistema UNIX da altri dischetti o da nastro magnetico. Segue quindi una fase di inizializzazione del sistema UNIX, che consente di definire la data e l'ora, installare le password per i login di sistema e aggiungere gli identificatori di login per gli utenti del sistema. Dopo di questo, dovete inizializzare ancora la macchina. A questo punto, potete entrare nel sistema con login e caricare altri pacchetti software addizionali usando i normali strumenti di gestione del sistema. Completate queste operazioni, potete installare il software Merge e il sistema operativo MS-DOS, se lo utilizzate. Se avete riservato sul disco una partizione MS-DOS, potete reinizializzare il sistema con un dischetto MS-DOS, accedere alla

Tabella 25.1 Pacchetti software per sistemi UNIX SVR4.

Nome	Sottopacchetti	Commenti
Foundation	Base	Necessario; include **uucp**,disk drive, ecc.
	Editing	**vi**
	terminfo	Descrizione generale terminali
	termcap	Retro-compatibilità; spesso non necessario
	lp	Necessario per la stampa
System administration		
	FMLI	Form interpreter; necessario per OA&M
	OA&M	Strumenti di gestione; occupano molto spazio
Networking		
	nsu	Utility di supporto; necessari per le comunicazioni
	dfs	File system distribuito; raccomandato con **nsu**
	inet	TCP/IP, utility Internet
	Ethernet driver	Dipende da hardware; necessario per Ethernet
	StarLan driver	Necessario per StarLan
	RPC	Remote Procedure Calls
	NFS	Network File System
	RFS	Remote File Sharing
Security		
	Crypt	Crittografia di file; disponibile solo in U.S.A.
	Users	Licenza per più di due utenti
	Enhanced security	Caratteristiche addizionali di sicurezza
X Windows		
	Mouse driver	Necessario per X
	geus	Sistema utenti X; necessario
Development		
	scde	Compilatore C ANSI, make, SCCS
	gpp	Strumenti di sviluppo X
	Kernel debugger	Non necessario
Documentation		
	man	Manuale On-line; può non essere disponibile
	Help	Può non essere disponibile
Document preparation		
	ditroff	Sistema base **troff**
	pic, grab	Funzioni estese
	wwb	Writer's Workbench; può non essere disponibile

Tabella 25.1 Pacchetti software per sistemi UNIX SVR4 (*continua*)

Nome	Sottopacchetti	Commenti
MS-DOS		
	Simultask o Merge	Funzioni e pacchetti possono variare
DOS server	Necessario per supportare macchine DOS in LAN	
Compatibility		
	BSD	Necessario per usare comandi BSD
	XENIX	Necessario per caricare applicazioni XENIX
Miscellaneous		
	FACE	Menu d'ufficio per terminali ASCII
	DMD windowing	Solo se avete il terminale che lo supporta
	Cartridge tape driver	Necessario per nastri a cartuccia
	Floppy tape	Necessario hardware nastro floppy

partizione dedicata a MS-DOS e caricare il sistema operativo MS-DOS per un ambiente di lavoro indipendente MS-DOS. L'intera procedura di caricamento può richiedere diverse ore, specialmente se eseguite all'inizio anche la verifica di regolarità della superficie del disco.

Il processo d'installazione inizia inserendo il primo dischetto nel drive e accendendo la macchina. Il disco corretto viene denominato *System Installation Disk* o *Floppy Boot Disk* e può essere il primo disco di *Base System Package*. Nella maggior parte delle versioni di UNIX, il Floppy Boot Disk deve essere accessibile in scrittura, ma in SVR4 *deve* essere protetto in scrittura. Invece dell'originale, utilizzate sempre una copia del disco di boot, che potete creare tramite la relativa procedura che si trova sul disco diagnostico fornito insieme alla macchina. Alcune release possono richiedere di caricare un secondo dischetto dopo avere trattato il primo.

Una versione speciale d'installazione del sistema UNIX inizializza dal dischetto, con una procedura che colloquia interattivamente con l'utente. Il dischetto non va rimosso fino a che non viene esplicitamente richiesto.

INSTALLAZIONE COMPLETA O SOVRAPPOSTA

Alcune release consentono la sovrapposizione (*overlay*) della nuova installazione sopra un sistema UNIX esistente. In questo caso, una delle prime domande dopo l'inizializzazione della macchina sarà:

```
Do you wish to overlay your existing UNIX System (y or n)?
```

Se sul vostro disco esiste già un sistema, potete rispondere **y** per accettare la sovrapposizione; questa scelta sovrascrive il vostro software di sistema senza alterare i file d'utente esistenti. In una installazione sovrapposta le scelte di opzioni sono ridotte, e non potete modificare la predisposizione del disco o delle partizioni senza eseguire un'installazione complete. Inoltre, le release SVR3 e precedenti non sono compatibili con la sovrapposizione di un nuovo sistema SVR4. La possibilità di sovrapposizione è progettata esclusivamente per l'aggiornamento di una versione SVR4 con una nuova versione SVR4. Oltretutto, l'integrità del sistema e le prestazioni complessive risulteranno migliori se l'installazione sovrapposta non viene utilizzata, anche se permessa.

SELEZIONE DI HARDWARE

Molte versioni iniziano richiedendovi immediatamente la configurazione hardware della vostra macchina e i pacchetti software che volete installare. In genere le predisposizioni hardware per i tipi di macchina fondamentali (EISA, MCA, ecc.) e per il tipo di video (VGA, ecc.) sono inclusi a questo punto; alcune release riconoscono automaticamente la configurazione hardware. Queste procedure di riconoscimento automatico non sempre funzionano correttamente; per esempio, un video super VGA (800×600) può essere riconosciuto come un normale VGA (640×480). Se ne avete la possibilità predisponete sempre la configurazione hardware manualmente. Di solito gli indirizzi di memoria e d'interruzione per le schede addizionali, tipo controllori di rete, sono selezionati quando installate il relativo pacchetto di software, invece che nella procedura d'installazione generale. In ogni caso, dovete sempre rispondere correttamente alle domande relative alle caratteristiche hardware, altrimenti la vostra macchina non funzionerà correttamente. In caso di dubbi, interrompete la procedura d'installazione e verificate il tipo di scheda e gli switch di predisposizione prima di procedere oltre.

SELEZIONE DEI PACCHETTI SOFTWARE

A questo punto, alcune release vi chiederanno di selezionare parti dell'intera release da installare; altre release, specie quelle distribuite su dischetti, possono omettere queste domande. In quest'ultimo caso dovrete installare ogni pacchetto separatamente dopo che il sistema di base è stato messo in funzione. Nel primo caso, invece, vedrete una domanda del tipo:

```
Do you wish to install any of these packages (y or n)?
```

Consultate la Tabella 25.1 per prendere le vostre decisioni, quindi scegliete l'adatto sottogruppo di pacchetti. Il produttore del vostro software potrebbe avere raggruppato i pacchetti della release in modo leggermente differente dalla tabella.

LA PROCEDURA D'INSTALLAZIONE fdisk

Se avete scelto un'installazione completa invece di una sovrapposizione, le domande vi guideranno a una sequenza di configurazione del disco. In alcune release, le prime informazioni che appaiono sul video provengono dal comando **fdisk** (fixed disk), che consente di configurare manualmente le partizioni sul disco rigido; in altre versioni, appare una serie di messaggi che chiedono all'utente informazioni per la creazione delle partizioni, quindi le procedure di caricamento eseguono automaticamente il comando **fdisk** per predisporre le partizioni.

Prima di continuare, dovete stabilire come utilizzare lo spazio disponibile sul disco rigido della macchina. Se intendete riservare una partizione a MS-DOS od OS/2, dovete decidere quanto spazio assegnare a ogni sistema operativo. Di solito si assegnano solo pochi megabyte alla partizione MS-DOS, specialmente se la macchina dispone del software Merge che consente di condividere il file system tra i sistemi UNIX e MS-DOS. Potete facilmente pensare alla suddivisione del disco nelle differenti partizioni in termine di percentuali. Per esempio, se avete un disco rigido da 100 MB, potete riservare il 10% a uso esclusivo di MS-DOS e il resto per il sistema UNIX. In questo caso, si ottiene una partizione MS-DOS indipendente di 10 MB, che soddisfa la maggior parte delle esigenze.

Se la vostra installazione inizia con il programma **fdisk**, appaiono sul video alcune informazioni di copyright, seguite dal messaggio

```
WARNING: A new installation of the UNIX System will destroy all files
currently on the system.
Do you wish to continue (y or n)?
```

A questo punto premete **y** per continuare, oppure reinizializzate immediatamente se non volete ricaricare il vostro disco rigido. Adesso, il programma **fdisk** parte, e l'immagine sul video assomiglia a quella di Figura 25.2. L'esempio nella figura mostra la presenza sul disco di due partizioni già definite, ma quando caricate per la prima volta il sistema, la tabella delle partizioni risulta probabilmente vuota. Nel nostro esempio, abbiamo utilizzato un disco da 200 MB; il numero di cilindri dipende dalla dimensione del disco.

Scegliendo un'opzione del menu, potete modificare la configurazione del disco, altrimenti scegliete l'opzione 5 per uscire senza apportare alcuna modifica. Effettuata la scelta, indicate il numero di selezione seguito dal RETURN.

```
                    Total hard disk size is 815 cylinders
                                                  Cylinders
Partition   Status     Type      Start   End    Length   %
=========   ========   =======   ======  =====  ========  ===
    1                  DOS          2    111    110      13
    2       Active     UNIX       112    814    703      86

SELECT ONE OF THE FOLLOWING:
     1. Create a partition
     2. Change Active (Boot from) partition
     3. Delete a partition
     4. Exit (Update disk configuration and exit)
     5. Cancel (Exit without updating disk configuration)
Enter Selection:
```

Figura 25.2 Informazioni di **fdisk** per la suddivisione del disco.

Per creare una nuova partizione, selezionate l'opzione 1. Il menu **fdisk** chiede di indicare il numero di cilindro di inizio e di fine della partizione. Riferitevi al numero totale di cilindri che appare in cima allo schermo e alla percentuale di disco che avete deciso di assegnare alla partizione per inserire da tastiera l'effettivo numero di cilindri. La prima partizione deve sempre iniziare dal cilindro 1 o 2 (mai dal cilindro 0) e l'ultima deve terminare in corrispondenza del penultimo cilindro definito sul disco rigido. Ogni partizione, a esclusione della prima, deve iniziare dal cilindro successivo all'ultimo cilindro riservato alla precedente partizione. Quando create le partizioni, riservate la prima partizione al sistema MS-DOS od OS/2 e l'ultima al sistema UNIX, come indica la Figura 25.2.

Se commettete un errore nel definire le partizioni, potete cancellarne alcune o tutte mediante l'opzione 3 e ripetere l'operazione. Accertatevi a questo punto che le partizioni siano corrette, poiché una volta che sono state assegnate, non potete cambiarle senza perdere il contenuto del disco rigido.

Per *partizione attiva* si intende la partizione che inizializza quando accendete la macchina, cioè la partizione che attiva l'ambiente iniziale di sistema (UNIX oppure MS-DOS); se volete inizializzare la macchina in sistema UNIX deve essere attiva la partizione UNIX; se volete inizializzare in sistema MS-DOS deve essere attiva la partizione MS-DOS. Per caricare il sistema UNIX e successivamente utilizzarlo, accertatevi di stabilire attiva a questo punto la partizione UNIX. Per scegliere la partizione del disco da rendere attiva, utilizzate l'opzione 2 del menu. Quando è attiva la partizione MS-DOS, per poter utilizzare il sistema UNIX dovete eseguire la versione MS-DOS del comando **fdisk**, per assegnare la partizione attiva al sistema UNIX e quindi reinizializzare.

Una volta assegnata con il comando **fdisk** una partizione al sistema MS-DOS indipendente, la partizione MS-DOS non è formattata; per utilizzarla, dovete attivare il sistema MS-DOS e seguire le solite procedure per inizializzare il disco rigido. La partizione destinata al sistema UNIX viene invece formattata come parte della procedura di caricamento, come descritto oltre.

PROCEDURA D'INSTALLAZIONE INTERATTIVA

In alcune versioni di sistema UNIX, la procedura di caricamento colloquia con l'utente e gli chiede come intenda configurare il sistema, mentre il comando **fdisk** esegue automaticamente. In queste versioni i messaggi che appaiono su video, subito dopo aver caricato il Floppy Boot Disk, sono analoghi a quelli indicati in Figura 25.3, e le uniche differenze sostanziali possono riguardare le informazioni relative al copyright. Dopo la fase di caricamento del dischetto, il sistema si arresta visualizzando il messaggio "Do

```
.............
boot: /unix

UNIX System V/386 Release 4.0
Node intel
total real memory = 4194304
total available memory = 3289088

Copyright (c) 1984, 1986, 1987, 1988, 1989, 1990 AT&T
Copyright (c) 1987, 1988 Microsoft Corp.
All Rights Reserved

386/ix Drivers Copyright (c) 1986 Interactive Systems Corp.
All Rights Reserved

About to install UNIX on your hard disk. This process will destroy
any and all data currently on that disk, INCLUDING DOS FILES!
If you have DOS files you would like to back up before proceeding
with system installation, enter n<RETURN> to the following question.

Do you wish to proceed with UNIX installation (y/n)?
y

Do you wish to (re)format your hard disk (y/n)?
y

Do you wish to do a complete surface analysis? Answer 'y' to write
and read every sector, 'n' to just read every sector. NOTE: writing
the whole disk takes a while... (y/n)?
y
```

Figura 25.3 Dialogo per la configurazione del sistema (Parte I).

you wish to proceed with UNIX installation?"; introducete la lettera "y" e premete il tasto di ritorno a capo. Il sistema risponde con la frase "Do you wish to (re)format your hard disk?"; la risposta "y" garantisce le migliori condizioni del disco per il caricamento. Il sistema poi visualizza il messaggio "Do you wish to do a complete surface analysis?"; è consigliabile la risposta "y" che lancia la procedura di verifica delle superfici e garantisce l'inclusione dei blocchi difettosi nell'elenco. In effetti, questa verifica non viene eseguita subito, ma in una fase successiva del processo di caricamento.

La Figura 25.4 mostra come procede il colloquio tra il sistema e l'utente.

```
Welcome to the Disksetup program. This program will allow you to
specify certain parameters about your
hard disk, as well as partitioning
the disk into Unix volumes and a DOS area (if desired).

This system uses 17 512-byte sectors per track.

Disk parameters currently configured are as follows:
Number of heads: 8. Number of cylinders: 940.
Is this correct (y/n)? y

Your disk drive will have 1 cylinder used for alternate sectors.

This leaves 939 cylinders (65384448 bytes) available, 69632 per
cylinder.

Do you wish to allocate any of your disk
to exclusive use by DOS (y/n)? y

Enter number of cylinders for DOS (0-577): 50

FDISK will be run to create the following disk allocation:
DOS starting at cylinder 1 for 50 cylinders, and
UNIX starting at cylinder 51 for 889 cylinders.

Is this the partitioning you want (y/n)?

You will now have the opportunity to enter known bad areas on
your disk. To do this, you will need to know the cylinder, head,
and byte offset from the index mark of each defect. This data
can usually be found on a label on the disk drive or on a defect
list which is shipped with the drive. If you have no such list,
only those sectors found to be bad during surface analysis will
be marked.

Do you wish to enter any more defect information (y/n)? n
Are you sure you've entered all defects (y/n)? y
```

Figura 25.4 Dialogo per la configurazione del sistema (Parte II).

Il programma *Disksetup* alloca le diverse partizioni e sottopartizioni del disco. Prima di tutto ricava direttamente dal disco i parametri relativi al numero e alla dimensione dei settori del disco, visualizzando il messaggio "This system uses....". Questo esempio si riferisce a un disco rigido da 67 MB; i valori numerici possono differire se la macchina dispone di un altro tipo di disco rigido, ma potete fidarvi delle informazioni che il programma visualizza. In ogni caso, se siete sicuri che i parametri riportati non siano corretti, potete rispondere negativamente con "n" al messaggio "Is this correct (y/n)?". Questa situazione si può creare a seguito di incompatibilità esistenti tra il formato a basso livello del disco rigido e il controller del disco, per cui spesso occorre riprogrammare la scheda del controller. Nella maggior parte dei casi, comunque, i valori numerici visualizzati sono corretti, per cui rispondete in modo affermativo con "y" alla domanda.

Il messaggio successivo consente la possibilità di riservare una parte del disco a uso esclusivo del sistema MS-DOS. Se rispondete in modo affermativo con "y" alla richiesta della procedura di caricamento, il sistema vi chiederà di specificare la quantità di spazio che intendete riservare, in termini di numero di cilindri, come indica l'esempio di Figura 25.4. Ricavate questo valore numerico dalla percentuale di disco che avete assegnato al sistema MS-DOS, dividendo la dimensione totale del disco (in cilindri) per la percentuale scelta. In questo caso, abbiamo riservato solo 50 cilindri (3.48 MB), in quanto la macchina viene utilizzata solo per semplici applicazioni MS-DOS indipendenti (tipo la copiatura di dischi o l'esecuzione di giochi). Il sistema restituisce i risultati delle sue elaborazioni e vi chiede di accettarli o meno con il messaggio "Is this the partitioning you want?". Se rispondete negativamente, la procedura di caricamento riparte dall'inizio.

La parte seguente della procedura consente di specificare manualmente i blocchi che, pur essendo difettosi, non sono stati ritenuti tali dalla procedura di verifica delle superfici del disco. L'elenco di questi blocchi si trova all'interno della macchina, su una etichetta attaccata direttamente al disco rigido ed è il risultato di una rigorosa verifica del disco rigido condotta dalla casa costruttrice. Molti dei blocchi elencati risultano difettosi solo marginalmente, per cui molti utenti lasciano alla procedura di controllo delle superfici del disco il compito di trovare i blocchi che risultano "realmente" difettosi. Alcune procedure d'installazione leggono la tabella originale dei blocchi difettosi direttamente dal disco, per cui non occorre specificarla manualmente. Altri tipi di dischi rigidi, come SCSI e IDE, usano uno schema logico di numerazione dei blocchi, in modo che il sistema non vede mai i blocchi difettosi esistenti; in questo caso potete omettere l'introduzione della tabella dei blocchi difettosi. Per non correre rischi, potete fornire manualmente sia l'elenco dei blocchi difettosi apposto sul disco, sia l'elenco fornito dalla procedura di controllo delle superfici. Nel nostro esempio in Figura 25.4, abbiamo evitato di specificare i blocchi difettosi, rispondendo negativamente con "n" al relativo messaggio; dopo di questo, un ultimo

messaggio chiede una conferma e potete rispondere con "y" se avete effettivamente concluso la fase di inserimento dei blocchi difettosi.

SUDDIVISIONE DEL FILE SYSTEM

In entrambi i tipi di configurazione, manuale e interattiva, la procedura d'installazione continua con la suddivisione del file system UNIX, come mostrato in Figura 25.5. Per ciascun file system che non sia del formato fisso *Boot* e *Swap* dovete scegliere un tipo di file system; il tipo **ufs** è preferibile ed è consigliato per tutti i file system che definite, salvo nel caso in cui alcune applicazioni richiedano di utilizzare il tipo **s5**. Introducete il tipo prescelto dopo il messaggio che inizia "Please press ENTER....".

Successivamente potete specificare file system addizionali, se volete; il messaggio di richiesta è tipo questo:

```
Do you wish to create any optional
disk slices or filesystems (y or n)?
```

Se volete aggiungere altri file system introducete "y" e il sistema proporrà una lista di possibili scelte. In un sistema personale raramente richiederete più di un file system sul disco rigido principale. Se a questo punto rispondete "n" tutto lo spazio disponibile sul disco viene assegnato al file system root. Quando avete terminato di usare la lista di file system disponibili, il sistema sottopone una richiesta di verifica, come segue:

```
The hard disk layout you have selected is:

Drive    Name              Type    File System/Slice
--------  ---------        -------  ----------------------------
0        Boot File System  bfs     /stand
0        Swap Slice        -       /dev/swap
0        Root File System  ufs     /

Is this correct (y or n)?
```

```
The following hard disk elements are required and
must reside on your primary (disk 0) hard disk:
Drive    Name              Type     File System/slice
0        Boot File System  bfs      /stand
0        Swap Slice                 /dev/swap
0        Root File System  s5, ufs  /

Please select the File System Type for / (Root File System      ) from
the following list:
            s5, ufs
Please press ENTER for the default type, s5
```

Figura 25.5 Suddivisione di file system.

Se volete cambiare le scelte precedenti, rispondete "n" e l'intero processo verrà ripetuto dall'inizio, altrimenti rispondete "y". Il sistema eseguirà adesso la verifica delle superfici del disco rigido, per esempio:

```
Checking for bad sectors in the UNIX System partition...

Checking cylinder: xxx

The following slice sizes are recommended for your disk.
A / filesystem of 616 cylinders (153 MB)
A /dev/swap slice of 65 cylinders (16 MB)
A /stand filesystem of 21 cylinders (5 MB)

Is this configuration acceptable? (y/n)
```

Potete verificare le dimensioni dei vari file system; di solito i default sono accettabili. Occorrono circa 5 MB per il file system **/stand** e lo spazio di swap deve essere predisposto al doppio della memoria RAM reale presente nella macchina. Altri file system, se esistono, possono avere qualunque dimensione a vostra scelta, tuttavia i file system **root** (/) e **/usr** possono richiedere una dimensione minima. Di solito i default sono la scelta migliore, ma se predisponete diversi file system su un piccolo disco (meno di 100 MB) i default possono non essere sufficienti a contenere tutti i file che il sistema dovrà installare.

Dopo il completamento della verifica delle superfici, il sistema crea realmente i file system, come in questo esempio:

```
Filesystems will now be created on the needed slices
Creating the / filesystem on /dev/rdsk/c0t0d0s1
Creating the /stand filesystem on /dev/rdsk/c0t0d0sa

A UNIX System will now be installed on your hard disk
Please standby.

When you are prompted to reboot your system,
remove the floppy disk from the diskette drive,
and strike CTRL-ALT-DEL.
Please wait for the prompt.

Reboot the system now.
```

Quando reinizializzate, il disco viene configurato.

CARICAMENTO DEL FOUNDATION SET

Il sistema quando viene reinizializzato, esegue una speciale versione d'installazione del sistema UNIX; il primo messaggio che appare sul video sarà simile a questo:

```
Please indicate the installation medium you intend to use.

Strike "C" to install from Cartridge TAPE
or "F" to install from FLOPPY DISKETTE.

Strike ESC to stop.
```

Selezionate la scelta adeguata per il vostro software e hardware; verrete richiesti di caricare il supporto adatto e di battere **go** o ENTER quando siete pronti. Assicuratevi di usare il corretto gruppo di dischetti o nastri del Foundation Set.

Se utilizzate i dischetti iniziate col primo disco in sequenza dopo i Boot Disk; quindi caricate in macchina uno per uno i rimanenti dischetti del software di sistema. Il sistema dopo aver terminato la lettura di un disco richiede di introdurre il successivo; non dovete rimuovere un dischetto prima del messaggio che lo richiede. Non interrompete il procedimento di caricamento in nessun modo, o dovrete ricominciare l'installazione dall'inizio. Se utilizzate il nastro magnetico il processo leggerà l'intero Foundation Set dal nastro.

Se avete già introdotto l'elenco dei pacchetti da installare, come descritto, le scelte fatte avranno il loro effetto; altrimenti il sistema può richiedere a questo punto la selezione dei pacchetti. Quindi viene letto il supporto e riceverete delle richieste per ogni questione relativa ai pacchetti da caricare. Vi possono essere richiesti gli indirizzi Ethernet, restrizioni o autorizzazioni di sicurezza e altre informazioni specifiche per i pacchetti; se non avete le risposte per le domande, accettate i valori di default. Di solito potete ancora modificare queste vostre scelte mediante strumenti di gestione del sistema, con la macchina normalmente in funzione. Se caricate il sistema con i dischetti, può darsi che dobbiate entrare nel sistema dopo averlo messo in funzione e caricare ciascun pacchetto dai dischetti come descritto in seguito in questo capitolo. Quando tutti i dischi, o l'intero nastro, del Foundation Set sono stati letti, il sistema UNIX entra in funzione.

LE PROCEDURE DI PERSONALIZZAZIONE

Una volta caricato il sistema sul disco rigido, vi trovate in uno speciale ambiente di personalizzazione che consente di eseguire gli iniziali compiti di gestione necessari perché la macchina funzioni correttamente. Si tratta di chiare procedure interattive, che tuttavia possono differire significativamente tra le diverse versioni del sistema. In alcune versioni, occorre entrare nel sistema utilizzando lo speciale identificatore di login **setup**, mentre, in altre versioni, le procedure di personalizzazione sono automatiche, una volta caricato il Foundation Set. Completate la personalizzazione del sistema in questa fase interattiva, per evitare di doverla riprendere in seguito con procedure manuali.

La prima parte della fase di personalizzazione consente di definire la data e l'ora nell'orologio interno. Le attività dei sistemi UNIX basano il loro corretto funzionamento sulle informazioni dell'orologio interno, che devono quindi essere predisposte correttamente. Alcune release richiedono esplicitamente una nuova data e ora, mentre altre presumono che questi dati siano già esatti; in ogni caso vedrete una visualizzazione di data e ora, e se non sono corrette, le potrete cambiare come opportuno.

I successivi dialoghi della procedura di personalizzazione consentono di creare password per gli utenti **root** e **install**; a volte a questo punto è possibile anche creare nuovi identificatori di login e nuove password per gli utenti della macchina. In questo caso dovreste creare almeno un login per vostro uso, dal momento che il login **root** dovrebbe essere riservato alla gestione e all'aggiornamento del sistema. Per definire un nuovo identificatore di login occorre specificare, oltre all'effettivo identificatore di login, il nome dell'utente e una password di partenza. Come identificatore di login potete scegliere qualsiasi parola che inizi con una lettera minuscola (da *a* a *z*) e abbia una lunghezza compresa tra 2 e 8 caratteri. La maggior parte degli utenti utilizza i loro nomi, o le loro iniziali, oppure parole originali. Il sistema assegna automaticamente all'utente un home directory, un numero di identificazione d'utente, un numero di identificazione del gruppo a cui l'utente appartiene; tutti questi assegnamenti vengono generalmente accettati come sono di default perché corrispondono sicuramente a valori corretti. Tutti gli identificatori di utente devono avere associata una password, per cui occorre sempre selezionare una password iniziale quando definite nuovi identificatori di login; per prevenire gli errori, la password deve essere inserita due volte. Nel creare le nuove password seguite rigorosamente tutte le misure di sicurezza descritte nel Capitolo 20.

Come ultimo passo, la procedura di personalizzazione richiede di scegliere un nome univoco per il sistema, noto nella terminologia UNIX come *nome di nodo* (node name) o **uname** (UNIX name) del sistema. Il nome della macchina deve essere univoco, in modo che le altre macchine possano inviare la posta senza rischio di ambiguità. Nel sistema può essere codificato un **uname** di default, ma occorre sempre cambiarlo per renderlo univoco. Assegnate alla macchina un identificatore di lunghezza compresa tra 4 e 8 caratteri, che inizi con una lettera minuscola (da *a* a *z*); il nome della macchina deve essere definito solo questa volta, e in seguito dovrebbe essere modificato solo eccezionalmente, perché verrà usato dagli utenti di altre macchine per dialogare con la vostra macchina. Se decidete di cambiare il nome, il sottosistema **uucp** non funzionerà più correttamente fino a che tutti i vostri corrispondenti non avranno apportato la stessa modifica nei loro dati di **uucp**.

La definizione di **uname** costituisce l'ultimo passo della procedura di personalizzazione. Se volete, potete interrompere a questo punto la procedura di personalizzazione e continuare il caricamento in un tempo successivo. Se

interrompete la procedura, è importante reinizializzare la macchina immediatamente, in modo da stabilire permanentemente la nuova configurazione del sistema. Quando reinizializzate dovreste vedere un messaggio simile a questo:

```
The UNIX Operating System will now be rebuilt.
This will take some time. Please wait.

The UNIX Kernel has been rebuilt.

Reboot the system now.
```

Questo messaggio vi avverte che il kernel UNIX è stato *relinked*, ovvero ricostituito con l'installazione di tutti i nuovi driver e device. Questo messaggio appare dopo che sono state installate le parti più semplici dei pacchetti del software.

25.10 Installazione di pacchetti software addizionali

A questo punto, invece di reinizializzare, il sistema può richiedere di iniziare il caricamento di pacchetti addizionali scelti in precedenza; se tale scelta non è stata ancora effettuata, può essere effettuata adesso. Il sistema presenterà una richiesta simile a questa:

```
Insert a cartridge tape into Cartridge Tape Drive.
Type [go] when ready,
or [q] to quit:
```

Se finora avete caricato tramite dischetti, il sistema richiederà un dischetto. Se rispondete **q** la macchina ricostruirà il kernel e reinizializzerà.

Potete caricare in questa fase tutti i pacchetti desiderati a patto che risiedano sullo stesso tipo di supporto. Se avete alcuni pacchetti su nastro e altri su disco, quando siete pronti a passare da un tipo all'altro dovete reinizializzare e caricare i rimanenti pacchetti da shell.

25.11 Installazione di software da shell

Quando la macchina è stata reinizializzata dopo la ricostruzione del kernel, potete collegarvi alla macchina come **root** e lanciare il comando **sysadm** o un'altra procedura utente di gestione (*user agent*). Di solito queste procedure includono un'opzione per l'installazione del software. In alternativa, per il caricamento di software potete utilizzare il comando **pkgadd** o **installpkg**, tuttavia è preferibile caricare la maggior parte possibile del software du-

rante il caricamento iniziale, perché si riduce significativamente il tempo totale di caricamento.

La scelta tra **pkgadd** e **installpkg** dipende dal particolare pacchetto softwtare da caricare. Usate:

```
# pkgadd -d diskette1
```

se il pacchetto risiede su dischetto, oppure

```
# pkgadd -d Ntape1
```

se risiede su nastro. Potete provare prima con **pkgadd**; se ottenete un messaggio d'errore provate con **installpkg**. Questi comandi richiedono di specificare durante la procedura d'installazione i pacchetti software che intendete caricare. Consultate la documentazione che riguarda la vostra macchina e il software addizionale da caricare, per determinare se occorrono particolari procedure od operazioni speciali.

USO DEI COMANDI D'INSTALLAZIONE

Le procedure **pkgadd** e **installpkg** spesso comportano molte domande addizionali per motivi di sicurezza. Per esempio, nel corso di queste procedure potreste incontrare richieste di questo genere:

```
The following files are already installed on the system and are being
used by another package:
    /usr/lib <attribute change only>

Do you want to install these conflicting files [y,n,?,q]

This package contains scripts which will be executed with super-user
permission during the process of installing this package.

Do you want to continue with the installation of this package [y,n,?]
```

Questi dialoghi sono previsti per migliorare la sicurezza del sistema, ma dovete rispondere **y** in tutti i casi, altrimenti quella installazione non viene eseguita. Se il pacchetto proviene su supporti originali del fornitore la soluzione migliore è accettare tutte le proposte offerte e procedere nell'installazione. Oltretutto, il software che intenzionalmente viola regole di sicurezza può facilmente superare anche questi controlli.

Alcuni pacchetti possono richiedere di selezionare varie caratteristiche di configurazione hardware, come indicare indirizzi di memoria o d'interruzione delle schede addizionali. Se compaiono questi messaggi e non conoscete le risposte potete prendere nota delle domande e abortire l'installazione; quindi potete spegnere la macchina, controllare i *jumper* e la documen-

tazione hardware, e riprendere l'installazione in un tempo successivo. Di solito la procedura d'installazione quando viene abortita lascia una situazione pulita.

A motivo della complessità intrinseca nell'integrazione di numerosi pacchetti in un unico sistema, installare un pacchetto può creare un conflitto con altri; queste situazioni avvengono e sono ufficialmente denominate *bug*. Potrete incontrare diversi messaggi simili:

```
Installation of <pkg> partially failed.
```

o equivalenti. In questo caso, potrete avere un'installazione non corretta per l'1%, ma spesso non potete fare nulla per evitare questi messaggi; se ne appaiono, continuate l'installazione comunque. In genere, il sistema rimane utilizzabile mentre procurate di ottenere dal fornitore una versione corretta del pacchetto.

25.12 Predisposizione dei terminali

Un ulteriore servizio che viene spesso configurato immediatamente dopo la personalizzazione del sistema, è *Service Access Facility*, che consente agli utenti di collegarsi al sistema da terminali remoti tramite porte di comunicazione asincrona. In SVR4 la procedura di configurazione viene eseguita tramite lo strumento per la gestione del sistema **sysadm**, trattato nel Capitolo 12.

Dopo avere completato questo passo delle procedure di configurazione, i terminali o i modem sono disponibili per le connessioni in ingresso. Inoltre, occorre anche configurare il sottosistema **uucp** in modo che la macchina possa chiamare le altre macchine. Le modalità dettagliate sono state descritte nel Capitolo 15.

25.13 Approfondimenti

Configurare correttamente il sistema richiede esperienza, in quanto trovare la soluzione più adatta per raggiungere l'efficienza ottimale del sistema è un'operazione complessa. Comunque, nella maggior parte dei microelaboratori, differenze nella configurazione non influenzano sensibilmente le prestazioni e poche condizioni sono critiche. A parte le dimensioni del file system, gli utenti di solito possono accettare anche piccole riduzioni nelle prestazioni del sistema, se in cambio possono non preoccuparsi troppo dei dettagli delle operazioni di configurazione del sistema; tuttavia, saranno utili alcune ulteriori considerazioni.

MESSAGGI DI PANIC

Quando il sistema UNIX e il kernel non funzionano correttamente, il sistema si può disattivare improvvisamente. Non si tratta di un evento frequente e si presenta come risultato della presenza di un errore (bug) nelle parti più interne del sistema operativo. Una situazione di questo tipo non si presenta quando tutto funziona correttamente, ma può presentarsi installando nella macchina in maniera scorretta nuovo hardware o altre schede di espansione.

Se si verifica la caduta (crash) del sistema, appare alla console un messaggio prima della disattivazione:

```
PANIC: system error type 0x0e
Attempting to dump 512 pages.....
```

Dopo questo messaggio la macchina rimane inoperativa fino alla successiva reinizializzazione.

Il messaggio visualizzato può differire, ma vi compare sempre la parola chiave "PANIC"; prendete nota del messaggio completo e reinizializzate il sistema. Se il sistema non si inizializza, probabilmente si è verificato qualche guasto di natura hardware, come per esempio nel disco o nella scheda madre.

Se la macchina si inizializza ma il messaggio di panico appare ancora, di solito qualche switch o jumper nelle schede addizionali è stato posizionato scorrettamente. Assicuratevi che tutti i componenti hardware addizionali funzionino correttamente, rimuovendo le schede una per una e osservando lo stato del sistema, oppure esaminando attentamente la documentazione delle schede e la posizione degli switch.

Se il messaggio di avvertimento PANIC appare solo una volta e non si ripresenta, allora non c'è motivo di preoccuparsi; avete solo scoperto un errore (raro) del sistema UNIX. In passato, gli esperti giudicavano la qualità di una versione di UNIX dalla frequenza di messaggi PANIC, ma nei recenti sistemi UNIX non se ne presentano mai, salvo in presenza di guasti nell'hardware.

INIZIALIZZAZIONE DA DISCHETTO

In un sistema UNIX correttamente in funzione, la macchina inizializza dal disco rigido quando nel drive non si trova alcun dischetto. Se il drive contiene un dischetto e lo sportellino è chiuso, la macchina cerca di inizializzare dal dischetto; se il dischetto che si trova nel drive non è inizializzante, allora la procedura d'inizializzazione riprende dal disco rigido e attiva la macchina come al solito. Tuttavia, in qualche caso la macchina si arresta e il processo di caricamento non si completa, per cui dovete rimuovere il dischetto e ripetere di nuovo l'inizializzazione.

Comunque, se necessario, è possibile inizializzare volutamente il sistema dal System Installation Disk, per esempio quando si hanno dei problemi nell'inizializzazione da disco rigido e occorre ripristinare manualmente la funzionalità del disco rigido. Le modalità d'inizializzazione sono le stesse che abbiamo seguito per caricare inizialmente il sistema UNIX dal System Installation Disk, utilizzando una copia del disco invece dell'originale.

Quando la procedura d'inizializzazione da dischetto si arresta alla visualizzazione di un messaggio, premete immediatamente il tasto DEL per abortire il processo di caricamento ed entrare subito in ambiente shell; comunque, invece di inizializzare il sistema dal disco rigido come consueto, il sistema procede in esecuzione da dischetto, o talvolta da un disco RAM temporaneo.

Esistono alcune differenze nell'ambiente di lavoro quando l'esecuzione procede da dischetto. Primo, il sistema è molto più lento del solito, dal momento che non utilizza il disco rigido. Secondo, non è possibile rimuovere il dischetto senza provocare la caduta del sistema. Terzo, l'insieme di comandi disponibili è notevolmente limitato, perché i comandi devono risiedere su un supporto di limitata capacità. Quarto, il sistema opera in modalità monoutente, per cui l'ambiente di processi differisce notevolmente dalle normali condizioni. Questa procedura d'inizializzazione da dischetto non è adatta per un utilizzo regolare della macchina; deve essere impiegata solo da un utente esperto che debba eseguire rare verifiche o riparazioni del sistema.

Una volta che la macchina è funzionante, è possibile montare il disco rigido per consentire l'accesso ai file e ai directory in esso contenuti. Il comando

```
# mount -F ufs /dev/dsk/0s1 /mnt
```

monta la partizione di disco rigido che contiene il file system **root**. Utilizzando il comando **cd**, potete accedere al directory **/mnt** per accedere al directory **root** del disco rigido e da quel punto percorrerne il file system. Potete lanciare i comandi che sono contenuti sul disco rigido, specificando i corrispondenti pathname completi preceduti da **/mnt**, oppure potete cambiare la vostra PATH per includere i directory **bin** del disco rigido. In questo modo potete ripristinare i file che ritenete siano difettosi. Sfortunatamente, non potete rimuovere il dischetto per copiare file da un dischetto di backup al disco rigido, né potete utilizzare il sottosistema **uucp** o strumenti LAN per trasferire file tramite una rete. In conclusione, la procedura di inizializzazione da dischetto presenta molti limiti; può avere maggiore utilità nelle operazioni di manutenzione, se il sistema dispone di due drive.

Quando inizializzate da dischetto con questa procedura, abbiate cura di disattivare correttamente la macchina. Il comando **shutdown** non è proba-

bilmente disponibile su dischetto, ma potete ricorrere alla seguente breve procedura:

```
# umount /mnt
# sync
# sync
# uadmin 2 0
Reboot the system now.
```

A questo punto potete spegnere la macchina o reinizializzare da disco rigido come consueto.

PASSAGGIO IN SVR4 DA VERSIONI PRECEDENTI

Se state utilizzando una versione meno recente di UNIX, dovete pianificare con attenzione le azioni prima di iniziare il passaggio a SVR4. Occorre ricaricare il software di sistema e probabilmente ridefinire le partizioni del disco rigido per riservare un'area di swap di dimensione maggiore che soddisfi le esigenze di SVR4. Conviene probabilmente riformattare il disco rigido e ricostruirlo ex novo. Inoltre, tra le versioni precedenti di sistema UNIX e SVR4, molti file chiave del sistema sono cambiati, talvolta anche profondamente. Non potete eseguire un'installazione in sovrapposizione passando in SVR4 da release precedenti; è possibile eseguire solo un'installazione completa.

Il primo passo nell'operazione di passaggio alla versione superiore (*upgrade*) consiste nel salvare scrupolosamente il sistema su qualche supporto sicuro, come il dischetto o il nastro. Salvate tutto l'intero sistema, perché se per qualche motivo non riuscite a concludere l'operazione dovete essere in grado di ricaricare la vecchia versione; per esempio, l'operazione può fallire perché uno dei dischetti della nuova versione non è leggibile e non può essere caricato correttamente. Assicuratevi di salvare tutti i vostri file e tutto il software che vi occorre. Il modo migliore per eseguire il salvataggio è di salvare ogni singolo directory su un dischetto distinto e, successivamente, quando siete pronti a ripristinare i dati nella nuova versione, caricare ogni disco separatamente in un directory temporaneo, trasferire attentamente i file a destinazione e poi cancellare le copie.

Questa procedura contiene a livelli tollerabili il maggiore utilizzo di spazio su disco, in quanto probabilmente la vostra macchina non supporterebbe contemporaneamente due versioni distinte dell'intero sistema sul disco. Oltretutto, non è possibile riscrivere la vecchia versione del sistema sulla versione SVR4, perché altrimenti si distrugge il sistema. Occorre salvare i file privati degli utenti separatamente dai file di sistema, perché non è possibile ripristinare con sicurezza nessun file system e nessun directory. Ciò riguarda in particolare i file **uucp** e il contenuto dei directory **/etc** e **var**, tra cui **/etc/profile**, **/etc/passwd**, **/etc/shadow**, **/etc/inittab**.

Dopo aver salvato tutto il necessario, lanciate il processo di caricamento secondo le modalità descritte precedentemente; al termine, potete collegarvi al sistema e iniziare a ripristinare il materiale salvato. Se stavate utilizzando la versione SVR3, la conversione risulta relativamente semplice, ma le versioni meno recenti di SVR2, oppure le versioni meno recenti di XENIX, richiedono sostanziali modifiche. In questi casi, non conviene probabilmente salvare niente del software meno recente, ma occorre ricostruire il sistema integralmente ogni volta che incontrate qualcosa che non risponde ai requisiti voluti. Comunque, è possibile ricaricare senza problemi i file di testo, fogli elettronici, database, ecc., e anche molti programmi eseguibili potranno funzionare correttamente. È molto importante ripristinare il vecchio sistema in un directory temporaneo, esaminare manualmente il contenuto e utilizzare **mv** per trasferire quello che interessa nel directory definitivo.

Se *non* siete sicuri, *non* riscrivete i file di sistema esistenti in un directory diverso dal vostro home directory o dagli altri directory privati. Dovete esaminare attentamente ogni file e decidere come trattarlo, considerando che generalmente non è possibile ripristinare i file di sistema che provengono da versioni meno recenti.

Le applicazioni eseguibili di SVR3 e i file di dati trasferiti da una macchina 80286 / 80386, possono continuare a funzionare correttamente su una macchina 80386 / 80486 con sistema SVR4, ma dovreste verificarne attentamente il comportamento. Se possibile, dovreste ricompilare le applicazioni che sono state trasferite a macchine SVR4 da release precedenti.

DEFAULT DI SISTEMA

In molti sistemi SVR4 nel directory **/etc/default** sono presenti diversi file che contengono i dati di configurazione della macchina, come nell'esempio:

```
# ls -F /etc/default
boot            default.att512  login       tar
cron            default.cpq     msdos       workstations
default.at386   dump            passwd      xrestor
default.att     init            su
#
```

Questi file contengono informazioni per il particolare hardware e ambiente della macchina; i file con nome che inizia con *default* contengono specifiche mappe della memoria e altre informazioni che differiscono in base al tipo di macchina; i rimanenti file controllano altre variabili di sistema. Potete esaminare questi file e modificare singoli elementi; le modifiche apportate avranno effetto quando la macchina verrà reinizializzata.

Esaminate attentamente ogni file per rendervi conto del genere di modifiche possibili in ogni area, ma siate molto accorti nell'eseguire modifiche;

verificate l'effetto di ogni modifica prima di "infliggere" ai vostri utenti una configurazione con nuove caratteristiche. In alcuni casi il campo delle alternative è limitato e non potete cambiare con valori a caso questi file.

Il file più importante per la gestione del sistema è **login**, che contiene diversi importanti parametri configurabili; per esempio:

```
$ cat /etc/default/login
TIMEZONE = MST7MDT
HZ = 100
ULIMIT = 80000
CONSOLE = /dev/console
PASSREQ = YES
ALTSHELL = YES
$
```

I valori *TIMEZONE* e *ULIMIT* dovrebbero essere predisposti esplicitamente per il vostro sistema; quando cambiate il valore *TIMEZONE* in **/etc/default/login** abbiate cura di cambiarlo anche in **/etc/TIMEZONE**. Il valore *CONSOLE* può essere cambiato in un device tty se volete controllare la macchina da una console remota; le altre variabili non dovrebbero in genere venire cambiate.

Potete predisporre un valore della variabile *TERM* per ciascuna singola porta nel file **/etc/ttytype**. Di solito è preferibile assegnare *TERM* nel **.profile** di ciascun utente, ma se avete un terminale in connessione diretta permanente a una porta seriale, o se usate spesso la modalità di console virtuale, potete predisporre un default per quel terminale in **/etc/ttytype**. Se volete sfruttare questa possibilità esaminate **/etc/ttytype**.

Potete riorganizzare totalmente, o rimappare, la posizione dei caratteri sulla tastiera di console; questo può permettere di rendere la tastiera in stile AT più comoda per le abitudini degli utenti di sistemi UNIX. Di solito i tasti CAPSLOCK e CTRL sinistro vengono invertiti, come pure i tasti ESC e ‘ (accento grave); sono possibili anche altre riorganizzazioni della tastiera. Il comando **mapkey** gestisce l'operazione di mappatura, ma tenete conto che le modifiche fatte da **mapkey** non influenzano le sessioni sotto X Window System; se usate X dovete fare i cambiamenti con **mapkey** per la console e con **xmodmap** per le sessioni X.

Per usare **mapkey** dovete scaricare la mappa corrente della tastiera in un file con

```
# mapkey -d > file
```

modificare la mappa con un editor secondo le vostre esigenze, e quindi ricaricarla con

```
# mapkey < file
```

Consultate le man page **mapkey(1)** e **kbd(7)** per dettagli maggiori, ma tenete conto che i cambiamenti alla mappa non sopravvivono a una reinizializzazione, per cui dovrete riportare le modifiche tutte le volte che inizializzate la macchina.

PARAMETRI DI AFFINAMENTO

Molti sistemi UNIX consentono a un gestore di sistema esperto di variare alcuni parametri di "affinamento" all'interno del nucleo. Si tratta di variabili che regolano la dimensione di alcune aree dati interne al sistema operativo e i cui valori possono essere variati per esigenze particolari, se necessario. Le versioni SVR4 per microcalcolatore dispongono di solito di alcuni strumenti che consentono di variare alcuni di questi parametri; altri parametri possono essere contenuti in **/etc/default**. In piccoli sistemi SVR4 spesso sono affinabili i parametri : **nbuf** (numero di buffer di sistema), **nclists** (numero di buffer di sistema per I/O di caratteri), **nproc** (numero di processi simultanei ammessi per utente) e **nfile** (numero di file che un processo può aprire).

Comunque, non occorre modificare i valori predefiniti di questi parametri, a meno che non sia richiesto per l'installazione di un particolare pacchetto applicativo.

Seguite attentamente le istruzioni del fornitore e non cambiate questi valori se non è necessario.

Capitolo 26

Conclusioni

Il sistema UNIX mette a disposizione un ambiente di elaborazione straordinariamente ricco e variato. Nessun singolo libro può esaurire la trattazione di tutte le caratteristiche del sistema UNIX in modo dettagliato e neanche esaminare adeguatamente l'intero insieme di comandi e le implicazioni connesse. In questo testo abbiamo trattato il *Foundation Set*, o complesso base, con un livello di dettaglio abbastanza accurato da far diventare un utente moderatamente esperto di microcalcolatori, utente e gestore competente di UNIX.

Oltre al *Foundation Set*, la versione completa del sistema UNIX SVR4 si compone di altre parti, di cui *Documenter's Workbench* e *C Compilation System* sono le più importanti.

Inoltre, molti altri pacchetti "semi-standard" fanno parte della versione ufficiale AT&T SVR4, ma generalmente vengono venduti separatamente. Si tratta per esempio del pacchetto *Writer's Workbench* (un insieme di strumenti addizionali che completano *Documenter's Workbench*), degli strumenti *System Activity Reporting* (**sar**) e degli strumenti *Remote Job Entry* (**rje**).

Un altro tipo di software comprende le applicazioni supportate dai numerosi appassionati utilizzatori del sistema UNIX in tutto il mondo. L'editor di testo EMACS, la rete delle notizie e alcuni giochi popolari sono esempi di questo tipo di programmi, disponibili per quasi tutte le versioni. In alcu-

ni casi non sono in vendita, e sono reperibili solo nel formato sorgente. Molti bollettini d'informazione di pubblico dominio forniscono versioni eseguibili o installabili dei programmi più diffusi, e una piccola partecipazione nella comunità dei sistemi UNIX vi fornirà le informazioni occorrenti per trovarli.

Inoltre, esiste in commercio una grande quantità di prodotti software disponibili per il sistema UNIX. Contrariamente all'opinione generale che non sia disponibile molto software per il sistema UNIX, in realtà si trovano migliaia di applicazioni usuali, tra cui molti word processor e fogli elettronici, un grande numero di sofisticati programmi scientifici e ingegneristici, strumenti grafici e un ricco insieme di pacchetti di comunicazione. Oggi la maggior parte delle case costruttrici di software e di hardware scrive le proprie applicazioni nel linguaggio C e quasi tutte supportano il sistema UNIX. A causa della crescente popolarità di UNIX, sul mercato appaiono regolarmente sempre nuove applicazioni; poiché i progettisti software preferiscono sviluppare le loro applicazioni in ambiente UNIX, la disponibilità di software continuerà a crescere.

26.1 Giochi e passatempi

Una delle più popolari aree di software non ufficiale è quella dei giochi e passatempi. Nel corso degli anni, è stata sviluppata una grande quantità di giochi innovativi per il sistema UNIX, come per altri sistemi operativi. In base alla filosofia del sistema UNIX di supportare terminali remoti ASCII, si tratta quasi esclusivamente di giochi che funzionano in modalità carattere. I giochi migliori sono diventati leggendari nel mondo dei calcolatori e si dice spesso che questi giochi abbiano consumato più risorse di CPU sulle macchine UNIX di ogni altra categoria di applicazioni, compreso lo sviluppo di software e l'elaborazione di testi.

I giochi, se sono disponibili, si trovano generalmente nel directory **/usr/games**. Il contenuto di questo directory differisce molto sulle diverse macchine, dal momento che i giochi non costituiscono una parte standard del sistema UNIX; comunque, la Sezione 6 del manuale di documentazione viene riservata ai giochi e il sistema spesso include un insieme di giochi originali che costituisce la base del directory **/usr/games**. I giochi originali consistono generalmente in semplici giochi per ragazzi, anche se alcuni di essi possono essere più sofisticati. Il gioco **chess** del sistema UNIX è in grado di sconfiggere giocatori di scacchi di media capacità; è stato superato negli ultimi anni da programmi allo stato dell'arte, come **gnuchess**, distribuito gratuitamente dalla Free Software Foundation, che si comporta come un eccellente giocatore a livello di maestro.

Il gioco più popolare tra quelli originali del sistema è noto come **adventure**, oppure "Zork" in ambiente MS-DOS, che consiste nell'esplorazione di

una prigione sotterranea da parte del giocatore. L'interfaccia utente di questo gioco non è grafica, ma orientata al linguaggio, per cui il giocatore si muove lungo la prigione specificando brevi frasi di comando come per esempio "Go West" oppure "Pick up the knife", ecc.

Nei primi anni Ottanta, **rogue** ha scalzato **adventure** quale gioco più popolare; consente anch'esso di esplorare una prigione sotterranea, ma con molti miglioramenti rispetto ad **adventure**. Il gioco **rogue** fornisce sul video una visualizzazione completa della prigione che viene esplorata, prevede molti livelli di difficoltà e il giocatore deve combattere 26 diversi tipi di mostri alla ricerca dell'oro e del favoloso amuleto di Yendor. Il gioco **rogue** si dimostra così difficile che solamente pochi giocatori sono riusciti a vincere, ma rappresenta comunque un passatempo particolarmente affascinante e fuori dal comune.

Nella seconda metà degli anni Ottanta da **rogue** sono derivati diversi altri giochi popolari, ognuno contraddistinto da maggiore complessità e numero di funzioni rispetto al precedente. Il gioco **hack** rappresenta una evoluzione di **rogue**, pur mantenendone lo stesso stile di gioco, mentre **larn** è un adattamento più libero di **rogue** e richiede una strategia diversa.

Attualmente, stanno comparendo molti giochi con interfacce utente grafiche bit-mapped, per i sistemi UNIX che le possono supportare, oppure alcuni giochi adatti a più giocatori. In alcuni casi, si tratta di giochi molto reattivi basati su comunicazioni in tempo reale tra gli utenti; in altri casi, consistono in giochi di strategia (come gli scacchi) che si avvantaggiano notevolmente dalle caratteristiche multiprocesso del sistema UNIX. Il principale esempio è il gioco **mazewar** in cui i gocatori si aggirano in un labirinto, guadagnando punti ogni volta che eliminano altri giocatori. Sono disponibili molti altri giochi che funzionano sotto X Window System e/o la rete Internet; uno dei più popolari è **nethack**, una versione di **hack** con più giocatori. Questi giochi, adatti a più giocatori, predomineranno sicuramente nei prossimi anni e saranno soggetti a evoluzioni e miglioramenti più di ogni altro settore software.

26.2 La comunità internazionale degli utenti

Nel corso della storia del sistema UNIX si è creata tra gli utenti una comunità estremamente compatta, favorita in parte dalla facilità di comunicazione coi sistemi **uucp** e, **mail** e in parte, dal fascino mistico che il sistema UNIX ha sempre avuto tra i progettisti software. Esistono diversi gruppi di utenti indipendenti, tra cui il principale viene chiamato *Uniforum* (in passato **/usr/group**); inoltre, diversi congressi e meeting nazionali sono dedicati a discussioni sul sistema UNIX, tra questi citiamo come più importante il meeting *Usenix*, che viene tenuto con regolarità. Tutte queste organizzazioni costituiscono eccellenti sorgenti di informazione e contatti. La maggior

parte dei vantaggi effettivi della tecnologia del sistema UNIX è stata originariamente diffusa verbalmente, o mediante posta elettronica, tra gli utenti interessati ed è stata poi introdotta nelle versioni ufficiali solo dopo che gli utenti hanno bene accettato le innovazioni.

Gli utenti del sistema UNIX dispongono di un catalogo generale a livello mondiale e di un sistema di posta elettronica che serve diverse migliaia di macchine e centinaia di migliaia di utenti. Nota sotto vari nomi, come *netnews*, *usenet*, *readnews*, o semplicemente *the net*, questa rete si basa principalmente sull'accesso mediante chiamata telefonica tra macchine vicine, che scambiano messaggi di posta elettronica, noti come *news items*, o articoli. Queste notizie possono essere raggruppate in più di 300 categorie diverse o gruppi di notizie, *newsgroups*.

Le macchine vicine ritrasmettono i messaggi alle altre macchine e così un messaggio alla fine si propaga dal mittente originario a tutte le macchine nella rete. Un utente collegato su una di queste macchine utilizza un particolare software per leggere le notizie e può definire un profilo che consente di richiamare solamente i gruppi di notizie che interessano; può quindi esaminare, salvare o rispondere ai messaggi. Questa rete può interfacciarsi mediante *gateway* a diverse altre reti comuni, come per esempio Internet, un'estesa rete universale, e BITNET, una rete di elaboratori di università e collegi.

Più di 1000 messaggi al giorno percorrono la rete e riguardano gli argomenti più disparati, principalmente il sistema UNIX e il linguaggio C; transita costantemente in rete anche il codice sorgente di dominio pubblico per nuovi giochi, nuovi algoritmi e recentissimi sistemi esperti. Si trattano argomenti che interessano nuove tecnologie e nuovi prodotti, mentre gruppi non tecnici trattano di politica, musica, sport, ecc.

La rete di messaggi è una struttura autosufficiente supportata da ogni macchina, che paga le bollette del telefono relative ai messaggi inoltrati alle sue macchine contigue. Comunque, recenti esperimenti di uno schema di comunicazione basato su satellite e una struttura di rete a pagamento per il collegamento tra macchine con funzioni di concentratori, può cambiare il carattere di libertà della rete, in quanto i costi dei collegamenti non possono essere agevolmente suddivisi tra le macchine. Alcuni progettisti creativi forniscono il software delle reti e migliorano costantemente gli strumenti. Sono disponibili diversi popolari lettori di messaggi, di solito distribuiti in forma di codice sorgente.

Per includere la vostra macchina nella rete di notizie, avete bisogno del software che consente di gestire i messaggi diretti alla macchina, in aggiunta al normale pacchetto **uucp**. Inoltre, dovete dedicare una notevole quantità di spazio su disco per i messaggi e localizzare una macchina vicina che sia disposta a trasmettere i messaggi alla vostra macchina. Alla vostra macchina possono essere consegnati da 2 a 3 megabyte di messaggi al giorno, se accettate tutti i diversi gruppi di notizie. Una volta che vi siete immessi

nella rete di circolazione delle notizie, potete essere collegati, con una connessione per rimbalzi, a quasi ogni altra macchina UNIX dislocata nel mondo.

26.3 Lettura delle notizie

Se nella vostra macchina disponete del sistema news, per leggere le notizie potete usare uno dei tanti programmi di dominio pubblico. Uno dei migliori è **rn** (read news); è disponibile anche una versione per X Window System, chiamata **xrn**. Questi programmi, come d'altra parte l'intero sistema news, non sono parte ufficiale di SVR4, pertanto possono non essere presenti nel vostro sistema.

Il programma **rn** visualizza gli articoli, consente all'utente di eseguire ricerche fra gli articoli e fornisce strumenti per creare e spedire articoli. Inoltre, **rn** gestisce un elenco di gruppi di notizie attivi e tiene conto degli articoli che non sono stati letti. Il programma **rn** prevede molte opzioni in linea di comando e mette a disposizione un modo comandi interno, con molti sottocomandi. Se usate la versione **xrn** molte funzioni di **rn** diventano disponibili tramite il click del mouse in luogo dei nomi dei comandi.

A inizio esecuzione, **rn** legge un proprio file nel vostro home directory, **.newsrc** (news run control), se il file non esiste viene creato alla prima esecuzione di **rn**. Questo file **.newsrc** contiene un elenco di tutti i gruppi di notizie supportati, un gruppo per ciascuna linea del file, come nell'esempio:

```
$ tail -6 .newsrc
comp.unix.wizards: 1-11402,12974-13606
rec.games.go: 1-1395
comp.unix.i386: 1-4476
comp.os.mach: 1-161
sci.virtual-worlds! 1-9
rec.arts.startrek.info!
$
```

Poiché esistono centinaia di gruppi, **.newsrc** può raggiungere grandi dimensioni. Ciascun gruppo appartiene a una gerarchia di nomi di classi. I gruppi **comp** riguardano la tecnologia degli elaboratori, **rec** riguarda molte forme di ricreazione, ecc.; all'interno di ciascuna classe esistono delle sottoclassi, come **unix** o **arts**; sono comuni anche ulteriori suddivisioni. A ogni articolo ricevuto dalla macchina viene assegnato un numero; la lista dei numeri che seguono il nome del gruppo rappresenta l'intera storia del sistema netnews nella macchina. Per esempio, nell'esempio precedente la macchina ha ricevuto 4476 articoli per il solo gruppo **comp.unix.i386**. Quando leggete degli articoli, il conteggio viene aggiornato e **rn** usa il contatore per visualizzare solo gli articoli non letti. Dopo il numero del gruppo il carattere **:** oppure

! richiede a **rn** di visualizzare o no quel gruppo; il carattere ! informa **rn** che quel gruppo non vi interessa. Per esempio, la linea

```
sci.virtual-worlds! 1 – 9
```

significa che l'utilizzatore ha esaminato i primi nove articoli di questo gruppo e ha quindi deciso di non ricevere ancora il gruppo.

Abitualmente, gli articoli vengono conservati nella macchina solo per un breve periodo (di solito una settimana o due), quindi potreste non vedere tutti gli articoli se rifiutate un gruppo, oppure trascurate di leggere gli articoli per un certo tempo. La linea

```
comp.unix.wizards: 1 – 11402,12974 – 13606
```

mostra un intervallo nella sequenza degli articoli letti; questi articoli sono stati cancellati dalla macchina prima che l'utente li potesse vedere.

Non dovete compiere nessuna operazione manuale di editor o modifica del file **.newsrc**; questo è compito del programma **rn**.

In pratica, **rn** è un programma complesso con molti differenti modi di operare e un gruppo separato di comandi per ciascun modo. Può lavorare a livello di gruppo di articoli, *newsgroup level*, nel senso che gestisce l'elenco di gruppi che ricevete, consentendovi di aggiungere o togliere gruppi se i vostri interessi cambiano. Quando cancellate un gruppo al vostro livello utente, il gruppo non viene realmente eliminato dalla macchina, ma **rn** cessa di visualizzare per voi gli articoli di quel gruppo; potete riaggiungere il gruppo in qualunque momento. Il livello d'articolo, *article level*, gestisce l'accesso agli articoli nel gruppo corrente, consentendo di selezionare o rifiutare singoli articoli. Il livello d'impaginatura, *paging level*, gestisce la visualizzazione dei singoli articoli, consentendo di spostarsi avanti e indietro nelle pagine dell'articolo. Inoltre, **rn** prevede modi per creare e spedire un articolo, rispondere per posta elettronica a un articolo e per configurare la vostra sessione.

Quando lo mettete in esecuzione, **rn** inizia col comparare gli articoli nell'albero di news (di solito **/usr/news** e suoi subdirectory) con gli articoli elencati nel vostro **.newsrc**. Se sono presenti più articoli di quelli che avete letto, **rn** visualizza il numero di articoli non letti di ciascun gruppo che avete dichiarato di vostro interesse, nell'ordine con cui appaiono nel vostro file **.newsrc**, come nell'esempio:

```
$ rn
Unread news in comp.sources.games                2 articles
Unread news in comp.unix.wizards                 1 article
Unread news in comp.windows.x                   16 articles
Unread news in rec.games.go                      1 article
Unread news in alt.sources                       1 article
etc.

******** 2 unread articles in comp.sources.games– –read now? [ynq]
```

Per ridurre la quantità dei dati in uscita **rn** visualizza solo pochi gruppi alla volta, quindi vi richiede di decidere se volete leggere adesso il primo gruppo; a questo punto vi trovate in modo gruppo. Potete rispondere **y** per entrare nel modo articolo per quel gruppo, **n** per passare al successivo gruppo, oppure **q** per uscire dal programma e ritornare in shell. Nel modo gruppo sono disponibili molti altri comandi; potete ottenere un elenco dei comandi introducendo **h** (help). Potete passare a un determinato gruppo con **g** *nome.gruppo*, evitare la successiva visualizzazione del gruppo con **u**, ricercare un gruppo il cui nome corrisponda a una stringa con *\stringa*, o compiere molte altre azioni relative alla gestione dei gruppi. Con **=** potete ottenere un elenco degli argomenti di tutti gli articoli non letti di un gruppo, con **c** (catch up) potete considerare tutti gli articoli di un gruppo come letti, se non avete interesse a leggerli.

Per leggere gli articoli di un gruppo, selezionate un numero di articolo (se avete usato = per ottenere l'elenco), oppure rispondete **y** per visualizzare il primo articolo ancora da leggere, come nell'esempio:

```
******** 2 unread articles in comp.sources.games--read now? [ynq] y

Article 493 (1 more) in comp.sources.games (moderated):
From: giorgio@my_sys.com (giorgio)
Subject: v98i085: go -- the oriental board game
Message-ID: <146@my_sys.com>
Reply-To: 6 Feb 90 17:43:12 GMT
Sender: giorgio@my_sys.com
Lines: 1989

Follows is a version of the game of "go" for UNIX SVR4.

The following is a "shar" archive; unpack with "/bin/sh" NOT csh

================ Cut here ================
--MORE--(2%)
```

All'inizio dell'articolo compaiono diverse intestazioni di articolo che contengono informazioni riguardanti l'articolo; sono simili a quelle di molti programmi di posta elettronica, anche se talvolta più numerose. Comprendono il mittente (Sender), la data e l'ora di creazione dell'articolo, il soggetto e un identificatore di messaggio attribuito dal sistema news al momento della creazione del messaggio. Notate che questo numero differisce dal contatore locale dei messaggi attuato dalla macchina ricevente. Il campo "Reply-To:" normalmente contiene il corretto indirizzo UNIX della posta, da usare per contattare il creatore del messaggio; talvolta l'indirizzo di risposta contiene dati errati, pur avendo le informazioni raggiunto la vostra macchina.

Nel precedente esempio il messaggio contiene più linee di quante ne possa contenere lo schermo; il programma **rn** riconosce questa condizione, e si arresta con il messaggio "MORE" quando lo schermo è pieno; adesso siete in modo pagina e potete introdurre comandi simili a quelli delle funzioni di **more**. Se introducete uno spazio, appare la successiva pagina del messaggio, se introducete un RETURN potete esaminare il messaggio linea per linea, se introducete **n** (next) potete considerare l'articolo come letto e passare all'articolo seguente; introducete **h** per ottenere un elenco completo dei comandi in modo pagina. Potete ricercare una stringa nell'articolo, oppure usare uno tra i vari comandi disponibili per arrivare alla fine dell'articolo.

Quando raggiungete la fine dell'articolo, sia per averlo letto completamente, sia per esserci arrivati in modo pagina, venite a trovarvi in modo articolo, come indicato in questo esempio:

```
End of article 493 (of 494)- -what next? [npq]
```

Potete introdurre **n** (next) o battere uno spazio per vedere l'articolo seguente del gruppo, **p** (previous) per vedere l'articolo precedente, **-** (meno) per rivedere l'ultimo articolo già visto, **q** (quit) per abbandonare il gruppo corrente e passare al gruppo successivo, **h** (help) per ottenere un elenco completo dei comandi in modo articolo. Potete eseguire una ricerca di un articolo con la stessa (approssimativamente) linea soggetto fra gli articoli successivi con CTRL-N, oppure fra gli articoli precedenti, con CTRL-P. Potete salvare un articolo in un file nel vostro directory col comando **s** (save); il comando accetta un argomento con un pathname e **rn** tenta di salvare l'articolo nella locazione specificata, come nell'esempio:

```
End of article 493 (of 494)- -what next? [npq] s $HOME/go.shar
File /usr/giorgio/go.shar doesn't exist- -
        use mailbox format? [ynq]
```

Come potete vedere, **rn** intende salvare l'articolo come se fosse un messaggio mail, accodandolo al file specificato; il programma può salvare l'articolo in due formati: se il file esiste già **rn** usa il medesimo formato, altrimenti richiede il formato da usare. Il formato mail può essere letto dal programma **mail**, al contrario dell'altro formato. Se decidete di abbandonare l'operazione di salvataggio introducete **q** (quit) e **rn** ritornerà in modo articolo. Se nel comando **s** non specificate un nome per il file di salvataggio, **rn** crea un file col nome del gruppo news nel vostro directory **$HOME/News**; se il subdirectory non esiste viene creato.

Potete anche salvare un articolo con un pipeline anziché un file; questo permette di trattare un articolo con una serie di comandi o di stamparlo. Occorre semplicemente usare | (pipe) nel comando **s** e quindi introdurre il pipeline come di consueto.

Per esempio con:

```
End of article 493 (of 494) - - what next? [npq] s |pr|lp
```

l'articolo viene elaborato con **pr** e quindi stampato con **lp**.

RISPOSTA VIA mail A UN ARTICOLO

Se introducete **r** o **R** (reply) potete rispondere a un articolo mediante i normali canali **mail** di UNIX. Il programma **rn** preleva l'indirizzo di "Reply-To:", quindi inzia una sessione di editor del messaggio; quando la sessione è conclusa, **rn** spedisce il messaggio e quindi ritorna in modo articolo per trattare altri articoli. Per scrivere il messaggio la procedura di risposta utilizza procedure di predisposizione del messaggio interne a **rn**, ma potete passare al vostro editor preferito. La prima azione di **rn** consiste nello scrivere intestazioni di messaggio che comprendono alcune delle informazioni chiave delle intestazioni dell'articolo; queste forniscono il riferimento all'articolo originale, come mostra l'esempio:

```
End of article 493 (of 494) - - what next? [npq] r
To: giorgio@my__sys.com (giorgio)
Subject: Re:v98i085: go - - the oriental board game
Newsgroups: comp.sources.games
In-Replay-To: <146@my__sys.com>
Organization: Human Interface Design Consulting
Cc:
Bcc:

(Above lines saved in file /home/giorgio/.rnhead)

(leaving cbreak mode; cwd = /home/giorgio)
Invoking command: mail - h /home/giorgio/.rnhead

Prepared file to include [none]:
```

In questo esempio, il programma spiega ciò che sta facendo per costituire il messaggio mail, quindi vi richiede il nome di un file preparato in precedenza; se introducete un pathname, quel file viene incluso, altrimenti potete introdurre RETURN per ignorare questa parte. Quindi **rn** inizia una sessione con l'editor indicato nella variabile di ambiente EDITOR, sul testo della versione corrente del messaggio. Potete modificare il messaggio e anche le intestazioni, ma alcune testate verranno incluse nella forma creata da **rn**, anche se le avete modificate. Quando scrivete il file e uscite da editor, **rn** richiede un'azione, come nell'esempio:

```
Send, abort, edit, or list?
```

Potete rispondere **send** per spedire il messaggio, **abort** per abbandonare il messaggio, **edit** per modificare ancora il messaggio, **list** per visualizzare il

messaggio. Quando avete finito l'operazione, **rn** ritorna in modo articolo per trattare l'articolo successivo.

INTRODUZIONE a xrn

Se disponete di X Window System troverete **xrn** di uso più agevole rispetto alla versione per tastiera **rn**; i due programmi hanno le stesse funzioni e modi d'azione e usano lo stesso file **.newsrc** nella stessa maniera. La Figura 26.1 mostra la videata iniziale di **xrn**; la linea superiore dello schermo mostra i nomi dei gruppi, in modo gruppo, oppure linee soggetto di articoli, in modo articolo. L'area sottostante è usata per il modo pagina e visualizza gli articoli selezionati. Potete selezionare un gruppo di articoli con il click del mouse sul bottone **read group**; la linea superiore mostrerà una lista di articoli non letti, in modo articolo. Il gruppo superiore dei bottoni di menu cambierà in un insieme di comandi appropriati per il modo articolo; in questa maniera potrete selezionare singoli articoli da visualizzare nella finestra inferiore. Usando il mouse e le barre di scorrimento sul lato sinistro della finestra potrete spostarvi all'interno dell'articolo; quando tutto un articolo è stato visualizzato, i bottoni grigi alla fine della finestra diventano attivi, permettendovi di predisporre il salvataggio o la risposta di un articolo. Oltre al controllo tramite mouse anziché tastiera, **xrn** differisce da **rn** anche in altri sottili dettagli, che potrete scoprire con un uso di entrambi i programmi.

CREAZIONE DI UN ARTICOLO

Quando vi trovate in modo articolo, potete introdurre i comandi **f** o **F** (follow up) per creare in articolo; sono previsti due comandi perché quando usate **F** il comando **rn** include nel messaggio l'articolo originale, mentre **f** crea un nuovo articolo. Invece di creare un messaggio mail privato per il mittente del messaggio originale, viene spedito nella rete un articolo come un qualsiasi altro articolo nella rete news. La creazione di un messaggio con i comandi "follow up" è simile alla creazione di un messaggio mail e vengono seguite le procedure già descritte per rispondere tramite mail; quando entrate nel vostro editor per creare il file, potete adattare le intestazioni del messaggio al vostro messaggio in luogo dell'originale. Probabilmente vorrete cambiare la parole chiave, il soggetto, i riferimenti. Se usate il comando **f** per creare un nuovo articolo che non ha riferimento a un articolo precedente cancellate la linea "References:" dalle intestazioni e create un nuovo soggetto. Tenete presente che alcune delle testate non possono essere alterate, perché **rn** le riscrive, rimpiazzando le modifiche da voi fatte con editor; le linee "From:" e "Organization:" non possono essere modificate, per impedire di ingannare il sistema news falsificando il nome o l'indirizzo. L'identificatore di messaggio viene attribuito dal programma **rn**.

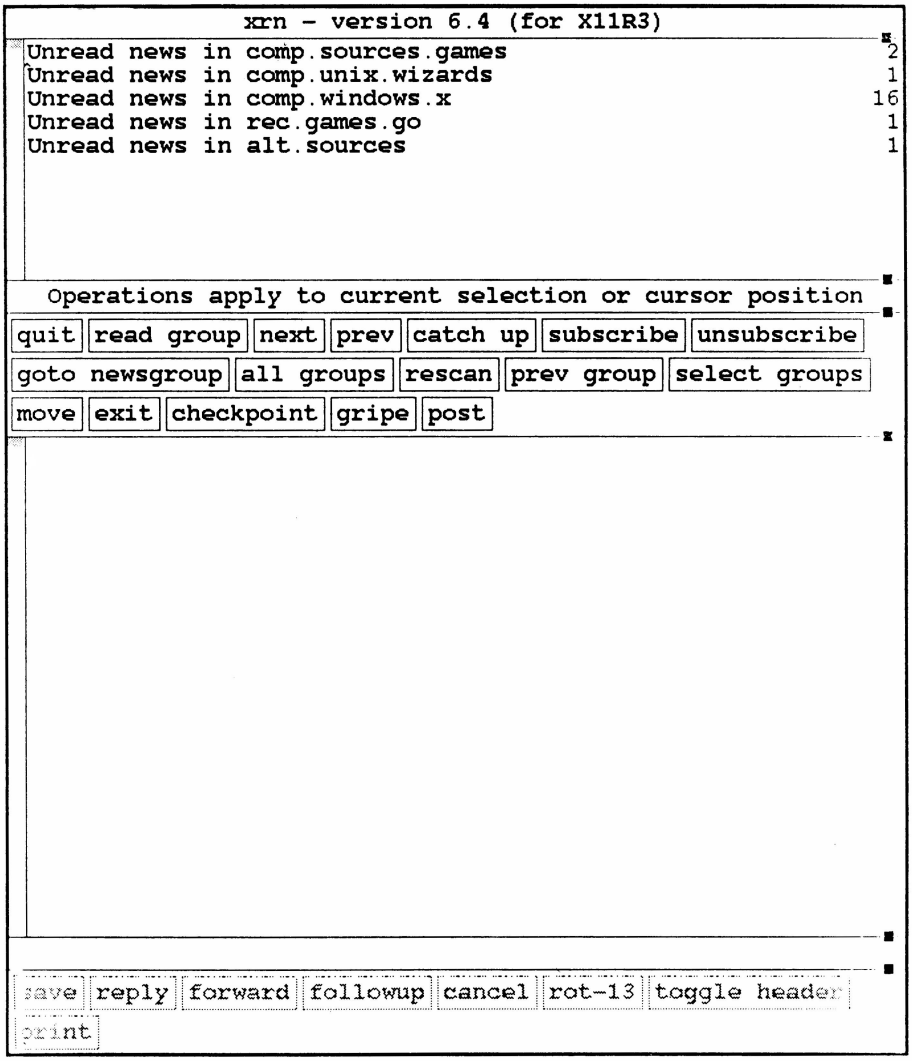


Figura 26.1 Aspetto dello schermo di **xrn**.

Alla fine del vostro articolo di nuova creazione, **rn** appende il contenuto del file **$HOME/.signature**; in questo file potete creare una firma personalizzata da apporre ai vostri invii in rete. La firma può avere qualsiasi lunghezza, ma ovviamente sono preferibili firme brevi.

La linea "Distribution:" limita la diffusione dell'articolo a una parte dell'intera rete; di solito viene indicata l'organizzazione di appartenenza, lo stato, la città e altre destinazioni. Dovrete consultare un utilizzatore abituale della rete news per conoscere l'elenco esatto.

USI E CONVENZIONI DI news

Diversamente dalle molte operazioni sbagliate che potete fare su UNIX, che coinvolgono tuttavia solo il vostro ambiente di login, o al massimo la vostra LAN, la spedizione di un articolo lo diffonde nella rete news mondiale; può essere copiato in migliaia di macchine in tutto il mondo e letto da moltissimi utilizzatori. Poiché la spedizione ha un effetto così vasto, è opportuno che limitiate la spedizione ad articoli importanti e di interesse generale; selezionate una diffusione la più ristretta possibile e rendete i vostri articoli brevi e concisi. Resistete all'impulso di contraddire un articolo che non condividete, usate una firma breve e pensate bene prima di fare un invio; naturalmente, non perdete il buon gusto in tutti gli aspetti del vostro messaggio. Molti dei nomi più stimati nell'ambiente UNIX non hanno mai inviato scritti in rete news, preferendo piuttosto condurre la loro corrispondenza privatamente via **mail**.

Se create un articolo per errore, risulta molto difficile arrestarne la distribuzione; tuttavia, l'opzione Cancel in **rn** consente di tentare di revocare un articolo. Se rileggete il vostro articolo e decidete che non dovrebbe essere distribuito, cancellatelo; solo il creatore e pochi gestori autorizzati della rete possono cancellare un articolo. La cancellazione può non avere effetto se un articolo si è oramai propagato largamente nella rete; notate che la revoca di un articolo provoca l'invio nella rete di un altro messaggio, raddoppiando così il traffico in rete.

Molti gruppi di notizie sono *moderati*, nel senso che quando inviate un articolo, viene in realtà inviato via mail a un moderatore designato per il gruppo; il moderatore legge l'articolo e decide se inviarlo nell'intera rete, o a una lista di distribuzione. In alcuni casi potete contattare il moderatore per revocare un articolo prima che venga distribuito.

Il sistema news è un libero servizio di portata internazionale; usatelo con discrezione.

26.4 La rete Internet e il comando ftp

Internet è una *Wide Area Network* (WAN), una rete di estensione mondiale ad alta velocità, originata come progetto di ricerca DARPA dell'U.S. Department of Defense. Negli ultimi anni è diventata di uso generale e il progetto DARPA ha ridotto la sua partecipazione dal momento che Internet è diventata autosufficente. Migliaia di macchine sono connesse con Internet e la

maggior parte può essere contattata usando l'indirizzo univoco Internet in quattro parti descritto nel Capitolo 24. Se la vostra macchina (o la vostra LAN) è connessa a Internet, potete usare applicazioni che lavorano come **rlogin** per connettervi con macchine in tutto il mondo. La sicurezza di Internet è naturalmente superiore che con la maggior parte delle LAN, e sono disponibili molti utili servizi. Larga parte dell'attività di Internet consiste nel trasferire mail e articoli news fra singoli utilizzatori, tuttavia attraverso la rete Internet è prevista anche la comunicazione real time. Consultatevi con il vostro esperto locale di reti per determinare se avete accesso alla rete Internet e con quali limitazioni d'uso.

Uno degli strumenti più utili di accesso a Internet è **ftp** (file transfer protocol), che consente a un utente il login con un'altra macchina attraverso la rete Internet e la copia di file tra i due sistemi. Il programma **ftp** funziona all'incirca come **cu**, ma con una velocità molto maggiore; inoltre, può trasferire tra le due macchine sia file ASCII, sia file binari. Il programma **ftp** può essere usato anche per connettere la vostra macchina ad altre macchine nella vostra LAN, come attraverso Internet, ma di solito viene principalmente usato per collegamenti Internet, in quanto **rlogin**, **rcp** e altri strumenti simili sono più efficienti e di più facile uso. Comunque, prima di utilizzare **ftp** con Internet potete sperimentarlo sulla vostra LAN; per contro, potete utilizzare **rlogin**, **rcp** e altri strumenti di uso nella LAN, in connessioni Internet, a condizione che il vostro host remoto consenta questo livello di accesso.

INDIRIZZAMENTO E LOGIN CON ftp

Il comando **ftp** accetta come suo argomento principale in linea di comando un nome di host, e tenta una connessione con l'elaboratore specificato; se il nome di host non viene indicato, **ftp** entra nel suo modo comandi e attende che venga selezionato un host. Il nome di host ha lo stesso formato usato da **rlogin**; può essere sia un nome logico di host, che un indirizzo Internet in quattro parti. Il nome logico di host viene ricercato nel file **/etc/hosts** (o nel suo equivalente **yp**); l'indirizzo numerico viene utilizzato direttamente. Se la vostra macchina non conosce l'elaboratore host che volete contattare, dovete utilizzare l'indirizzo numerico in quattro parti. Le linee di comando seguenti sono tutte accettabili:

```
$ ftp yoursys
$ ftp gateway.hid.com
$ ftp 127.0.0.1
$ ftp
```

Se **ftp** può connettersi allo host specificato, tenta il login con quella macchina, come nell'esempio:

```
$ ftp yoursys
Connected to yoursys.
220 yoursys FTP server (UNIX System V Release 4.0) ready.
Name (yoursys:giorgio):
```

Dovete fornire un identificatore di login e una password; se **ftp** vi conosce, può usare di default il vostro login corrente e voi potete accettarlo con RETURN, oppure introdurre un login differente. In quest'ultimo caso, di solito dovrete introdurre la vostra password e, se sarà accettata, verrete collegati con l'elaboratore remoto e **ftp** entrerà nel suo modo comandi; in caso contrario, **ftp** darà un messaggio d'errore. Per esempio:

```
$ ftp yoursys
Connected to yoursys.
220 yoursys FTP server (UNIX System V Release 4.0) ready.
Name (yoursys:giorgio): nouser
530 User nouser access denied.
Login failed.
ftp>
```

In ambedue i casi, verrete lasciati in modo comandi.

Per uscire dal programma **ftp**, chiudere la connessione con lo host remoto e ritornare in shell, introducete **quit** al prompt **ftp>**, come nell'esempio:

```
ftp> quit
221 Goodbye.
$
```

Molti host Internet consentono di accedere con **ftp** alla macchina e copiare file nel vostro sistema; le locazioni di host e i file disponibili sono spesso comunicati nella rete o con altri mezzi. Poiché anche utenti sconosciuti possono cercare di raccogliere file con **ftp**, molti host che consentono questo accesso prevedono la possibilità di un login anonimo; i permessi per **ftp** anonimi sono di solito notevolmente limitati rispetto ai normali login, ma consentono normalmente di spostarsi in un ridotto albero di directory, di copiare file e usare pochi semplici comandi. Per usare un **ftp** anonimo, usate l'identificatore **anonymous** al prompt di login e introducete il vostro nome e il nome dalla vostra macchina alla richiesta di password, come nell'esempio:

```
$ ftp yoursys
Connected to yoursys.
220 yoursys FTP server (UNIX System V Release 4.0) ready.
Name (yoursys:giorgio): anonymous
331 Guest login ok, send ident as password.
Password:
230 Guest login ok, access restrictions apply.
ftp>
```

Usate una password nella forma:

```
login@macchina.rete
```

dove *login* è il vostro identificatore di login e *macchina.rete* è il vostro indirizzo logico Internet o di **mail**, oppure l'indirizzo numerico in quattro parti. Lo scopo di questa "password" di login anonimo è quello di consentire al gestore dell'elaboratore host di avere una traccia degli utilizzatori "anonimi" e di poterli contattare, se necessario.

Quando il collegamento con la macchina remota ha successo, non venite connessi in un normale shell di quella macchina, ma in uno speciale processo server di **ftp** che è completamente diverso da uno shell. Questo processo non supporta l'intero insieme di comandi UNIX, ma solo un limitato gruppo di comandi del modo comandi di **ftp**.

MODO COMANDI DI ftp

Al prompt **ftp>** sono disponibili molti comandi per controllare la vostra sessione e copiare file tra le due macchine. Il comando più importante è **quit**, che chiude la connessione con l'elaboratore remoto prima di ritornare in shell; il comando **!** seguito da una linea di comando esegue quella linea di comando sulla macchina locale; il comando **!** senza linea di comando invoca uno shell interattivo e, quando uscite da quello shell, **ftp** ritorna nel suo modo comandi; il comando **help** o **?** fa ottenere un elenco di tutti i comandi **ftp**. La forma:

```
ftp> help cmd
```

fornisce una breve spiegazione del comando di nome *cmd*.

Il programma **ftp** consente di cambiare directory in host remoto e di listarne il contenuto. Per cambiare directory in host remoto usate il comando **cd** di **ftp**; usate il comando **ls** per visualizzare il contenuto dei directory; per cambiare directory corrente nella macchina locale dovete usare il comando **lcd** (local cd). Poiché questi comandi possono essere interni a **ftp**, possono fornire risultati leggermenti diversi da quelli che potreste aspettarvi d'abitudine, per esempio:

```
ftp> cd pub/src
250 CWD command successful.
ftp> pwd
257 "pub/src" is current directory
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (127.0.0.1,2390) (0 bytes).
mystuff
```

```
patches
contrib
226 ASCII Transfer complete.
170 bytes received in 0.08 seconds (2.1 Kbytes/s)
ftp>
```

Come potete vedere, **pwd** visualizza il directory corrente.
Linee della forma:

```
200 PORT command successful.
```

compaiono di frequente in uscita di **ftp**; si tratta di messaggi da parte di **ftp** ogni volta che crea o interrompe canali di comunicazione separati all'interno della connessione, per trasferire i dati.

TRASFERIMENTO DI FILE CON ftp

Il programma **ftp** fornisce diversi comandi per il trasferimento di file tra due macchine; potete copiare sia file binari (eseguibili, oppure compressi), che file di testo (ASCII). Il programma ha due modi per i due tipi di file; usate i comandi **binary** e **ascii** per passare dall'uno all'altro, come nell'esempio:

```
ftp> binary
200 Type set to I:
ftp> ascii
200 Type set to A.
ftp>
```

Il default è il modo ASCII, assicuratevi quindi di predisporre il modo binario prima di trasferire file binari. Comunque, il modo binario di solito copia correttamente anche file ASCII, pertanto potete predisporre il modo binario per tutti i trasferimenti, specialmente se non conoscete il tipo del file.

Per copiare un file da una macchina remota, utilizzate i comandi **get** oppure **mget** (multiple get). Il comando **get** accetta come argomento il nome del file da copiare dalla macchina remota alla macchina locale; per esempio:

```
ftp> get nome.file
```

Il comando **mget** consente di copiare file multipli determinati mediante la familiare regola dei metacaratteri; per esempio:

```
ftp> mget *xbm*
mget xwatch.xbm?
```

Il comando **mget** vi richiederà conferma per ciascun file da copiare; se rispondete **y**, il file verrà copiato nel directory corrente della macchina locale, come mostrato nell'esempio:

```
ftp> mget *xbm*
mget xwatch.xbm? y
200 PORT command successful.
150 ASCII data connection for xwatch.xbm (127.0.0.1) (1886 bytes).
226 ASCII Transfer complete.
local: xwatch.xbm remote: xwatch.xbm
1913 bytes received in .11 seconds (17 Kbytes/s)
mget xwatch_empty.xbm?
```

Sono state visualizzate molte informazioni, ma il trasferimento del file è stato eseguito; come potete vedere, **mget** può trasferire i file molto velocemente (17 kB per secondo in questo esempio).

Se volete copiare diversi file tutti in un colpo, potete disabilitare la richiesta di conferma per ciascun file, col comando

```
ftp> prompt
```

Per riabilitare la richiesta di conferma dovete eseguire di nuovo il comando **prompt**.

Per trasferire file dalla macchina locale alla macchina remota, viene usata la stessa procedura; **put** accetta un argomento col nome di un file, **mput** accetta un argomento composto da una lista di nomi o da una specificazione con metacaratteri; ambedue i comandi copiano il o i file dalla macchina locale a quella remota. I comandi **put** e **mput** utilizzano i colandi **binary** e **prompt** come **get** e **mget**. Ovviamente, potrebbe non esservi permesso creare file sulla macchina remota; in ogni caso dovreste utilizzare questi comandi con attenzione.

Il comando **ftp** include molte caratteristiche progettate per esperti, compresa la capacità di creare macro, ovvero sequenze di comandi usati di frequente, identificate da un nome. Potete inoltre scegliere se fare eseguire a **ftp** l'espansione dei metacaratteri, oppure se fare accettare i nomi alla lettera; potete anche controllare esplicitamente la creazione dei canali di comunicazione (i comandi PORT) e altre caratteristiche. La maggior parte degli utenti usa raramente questi servizi evoluti. Se volete maggiori informazioni consultate la man page **ftp(1)**.

26.5 Lo sviluppo del software

L'area di lavoro più ricca all'interno del sistema UNIX è lo sviluppo di software. Il linguaggio di programmazione C e l'eccellente ambiente di suppor-

to rendono il sistema molto popolare tra i progettisti software. Nel sistema UNIX quasi tutti gli strumenti e la maggior parte del kernel sono scritti in linguaggio C. Il linguaggio C negli ultimi tempi ha preso campo anche in ambienti diversi dal sistema UNIX e oggi compilatori e strumenti C sono disponibili in quasi tutti i sistemi operativi. Il C è oramai universalmente considerato uno dei migliori linguaggi in uso per la programmazione di sistema, perché consente all'implementatore una grandissima flessibilità, che copre caratteristiche che vanno da quelle di un codice a livello assembly a quelle di un vero linguaggio di alto livello, con concetti di programmazione astratta.

La versione Standard C Development Environment ufficiale di SVR4 implementa un nuovo compilatore di alta efficienza, capace di produrre codice che di solito viene eseguito dal 10 al 20% più velocemente rispetto alle precedenti release System V. In aggiunta, i nuovi strumenti supportano le recenti specifiche ANSI per il linguaggio C in una maniera che è particolarmente compatibile con il software precedente.

Per ridurre i tempi di compilazione (e di gestione) necessari nello sviluppo di grandi progetti potete usare il comando **make**. Il programma **make** usa un elenco predefinito di dipendenze fra i file sorgente e le applicazioni oggetto per determinare la minima ricompilazione occorrente per rigenerare l'applicazione quando una qualsiasi parte è stata cambiata. La data di modifica del file da parte del sistema viene usata per determinare quali file sono più recenti dei file da loro dipendenti; quindi **make** produce questi moduli e ricostruisce l'intera applicazione. In complessi progetti di sviluppo di software, costituiti da molti file sorgente sotto diversi directory, sviluppati da progettisti diversi, **make** può ridurre in maniera significativa i costi di sviluppo.

Il programma **make** ha molti impieghi oltre che nello sviluppo di software; può essere utile in ogni applicazione in cui l'output dipende da numerosi file in input, o dove per la creazione dell'output sono possibili differenti soluzioni. Il programma **make** viene usato spesso per la realizzazione di documentazione; può essere impiegato sia in piccoli che in grandi progetti.

Il pacchetto di sviluppo standard include due strumenti per l'analisi lessicale e per lo sviluppo di programmi dalla descrizione linguistica degli effetti che il programma dovrà avere. I programmi **lex** (lexical analysis) e **yacc** (yet another compiler-compiler) sono strumenti di sviluppo di quarta generazione, che consentono ai programmatori di sviluppare macchine a stato lessicale e compilatori, descrivendo *come* le applicazioni debbono agire, anziché *cosa* debbono fare. Si tratta di strumenti progettati per esperti e richiedono una considerevole esperienza per essere usati con successo.

Compilatori FORTRAN e altri linguaggi e strumenti di sviluppo sono disponibili per sistemi UNIX da diverse fonti. Quasi ogni strumento che possiate desiderare è disponibile per sistemi UNIX, inclusi procedimenti statistici, pacchetti grafici, fogli elettronici e calcolatori a pieno video per numerosi terminali e display.

Il C Development Set ufficiale è di solito un'aggiunta al sistema base SVR4, con un costo addizionale, ma il popolare **gnu C** è disponibile gratuitamente dalla Free Software Foundation. Viene distribuito in forma sorgente, per cui dovete avere accesso a un compilatore C per ottenere la forma eseguibile, ma è un'alternativa affidabile e conveniente alla versione standard ufficiale. Il pacchetto **gnu C** include molti strumenti addizionali di sviluppo, comprese versioni degli strumenti **c++** e **yacc**, inoltre supporta sia il linguaggio C ANSI che l'originale sintassi "K&R".

Molti fornitori supportano diversi sistemi di programmazione per l'ambiente UNIX, come LISP, ADA, Prolog e anche COBOL. Sono inoltre disponibili anche linguaggi database query, interpreti tipo shell, sistemi esperti, e strumenti di sviluppo di interfacce utente. Per contro, può essere molto difficile trovare un'implementazione di BASIC per SVR4.

Lo sviluppo di software oltrepassa lo scopo di questo libro, tuttavia esiste una vasta letteratura che tratta l'argomento; se avete un interesse nella programmazione, troverete che il sistema UNIX è particolarmente adatto a questo compito.

26.6 Source Code Control System

Una parte dell'ambiente di sviluppo software di utilità generale per gli utenti è il *Source Code Control System* , o SCCS, il sistema di gestione del codice sorgente. In origine progettati per la gestione del software, gli strumenti SCCS consentono anche la gestione di documenti e forniscono servizi per la gestione delle versioni per qualsiasi documento di testo. Poiché i documenti cambiano nel tempo, può essere difficile archiviare tutte le versioni, tenendo conto delle relative date, di commenti e del documento da cui hanno avuto origine; il sistema SCCS semplifica questo procedimento, gestendo un file unico che contiene tutte le modifiche a un documento. Voi potete in ogni momento aggiungere al file le versioni cambiate, o *delta*, senza distruggere le versioni precedenti; in seguito, potete recuperare, con *get*, qualsiasi versione dal file principale, per ulteriori modifiche, stampe, o compilazioni. Il sistema SCCS non fa parte del sistema operativo UNIX di base, ma è incluso nel C Compilation System.

Il sistema SCCS lavora mantenendo i *numeri di versione* di ogni nuovo delta aggiunto al database. Quando viene creato un file, gli viene attribuito il numero di versione 1.1; quanto riprendete quella versione dal database, la aggiornate e la reinserite nel database, a questa nuova versione viene attribuito il numero 1.2; la volta successiva il numero di versione diventerà 1.3 e così di seguito. In questo modo, tutte le precedenti versioni restano disponibili e le potete richiamare in qualunque momento. SCCS fornisce anche un meccanismo per associare commenti di annotazioni alle versioni, per mantenere una descrizione dei motivi delle modifiche.

Il database SCCS di un file ha come nome lo stesso nome del file, preceduto da **s.** (esse punto); per esempio, un file di nome **mydoc** avrà un database SCCS col nome **s.mydoc**. Se il vostro sistema è soggetto alla limitazione a 14 caratteri dei nomi di file, dovrete attribuire nomi di 12 caratteri, o meno, ai vostri documenti originali.

Per creare un nuovo database SCCS per un file, usate il comando **admin**; la linea di comando deve contenere l'opzione **-n** (new), l'opzione **-i** (input) seguita dal nome del file e il nome del database SCCS; per esempio:

```
$ ls
mydoc
$ admin -n -imydoc s.mydoc
No id keywords (cm7)
$ ls
mydoc        s.mydoc
$
```

Notate che fra l'opzione **-i** e il nome del file non sono ammessi spazi.

Il file **s.mydoc** contiene il contenuto del file originale, in formato SCCS; il messaggio "No id keywords" è un semplice avviso, non un errore. Le keyword verranno trattate in seguito. Il comando **admin** è uno strumento di uso generale nella gestione di file SCCS; prevede un grande numero di funzioni addizionali e di argomenti in linea di comando, che non vengono qui descritti. Le sue caratteristiche includono la capacità di fornire rapporti della storia dei delta di un file e la possibilità di ricostruire un database danneggiato.

Per richiamare il testo originale dal database, usate il comando **get** col nome del file SCCS come argomento, per esempio:

```
$ rm mydoc
$ get s.mydoc
1.1
12 lines
No id keywords (cm7)
$ ls
mydoc        s.mydoc
$
```

Questa operazione crea un file col nome originale, che contiene il contenuto del file originale; il contenuto del database SCCS non viene alterato.

Per default, **get** richiama l'ultima versione esistente nel database; potete richiamare una precedente versione, se esiste, usando l'opzione **-r** (release), seguita dal numero della versione; per esempio:

```
$ get -r1.1 s.file
```

Questa operazione consente di ottenere un'immagine del file al solo scopo di stamparlo, compilarlo, o scopi simili; non è possibile reinserire questo file nel database SCCS come delta. In pratica, il file creato non possiede i permessi di scrittura, per ricordarvi che non potete cambiarlo, e neanche il database SCCS è scrivibile; per eseguire cambiamenti dovete sempre utilizzare strumenti SCCS, che superano i controlli dei diritti di scrittura. Se volete richiamare un file, aggiornarlo con editor e reinserirlo nel database dopo le modifiche, dovete usare in **get** l'opzione **-e** (edit); esempio:

```
$ get -e s.mydoc
1.1
new delta 1.2
12 lines
$ ls
mydoc        p.mydoc        s.mydoc
$
```

In questo caso il file **mydoc** è accessibile in scrittura; il file aggiuntivo **p.mydoc** creato in questa fase è un file di controllo che informa il sistema SCCS che un file è stato estratto per l'editing. La gestione delle versioni richiede che un solo utente alla volta venga autorizzato ad aggiornare una data versione, perciò deve essere proibito l'accesso a una versione che qualcun altro sta già aggiornando; abbiate cura di non cancellare mai i p.file, perché questo creerebbe confusione nelle operazioni di SCCS.

Se volete creare una nuova versione del file, anziché quella che incrementa il numero minore di versione, usate **get** con l'opzione **-r** seguita dal numero di versione che volete creare; per esempio:

```
$ get -r2.0 -e s.mydoc
1.2
new delta 2.1
12 lines
$
```

La versione successiva adesso diventa 2.1, invece di 1.3.

Per reintrodurre una versione aggiornata nel database SCCS, usate il comando **delta**; potete usare questo comando solo se avete in precedenza richiamato il file per l'editing; ecco un esempio:

```
$ ls
mydoc     p.mydoc     s.mydoc
$ delta s.mydoc
comments? incorporati i suggerimenti del capo.
No id keywords (cm7)
1.2
1 inserted
0 deleted
```

```
12 unchanged
$ ls
s.mydoc
$
```

Come argomento di **delta** dovete indicare il nome del file SCCS, **s.mydoc** nell'esempio; come potete vedere, il file originale scrivibile e il p.file sono stati ambedue cancellati, ma il database SCCS è stato aggiornato e contiene l'ultima versione. Il messaggio di richiesta "comments?" consente di introdurre annotazioni alla nuova versione, e in questo modo **delta** contiene la storia dei cambiamenti apportati.

I delta di SCCS possono costituire una gerarchia di versioni complessa. Per esempio, se nel vostro database SCCS avete le versioni 1.1, 1.2, 2.1 e 2.2, potete richiamare la versione 1.1, modificarla e reinserirla come delta nel database; il sistema SCCS è a conoscenza della presenza delle altre versioni e, invece di rimpiazzare la versione 1.2, crea la versione 1.1.1. Quando voi successivamente modificate la versione 1.1.1 e la reinserite, la nuova versione diventerà 1.1.2; se modificate la versione 1.2 e la reinserite, la nuova versione diventerà 1.2.1. Non esistono limiti pratici al numero delle versioni possibili.

Il sistema SCCS supporta inoltre delle parole chiave, o *keyword*, stringhe speciali che voi potete introdurre nel file. Quando richiamate una versione dal database, SCCS automaticamente sostituisce le stringhe con i loro valori correnti; questo permette di includere il numero della versione nel file stesso. Per esempio, la stringa **%M%** (percento M percento) viene rimpiazzata dal nome del file, la stringa **%R%** viene rimpiazzata dal numero maggiore della versione e la stringa **%I%** viene rimpiazzata da un'utile stringa d'identificazione di SCCS, contenente la data, il numero della versione e altre informazioni. La man page **get(1)** contiene l'elenco completo delle keyword; potete usare queste stringhe nel vostro testo, se lo ritenete necessario. Notate che le parole chiave non vengono rimpiazzate quando richiamate una versione con l'opzione **–e**, ma solo quando il file non può essere reinserito come delta.

26.7 La crescente influenza del sistema UNIX

Oltre agli strumenti di sviluppo di software, il sistema UNIX stesso si è dimostrato una sorgente estremamente fertile per fornire idee e algoritmi ai progettisti software. La maggior parte del software adatto ai microcalcolatori moderni ha derivato liberamente molte idee apparse inizialmente nel sistema UNIX. Per esempio, il sistema operativo OS/2 ha estratto molti concetti dal sistema UNIX, dal suo kernel alla sua versione di shell, **cmd.exe**. Per contro, X Window System ha adottato molte idee dall'interfaccia utente Macintosh e dai prodotti Microsoft Windows. È opinione comune che la co-

noscenza del sistema UNIX renda più semplice l'utilizzo di quasi tutti gli altri sistemi, anche se, talvolta, rende difficile accettare che non sia possibile trasportare in altri sistemi operativi gli strumenti e le procedure di UNIX.

26.8 Considerazione finale

Finalmente, siete arrivati alla fine del libro. Se avete letto con attenzione e appreso il significato dei concetti presentati in tutti i capitoli siete sicuramente un utente competente che può, in modo indipendente, utilizzare il sistema UNIX per le proprie esigenze, ora e in futuro. Ci auguriamo che siate diventati entusiasti del sistema UNIX come molti altri utenti dislocati in ogni parte del mondo. Ricordatevi che l'apprendimento del sistema UNIX è un processo continuo e perfino il più grande esperto può conoscere solo una parte del ricco ambiente UNIX. Continuate quindi a incrementare la vostra conoscenza di UNIX, sperimentando nuove idee e procedure, e nello stesso tempo insegnate ai principianti i concetti base perché possano anche loro diventare presto esperti di questo sistema operativo.

Nessun libro può esaurire la trattazione di tutti gli argomenti relativi a UNIX in modo dettagliato o anche esaminare adeguatamente l'intero set di comandi e le implicazioni connesse. In questo testo abbiamo trattato l'insieme base dei comandi con un livello di dettaglio abbastanza accurato da far diventare un utente moderatamente esperto di microcalcolatori un utente e gestore competente di UNIX.

Indice analitico

La McGraw-Hill pubblica in tutto il mondo centinaia di libri di informatica per lo studio, la professione, il tempo libero. La produzione in lingua italiana comprende:

Come usare il calcolatore senza fatica

88 386 0155-0 D. Gookin, A. Rathbone *Usare il personal senza fatica e vivere felici*
88 386 0192-5 D. Gookin *Usare WordPerfect senza fatica e vivere felici*
88 386 0233-6 D. Pogue *Usare il Macintosh senza fatica e vivere felici*
88 386 0262-X A. Rathbone *Usare Windows 3.1 senza fatica e vivere felici*
88 386 0260-3 G. Harvey *Usare Lotus 1-2-3 senza fatica e vivere felici*
88 386 0261-1 G. Harvey *Usare Excel 4 senza fatica e vivere felici*
88 386 0216-6 D. Gookin *Usare DOS 6 senza fatica*

Guide complete

88 386 0156-9 N. Johnson, *Guida completa AutoCAD Versione 10 inglese*
88 386 0229-8 N. Johnson, *Guida completa AutoCAD Versione 11 inglese*
88 386 0082-1 H. Schildt, *Guida completa C ANSI C e C++*
88 386 0066-X J.D. Carrabis, *Guida completa dBASE III PLUS*
88 386 0182-8 G.T. Le Blond, W.B. Le Blond, B. Heslop, *Guida completa dBASE IV*
88 386 0081-3 K. Jamsa, *Guida completa DOS*
88 386 0147-X K. Jamsa, *Guida completa DOS Versione 3.30 italiana*
88 386 0228-X K. Jamsa, *Guida completa DOS 5*
88 386 0140-2 M.S. Matthews, S. Seymour, *Guida completa Excel 4 per windows*
88 386 0065-1 M. Campbell, *Guida completa Lotus 1-2-3*
88 386 0161-5 M. Campbell, *Guida completa Lotus 1-2-3 Versione 2.2 inglese*
88 386 0163-1 M. Campbell, *Guida completa Lotus 1-2-3 Versione 3 inglese*
88 386 0099-6 E. Alderman, *Guida completa Microsoft Word*
88 386 0130-5 J. Keogh, *Guida completa Paradox 2.0*
88 386 0110-0 Y. McCoy, *Guida completa Quattro*
88 386 0124-0 H. Schildt, *Guida completa Turbo C*
88 386 0078-3 S.K. O'Brien, *Guida completa Turbo Pascal*
88 386 0162-3 S.K. O'Brien, *Guida completa Turbo Pascal Versione 5.5 inglese*
88 386 0255-7 S.K. O'Brien, *Guida completa Turbo Pascal Versione 6 inglese*
88 386 0181-X S. Coffin, *Guida completa UNIX System V Release IV*
88 386 0157-7 M. Holt, R. Birmele, *Guida completa Ventura 2*
88 386 0123-2 G. Todd, *Guida completa Videoscrittura 4*
88 386 0187-9 T. Sheldon, *Guida completa Windows Versione 3.0 italiana*
88 386 0242-5 T. Sheldon, *Guida completa Windows Versione 3.1*

Guide tascabili

88 386 0246-8 V. Wolverton, *Guida rapida Hard Disk Tecniche di gestione*
88 386 0252-2 Rinearon, *Guida rapida Microsoft Word 5.5*
88 386 0244-1 V. Wolwerton, *Guida rapida MS-DOS 5*
88 386 0245-X K. Jamsa, *Guida rapida MS-DOS Batch file*
88 386 0247-6 K. Jamsa, *Guida rapida MS-DOS QBasic*
88 386 0251-4 J.L. Viescas, *Guida rapida Norton Utilities versione 5*
88 386 0250-6 C. Townsend, *Guida rapida PCTools Versione 6*
88 386 0100-3 H. Schildt, *Guida tascabile C*
88 386 0089-9 M. Liskin, *Guida tascabile dBASE III Plus*
88 386 0178-X *M. Liskin, Guida tascabile dBASE IV*
88 386 0102-X M.K. Jamsa, *Guida tascabile DOS Versione 3.30*

88 386 0105-4 H. Schildt, *Guida tascabile Turbo C*
88 386 0104-6 K. Jamsa, *Guida tascabile Turbo Pascal 4.0*
88 386 0098-8 M. Campbell, *Guida tascabile Lotus 1-2-3*
88 386 0179-8 M. Campbell, *Guida tascabile Lotus 1-2-3 Versioni 2 e 3 inglesi*
88 386 0108-9 E. Alderman, *Guida tascabile Microsoft Word*
88 386 0107-0 E. Jones, *Guida tascabile Paradox*
88 386 0106-2 S. Cobb, *Guida tascabile Quattro*
88 386 0127-5 K. Jamsa, *Guida tascabile OS/2 API*
88 386 0103-8 K. Jamsa, *Guida tascabile OS/2*
88 386 0122-4 C. Gilbert, *Guida tascabile Wordstar Versione 5.5*

Informatica professionale

88 386 0202-6 M. Salin, *Applicazioni statistiche con SPSS Versione 4.01*
88 386 0220-4 J. Occhiogrosso, *Clipper 5.0 Le librerie (libro + disco)*
88 386 0186-0 E. Teja, L. Johnson, *Computer Graphics con PC IBM e PS/2 Hardware e software*
88 386 0160-7 Price Waterhouse, *Computer Virus Analisi, prevenzione, metodi di difesa*
88 386 0169-0 P.A. Darnell, P.E. Margolis, *C Manuale di programmazione Linguaggio e tecniche di ingegnerizzazione del software*
88 386 0153-4 D.M. Kalman, *Il manuale dei linguaggi dBASE*
88 386 0238-7 P. Sanasi *Il sistema informativo in rete Analisi, implementazione, gestione*
88 386 0167-4 R.W. Ridinghton, S. Tucker, *Lotus 1-2-3 macro Versione 2.2 inglese e italiana*
88 386 0175-5 H. Schildt *L'arte della programmazione in C*
88 386 0165-8 K.W. Christopher, B.A. Feigenbaum, S.O. Saliga, *MS-DOS Manuale di programmazione*
88 386 0094-5 E. Jacobucci, *OS/2 Manuale di programmazione*
88 386 0224-7 R. Pressman, *Principi di ingegneria del software*
88 386 0185-2 M.G. Naugle *Reti locali*
88 386 0136-4 B. Eckel *Programmare in C++*
88 386 0154-2 R. Spence, *Programmare in Clipper*
88 386 0219-6 R. Spence, *Programmare in Clipper Versione 5.01*
88 386 0259-X R. Spence, *Programmare in Clipper Versione 5.01 Seconda edizione*
88 386 0173-9 R. Rocchetti, A. Moroni, *Programmare in Excel 4 e Q+E per Windows*
88 386 0199-2 D. Ince *Programmazione a oggetti in C++ Metodi di ingegneria del software*
88 386 0059-7 M. Liskin, *Programmazione avanzata in dBASE III Plus*
88 386 0256-5 R. Salcedo, *Programmare in Paradox 3.5*
88 386 0241-7 B. Livingston, *I segreti di Windows 3.1*

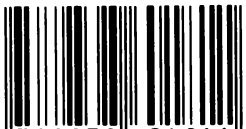
UNIX System V/La guida completa

Il sistema operativo UNIX, particolarmente diffuso nelle università per le notevoli potenzialità offerte in ambienti di lavoro diversi e con le aspettative più disparate, sta raccogliendo un consenso sempre più unanime anche in ambito professionale, dove sta diventando l'ambiente operativo più utilizzato.

La nuova edizione del volume di Stephen Coffin lo conferma come testo di riferimento per chiunque lavori con questo sistema operativo, sia perché presenta le più complete e aggiornate informazioni sulle nuove caratteristiche della release 4 di UNIX System V, sia perché offre l'occasione di approfondire aspetti significativi del sistema.

Oltre a una vasta trattazione dei fondamenti di UNIX, il volume comprende alcuni capitoli sull'interfaccia X Windows, gli shell C e Korn, l'utilizzo in rete di UNIX con Ethernet e NFS.

ISBN 88-386-0181-X



Lire 75.000 (i.i.)

Guida completa

UNIX System V

0181-X