Fondamenti di GNU/Linux

basato su GNU/Linux Debian

di Michele Marcucci (http://www.michelem.org)

liberamente tratto da "Appunti di informatica libera" di Daniele Giacomini (http://a2.swlibero.org)

Indice Generale

- Directory
- Collegamenti (link)
- Nomi dei file
- Permessi
- Permessi espressi in forma di stringa
- Permessi espressi in forma numerica
- S-bit
- Date
- Utenza e accesso
- adduser o useradd
- exit
- Interpretazione dei comandi
- File globbing
- Tilde
- Variabili di ambiente
- Ridirezione e pipeline
- Ridirezione dello standard input
- Ridirezione dello standard output
- Ridirezione dello standard error
- Pipeline
- Comandi e programmi di servizio di uso comune
- Interpretazione della sintassi
- Organizzazione tipica
- Programma o eseguibile
- L'ABC dei comandi GNU/Linux
- ls, cd, mkdir, cp, ln, rm, mv, cat

cap 3) Documentazione

- Testo puro
- Pagine di guida
- /etc/man.config
- $ man
- $ whatis
- $ apropos
- Documentazione allegata ai pacchetti
- HOWTO
- FAQ
- LPD
- Ricerche nella rete

cap 4) LILO: Intoduzione

- Organizzazione essenziale

- Installazione del meccanismo di caricamento del sistema operativo
- /etc/lilo.conf
- # lilo
- LILO su un disco differente
- Boot prompt

cap 5) Kernel LINUX

- Ricompilazione del kernel
- Kernel monolitico
- Kernel modulare
- Compilazione del kernel in una distribuzione GNU/Linux Debian
- Elementi della configurazione
- Code maturity level options
- Loadable module support
- Processor type and features
- General setup
- Parallel port support
- Plug and Play configuration
- Block devices
- Multi-device support (RAID and LVM)
- Networking options
- ATA/IDE/MFM/RLL support
- SCSI support
- Network device support
- Character devices
- Filesystems
- Console drivers
- Sound
- USB support
- Come fare per configurare correttamente il kernel che si vuole compilare

cap 6) Moduli

- Gestione dei moduli
- Funzionamento in breve
- Aspetto e collocazione
- Caricamento guidato
- Parametri
- Gestione automatica
- # insmod
- # rmmod
- $ lsmod
- # depmod

- # modprobe
- # kerneld
- Configurazione dei moduli
- alias
- options

cap 7) Introduzione ai processi di elaborazione

- Tabella dei processi
- /proc/
- Nascita e morte di un processo
- Core dump
- Comunicazione tra processi
- Segnali
- Pipe
- IPC di System V
- Scheduling e priorita`
- Privilegi dei processi

cap 8) Procedura di inizializzazione del sistema (System V)

- Init
- # init
- /etc/inittab
- Script della procedura di inizializzazione del sistema
- Procedura di attivazione e disattivazione dei servizi
- Script di avvio e interruzione di un servizio
- Collegamenti simbolici per ogni livello di esecuzione

cap 9) Situazione dei processi

- Process status
- Intestazioni
- $ ps
- $ pstree
- $ top
- Accesso ai file
- # fuser
- Informazioni riepilogative
- $ uptime
- $ free
- Processi e shell
- Controllo dei job di shell
- Processi in primo piano e processi sullo sfondo
- Avvio di un job sullo sfondo

- Sospensione di un job in primo piano
- jobs
- Riferimenti ai job
- fg
- bg
- kill
- Cattura dei segnali
- trap

cap 10) VI: cenni sull'utilizzo

- VI
- Avvio
- Modalita' di funzionamento
- Posizione attiva
- Moltiplicatori
- Inserimento
- Modificatori
- Cancellazione
- Sostituzione
- Copia e spostamento di porzioni di testo
- Copia e spostamento con nome
- Ricerche
- Ricerche e sostituzioni
- Annullamento dell'ultimo comando
- Caricamento, salvataggio e conclusione
- Variabili
- Comandi particolari
- File di configurazione
- Problemi di portabilita`

cap 11) Applicativi distribuiti in forma sorgente o compilata

- Struttura tipica di un pacchetto sorgente
- Documentazione necessaria alla compilazione
- ./configure
- File-make
- Fasi tipiche di una compilazione e installazione
- Problemi
- Installazione di programmi gia` compilati
- Scelta della piattaforma
- Eseguibili e variabili di ambiente
- Librerie dinamiche
- $ ldd

- File di differenze, o patch
- Aggiornamento delle librerie standard

cap 12) Pacchetti Debian

- Caratteristiche dei pacchetti Debian
- Priorita` di un pacchetto
- Dipendenze secondo i pacchetti Debian
- Stato di un pacchetto
- Disponibilita` di un pacchetto
- Stratificazione degli strumenti di gestione dei pacchetti Debian
- Gestione elementare attraverso gli strumenti fondamentali
- Gestione piu` evoluta dei pacchetti: organizzazione di una copia della distribuzione
- APT a livello essenziale
- Ricerca dei file che apparentemente non appartengono ad alcun pacchetto
- Pacchetti Debian sorgenti

cap 13) IPv4: configurazione delle interfacce di rete

- Configurazione delle interfacce di rete
- Loopback
- Ethernet
- Connessioni punto-punto
- Configurazione delle interfacce di rete con un sistema GNU/Linux
- # ifconfig
- Alias IP
- IPv4: instradamento locale
- Rete locale
- Loopback
- Ethernet
- # route
- Utilizzo di Route
- Verifica di un instradamento
- $ ping
- ARP
- # arp

cap 14) Indirizzi e nomi

- Configurazione del tipo di conversione
- /etc/host.conf
- Variabili di ambiente
- File per la conversione
- /etc/hosts

- /etc/resolv.conf

cap 15) Servente HTTP: Apache

- Visione generale
- Struttura di directory
- Avvio e conclusione dell'attivita` del servente
- Configurazione essenziale con httpd.conf
- Impostazioni varie
- Identificazione
- Utenti
- Collocazione e denominazione di file e directory
- Configurazione delle risorse con srm.conf
- Documenti HTML
- Indici e file di informazioni
- Tipi di file
- Directory alias
- Gestori specifici in base all'estensione
- File di messaggi
- Controllare l'accesso con access.conf
- Sezioni di controllo
- Sezione “Directory”
- Sezione “Limit”
- Sezione “Location”
- Controllare l'accesso con .htaccess
- Considerazioni sulla sicurezza
- Utilizzo del sistema di autenticazione
- Configurazione
- Siti virtuali

cap 16) Servente Samba

- I demoni del servente
- Attivazione del servente Samba
- Configurazione di un servente Samba
- Livello di sicurezza “share”
- Livello di sicurezza “user”
- Livello di sicurezza “server”
- Livello di sicurezza “domain”
- smb.conf: sezioni generiche di condivisione
- smb.conf: sezione “homes”
- smb.conf: sezione “printers”
- Programmi ausiliari per un servente Samba

cap 1) Storia breve del software libero

L'esigenza di liberta` nel settore del software e` sempre stata sentita. Ma se oggi questo tipo di software rappresenta concretamente una scelta possibile, lo si deve all'azione di persone che con impegno hanno agito, legalmente, verso il raggiungimento di questo obiettivo.

BSD

I primi utenti di UNIX sono state le universita`, a cui in particolare questo sistema operativo veniva fornito a costo contenuto, con i sorgenti, ma senza alcun tipo di supporto tecnico, ne' alcuna garanzia. Proprio questa assenza di sostegno da parte della casa che lo aveva prodotto, stimolava la cooperazione tra gli utenti competenti, in pratica tra le universita`.

Il maggior fermento intorno a UNIX si concentro` presso l'universita` della California a Berkeley, dove a partire dal 1978 si comincio` a distribuire una variante di questo sistema operativo: BSD (Berkeley software distribution).

Per questo software, nacque una licenza d'uso che rimane il progenitore della filosofia del software libero: la licenza BSD.

Per molto tempo, la variante BSD di UNIX rimase relegata all'ambito universitario o a quello di aziende che avevano acquistato i diritti per utilizzare il codice sorgente dello UNIX originale. Cio` fino a quando si decise di ripulire lo Unix BSD dal codice proprietario.

Il risultato iniziale fu 386BSD, che venne rilasciato nel 1992 con la versione 0.1. Tuttavia, questa edizione libera dello Unix BSD non ebbe vita facile, dal momento che da quel punto iniziarono delle contese giudiziarie sulla proprieta` di alcune porzioni di codice ritenute libere (a torto o a ragione che fosse).

Dai problemi di 386BSD che causarono la sua eliminazione dalla distribuzione pubblica, si svilupparono altri progetti indipendenti per ottenere, finalmente, un sistema BSD

libero. Il primo di questi fu nominato NetBSD, al quale si aggiunse subito dopo FreeBSD; piu` tardi, apparve anche OpenBSD.

Tuttavia, i problemi legali non erano finiti. In particolare, per quanto riguarda FreeBSD, questa versione di BSD fu ?libera? solo all'inizio del 1995 con la versione 2.0.

Allo stato attuale, le tre varianti *BSD sono tutte riconducibili a BSD 4.4-Lite, dove le differenze piu` importanti riguardano le piattaforme hardware in cui possono essere installate e l'origine della distribuzione. Infatti, il punto di forza della variante OpenBSD, sta nel fatto di essere realizzata in Canada, da dove possono essere distribuiti anche componenti per la comunicazione crittografica.

GNU

Nel 1985, Richard Stallman ha fondato la FSF, Free software foundation, con lo scopo preciso di creare e diffondere la filosofia del ?software libero?. Liberta` intesa come la possibilita` data agli utenti di distribuire e modificare il software a seconda delle proprie esigenze e di poter distribuire anche le modifiche fatte.

Queste idee filosofiche si tradussero in pratica nella redazione di un contratto di licenza d'uso, la General Public License (GPL), studiato appositamente per proteggere il software libero in modo che non potesse essere accaparrato da chi poi avrebbe potuto impedirne la diffusione libera. Per questo motivo, oggi, il copyright di software protetto in questo modo, viene definito copyleft.

Il software libero richiede delle basi, prima di tutto il sistema operativo. In questo senso, l'obiettivo pratico che si prefiggeva Richard Stallman era quello di realizzare, con l'aiuto di volontari, un sistema operativo completo.

Nacque cosi` il progetto GNU (Gnu's not Unix), con il quale, dopo la realizzazione di un compilatore C, si volevano costruire una serie di programmi di servizio necessari nel momento in cui il cuore del sistema fosse stato completo.

Il progetto GNU diede vita cosi` a una grande quantita` di software utilizzabile sulla maggior parte delle piattaforme Unix, indirizzando implicitamente il software libero nella direzione dei sistemi di questo tipo.

Nel 1990 inizia lo sviluppo del kernel Hurd e intorno al 2000 inizia la distribuzione del sistema GNU/Hurd (sistema GNU basato su kernel Hurd).

Minix

Alla fine degli anni 1980, il professor Andrew S. Tanenbaum sviluppa , un sistema operativo Unix per elaboratori i86, realizzato specificamente per uso didattico. Era sufficiente acquistare il libro a cui era abbinato e si otteneva un sistema completo di sorgenti. Tuttavia, Minix aveva un problema: poteva essere usato, distribuito e modificato, solo per fini didattici.

I diritti di questo sistema operativo sono stati ceduti inizialmente alla casa editrice del libro con il quale questo veniva diffuso. Nell'anno 2000, Andrew S. Tanenbaum ha concordato che la licenza di Minix diventasse meno restrittiva della precedente, portandola ad assomigliare a quella di BSD.

Linux

Linux e` nato come un progetto personale di studio delle funzionalita` di multiprogrammazione dei microprocessori i386 da parte di Linus Torvalds, all'epoca uno studente all'universita` di Helsinki, in Finlandia.

Linus Torvalds decise di trasferire il suo studio dei microprocessori i386 su Minix, con l'idea di realizzare qualcosa di simile a Minix, anzi, qualcosa di migliore (a better Minix than Minix), cominciando da quel sistema operativo, per poi staccarsene completamente.

Dopo molto lavoro, Linus Torvalds riusci` ad arrivare a un sistema minimo e soprattutto autonomo da Minix. Il 5 ottobre 1991 invio` il messaggio seguente su comp.os.minix.

Do you pine for the nice days of Minix-1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on Minix? No more all-nighters to get a nifty program working? Then this post might be just for you.

As I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02...but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it.

L'anno di nascita di un sistema operativo basato sul kernel Linux e` quindi il 1991, anche se non e` il caso di tentare di stabilire una data esatta della nascita della prima versione, la 0.01. Infatti, in quel momento non si poteva ancora parlare di sistema operativo vero e proprio; era solo la dimostrazione che la strada era giusta.

Linux non e` rimasto il progetto personale di una persona; in breve tempo ha coinvolto un numero molto grande di persone, unite dal fatto che si trattava di un progetto libero da qualunque restrizione legale al suo utilizzo, alla sua diffusione, alla possibilita` di modificarlo ecc. In pratica, la fortuna di Linux rispetto a Minix, e` stata quella di avere scelto subito la licenza GNU-GPL, quella che ancora oggi rappresenta la difesa ideale per il software che viene scritto perche' sia a disposizione di tutti. In questo modo si e` superato il limite originale di Minix che lo rendeva interessante solo per professori e studenti. La licenza GPL rende Linux interessante per chiunque.

Tuttavia non bisogna trascurare l'importanza del progetto GNU, che ha dato al kernel Linux tutto quello che serve per arrivare a un sistema operativo completo: GNU/Linux appunto.

Open Source

Una volta compresa l'importanza del software libero, nel momento in cui hanno cominciato a giocarsi interessi economici, o di altro genere, si e` posto il problema di definire in modo preciso e inequivocabile cosa sia effettivamente il ?software libero?.

In questa direzione si e` distinto particolarmente il gruppo che pubblica la distribuzione GNU/Linux Debian, nel definire una serie di punti che devono essere rispettati per l'inserimento del software nella distribuzione stessa.

Al problema dell'ambiguita` del concetto, si affiancava l'ambiguita` della denominazione: in inglese, free software poteva essere inteso come software gratuito (free of charge).

Cosi`, nel 1998, nasce la definizione Open Source, a identificare i principi secondo cui il software puo` essere ritenuto ?libero?, riutilizzando gran parte del lavoro del gruppo Debian, ma dandogli un nome inequivocabile e non modificabile (<http://www.opensource.org>).

Tuttavia, nonostante le buone intenzioni, il nome di questa definizione e` ancora piu` ambiguo, dal momento che non sintetizza il significato che vorrebbe avere. In breve: Open Source, ovvero ?sorgente aperto?, non fa pensare alla ?liberta`? che invece e` il motivo alla base del software libero. In tal senso, benche' la definizione Open Source sia un marchio registrato, non si riesce a impedire l'utilizzo di questi termini, in inglese, slegati da un contesto preciso. Cosi` si permette di sfruttarli per ?illudere? gli ingenui sulle qualita` ?open? del sorgente (?source?) di un certo prodotto commerciale (proprietario) che non ha nulla a che vedere con il software libero. Il vero problema, come sempre, e` l'ignoranza: il software libero non e` un concetto radicato e compreso a sufficienza.

Futuro del software libero

Da un punto di vista ideale, il futuro del software libero non e` cosi` roseo come sembrerebbe, a seguito dell'attenzione che viene data a livello commerciale al sistema operativo GNU/Linux e dall'euforia che ne deriva. Di per se', cio` non dovrebbe essere un male, ma in questa situazione diventa difficile per l'utente comune riuscire a comprendere il significato e il valore del software libero; soprattutto diventa difficile distinguere facilmente quale software sia veramente ?software libero?.

In questo senso, chi crede nella filosofia che ha dato vita a tutto questo, non puo` esserne soddisfatto. Come scrive Richard Stallman in Why ``Free Software'' is better than ``Open

Source":

We have to say, ``It's free software and it gives you freedom!" -- more and louder than ever before.

Chi utilizza GNU/Linux e il software che puo` funzionare con questo sistema operativo, deve impegnarsi a leggere le licenze d'uso: tutto quello che porta il marchio ?Linux? non e` necessariamente ?software libero?. Questo non significa essere contrari all'utilizzo del software proprietario, ma diventa indispensabile distinguere le cose, soprattutto per il rispetto delle leggi.

L'ultima cosa da considerare nei confronti del futuro del software libero e` il problema del brevetto sugli algoritmi e su altri concetti legati al software. Il brevetto impedisce la produzione di software libero che utilizzi algoritmi brevettati, anche se per la realizzazione dei programmi non si copia del codice protetto.

cap 2) Introduzione a GNU/Linux

Il sistema operativo GNU/Linux e` il risultato di una serie molto grande di apporti da diversi ambienti Unix. Quindi, gran parte di cio` che riguarda o compone GNU/Linux, non e` esclusivo di questo ambiente.

Questo capitolo introduttivo e` rivolto a tutti i lettori che non hanno avuto esperienze con Unix, ma anche chi ha gia` una conoscenza di Unix farebbe bene a darci un'occhiata.

Distinzione tra lettere maiuscole e lettere minuscole

I sistemi operativi Unix, come GNU/Linux, sono sensibili alla differenza tra le lettere maiuscole e minuscole. La differenza e` sostanziale, per cui gli ipotetici file denominati: Ciao, cIao, CIAO, ecc. sono tutti diversi.

Non bisogna confondere questa caratteristica con quello che puo` succedere in altri ambienti, come per esempio MS-Windows 95/98/NT/2000, che preservano l'indicazione delle lettere maiuscole o minuscole, ma che poi non fanno differenza quando si vuole fare riferimento a quei file.

Quando in un contesto si fa differenza tra maiuscole e minuscole, capita spesso di vederlo definito come case sensitive, mentre per converso, quando non si fa differenza, come case insensitive.

Root

Negli ambienti Unix si fa spesso riferimento al termine root in vari contesti e con significati differenti. Root e` la radice, o l'origine, senza altri significati. A seconda del contesto, ne rappresenta l'origine, o il punto iniziale. Per esempio, si puo` avere:

- una directory root, che e` la directory principale di un file system, ovvero la directory radice;
- un file system root, che e` il file system principale di un gruppo che si unisce insieme;
- un utente root, che e` l'amministratore;
- un dominio root, che e` il dominio principale;
- una finestra root che e` quella principale, ovvero la superficie grafica (desktop) su cui si appoggiano le altre finestre del sistema grafico X.

Le situazioni in cui si presenta questa definizione possono essere molte di piu`. L'importante, per ora, e` avere chiara l'estensione del significato di questa parola.

Utenti

GNU/Linux, come gli altri sistemi derivati da Unix, e` multiutente. La multiutenza implica una distinzione tra i vari utenti. Fondamentalmente si distingue tra l'amministratore del sistema, o superuser, e gli altri utenti.

L'amministratore del sistema e` quell'utente che puo` fare tutto cio` che vuole, soprattutto rischia di produrre gravi danni anche solo per piccole disattenzioni.

L'utente comune e` quello che utilizza il sistema senza pretendere di organizzarlo e non gli e` possibile avviare programmi o accedere a dati che non lo riguardano.

Registrazione dell'utenza

Per poter utilizzare un sistema di questo tipo, occorre essere stati registrati, ovvero, e` necessario avere ottenuto un account.

Dal punto di vista dell'utente, l'account e` un nome abbinato a una parola d'ordine che gli permette di essere riconosciuto e quindi di poter accedere. Oltre a questo, l'account stabilisce l'appartenenza a un gruppo di utenti.

Il nome dell'amministratore e` sempre root, quello degli altri utenti viene deciso di volta in volta.

Monoutenza

I sistemi Unix e i programmi che su questi sistemi possono essere utilizzati, non sono predisposti per un utilizzo distratto: gli ordini non vengono discussi. Molti piccoli errori possono essere disastrosi se sono compiuti dall'utente root.

E` molto importante evitare il piu` possibile di utilizzare il sistema in qualita` di utente amministratore (root) anche quando si e` l'unico utilizzatore del proprio elaboratore.

Composizione

Il sistema operativo GNU/Linux, cosi` come tutti i sistemi operativi Unix, e` composto essenzialmente da:

- un sistema di avvio o boot;
- un kernel;
- un file system;
- un sistema di inizializzazione e gestione dei processi in esecuzione;
- un sistema di gestione della rete;
- un sistema di gestione delle stampe;
- un sistema di registrazione e controllo degli accessi;
- una shell (interprete dei comandi);
- alcuni programmi di servizio (utility) per la gestione del sistema;
- strumenti di sviluppo software (C/C++).

Avvio

Il boot e` il modo con cui un sistema operativo puo` essere avviato quando l'elaboratore viene acceso. Di solito, il software registrato su ROM degli elaboratori basati sull'uso di dischi, e` fatto in modo da eseguire le istruzioni contenute nel primo settore di un dischetto, oppure, in sua mancanza, del cosiddetto MBR (Master boot record) che e` il primo settore del primo disco fisso. Il codice contenuto nel settore di avvio di un dischetto o del disco fisso, provvede all'esecuzione del kernel (lo avvia).

Con GNU/Linux installato in un elaboratore i386, la configurazione e la gestione del sistema di avvio viene fatta principalmente attraverso tre modi possibili:

- LILO, che e` in grado di predisporre un settore di avvio su un dischetto, sull'MBR o sul primo settore della partizione contenente GNU/Linux;
- GRUB, che e` funzionalmente simile a LILO;
- Loadlin, che permette di avviare l'esecuzione di un kernel Linux da una sessione Dos.

Kernel

Il kernel, come suggerisce il nome, e` il nocciolo del sistema operativo. I programmi utilizzano il kernel per le loro attivita` e in questa maniera sono sollevati dall'agire direttamente con la CPU. Di solito, e` costituito da un file unico, il cui nome potrebbe essere vmlinuz (oppure zImage, bzImage e altri), ma puo` comprendere anche moduli aggiuntivi, per la gestione di componenti hardware specifici che devono poter essere attivati e disattivati durante il funzionamento del sistema.

Quando il kernel viene avviato (attraverso il sistema di avvio), esegue una serie di controlli diagnostici in base ai tipi di dispositivi (componenti hardware) per il quale e`

stato predisposto, quindi monta (mount) il file system principale (root) e infine avvia la procedura di inizializzazione del sistema (Init).

File system

Il file system e` il modo con cui sono organizzati i dati all'interno di un disco o di una sua partizione. Nei sistemi operativi Unix non esiste la possibilita` di distinguere tra un'unita` di memorizzazione e un'altra, come avviene nel Dos, in cui ogni disco o partizione sono contrassegnati da una lettera dell'alfabeto (A:, B:, C:). Nei sistemi Unix, tutti i file system cui si vuole poter accedere devono essere concatenati assieme, in modo da formare un solo file system globale.

Quando un sistema Unix viene avviato, si attiva il file system principale, o root, quindi possono essere collegati a questo altri file system a partire da una directory o sottodirectory di quella principale. Dal momento che per accedere ai dati di un file system diverso da quello principale occorre che questo sia collegato, nello stesso modo, per poter rimuovere l'unita` di memorizzazione contenente questo file system, occorre interrompere il collegamento. Cio` significa che, nei sistemi Unix, non si puo` inserire un dischetto, accedervi immediatamente e toglierlo quando si vuole: occorre dire al sistema di collegare il file system del dischetto, quindi lo si puo` usare come parte dell'unico file system globale. Al termine si deve interrompere questo collegamento e solo allora si puo` rimuovere il dischetto.

L'operazione con cui si collega un file system secondario nel file system globale viene detta mount, per cui si utilizza normalmente il verbo montare con questo significato; l'operazione inversa viene detta unmount e conseguentemente si utilizza il verbo smontare. La directory a partire dalla quale si inserisce un altro file system e` il mount point, che potrebbe essere definito come il punto di innesto.

Inizializzazione e gestione dei processi

GNU/Linux, come tutti i sistemi Unix, e` in multiprogrammazione, ovvero multitasking, cioe` in grado di eseguire diversi programmi, o processi elaborativi, contemporaneamente. Per poter realizzare questo, esiste un gestore dei processi elaborativi: Init, realizzato in pratica dall'eseguibile init, che viene avviato subito dopo l'attivazione del file system principale, allo scopo di occuparsi di eseguire la procedura di inizializzazione del sistema. In pratica, esegue una serie di istruzioni necessarie alla configurazione corretta del sistema particolare che si sta avviando.

Demone

Molti servizi sono svolti da programmi che vengono avviati durante la fase di inizializzazione del sistema e quindi compiono silenziosamente la loro attivita`. Questi programmi sono detti demoni (daemon) e questo termine va considerato come equivalente a ?servente? o ?esperto?.

Gestione dei servizi di rete

Nei sistemi Unix la gestione della rete e` un elemento essenziale e normalmente presente. I servizi di rete vengono svolti da una serie di demoni attivati in fase di inizializzazione del sistema. Nei sistemi GNU/Linux, i servizi di rete sono controllati fondamentalmente da tre demoni:

- inetd che si occupa di attivare di volta in volta, quando necessario, alcuni demoni che poi gestiscono servizi specifici;
- tcpd che si occupa di controllare e filtrare l'utilizzazione dei servizi offerti dal proprio sistema contro gli accessi indesiderati;
- rpc.portmap (oppure solo portmap) che si occupa del protocollo RPC (Remote procedure call).

Un servizio molto importante nelle reti locali consente di condividere porzioni di file system da e verso altri elaboratori connessi. Questo si ottiene con il protocollo NFS che permette quindi di realizzare dei file system di rete.

Gestione della stampa

Tutti i sistemi operativi in multiprogrammazione (multitasking) hanno un sistema di coda di stampa (spool). GNU/Linux utilizza normalmente il demone lpd che in particolare e` anche in grado di ricevere richieste di stampa remote e di inviare richieste di stampa a elaboratori remoti.

Registrazione e controllo degli accessi

I sistemi Unix, oltre che essere in multiprogrammazione sono anche multiutente, cioe` possono essere usati da piu` utenti contemporaneamente. La multiutenza dei sistemi Unix e` da considerare nel modo piu` ampio possibile, nel senso che si puo` accedere all'utilizzo dell'elaboratore attraverso la console, terminali locali connessi attraverso porte seriali, terminali locali connessi attraverso una rete locale e terminali remoti connessi attraverso il modem.

In queste condizioni, il controllo dell'utilizzazione del sistema e` essenziale. Per questo, ogni utente che accede deve essere stato registrato precedentemente, con un nome e una parola d'ordine, o password.

La fase in cui un utente viene riconosciuto e quindi gli viene consentito di agire, e` detta login. Cosi`, la conclusione dell'attivita` da parte di un utente e` detta logout.

Shell: interprete dei comandi

Cio` che permette a un utente di interagire con un sistema operativo e` la shell, che si occupa di interpretare ed eseguire i comandi dati dall'utente.

Dal punto di vista pratico, il funzionamento di un sistema Unix dipende molto dalla shell utilizzata, di conseguenza, la scelta della shell e` molto importante. La shell standard del sistema GNU/Linux e` Bash (il programma bash).

Una shell Unix normale svolge i compiti seguenti:

- mostra l'invito, o prompt, all'inserimento dei comandi;
- interpreta la riga di comando data dall'utente;
- esegue delle sostituzioni, in base ai caratteri jolly e alle variabili di ambiente;(1)
- mette a disposizione alcuni comandi interni;
- mette in esecuzione i programmi;
- gestisce la ridirezione dell'input e dell'output;
- e` in grado di interpretare ed eseguire dei file script di shell.

Programmi di servizio per la gestione del sistema

I comandi interni di una shell non bastano per svolgere tutte le attivita` di amministrazione del sistema. I programmi di servizio sono quelli che di solito hanno piccole dimensioni, sono destinati a scopi specifici di amministrazione del sistema o anche solo di uso comune.

I programmi di servizio di uso comune sono contenuti solitamente all'interno delle directory /bin/ e /usr/bin/. Quelli riservati all'uso da parte dell'amministratore del sistema, l'utente root, sono contenuti normalmente in /sbin/ e /usr/sbin/ dove la lettera ?s? iniziale, sta per superuser, con un chiaro riferimento all'amministratore.

Strumenti di sviluppo software

Tutti i sistemi operativi devono avere un mezzo per produrre del software. In particolare, un sistema operativo Unix deve essere in grado di compilare programmi scritti in linguaggio C/C++. Gli strumenti di sviluppo del sistema GNU/Linux, composti da un compilatore in linguaggio C/C++ e da altri programmi di contorno, sono indispensabili per poter installare del software distribuito in forma sorgente non compilata.

Arresto o riavvio del sistema

Qualunque sistema operativo in multiprogrammazione, tanto piu` se anche multiutente, deve prevedere una procedura di arresto del sistema che si occupi di chiudere tutte le attivita` in corso prima di consentire lo spegnimento fisico dell'elaboratore.

GNU/Linux permette solo all'utente root di avviare la procedura di arresto del sistema con il comando seguente:

```
# shutdown -h now
```

In teoria, negli elaboratori i386 e` possibile utilizzare la combinazione [Ctrl+Alt+Canc] per riavviare il sistema, ma e` sempre preferibile richiamare esplicitamente la procedura di arresto del sistema, specificando che si vuole il riavvio finale.

```
# shutdown -r now
```

Generalmente, l'unico modo per un utente comune di spegnere il sistema, e` quello di riavviare attraverso la combinazione di tasti [Ctrl+Alt+Canc]. Non e` elegante, ma e` il modo migliore per risolvere il problema.

Dispositivi

I vari componenti hardware di un elaboratore, sono rappresentati in un sistema Unix come file di dispositivo, contenuti normalmente nella directory /dev/ (device). Quando si vuole accedere direttamente a un dispositivo, lo si fa utilizzando il nome del file di dispositivo corrispondente.

Tipi

Esistono due categorie fondamentali di dispositivi:

- a carattere, cioe` in grado di gestire i dati in blocchetti di un solo byte per volta;
- a blocchi, cioe` in grado di gestire i dati solo in blocchi (settori) di una dimensione fissa.

Il dispositivo a caratteri tipico e` la console o la porta seriale, mentre il dispositivo a blocchi tipico e` un'unita` a disco. A titolo di esempio, la tabella 5.1 mostra l'elenco di alcuni nomi di dispositivo di GNU/Linux.

Tabella 5.1. Alcuni nomi di dispositivo utilizzati da GNU/Linux.
dispositivo descrizione

/dev/fd0 la prima unita` a dischetti
/dev/fd0u1440 unita` a dischetti con l'indicazione esplicita del formato: 1 440 Kibyte
/dev/hda il primo disco fisso ATA (IDE)
/dev/hda1 la prima partizione del primo disco fisso ATA (IDE)
/dev/hdb il secondo disco fisso ATA (IDE)
/dev/sda il primo disco SCSI
/dev/sda1 la prima partizione del primo disco SCSI
/dev/lp0 la prima porta parallela dal punto di vista di GNU/Linux
/dev/lp1 la seconda porta parallela dal punto di vista di GNU/Linux
/dev/ttyS0 la prima porta seriale

Alcuni file di dispositivo non fanno riferimento a componenti hardware veri e propri. Il piu` noto di questi e` /dev/null utilizzato come fonte per il ?nulla? o come pattumiera senza fondo.

Nomi

I nomi utilizzati per distinguere i file di dispositivo, sono stati scelti in base a qualche criterio mnemonico e all'uso piu` frequente. Tuttavia non e` detto che un dispositivo debba chiamarsi in un modo rispetto a un altro.

Sotto questo aspetto, le distribuzioni GNU/Linux non sono tutte uguali: ognuna interpreta in qualche modo questi nomi. Per fare un esempio, il dispositivo corrispondente all'unita` a dischetti da 1 440 Kibyte, puo` corrispondere a questi nomi differenti:

- /dev/fd0H1440 per la distribuzione Red Hat;
- /dev/fd0u1440 per le distribuzioni Slackware, Debian e SuSE (e` anche la sigla indicata nei sorgenti del kernel).

Le cose si complicano ancora di piu` quando si ha a che fare con sistemi Unix differenti. Quindi: attenzione.

Dischi

Le unita` di memorizzazione a dischi sono dispositivi come gli altri, ma possono essere trattati in due modi diversi a seconda delle circostanze: i dischi, o le partizioni, possono essere visti come dei file enormi o come contenitori di file (file system).

Questa distinzione e` importante perche' capita spesso di utilizzare dischetti che non hanno alcuna struttura di dati essendo stati usati come se si trattasse di un file unico. Il caso piu` comune e` dato dai dischetti di avvio contenenti solo il kernel: non si tratta di dischetti all'interno dei quali e` stato copiato il file del kernel, ma si tratta di dischetti che sono il kernel.

La visione che normalmente si ha delle unita` di memorizzazione contenenti file e directory e` un'astrazione gestita automaticamente dal sistema operativo. Questa astrazione si chiama file system.

Organizzazione di un file system Unix

Tutto cio` che e` contenuto in un file system Unix e` in forma di file: anche una directory e` un file.

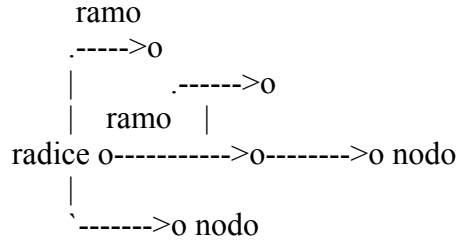
File normale

Quando si vuole fare riferimento a un file nel senso stretto del termine, ovvero un archivio di dati, se si vuole evitare qualunque ambiguita` si utilizza il termine file normale, o regular file.

Directory

Una directory e` un file speciale contenente riferimenti ad altri file. I dati contenuti in un file system sono organizzati in forma gerarchica schematizzabile attraverso un albero, ovvero un tipo particolare di grafo orientato che parte da una radice e si sviluppa in rami e nodi. La figura 5.2 mostra uno schema di un albero.

Figura 5.2. Albero.



La radice e` il nodo principale di questo grafo orientato, i rami rappresentano il collegamento (la discendenza) dei nodi successivi con quello di origine (il genitore). La radice corrisponde a una directory, mentre i nodi successivi possono essere directory, file di dati o file di altro genere.

Per identificare un nodo (file o directory) all'interno di questa gerarchia, si definisce il percorso (path). Il percorso e` espresso da una sequenza di nomi di nodi che devono essere attraversati, separati da una barra obliqua (/). Il percorso idrogeno/carbonio/ossigeno rappresenta un attraversamento dei nodi idrogeno, carbonio e ossigeno.

Dal momento che il grafo di un sistema del genere ha un nodo di origine corrispondente alla radice, si distinguono due tipi di percorsi: relativo e assoluto.

Percorso relativo

Un percorso e` relativo quando parte dalla posizione corrente (o attuale) del grafo per raggiungere la destinazione desiderata. Nel caso dell'esempio precedente, idrogeno/carbonio/ossigeno indica di attraversare il nodo idrogeno inteso come discendente della posizione corrente e quindi gli altri.

Percorso assoluto

Un percorso e` assoluto quando parte dalla radice.

Il nodo della radice non ha un nome come gli altri: viene rappresentato con una sola barra obliqua (/), di conseguenza, un percorso che inizia con tale simbolo, e` un percorso assoluto. Per esempio, /cloro/sodio indica un percorso assoluto che parte dalla radice per poi attraversare cloro e quindi raggiungere sodio.

Un albero e` un grafo orientato, nel senso che i rami hanno una direzione (archi orientati), ovvero ogni nodo ha un genitore e puo` avere dei discendenti e il nodo radice rappresenta l'origine. Quando in un percorso si vuole tornare indietro verso il nodo genitore, non si usa il nome di questo, ma un simbolo speciale rappresentato da due punti in sequenza (..). Per esempio, ../potassio rappresenta un percorso relativo in cui si raggiunge il nodo finale, potassio, passando prima per il nodo genitore della posizione corrente.

In alcuni casi, per evitare equivoci, puo` essere utile poter identificare il nodo della posizione corrente. Il simbolo utilizzato e` un punto singolo (.). Per cui, il percorso

idrogeno/carbonio/ossigeno e` esattamente uguale a ./idrogeno/carbonio/ossigeno.

Collegamenti (link)

Un albero e` tale purche' esista uno e un solo percorso dalla radice a un qualunque altro nodo. Nei file system Unix non e` necessariamente cosi`; pertanto sono schematizzabili attraverso grafi orientati, ma non necessariamente degli alberi. Infatti e` possibile inserire dei collegamenti aggiuntivi, o link, che permettono l'utilizzo di percorsi alternativi. Si distinguono due tipi di questi collegamenti: simbolici e fisici (hard).

Collegamenti fisici, hard link

Un collegamento fisico, o hard link, e` un collegamento che una volta creato ha lo stesso livello di importanza di quelli originali e non e` distinguibile da quelli.

Collegamento simbolico, link simbolico, symlink

Il collegamento simbolico, o link simbolico, e` un file speciale contenente un riferimento a un altro percorso e quindi a un altro nodo del grafo di directory e file.

In generale si preferisce l'uso di collegamenti simbolici per poter distinguere la realta` (o meglio l'origine) dalla finzione. Utilizzando un collegamento simbolico si dichiara apertamente che si sta indicando una scorciatoia e non si perde di vista il percorso originale.

Nomi dei file

Non esiste una regola generale precisa che stabilisca quali siano i caratteri che possono essere usati per nominare un file. Esiste solo un modo per cercare di stare fuori dai guai: il simbolo / non deve essere utilizzato essendo il modo con cui si separano i nomi all'interno di un percorso; inoltre conviene limitarsi all'uso delle lettere dell'alfabeto inglese non accentate, dei numeri, del punto e del trattino basso.

Per convenzione, nei sistemi Unix i file che iniziano con un punto sono classificati come nascosti, perche' vengono mostrati e utilizzati solo quando sono richiesti espressamente.

Questi file, quelli che iniziano con un punto, sono nascosti per una buona ragione: si vuole evitare che utilizzando i caratteri jolly si faccia riferimento alla directory stessa (.) e alla directory genitrice (..). Nello stesso modo si deve fare molta attenzione quando si vuole fare riferimento a questi file nascosti. Il comando rm -r .* non si limita a eliminare i file e le directory che iniziano con un solo punto iniziale, ma elimina anche . e .., cioe`, alla fine, l'intero file system!

Permessi

I file di un file system Unix appartengono simultaneamente a un utente e a un gruppo di utenti. Per questo si parla di utente e gruppo proprietari, oppure semplicemente di proprietario e di gruppo.

L'utente proprietario puo` modificare i permessi di accesso ai suoi file, limitando questi anche per se stesso. Si distinguono tre tipi di accesso: lettura, scrittura, esecuzione. Il significato del tipo di accesso dipende dal file cui questo si intende applicare.

Per i file normali:

- l'accesso in lettura permette di leggerne il contenuto;
- l'accesso in scrittura permette di modificarne il contenuto;
- l'accesso in esecuzione permette di eseguirlo, se si tratta di un eseguibile binario o di uno script di qualunque tipo.

Per le directory:

- l'accesso in lettura permette di leggerne il contenuto, ovvero di poter conoscere l'elenco dei file in esse contenuti (di qualunque tipo essi siano);
- l'accesso in scrittura permette di modificarne il contenuto, ovvero di creare, eliminare e rinominare dei file;
- l'accesso in esecuzione permette di attraversare una directory.

I permessi di un file permettono di attribuire privilegi differenti per gli utenti, a seconda che si tratti del proprietario del file, di utenti appartenenti al gruppo proprietario(3), oppure si tratti di utenti diversi. Cosi`, per ogni file, un utente puo` ricadere in una di queste tre categorie: proprietario, gruppo o utente diverso.

I permessi si possono esprimere in due forme diverse: attraverso una stringa alfabetica o un numero.

Permessi espressi in forma di stringa

I permessi possono essere rappresentati attraverso una stringa di nove caratteri in cui possono apparire le lettere r, w, x, oppure un trattino (-). La presenza della lettera r indica un permesso di lettura, la lettera w indica un permesso di scrittura, la lettera x indica un permesso di esecuzione.

I primi tre caratteri della stringa rappresentano i privilegi concessi al proprietario stesso, il gruppetto di tre caratteri successivo rappresenta i privilegi degli utenti appartenenti al gruppo, il gruppetto finale di tre caratteri rappresenta i privilegi concessi agli altri utenti.

Esempi

rw-r--r--

L'utente proprietario puo` accedervi in lettura e scrittura, mentre sia gli appartenenti al gruppo che gli altri utenti possono solo accedervi in lettura.

rwxr-x---

L'utente proprietario puo` accedervi in lettura, scrittura ed esecuzione; gli utenti appartenenti al gruppo possono accedervi in lettura e in esecuzione; gli altri utenti non possono accedervi in alcun modo.

rw-------

L'utente proprietario puo` accedervi in lettura e scrittura, mentre tutti gli altri non possono accedervi affatto.

Permessi espressi in forma numerica

I permessi possono essere rappresentati attraverso una serie di tre cifre numeriche, in cui la prima rappresenta i privilegi dell'utente proprietario, la seconda quelli del gruppo e la terza quelli degli altri utenti. Il permesso di lettura corrisponde al numero quattro, il permesso di scrittura corrisponde al numero due, il permesso di esecuzione corrisponde al numero uno. Il numero che rappresenta il permesso attribuito a un tipo di utente, si ottiene sommando i numeri corrispondenti ai privilegi che si vogliono concedere.

Esempi

644

L'utente proprietario puo` accedervi in lettura e scrittura (4+2), mentre sia gli appartenenti al gruppo che gli altri utenti possono solo accedervi in lettura.

750

L'utente proprietario puo` accedervi in lettura, scrittura ed esecuzione (4+2+1); gli utenti appartenenti al gruppo possono accedervi in lettura e in esecuzione (4+1); gli altri utenti non possono accedervi in alcun modo.

600

L'utente proprietario puo` accedervi in lettura e scrittura (4+2), mentre tutti gli altri non possono accedervi affatto.

S-bit

I permessi dei file sono memorizzati in una sequenza di 9 bit, dove ogni gruppetto di tre rappresenta i permessi per una categoria di utenti (il proprietario, il gruppo, gli altri).

Assieme a questi 9 bit ne esistono altri tre, posti all'inizio, che permettono di indicare altrettante modalita`: SUID (Set user identifier), SGID (Set group identifier) e Sticky (Save text image). Si tratta di attributi speciali che riguardano prevalentemente i file eseguibili. Solitamente non vengono usati e per lo piu` gli utenti comuni ignorano che esistano.

Tutto questo serve adesso per sapere il motivo per il quale spesso i permessi espressi in forma numerica (ottale) sono di quattro cifre, con la prima che normalmente e` azzerata.

Per esempio, la modalita` 0644 rappresenta il permesso per l'utente proprietario di accedervi in lettura e scrittura, mentre agli altri utenti si concede di accedervi in sola lettura.

L'indicazione della presenza di questi bit attivati puo` essere vista anche nelle rappresentazioni in forma di stringa. L'elenco seguente mostra il numero ottale e la sigla corrispondente.

- SUID = 4 = --s------
- SGID = 2 = -----s---
- Sticky = 1 = --------t

Come si puo` osservare, questa indicazione prende il posto del permesso di esecuzione. Nel caso in cui il permesso di esecuzione corrispondente non sia attivato, la lettera (s o t) appare maiuscola.

Date

Tra gli attributi di un file ci sono anche tre indicazioni data-orario:

- la data e l'ora di creazione: viene modificata in particolare quando si cambia lo stato del file (permessi e proprieta`) e si riferisce precisamente al cambiamento di inode (che verra` descritto piu` avanti);
- la data e l'ora di modifica: viene modificata quando si modifica il contenuto del file;
- la data e l'ora di accesso: viene modificata quando si accede al file anche solo in lettura.

Utenza e accesso

Una volta avviato un sistema Unix, prima che sia disponibile l'invito della shell, ovvero il prompt, occorre che l'utente sia riconosciuto dal sistema, attraverso la procedura di accesso (login). Quello che viene chiesto e` l'inserimento del nome dell'utente (cosi` come e` stato registrato) e subito dopo la parola d'ordine (password) abbinata a quell'utente. Eccezionalmente puo` trattarsi di un utente senza parola d'ordine, cosi` come avviene per i mini sistemi a dischetti fatti per consentire le operazioni di manutenzione eccezionale.

Si distingue solo tra due tipi di utenti: l'amministratore, il cui nome e` root, e gli altri utenti comuni. L'utente root non ha alcun limite di azione, gli altri utenti dipendono dai permessi attribuiti ai file (e alle directory) oltre che dai vincoli posti direttamente da alcuni programmi.

In teoria, e` possibile usare un elaboratore personale solo utilizzando i privilegi

dell'utente root. In pratica, questo non conviene perche' si perde di vista il significato della gestione dei permessi sui file e sulle directory, ma soprattutto si rendono vani i sistemi di sicurezza predefiniti contro gli errori. Per comprendere meglio questo concetto, basta pensare a cosa succede in un sistema Dos quando si esegue un comando come quello seguente:

C:\> DEL *.*

Prima di iniziare la cancellazione, il Dos chiede una conferma ulteriore, proprio perche' non esiste alcun tipo di controllo. In un sistema Unix, di solito cio` non avviene: la cancellazione inizia immediatamente senza richiesta di conferme. Se i permessi consentono la cancellazione dei file solo all'utente root, un utente registrato in modo diverso non puo` fare alcun danno.

In conclusione, l'utente root deve stare molto attento a quello che fa proprio perche' puo` accedere a qualunque funzione o file del sistema, inoltre il sistema non pone alcuna obiezione al suo comportamento. Invece, un utente comune e` vincolato dai permessi sui file e dai programmi che possono impedirgli di eseguire certe attivita`, di conseguenza, e` possibile lavorare con meno attenzione.

adduser o useradd

Di solito, nelle distribuzioni GNU/Linux si trova il programma di servizio adduser, oppure useradd, che consente all'utente root di aggiungere un nuovo utente. Il nome dell'utente non deve superare gli otto caratteri e tutti gli altri dati richiesti possono essere lasciati semplicemente al loro valore predefinito. Dopo la prima installazione del sistema GNU/Linux, e` importante creare il proprio utente personale per poterlo usare senza i privilegi che ha l'amministratore.

exit

La shell comprende solitamente il comando exit che ne termina l'esecuzione. Se si tratta di una shell avviata automaticamente subito dopo l'accesso, il sistema provvedera` ad avviare nuovamente la procedura di accesso.

Interpretazione dei comandi

Come gia` e` stato indicato, l'interpretazione dei comandi e` compito della shell. L'interpretazione dei comandi implica la sostituzione di alcuni simboli che hanno un significato speciale.

File globbing

Il glob (o globbing) e` il metodo attraverso il quale, tramite un modello simbolico, e` possibile indicare un gruppo di nomi di file. Corrisponde all'uso dei caratteri jolly del Dos, con la differenza fondamentale che e` la shell a occuparsi della loro sostituzione e non i programmi. Di solito, si possono utilizzare i simboli seguenti:

*

*

l'asterisco rappresenta un gruppo qualsiasi di caratteri, compreso il punto, purche' questo punto non si trovi all'inizio del nome;

*

?

il punto interrogativo rappresenta un carattere qualsiasi, compreso il punto, purche' questo punto non si trovi all'inizio del nome;

*

[...]

le parentesi quadre permettono di rappresentare un carattere qualsiasi tra quelli contenuti al loro interno, o un intervallo di caratteri possibili.

Dal momento che e` la shell a eseguire la sostituzione dei caratteri jolly, la sintassi tipica di un programma di servizio e` la seguente:

programma [opzioni] [file...]

Nei sistemi Dos si usa spesso la convenzione inversa, secondo cui l'indicazione dei file avviene prima delle opzioni. Da un punto di vista puramente logico, potrebbe sembrare piu` giusto l'approccio del Dos: si indica l'oggetto su cui agire e quindi si indica il modo. Facendo cosi` si ottengono pero` una serie di svantaggi:

- ogni programma deve essere in grado di espandere i caratteri jolly per conto proprio;
- non e` possibile utilizzare l'espansione delle variabili di ambiente e nemmeno di altri tipi;
- se si vogliono indicare elenchi di file che non possono essere espressi con i caratteri jolly, occorre che il programma sia in grado di gestire questa possibilita`, di solito attraverso la lettura di un file esterno.

In pratica, il tipo di semplificazione utilizzato dal Dos e` poi la fonte di una serie di complicazioni per i programmatori e per gli utilizzatori.

Tilde

Di solito, la shell si occupa di eseguire la sostituzione del carattere tilde (~). Nei sistemi Unix, ogni utente ha una directory personale, conosciuta comunemente come directory home. Il simbolo ~ da solo viene sostituito dalla shell con la directory personale dell'utente che sta utilizzando il sistema, mentre un nominativo-utente preceduto dal simbolo ~, viene sostituito dalla shell con la directory personale dell'utente indicato.

Variabili di ambiente

Le variabili di ambiente sono gestite dalla shell e costituiscono uno dei modi attraverso cui si configura un sistema. I programmi possono leggere alcune variabili di loro interesse e modificare il proprio comportamento in base al loro contenuto.

Una riga di comando puo` fare riferimento a una variabile di ambiente: la shell provvede a sostituirne l'indicazione con il suo contenuto.

Ridirezione e pipeline

I programmi, quando vengono eseguiti, hanno a disposizione alcuni canali standard per il flusso dei dati (input/output). Questi sono: standard input, standard output e standard error.

Standard input

Lo standard input viene utilizzato come fonte standard per i dati in ingresso (input) nel programma.

Standard output

Lo standard output viene utilizzato come destinazione standard per i dati in uscita (output) dal programma.

Standard error

Lo standard error, viene utilizzato come destinazione standard per i dati in uscita dal programma derivati da situazioni anomale.

Lo standard input e` rappresentato di norma dai dati provenienti dalla tastiera del terminale. Lo standard output e lo standard error sono emessi normalmente attraverso lo schermo del terminale.

Per mezzo della shell si possono eseguire delle ridirezioni di questi flussi di dati, per esempio facendo in modo che lo standard output di un programma sia inserito come standard input di un altro, creando cosi` una pipeline.

Ridirezione dello standard input

programma < file_di_dati

Si ridirige lo standard input utilizzando il simbolo minore (<) seguito dalla fonte alternativa di dati. Il programma a sinistra del simbolo < riceve come standard input il contenuto del file indicato a destra.

Esempi

$ sort < elenco.txt

Visualizza il contenuto del file elenco.txt dopo averlo riordinato.

Ridirezione dello standard output

programma > file_di_dati

Si ridirige lo standard output utilizzando il simbolo maggiore (>) seguito dalla destinazione alternativa dei dati. Il programma a sinistra del simbolo > emette il suo standard output all'interno del file indicato a destra che viene creato per l'occasione.

Lo standard output puo` essere aggiunto a un file preesistente; in tal caso si utilizza il simbolo >>.

Esempi

$ ls > elenco.txt

Genera il file elenco.txt con il risultato dell'esecuzione di ls.

$ ls >> elenco.txt

Aggiunge al file elenco.txt il risultato dell'esecuzione di ls.

Ridirezione dello standard error

programma 2> file_di_dati

Si ridirige lo standard error utilizzando il simbolo 2> seguito dalla destinazione alternativa dei dati. Il programma a sinistra del simbolo 2> emette il suo standard error all'interno del file indicato a destra che viene creato per l'occasione.

Lo standard error puo` essere aggiunto a un file preesistente; in tal caso si utilizza il simbolo 2>>.

Esempi

$ controlla 2> errori.txt

Genera il file errori.txt con il risultato dell'esecuzione dell'ipotetico programma controlla.

$ controlla 2>> errori.txt

Aggiunge al file errori.txt il risultato dell'esecuzione dell'ipotetico programma controlla.

Pipeline

programma1 | programma2 [| programma3...]

Si ridirige lo standard output di un programma nello standard input di un altro, utilizzando il simbolo barra verticale (|). Il programma a sinistra del simbolo | emette il suo standard output nello standard input di quello che sta a destra.

Nella rappresentazione schematica delle sintassi dei programmi, questo simbolo ha normalmente il significato di una scelta alternativa tra opzioni diverse, parole chiave o altri argomenti. In questo caso fa proprio parte della costruzione di una pipeline.

Esempi

$ ls | sort

Riordina il risultato del comando ls.

$ ls | sort | less

Riordina il risultato del comando ls e quindi lo fa scorrere sullo schermo con l'aiuto del programma less.

Comandi e programmi di servizio di uso comune

In linea di principio, con il termine comando ci si dovrebbe riferire ai comandi interni di una shell, mentre con il termine utility, o semplicemente programma, si dovrebbe fare riferimento a programmi eseguibili esterni alla shell. Di fatto pero`, dal momento che si mette in esecuzione un programma impartendo un comando alla shell, con questo termine (comando) si fa spesso riferimento in maniera indistinta a comandi interni di shell o (in mancanza) a comandi esterni o programmi di servizio.

Naturalmente, questo ragionamento vale fino a quando si tratta di programmi di servizio di uso comune, non troppo complessi, che usano un sistema di input/output elementare. Sarebbe un po' difficile definire comando un programma di scrittura o un navigatore di Internet.

Interpretazione della sintassi

La sintassi di un programma o di un comando segue delle regole molto semplici.

- Le metavariabili, scritte in questo modo, descrivono l'informazione che deve essere inserita al loro posto.
- Le altre parole rappresentano dei termini chiave che, se usati, devono essere indicati cosi` come appaiono nello schema sintattico.
- Quello che appare racchiuso tra parentesi quadre rappresenta una scelta facoltativa: puo` essere utilizzato o meno.

- La barra verticale (|) rappresenta la possibilita` di scelta tra due possibilita` alternative: quello che sta alla sua sinistra e quello che sta alla sua destra. Per esempio, uno | due rappresenta la possibilita` di scegliere una tra le parole uno e due.
- Quello che appare racchiuso tra parentesi graffe rappresenta una scelta obbligatoria e serve in particolare per evitare equivoci nell'interpretazione quando si hanno piu` scelte alternative, separate attraverso il simbolo |. Seguono alcuni esempi.

  {uno | due | tre}

  Rappresenta la scelta obbligatoria di una tra le tre parole chiave: uno, due e tre.

  {-f file | --file=file}

  Rappresenta la scelta obbligatoria di una tra due opzioni equivalenti.
- I puntini di sospensione rappresentano la possibilita` di aggiungere altri elementi dello stesso tipo di quello che li precede. Per esempio, file... rappresenta la metavariabile file che puo` essere seguita da altri valori dello stesso tipo rappresentato dalla metavariabile stessa.

Naturalmente, puo` capitare che i simboli utilizzati per rappresentare la sintassi, servano negli argomenti di un comando o di un programma. I casi piu` evidenti sono:

- le pipeline che utilizzano la barra verticale per indicare il flusso di dati tra un programma e il successivo;
- le parentesi graffe usate dalla shell Bash come tipo particolare di espansione.

Quando cio` accade, occorre fare attenzione al contesto per poter interpretare correttamente il significato di una sintassi, osservando gli esempi eventualmente proposti.

Organizzazione tipica

Il programma di servizio tipico ha la sintassi seguente:

programma [opzioni] [file...]

In questo caso, il nome del programma e` proprio programma.

Opzioni

Normalmente vengono accettate una o piu` opzioni facoltative, espresse attraverso una lettera dell'alfabeto preceduta da un trattino (-a, -b,...). Queste possono essere usate separatamente oppure raggruppandole con un solo trattino seguito da tutte le lettere delle opzioni che si intendono selezionare. Quindi:

programma -a -b

e` traducibile nel comando seguente:

programma -ab

I programmi piu` recenti includono opzioni descrittive formate da un nome preceduto da due trattini. In presenza di questi tipi di opzioni, non si possono fare aggregazioni nel modo appena visto.

A volte si incontrano opzioni che richiedono l'indicazione aggiuntiva di un altro argomento.

File

La maggior parte dei programmi di servizio esegue delle elaborazioni su file, generando un risultato che viene emesso normalmente attraverso lo standard output. Spesso, quando non vengono indicati file negli argomenti, l'input per l'elaborazione viene ottenuto dallo standard input.

Alcuni programmi permettono l'utilizzo del trattino (-) in sostituzione dell'indicazione di file in ingresso o in uscita, allo scopo di fare riferimento, rispettivamente, allo standard input e allo standard output.

Programma o eseguibile

In generale, quando si usa il termine ?programma? non si chiarisce quale sia la sua estensione reale. Si puo` usare questo termine per identificare qualcosa che si compone di un solo file eseguibile, oppure un piccolo sistema composto da piu` componenti, che vengono comandate da un solo sistema frontale.

Spesso, in particolare all'interno di questo documento, quando si vuole fare riferimento a un programma inteso come un insieme di componenti, oppure come qualcosa di astratto per il quale nel contesto non conta il modo in cui viene avviato, lo si indica con un nome che non ha enfatizzazioni particolari e generalmente ha l'iniziale maiuscola. Per esempio, questo e` il caso della shell Bash, a cui si e` accennato, il cui eseguibile e` in realta` bash.

Per evitare ambiguita`, quando si vuole essere certi di fare riferimento a un programma eseguibile, si specifica proprio che si tratta di questo, cioe` di un ?eseguibile?, mostrandolo attraverso enfatizzazioni di tipo dattilografico, scrivendo il nome esattamente nel modo in cui cio` va fatto per avviarlo.

L'ABC dei comandi GNU/Linux

Nelle sezioni seguenti vengono descritti in modo sommario alcuni programmi di servizio fondamentali. Gli esempi mostrati fanno riferimento all'uso della shell Bash che costituisce attualmente lo standard per GNU/Linux.

E` importante ricordare che negli esempi viene mostrato un invito differente a seconda che ci si riferisca a un comando impartito da parte di un utente comune o da parte dell'amministratore: il dollaro ($) rappresenta un'azione di un utente comune, mentre il simbolo # rappresenta un'azione dell'utente root.

ls

ls [opzioni] [file...]

Elenca i file contenuti in una directory.

Esempi

$ ls

Elenca il contenuto della directory corrente.

$ ls -l *.doc

Elenca tutti i file che terminano con il suffisso .doc che si trovano nella directory corrente. L'elenco contiene piu` dettagli sui file essendoci l'opzione -l.

cd

cd [directory]

Cambia la directory corrente.

Esempi

$ cd /tmp

Cambia la directory corrente, facendola diventare /tmp/.

$ cd ciao

Cambia la directory corrente, spostandosi nella directory ciao/ che discende da quella corrente.

$ cd ~

Cambia la directory corrente, spostandosi nella directory personale dell'utente.

$ cd ~daniele

Cambia la directory corrente, spostandosi nella directory personale dell'utente daniele.

mkdir

mkdir [opzioni] directory...

Crea una directory.

Esempi

$ mkdir cloro

Crea la directory cloro/, come discendente di quella corrente.

$ mkdir /sodio/cloro

Crea la directory cloro/, come discendente di /sodio/.

$ mkdir ~/cloro

Crea la directory cloro/, come discendente della directory personale dell'utente attuale.

cp

cp [opzioni] origine... destinazione

Copia uno o piu` file (incluse le directory) in un'unica destinazione.

La copia in un sistema Unix non funziona come nei sistemi Dos e cio` principalmente a causa di due fattori: i caratteri jolly (ovvero il file globbing) vengono risolti dalla shell prima dell'esecuzione del comando e i file system Unix possono utilizzare i collegamenti simbolici.

Se vengono specificati solo i nomi di due file normali, il primo viene copiato sul secondo, viene cioe` generata una copia che ha il nome indicato come destinazione. Se il secondo nome indicato e` una directory, il file viene copiato nella directory con lo stesso nome di origine. Se vengono indicati piu` file, l'ultimo nome deve essere una directory e verranno generate le copie di tutti i file indicati, all'interno della directory di destinazione. Di conseguenza, quando si utilizzano i caratteri jolly, la destinazione deve essere una directory. In mancanza di altre indicazioni attraverso l'uso di opzioni adeguate, le directory non vengono copiate.

Chi utilizzava il Dos potrebbe essere abituato a usare il comando COPY per copiare un gruppo di file in un altro gruppo di file con i nomi leggermente modificati, come in questo esempio: COPY *.bak *.doc. Con i sistemi Unix, questo tipo di approccio non puo` funzionare.

I file elencati nell'origine potrebbero essere in realta` dei collegamenti simbolici. Se non viene specificato diversamente attraverso l'uso delle opzioni, questi vengono copiati cosi` come se fossero file normali; cioe` la copia sara` ottenuta a partire dai file originali e non si otterra` quindi una copia dei collegamenti.

Alcune opzioni

-a

Equivalente a -dpR, utile per l'archiviazione o comunque per la copia di collegamenti simbolici cosi` come sono.

-d

Copia i collegamenti simbolici mantenendoli come tali, invece di copiare il file a cui i collegamenti si riferiscono.

-f

Sovrascrittura forzata dei file di destinazione.

-l

Crea un collegamento fisico invece di copiare i file.

-p

Mantiene le proprieta` e i permessi originali.

-r

Copia file e directory in modo ricorsivo (includendo le sottodirectory), considerando tutto cio` che non e` una directory come un file normale.

-R

Copia file e directory in modo ricorsivo (includendo le sottodirectory).

Esempi

$ cp -r /test/* ~/prova

Copia il contenuto della directory /test/ in ~/prova/ copiando anche eventuali sottodirectory contenute in /test/.

Se ~/prova esiste gia` e non si tratta di una directory, questo file viene sovrascritto, perdendo quindi il suo contenuto originale.

$ cp -r /test ~/prova

Copia la directory /test/ in ~/prova/ (attaccando test/ a ~/prova/) copiando anche eventuali sottodirectory contenute in /test/.

$ cp -dpR /test ~/prova

Copia la directory /test/ in ~/prova/ (attaccando test/ a ~/prova/) copiando anche eventuali sottodirectory contenute in /test/, mantenendo inalterati i permessi e riproducendo i collegamenti simbolici eventuali.

ln

ln [opzioni] origine... destinazione

Crea uno o piu` collegamenti di file (incluse le directory) in un'unica destinazione.

La creazione di un collegamento e` un'azione simile a quella della copia. Di conseguenza valgono le stesse considerazioni fatte in occasione del comando cp per quanto riguarda la differenza di comportamento che c'e` tra Unix e Dos.

Se vengono specificati solo i nomi di due file normali, il secondo diventa il collegamento del primo. Se il secondo nome indicato e` una directory, al suo interno verranno creati altrettanti collegamenti quanti sono i file e le directory indicati come origine. I nomi utilizzati saranno gli stessi di quelli di origine. Se vengono indicati piu` file, l'ultimo nome deve corrispondere a una directory.

E` ammissibile la creazione di collegamenti che fanno riferimento ad altri collegamenti.

Se ne possono creare di due tipi: collegamenti fisici e collegamenti simbolici. Questi ultimi sono da preferire (a meno che ci siano delle buone ragioni per utilizzare dei collegamenti fisici). Se non viene richiesto diversamente attraverso le opzioni, si generano dei collegamenti fisici invece che i consueti collegamenti simbolici.

Alcune opzioni

-s

Crea un collegamento simbolico.

-f

Sovrascrittura forzata dei file o dei collegamenti gia` esistenti nella destinazione.
Esempi

$ ln -s /test/* ~/prova

Crea, nella destinazione ~/prova/, una serie di collegamenti simbolici corrispondenti a tutti i file e a tutte le directory che si trovano all'interno di /test/.

$ ln -s /test ~/prova

Crea, nella destinazione ~/prova, un collegamento simbolico corrispondente al file o

alla directory /test. Se ~/prova e` una directory, viene creato il collegamento ~/prova/test; se ~/prova non esiste, viene creato il collegamento ~/prova.

rm

rm [opzioni] nome...

Rimuove i file indicati come argomento. In mancanza dell'indicazione delle opzioni necessarie, non vengono rimosse le directory.

Alcune opzioni

-r | -R

Rimuove il contenuto delle directory in modo ricorsivo.

Esempi

$ rm prova

Elimina il file prova.

$ rm ./-r

Elimina il file -r che inizia il suo nome con un trattino, senza confondersi con l'opzione -r (ricorsione).

$ rm -r ~/varie

Elimina la directory varie/ che risiede nella directory personale, insieme a tutte le sue sottodirectory eventuali.

Attenzione

rm -r .*

Elimina tutti i file e le directory a partire dalla directory radice! In pratica elimina tutto.

Questo e` un errore tipico di chi vuole cancellare tutte le directory nascoste (cioe` quelle che iniziano con un punto) contenute nella directory corrente. Il disastro avviene perche' nei sistemi Unix, .* rappresenta anche la directory corrente (.) e la directory precedente o genitrice (..).

mv

mv [opzioni] origine... destinazione

Sposta i file e le directory. Se vengono specificati solo i nomi di due elementi (file o directory), il primo viene spostato e rinominato in modo da ottenere quanto indicato come destinazione. Se vengono indicati piu` elementi (file o directory), l'ultimo attributo deve

essere una directory: verranno spostati tutti gli elementi elencati nella directory di destinazione. Nel caso di spostamenti attraverso file system differenti, vengono spostati solo i cosiddetti file normali (quindi: niente collegamenti e niente directory).

Nei sistemi Unix non esiste la possibilita` di rinominare un file o una directory semplicemente come avviene nel Dos. Per cambiare un nome occorre spostarlo. Questo fatto ha poi delle implicazioni nella gestione dei permessi delle directory.

Esempi

$ mv prova prova1

Cambia il nome del file (o della directory) prova in prova1.

$ mv * /tmp

sposta, all'interno di /tmp/, tutti i file e le directory che si trovano nella directory corrente.

cat

cat [opzioni] [file...]

Concatena dei file e ne emette il contenuto attraverso lo standard output. Il comando emette di seguito i file indicati come argomento attraverso lo standard output (sullo schermo), in pratica qualcosa di simile al comando TYPE del Dos. Se non viene fornito il nome di alcun file, viene utilizzato lo standard input.

Tabella 5.2. Comparazione tra alcuni comandi Dos e gli equivalenti per GNU/Linux attraverso degli esempi.

Dos	GNU/Linux
DIR	ls -l
DIR /W	ls
MD PIPPO	mkdir pippo
CD PIPPO	cd pippo
RD PIPPO	rmdir pippo
COPY *.* \PROVA	cp * /prova
XCOPY *.* \PROVA /E /S	cp -dpR * /prova
REN ARTICOLO LETTERA	mv articolo lettera
MOVE *.* \PROVA	mv * /prova
DEL ARTICOLO	rm articolo
DELTREE TEMP	rm -R temp
TYPE LETTERA	cat lettera
TYPE LETTERA | MORE	cat lettera | more
HELP DIR	man ls
FORMAT A: /N:18 /T:80	fdformat /dev/fd0u1440
FORMAT A: /N:9 /T:80	fdformat /dev/fd0u720
DISKCOPY A: B:	cp /dev/fd0 /dev/fd1

```
DISKCOPY A: A: cp /dev/fd0 /tmp/pippo ; cp /tmp/pippo /dev/fd0
KEYB IT loadkeys it
CLS clear
BACKUP C:\DATI\*.* A: /S tar cvf /dev/fd0 -L 1440 -M /dati
FIND "saluti" PRIMO.DOC grep "saluti" primo.doc
FOR %A IN (*.DOC) DO FIND "saluti" %A grep "saluti" *.doc
MEM free
```

Esempi

$ cat prova prova1

Mostra di seguito il contenuto di prova e prova1.

$ cat prova prova1 > prova2

Genera il file prova2 come risultato del concatenamento in sequenza di prova e prova1.

cap 3) Documentazione

Esistono diverse fonti di documentazione su GNU/Linux. La maggior parte di questa e` normalmente disponibile all'interno delle distribuzioni, ma la consultazione puo` risultare un problema per chi non ha esperienza.

Testo puro

Il modo piu` semplice con cui puo` essere stato scritto qualcosa e` quello del testo puro. Questo e` il caso dei file ?leggimi? (readme) e simili, oppure di tutta quella documentazione che, per semplicita`, e` stata convertita anche in questo formato.

La lettura di questi file puo` essere fatta attraverso due programmi ben conosciuti: more oppure less. more e` quello tradizionalmente piu` diffuso negli ambienti Unix; less e` il piu` pratico ed e` decisamente piu` ricco di piccoli accorgimenti. Le funzionalita` essenziali di questi due programmi sono simili, anche se il secondo offrirebbe un gran numero di funzioni aggiuntive che qui non vengono considerate.

La sintassi di questi due programmi e` la seguente:

more [opzioni] [file]...

less [opzioni] [file]...

Nell'utilizzo normale non vengono fornite opzioni e se non viene indicato alcun file negli argomenti, viene fatto lo scorrimento di quanto ottenuto dallo standard input.

Una volta avviato uno di questi due programmi, lo scorrimento del testo dei file da visualizzare avviene per mezzo di comandi impartiti attraverso la pressione di tasti. Il meccanismo e` simile a quello utilizzato da VI: alcuni comandi richiedono semplicemente la pressione di uno o piu` tasti in sequenza; altri richiedono un argomento e in questo caso, la digitazione appare nell'ultima riga dello schermo o della finestra a disposizione.

La differenza fondamentale tra questi due programmi sta nella possibilita` da parte di less di scorrere il testo all'indietro anche quando questo proviene dallo standard input, mentre per more non e` possibile. less permette inoltre di utilizzare i tasti freccia e i tasti pagina per lo scorrimento del testo, consentendo anche di effettuare ricerche all'indietro.

less e` un programma che permette un utilizzo molto piu` complesso di quanto descritto qui, ma questo va oltre l'uso che se ne fa normalmente.

Esempi

```
$ ls -l | more
```

$ ls -l | less

Scorre sullo schermo l'elenco del contenuto della directory corrente che probabilmente e` troppo lungo per essere visualizzato senza l'aiuto di uno tra questi due programmi.

$ more README

$ less README

Scorre sullo schermo il contenuto del file README.

Pagine di guida

Quasi tutti i programmi sono accompagnati da una pagina di manuale, ovvero man page. Si tratta di un documento stringato sull'uso di quel programma particolare, scritto in uno stile abbastanza uniforme.

Si distinguono diverse sezioni di queste pagine di manuale, a seconda del genere di informazioni in esse contenute. Puo` infatti accadere che esistano piu` pagine con lo stesso nome, appartenenti pero` a sezioni diverse. La tabella 8.3 riporta l'elenco di queste sezioni.

Tabella 8.3. Sezioni delle pagine di manuale.

Sezione	Contenuto	Descrizione
1	comandi utente	Comandi (di uso comune) disposizione dell'utente.
2	chiamate di sistema	Funzioni messe a disposizione dal kernel ai programmi.
3	chiamate di libreria	Funzioni messe a disposizione da librerie esterne ai programmi.
4	dispositivi	File speciali che si trovano solitamente nella directory /dev/.
5	formati dei file	Sintassi dei file di configurazione.
6	giochi	
7	varie	Convenzioni e altre informazioni.
8	amministrazione	Comandi per l'amministrazione del sistema.
9	routine del kernel	Non standard

Quando si vuole fare riferimento a una pagina di manuale, se ne indica il nome seguito da un numero tra parentesi, che ne esprime la sezione. Per cui, man(1) indica la pagina di manuale di nome man nella prima sezione. Spesso, nella documentazione, si fa riferimento ai programmi in questo modo, per dare istantaneamente l'informazione di dove raggiungere le notizie che riguardano quel programma particolare.

Questa documentazione viene consultata normalmente attraverso il programma man che a sua volta, solitamente, si avvale di less, oppure more, per scorrere il documento.

/etc/man.config

Il comportamento di man puo` essere configurato attraverso il file /etc/man.manconfig. Tra le tante cose, questo file contiene gli argomenti da fornire ai programmi utilizzati per la formattazione del testo da visualizzare o da stampare. Si osservi l'estratto seguente:

```
TROFF        /usr/bin/groff -Tps -mandoc
NROFF        /usr/bin/groff -Tlatin1 -mandoc
EQN         /usr/bin/geqn -Tps
NEQN         /usr/bin/geqn -Tlatin1
TBL         /usr/bin/gtbl
# COL        /usr/bin/col
REFER        /usr/bin/grefer
PIC        /usr/bin/gpic
VGRIND
GRAP
PAGER        /usr/bin/less -is
CAT         /bin/cat
```

Una cosa che conviene modificare, se non fosse gia` impostata correttamente, e` l'opzione -T di groff e di geqn. Utilizzandola nel modo che si vede nell'esempio (-Tlatin1), serve a consentire la visualizzazione dei caratteri accentati delle pagine di manuale tradotte in italiano.

$ man

man [opzioni] nome...

L'eseguibile man formatta ed emette attraverso lo standard output la pagina di manuale indicata dal nome. Lo scorrimento del testo che compone le pagine di manuale indicate negli argomenti viene fatto attraverso un programma esterno, richiamato automaticamente da man. Solitamente si tratta di more o di less. Di conseguenza, i comandi per lo scorrimento del testo dipendono dal tipo di programma utilizzato. Se si tratta di uno di questi due appena citati, sono sempre validi almeno quelli riportati nella tabella 8.1.

Alcune opzioni

numero_di_sezione

Se prima del nome del comando o dell'argomento appare un numero, si intende che si vuole ottenere la pagina di manuale da una sezione determinata, come riportato nella tabella 8.3.

-f

Si comporta come whatis.

-h

Visualizza una breve guida di se stesso.

-k

Equivalente a apropos.

Esempi

$ man ls

Visualizza la pagina di manuale del programma ls.

$ man 8 lilo

Visualizza la pagina di manuale nell'ottava sezione, del programma lilo.

$ whatis

whatis parola...

Cerca la descrizione di una o piu` pagine di manuale corrispondenti ai nomi indicati come argomento.(1)

Il risultato della ricerca viene emesso attraverso lo standard output. Sono visualizzate solo le corrispondenze con parole intere.

Esempi

$ whatis ls

Visualizza le descrizioni aventi la parola ls nel nome della pagina.

$ apropos

apropos stringa...

Cerca la descrizione di una o piu` pagine di manuale corrispondenti, che contengono al loro interno la stringa indicata (la ricerca viene fatta solo nella descrizione, nome compreso, e non nel corpo della pagina di manuale)(2)

Il risultato della ricerca viene emesso attraverso lo standard output.

Esempi

$ apropos keyboard

Visualizza le descrizioni contenenti la stringa keyboard.

Documentazione allegata ai pacchetti

I pacchetti di programmi piu` importanti sono accompagnati da documentazione scritta in vario modo (testo, LaTeX, TeX, SGML, PostScript, ecc.). Questa si trova collocata normalmente in sottodirectory discendenti da /usr/share/doc/.

HOWTO

I documenti HOWTO non accompagnano i pacchetti di programmi come loro parte integrante, essendo delle guide aggiuntive con scopi che vanno oltre la semplice documentazione del funzionamento di un solo pacchetto particolare. La maggior parte delle distribuzioni GNU/Linux include anche i file di documentazione HOWTO. Solitamente, questi vengono installati al di sotto della directory /usr/share/doc/HOWTO/.

FAQ

Un'altra fonte di documentazione su GNU/Linux sono le cosiddette FAQ o Frequently asked questions. Si tratta di informazioni disordinate in forma di botta e risposta. Solitamente si trovano al di sotto della directory /usr/share/doc/FAQ/.(3)

LDP

A fianco della documentazione standard fornita piu` o meno con tutte le distribuzioni GNU/Linux, ci sono dei libri veri e propri disponibili liberamente. Questi sono raccolti all'interno del progetto LDP, o Linux documentation project.

Questi documenti, normalmente disponibili sia in PostScript che in HTML, sono raggiungibili a partire da <http://www.ibiblio.org/pub/Linux/docs/LDP/> e dai siti speculari relativi.

Ricerche nella rete

Alle volte non si riesce a trovare l'informazione che si cerca all'interno della documentazione di cui si dispone, cosi` capita di rivolgersi ai gruppi di discussione di Usenet con richieste di aiuto disperate. Altre volte non si riesce a trovare il pacchetto per GNU/Linux che si sta cercando, cosi` ancora una volta si inviano richieste, con la speranza che qualcuno di buon cuore risponda.

La rete offre strumenti e servizi che e` bene conoscere e usare prima di tentare l'ultima carta delle richieste ?circolari?.

Ricerche sul web

Le ricerche attraverso i motori di ricerca generici, permettono di trovare le pagine pubblicate contenenti una combinazione di parole particolare, secondo la stringa di ricerca fornita. Questo tipo di ricerca permette normalmente di ottenere informazioni non molto recenti (cio` inteso in senso relativo) essendoci voluto il tempo necessario a

pubblicarle e a catalogarle nei sistemi di ricerca automatica.

I motori di ricerca disponibili sono diversi e di solito conviene concentrarsi su un paio di questi, studiandone la sintassi corretta per le espressioni di ricerca. Generalmente si pone il problema di conoscere in che modo indicare la presenza simultanea di due parole particolari, o la presenza di alcune parole escludendone altre.

Supponendo di voler cercare informazioni sulla masterizzazione di CD-R con GNU/Linux, si potrebbe indicare un'espressione per la ricerca simultanea delle parole Linux e CD-R. Nel caso di Google, si puo` usare l'espressione Linux CD-R, senza bisogno di operatori (simboli) particolari.

Ricerche nei gruppi di discussione

I gruppi di discussione sono spesso la destinazione di domande e risposte di ogni tipo, soprattutto di quelle banali dei principianti di ogni genere. Quando sorge un problema per il quale ci si vorrebbe rivolgere a un gruppo di discussione, nella maggior parte dei casi qualcun altro ha gia` posto la stessa domanda che vorremmo fare. Per questo, invece di affollare ulteriormente la rete di altre domande ridondanti, conviene provare prima a scandagliare i gruppi di Usenet alla ricerca dell'argomento del proprio problema, per vedere se esiste gia` la risposta che si desidera ottenere.

Indubbiamente cio` non e` facile e per questo vengono in aiuto dei servizi simili ai motori di ricerca della rete, ma specializzati nei gruppi di Usenet. La figura 9.3 mostra una parte della pagina introduttiva del servizio offerto da Google, <http://groups.google.com>, contenente il necessario per inviare un'espressione di ricerca.

Anche in questo caso si suppone di voler cercare informazioni sulla masterizzazione di CD-R con GNU/Linux; come prima ci si vuole concentrare sulle parole chiave Linux e CD-R. Nel caso di Google si deve usare semplicemente l'espressione Linux CD-R senza bisogno di aggiungere operatori particolari. La figura 9.4 mostra il risultato della ricerca.

Naturalmente, e` possibile leggere i messaggi di richiesta e le risposte; cio` che si desiderava.

Ricerche nelle liste di posta elettronica

Le liste di posta elettronica, o piu` amichevolmente ?liste?, sono dei gruppi di discussione a cui ci si iscrive e da cui si ricevono regolarmente tutti i messaggi attraverso la posta elettronica. Non esiste un servizio che permetta di consultare questi messaggi, perche' non esiste una forma di pubblicizzazione standard.

Alcune di queste liste sono pubblicate automaticamente in forma di pagine HTML sulla rete ipertestuale (il web). Quando cio` accade, e` probabile che si riesca a raggiungere queste notizie attraverso i motori di ricerca normali.

Ricerche negli FTP

Quando si cerca qualcosa che dovrebbe trovarsi in un servizio FTP, come un pacchetto applicativo di cui si e` sentito parlare ma non si sa dove sia, e` possibile utilizzare uno dei servizi di ricerca che permette di trovare un file a partire dalle informazioni del nome.

Il servizio piu` popolare a questo proposito e` FTPSearch <http://www.alltheweb.com/?c=ftp>. Di solito, la cosa piu` conveniente da definire e` il tipo di ricerca; il tipo wildcard search rappresenta una ricerca per i nomi di file secondo un modello composto con caratteri jolly (asterisco e punto interrogativo), senza tenere conto della differenza tra lettere maiuscole e minuscole.

cap 4) LILO: introduzione

LILO e` una procedura molto comune per il caricamento di GNU/Linux negli elaboratori con architettura i386. Permette di avviare anche altri sistemi operativi eventualmente residenti nello stesso elaboratore in cui si usa GNU/Linux.

Organizzazione essenziale

La procedura LILO e` composta essenzialmente da:

- la directory /boot/ e dal suo contenuto;

- l'eseguibile lilo;
- il file di configurazione /etc/lilo.conf.

La directory /boot/ contiene i file utilizzati per effettuare l'avvio del sistema: sia per avviare GNU/Linux, sia per altri sistemi operativi eventuali. Puo` contenere anche il file del kernel, o piu` file di kernel differenti, quando per questo non si usa semplicemente la directory radice. Piu` precisamente, contiene almeno i file seguenti:

- boot.b;
- map che viene creato da LILO;
- boot.n, dove l'estensione e` un numero esadecimale, che viene creato da LILO e contiene il settore di avvio dell'unita` rappresentata dal numero stesso (non si tratta necessariamente di un solo file);
- il kernel se non risiede gia` nella directory radice.

Installazione del meccanismo di caricamento del sistema operativo

L'installazione del meccanismo di caricamento del sistema operativo avviene modificando il contenuto di uno di questi settori:

- MBR o Master boot record;
- il primo settore di una partizione;
- il primo settore di un dischetto.

Nel primo caso, LILO ha il controllo su tutti i sistemi operativi per il loro caricamento; nel secondo, LILO dipende da un sistema di avviamento di un altro sistema operativo che, a sua volta, passa a LILO il controllo quando cio` viene richiesto; nel terzo caso si utilizza un dischetto in modo da non alterare il sistema di avvio gia` presente.

L'installazione avviene per mezzo dell'eseguibile lilo, che a sua volta si basa sulla configurazione stabilita attraverso /etc/lilo.conf. Ogni volta che si cambia qualcosa all'interno della directory /boot/, o si modifica, o si sposta il file del kernel, e` necessario ripetere l'installazione attraverso l'eseguibile lilo.

/etc/lilo.conf

/etc/lilo.conf e` il file di configurazione utilizzato da LILO per installare il sistema di avvio. Si tratta di una sorta di script contenente solo assegnamenti a variabili. Ne viene descritto il funzionamento in modo sommario partendo da un esempio in cui si ha un solo disco fisso, dove la prima partizione e` riservata al Dos e la seconda a GNU/Linux. L'esempio permette di avviare GNU/Linux e il Dos selezionando una tra le parole linux o dos al momento dell'avvio. Il simbolo # rappresenta l'inizio di un commento che viene ignorato.

```
# Prima parte generale
boot=/dev/hda
```

```
prompt
timeout=50

# Caricamento di Linux
image=/boot/vmlinuz
   label=linux
   root=/dev/hda2
   read-only

# Caricamento del Dos
other=/dev/hda1
   label=dos
   table=/dev/hda
```

Segue la descrizione delle direttive che appaiono nell'esempio.

boot=/dev/hda

Nella prima parte viene specificato che il settore di avvio deve essere collocato nel primo disco ATA, di conseguenza nell'MBR. Se fosse stata indicata una partizione specifica, si sarebbe trattato del primo settore di quella partizione (per esempio: boot=/dev/hda2). Volendo si poteva indicare anche un'unita` per i dischetti, in modo da installare tale settore di avvio in quel dischetto (per esempio: boot=/dev/fd0).

prompt

Si tratta di un'opzione (una variabile booleana) la cui presenza fa si` che all'atto del caricamento venga richiesto di inserire il nome del sistema che si desidera avviare (per la precisione, la parola chiave che vi fa riferimento).

timeout=50

Dopo 50 decimi di secondo (cinque secondi), senza che sia stato selezionato alcunche', viene avviato il sistema predefinito (in questo caso linux).

image=/boot/vmlinuz

Inizia la definizione di un kernel da avviare: /boot/vmlinuz.

Si tratta del file che si trova nel file system in funzione nel momento in cui si avvia l'eseguibile lilo. Questo particolare potrebbe sembrare ovvio, ma non e` sempre cosi`. Se si vuole preparare un sistema di avvio per un sistema GNU/Linux residente in un'altra partizione (magari un dischetto), si vuole forse fare riferimento a un kernel che si trova li`. La cosa potrebbe non essere tanto intuitiva e viene descritta piu` avanti.

label=linux

Definisce il nome utilizzato per fare riferimento a questo kernel. Poteva essere

qualunque cosa, in questo caso il nome linux e` utile per ricordare che si tratta dell'avvio di quel sistema operativo.

root=/dev/hda2

Indica la partizione da utilizzare come file system principale (root).

read-only

La presenza di questa opzione fa si` che la partizione specificata venga montata inizialmente in sola lettura, in modo da permettere al kernel di eseguire un controllo prima di avviare il resto del sistema. Al termine del controllo, la partizione viene rimontata regolarmente in lettura e scrittura, ma questo per opera della procedura di inizializzazione del sistema.

other=/dev/hda1

Inizia la definizione dell'avvio di un altro sistema operativo, per il quale non e` LILO a prendersi cura dell'avvio del kernel, ma un altro settore di avvio. In questo caso il settore di avvio deve trovarsi all'inizio della partizione /dev/hda1.

label=dos

Definisce il nome utilizzato per fare riferimento a questo sistema operativo. La parola dos e` utile per ricordare che si tratta dell'avvio di quel sistema operativo.

table=/dev/hda

Specifica il file di dispositivo che si riferisce all'unita` che contiene l'indicazione della tabella delle partizioni. In effetti, questa e` contenuta nella parte iniziale del disco fisso, quindi si fa riferimento all'intera unita` /dev/hda.

Volendo, e` possibile avviare lo stesso file system con kernel differenti a seconda delle necessita`. In tal caso si possono aggiungere al file /etc/lilo.conf altri blocchetti come quello seguente:

```
# Caricamento di Linux con un kernel sperimentale
image=/boot/vmlinuz-prova
   label=prova
   root=/dev/hda2
   read-only
```

Se si vuole la possibilita` di utilizzare come file system principale una partizione diversa da quella normale, magari per fare delle prove, o per qualunque altro motivo, si puo` indicare una voce alternativa come quando si vuole avviare con diversi kernel possibili.

```
# Caricamento di una partizione alternativa in un disco SCSI
image=/boot/vmlinuz
```

```
   label=extra
   root=/dev/sda3
   read-only
```

Quello che conta e` comprendere che il sistema di avvio resta nella directory /boot/ e senza il disco che la contiene, i file system in /dev/hda2 o /dev/sda3 non possono essere montati. Inoltre, senza /dev/hda (in questi esempi), non si avvierebbe alcunche'. Per comprendere meglio il problema, si pensi a questo esempio:

- GNU/Linux sia avviato e stia utilizzando la partizione /dev/hda2 come file system principale;
- la directory /boot/ sia vuota e sia stata utilizzata per montare un dischetto corrispondente al dispositivo /dev/fd0;
- la directory radice del dischetto corrisponda esattamente a /boot/;
- il dischetto contenga i file gia` visti, necessari per l'avvio (il kernel, boot.b, map, ecc.);
- il file /etc/lilo.conf sia come quello visto sopra, per cui il settore di avvio si deve trovare nell'MBR del primo disco fisso (/dev/hda).

In questo modo, se si esegue lilo, viene creato un settore di avvio nell'MBR di /dev/hda che fa riferimento ai file di avvio (kernel incluso) contenuti nel dischetto. Cioe`, senza quel dischetto (proprio quello), il sistema non potrebbe avviarsi. Questo problema viene rivisto piu` avanti dove viene spiegato come costruire un dischetto contenente sia un settore di avvio che il kernel e i file di LILO.

Alle volte e` necessario informare il kernel di qualche particolarita` dell'hardware installato. In tal caso si utilizza la variabile append alla quale si assegna la stringa necessaria. Nell'esempio seguente si invia la stringa cdu31a=0x340,0 necessaria per poter attivare un vecchio lettore CD-ROM Sony.

```
# Caricamento di Linux con l'attivazione del CD-ROM
image=/boot/vmlinuz
   label=sony
   root=/dev/hda2
   append="cdu31a=0x340,0"
   read-only
```

 # lilo

lilo [opzioni]

L'eseguibile lilo permette di installare il sistema di avvio basato sulla procedura LILO. Per farlo, legge il contenuto del file /etc/lilo.conf o di quello indicato attraverso l'opzione -C.

Alcune opzioni

-C file_di_configurazione

Permette di indicare un file di configurazione differente rispetto al solito /etc/lilo.conf.

-r directory_di_partenza

Permette di definire una pseudo directory radice in modo da poter utilizzare quanto contenuto in un dischetto o in un altro disco montato da qualche parte.
Esempi

lilo -C ./mia.conf

Installa il sistema di avvio utilizzando la configurazione del file mia.conf contenuto nella directory corrente.

lilo -r /mnt/floppy

Utilizza la configurazione del file /mnt/floppy/etc/lilo.conf, facendo riferimento (probabilmente) ai file contenuti in /mnt/floppy/boot/, utilizzando i file di dispositivo in /mnt/floppy/dev/.

LILO su un disco differente

LILO parte dal presupposto che si stia operando sempre all'interno del file system attivo nel momento in cui si avvia l'eseguibile lilo. Si potrebbe pensare che per fare in modo di sistemare l'avvio su un altro disco, come un dischetto o un'altra unita` rimovibile, si debba agire semplicemente sulla direttiva boot=dispositivo; ma questo non basta. Si deve utilizzare l'opzione -r per fare riferimento a una pseudo directory radice, a partire dalla quale LILO deve trovare tutto quello che gli serve, compreso il file di configurazione.

Di seguito viene mostrato l'esempio della preparazione di un dischetto contenente il kernel avviato da LILO, in modo completamente indipendente dal file system attivo nel momento in cui lo si realizza, con una configurazione simile a quella mostrata in precedenza, nella sezione

1.

All'interno di un dischetto inizializzato e contenente un file system Second-extended (Ext2) si riproduce tutto quello che serve a LILO per definire il sistema di avvio. Si tratta della directory boot/ contenente gli stessi file della stessa directory appartenente al file system generale, insieme al kernel; della directory etc/ con il file lilo.conf; della directory dev/ con i file di dispositivo corrispondenti alle unita` di memorizzazione cui si fa riferimento. Si suppone di avere montato il dischetto utilizzando la directory /mnt/floppy/ come punto di innesto.

fdformat /dev/fd0u1440

```
# mke2fs /dev/fd0

# mount -t ext2 /dev/fd0 /mnt/floppy

# cp -dpR /boot /mnt/floppy

# mkdir /mnt/floppy/etc

# cp /etc/lilo.conf /mnt/floppy/etc/lilo.conf

# mkdir /mnt/floppy/dev

# cd /mnt/floppy/dev/

# /dev/MAKEDEV fd0 fd1 hda hdb hdc hdd sda sdb sdc sdd
```

2. Il file /mnt/floppy/etc/lilo.conf viene modificato in modo da fare riferimento al dispositivo /dev/fd0.

```
boot=/dev/fd0
```

3. Si utilizza l'eseguibile lilo con l'opzione -r in modo da fargli usare i file nel dischetto e non quelli contenuti nel file system principale.

```
# lilo -r /mnt/floppy
```

Il problema puo` presentarsi anche in modo inverso, quando si avvia il sistema attraverso dischetti di emergenza e si vuole sistemare l'avvio di GNU/Linux attraverso il disco fisso. La partizione principale del disco fisso potrebbe essere montata nel sistema di emergenza, per esempio in corrispondenza della directory /mnt/, mentre per il resto non dovrebbe essere necessario preoccuparsi d'altro, a parte la versione di LILO presente nel dischetto, che deve essere compatibile con i file di avvio del disco fisso.

```
# lilo -r /mnt
```

Boot prompt

Subito dopo la prima fase dell'avvio del sistema, quella gestita da LILO, prima dell'avvio vero e proprio del kernel, in presenza di determinate condizioni viene visualizzato un invito particolare a inserire delle opzioni: il boot prompt. Questo appare:

- se era stata indicata l'istruzione prompt nel file /etc/lilo.conf;
- se viene premuto il tasto [Maiuscole], oppure [Ctrl], oppure [Alt];
- se il tasto [Fissamaiuscole] oppure [BlocScorr] risultano inseriti.

Il boot prompt, ovvero l'invito dell'avvio, ha l'aspetto seguente:

boot:

Normalmente si utilizza la riga di comando di avvio per indicare il nome di una configurazione particolare. In altri casi e` il mezzo per specificare un'opzione che per qualche motivo non e` attiva automaticamente e si vuole che LILO la passi al kernel.

La digitazione all'interno di questa riga di comando e` abbastanza intuitiva: per cancellare si possono usare i tasti [Backspace], [Canc] e le combinazioni [Ctrl+u] e [Ctrl+x]. Eventualmente, si puo` ottenere un elenco delle configurazioni, riferite a diverse voci del file /etc/lilo.conf, attraverso la pressione del tasto [Tab]. Si conferma con il tasto [Invio]. Il vero problema e` la tastiera: si deve considerare che la disposizione dei tasti e` quella statunitense.

La sintassi di quanto si puo` inserire attraverso la riga di comando e` la seguente:

[configurazione [opzione...]]

Se si preme semplicemente [Invio] viene avviata la configurazione predefinita, altrimenti e` obbligatorio l'inserimento del nome di questa, seguita eventualmente da altre opzioni.

I vari argomenti inseriti attraverso la riga di comando (il nome della configurazione e le altre opzioni eventuali) sono separati tra loro attraverso uno spazio. Per questo, un argomento non puo` contenere spazi.

Nella sezione 16.4 vengono descritti alcuni tipi di parametri che possono essere inseriti in una riga di comando di avvio. Per una descrizione piu` ampia conviene consultare il capitolo 30 ed eventualmente il BootPrompt HOWTO.

cap 5) Kernel Linux

Il kernel e` il nocciolo del sistema operativo. I programmi utilizzano le funzioni fornite dal kernel e in questa maniera sono sollevati dall'agire direttamente con la CPU.

Il kernel Linux e` costituito normalmente da un file soltanto, il cui nome puo` essere vmlinuz, oppure zImage, bzImage e altri ancora, ma puo` comprendere anche moduli aggiuntivi per la gestione di componenti hardware specifici che devono poter essere attivati e disattivati durante il funzionamento del sistema.

Quando si fa riferimento a un kernel in cui tutte le funzionalita` che servono sono incluse nel file principale, si parla di kernel monolitico, mentre quando parte di queste sono poste all'interno di moduli esterni, si parla di kernel modulare. Il kernel monolitico ha il vantaggio di avere tutto in un file, ma nello stesso modo e` rigido e non permette di liberare risorse quando le unita` periferiche gestite non servono. Il kernel modulare ha il vantaggio di poter disattivare e riattivare i moduli a seconda delle esigenze, in particolare quando moduli distinti gestiscono in modo diverso lo stesso tipo di unita` periferica. Tuttavia, a causa della frammentazione in molti file, l'uso dei moduli puo` essere fonte di errori.

In generale, l'uso dei kernel modulari dovrebbe essere riservato agli utilizzatori che hanno gia` un'esperienza sufficiente nella gestione dei kernel monolitici. In ogni caso, ci possono essere situazioni in cui l'uso di un kernel modulare e` praticamente indispensabile, per esempio quando un certo tipo di dispositivo fisico puo` essere gestito in vari modi differenti e conflittuali, ma si tratta di situazioni rare.

Ricompilazione del kernel

Le distribuzioni GNU/Linux tendono a fornire agli utilizzatori un kernel modulare per usi generali. Anche se questo si adattera` sicuramente alla maggior parte delle configurazioni, ci sono situazioni particolari dove e` preferibile costruire un proprio kernel, monolitico o modulare che sia.

Per poter comprendere il procedimento di compilazione descritto in questo capitolo, occorre sapere come si compila e si installa un programma tipico distribuito in forma sorgente, come descritto nel capitolo 20.

Kernel monolitico

Il procedimento descritto in questa sezione serve per generare un kernel monolitico, cioe` un kernel in un solo file.

Per poter procedere alla compilazione del kernel e` necessario avere installato gli strumenti di sviluppo software, cioe` il compilatore e i sorgenti del kernel. In particolare, i sorgenti del kernel possono anche essere reperiti presso vari siti che offrono l'accesso attraverso il protocollo FTP. In tal caso si puo` fare una ricerca per i file che iniziano per linux-x.y.z.tar.gz, dove x.y.z sono i numeri della versione.

Il numero di versione del kernel Linux e` strutturato in tre livelli: x.y.z, dove il primo, x, rappresenta il valore piu` importante, mentre l'ultimo, z, rappresenta quello meno importante. Quello che conta, e` porre attenzione al valore intermedio: y. Se si tratta di un numero pari, la versione si riferisce a un kernel ritenuto sufficientemente stabile, mentre un numero dispari rappresenta una versione destinata agli sviluppatori e non ritenuta adatta per l'utilizzo normale.

Se i sorgenti sono stati installati attraverso un disco (un CD-ROM) di una distribuzione, questi si troveranno al loro posto, altrimenti occorre provvedere a installarli manualmente. La posizione standard in cui devono trovarsi i sorgenti del kernel e` la directory /usr/src/linux/. Se si utilizza un'altra posizione e` necessario un collegamento simbolico che permetta di raggiungere i sorgenti nel modo prestabilito.

Occorre inoltre verificare che tre collegamenti simbolici contenuti nella directory /usr/include/ siano corretti.

- asm --> /usr/src/linux/include/asm-i386/
- linux --> /usr/src/linux/include/linux/
- scsi --> /usr/src/linux/include/scsi

E` evidente che il primo, asm, dipende dal tipo di piattaforma hardware utilizzato.

Una volta installati i sorgenti del kernel, si puo` passare alla configurazione che precede la compilazione. Per questo, ci si posiziona nella directory dei sorgenti; quindi, dopo aver letto il file README, si puo` procedere come mostrato nel seguito.

cd /usr/src/linux

La directory corrente deve essere quella a partire dalla quale si diramano i sorgenti del kernel.

make mrproper

Serve a eliminare file e collegamenti vecchi che potrebbero interferire con una nuova compilazione.

make config

E` l'operazione piu` delicata attraverso la quale si definiscono le caratteristiche e i componenti del kernel che si vuole ottenere. Ogni volta che si esegue questa operazione viene riutilizzato il file .config contenente la configurazione impostata precedentemente, mentre alla fine la nuova configurazione viene salvata nello stesso file. Di conseguenza, ripetendo il procedimento make config, le scelte predefinite corrisponderanno a quelle effettuate precedentemente.

Il comando make mrproper elimina il file .config, quindi si deve fare attenzione a non eseguire tale comando se non e` questa l'intenzione.

Se si dispone di un kernel recente, in alternativa a make config che e` un metodo piuttosto spartano di configurare il sistema, si possono utilizzare:

make menuconfig

un sistema di configurazione a menu` basato su testo;

make xconfig

un sistema di configurazione a menu` grafico per X.

Dopo la definizione della configurazione, si puo` passare alla compilazione del kernel relativo, utilizzando la sequenza di comandi seguente:

make dep ; make clean ; make zImage

Si tratta di tre operazioni che si possono avviare tranquillamente in questo modo perche' non richiedono nessun tipo di interazione con l'utente. Al termine della compilazione, se questa ha avuto successo, il nuovo kernel si trova nella directory /usr/src/linux/arch/i386/boot/ con il nome zImage (questo vale naturalmente nel caso si utilizzi l'architettura i386).

Se il kernel che si genera e` troppo grande, potrebbe non funzionare l'istruzione make zImage. In tal caso si deve sostituire con make bzImage e alla fine si otterra` un file con quel nome, cioe` bzImage. Si deve tenere in considerazione che anche in questo secondo

caso esiste un limite massimo alla grandezza del kernel, per cui, potrebbe essere necessario l'utilizzo di moduli per le funzionalita` che non sono indispensabili al caricamento del sistema.

Naturalmente, per fare in modo che il kernel possa essere utilizzato, questo andra` collocato dove e` necessario che si trovi perche' il sistema che si occupa del suo avvio possa trovarlo (parte vi). Di solito lo si copia nella directory radice o in /boot/, dandogli il nome vmlinuz (come di consueto), sistemando poi cio` che serve per il sistema di avvio che si utilizza.

Una volta realizzato un kernel e` necessario fare una prova per vedere se funziona. Il modo migliore (nel senso che e` meno pericoloso) per verificarne il funzionamento e` quello di farne una copia in un dischetto di avvio (ovvero un dischetto di boot).(2)

```
# cp /usr/src/linux/arch/i386/boot/zImage /dev/fd0
```

Per utilizzare correttamente questo dischetto di avvio e` molto probabile che si debba intervenire prima con il programma rdev (16.1.1).

Kernel modulare

Il procedimento per la creazione di un kernel modulare inizia nello stesso modo di quello monolitico e giunge alla creazione di un file che in piu` ha dei riferimenti a moduli esterni che vengono compilati a parte. Questi moduli, per poter essere gestiti correttamente, necessitano di programmi di servizio che si occupano della loro attivazione e disattivazione.

In questo caso, oltre ai sorgenti del kernel sono necessari i programmi per la gestione dei moduli. Questi si trovano normalmente in archivi il cui nome e` organizzato in modo simile a quello dei sorgenti del kernel: modules-x.y.z.tar.gz. La struttura della versione rappresentata dai numeri x.y.z rispecchia lo stesso meccanismo utilizzato per i sorgenti del kernel, pero` non ne vengono prodotte altrettante versioni: si dovra` badare a utilizzare la versione piu` vicina a quella del kernel che si utilizza. Questo archivio si trova normalmente nella stessa directory del sito dal quale si ottengono i sorgenti del kernel.

Anche i programmi contenuti nell'archivio modules-x.y.z.tar.gz sono in forma sorgente e prima di poterli utilizzare devono essere compilati e installati.

Se si sta ricompilando il kernel attraverso i sorgenti della distribuzione GNU/Linux che si utilizza, e` ragionevole supporre che questi programmi di gestione dei moduli siano gia` stati installati correttamente.

Per ottenere un kernel modulare, dopo la preparazione del file principale del kernel attraverso lo stesso procedimento visto nel caso di un kernel monolitico, si devono compilare i moduli.

```
# make modules ; make modules_install
```

Quello che si ottiene sono una serie di file oggetto, il cui nome ha un'estensione .o, raggruppati ordinatamente all'interno di directory discendenti da /lib/modules/x.y.z/, dove x.y.z rappresenta il numero della versione dei sorgenti del kernel. La posizione di questi file non deve essere cambiata.

Compilazione del kernel in una distribuzione GNU/Linux Debian

La distribuzione GNU/Linux Debian mantiene una separazione netta tra i file di intestazione dei sorgenti del kernel e quelli delle librerie di sviluppo. In questo modo, non si deve piu` provvedere a sistemare i collegamenti simbolici nella directory /usr/include/, al massimo ci puo` essere la necessita` di aggiornare le librerie di sviluppo.

Per il resto, la procedura per la compilazione del kernel e dei moduli potrebbe essere svolta nello stesso modo gia` descritto. Tuttavia, questa distribuzione mette a disposizione uno strumento accessorio, molto utile, per facilitare questa operazione, passando per la creazione di un pacchetto Debian vero e proprio. Il pacchetto in questione e` denominato kernel-package e per questo scopo puo` essere usato direttamente senza bisogno di alcuna configurazione. E` sufficiente procedere nel modo seguente:

1. cd directory_iniziale_dei_sorgenti

   ci si sposta nella directory iniziale dei sorgenti del kernel;
2. make {config|menuconfig|xconfig}

   si procede con la configurazione del kernel che si vuole ottenere;
3. make-kpkg clean

   ci si prepara alla compilazione;
4. make-kpkg --revision=versione kernel_image

   si esegue la compilazione generando l'archivio Debian corrispondente, nella directory precedente.

L'esempio seguente si riferisce alla compilazione di un kernel 2.2.15 (compresi i moduli eventuali) collocato nella directory /usr/src/linux-2.2.15.

```
# cd /usr/src/linux-2.2.15

# make-kpkg clean
```

```
# make-kpkg --revision=custom.1.0 kernel_image
```

Si puo` osservare che la versione e` stata definita dalla stringa custom.1.0. Questo e` cio` che viene suggerito nella documentazione originale. In particolare, il numero ?1.0? va incrementato ogni volta che si predispone una versione successiva.

Al termine si ottiene l'archivio kernel-image-2.2.15_custom.1.0_i386.deb, collocato nella directory precedente a quella dei sorgenti da cui e` stato ottenuto; per installarlo basta procedere come segue:

```
# dpkg -i ../kernel-image-2.2.15_custom.1.0_i386.deb
```

Elementi della configurazione

Gli elementi richiesti per la configurazione del kernel prima della sua compilazione, dipendono molto dalla versione che si possiede. In particolare, puo` capitare che alcune voci vengano spostate da una versione all'altra del kernel.

Le varie opzioni sono raggruppate in alcuni gruppi principali, che dovrebbero guidare intuitivamente nella configurazione prima della compilazione del kernel:

Code maturity level options
selezione del dettaglio con cui si vogliono definire le voci della configurazione;

Loadable module support
gestione dei moduli del kernel;

Processor type and features
caratteristiche del microprocessore o dei microprocessori;

General setup
caratteristiche generali, sia fisiche, sia logiche del sistema;

Memory Technology Devices (MTD)
gestione di memorie MTD, ovvero memoria speciale che ha la proprieta` di non essere volatile come la RAM comune;

Parallel port support
gestione delle porte parallele;

Plug and Play configuration
gestione del Plug & Play;

Block devices
gestione dei dispositivi a blocchi;

Multi-device support (RAID and LVM)

gestione di unita` multiple di memorizzazione, come nel caso dei dischi RAID;

Networking options
funzionalita` di rete in generale;

Telephony Support
gestione di hardware speciale per la telefonia digitale su IP;

ATA/IDE/MFM/RLL support
gestione di dischi ATA/ATAPI e simili;

SCSI support
gestione di unita` SCSI;

IEEE 1394 (FireWire) support
gestione di un bus IEEE 1394, noto anche con il nome FireWire;

I2O device support
gestione di dispositivi periferici speciali denominati ?I2O?;

Network device support
gestione specifica delle interfacce di rete, includendo alcuni protocolli usati comunemente come tunnel (per esempio PPP, SLIP e PLIP), in quanto acquistano nel sistema un nome di interfaccia;

Amateur Radio support
gestione di hardware e protocolli per le comunicazioni via radio;

IrDA subsystem support
gestione del sistema IrDA di comunicazione a raggi infrarossi;

ISDN subsystem
gestione di alcune schede speciali per la connessione a una rete ISDN (non e` necessario utilizzare queste opzioni se si dispone di un ?modem? ISDN esterno);

Old CD-ROM drivers (not for SCSI or IDE/ATAPI drives)
gestione di vecchi tipi di lettori CD-ROM, che non sono compatibili con gli standard ATA/ATAPI e nemmeno con lo standard SCSI;

Input core support

Character devices
gestione dei dispositivi a caratteri (terminali, porte seriali, porte parallele, mouse, ecc.);

Multimedia devices
gestione di dispositivi multimediali;

File systems
gestione di file system, con le problematiche relative;

Console devices
gestione particolare della console;

Sound
gestione dell'audio;

USB support
gestione del bus USB e delle unita` periferiche relative;

Kernel hacking
configurazione particolare per chi vuole lavorare attivamente allo sviluppo del kernel.

Nelle sezioni seguenti vengono descritte in parte solo alcuni di questi gruppi di configurazione, mostrando quale esempio che comunque non puo` esaurire il problema.

Code maturity level options

Questa sezione della procedura di configurazione si riferisce al livello di dettaglio a cui si e` interessati, per quanto riguarda le opzioni di configurazione che possono essere richieste. Se si e` interessati a funzionalita` relativamente nuove, conviene abilitare il dettaglio massimo nella selezione delle opzioni di configurazione.

Loadable module support

Questa sezione della procedura di configurazione permette di attivare il sistema di gestione dei moduli. I moduli sono blocchetti di kernel precompilati che possono essere attivati e disattivati durante il funzionamento del sistema. Solo alcune parti del kernel possono essere gestite in forma di modulo.

Se si intende creare un kernel modulare, e` evidente la necessita` di attivare questa gestione all'interno della parte principale del kernel stesso.

[*] Enable loadable module support
[*] Set version information on all module symbols
[*] Kernel module loader

Processor type and features

Questa sezione serve a definire il tipo di microprocessore utilizzato. In generale, se si utilizza un'architettura di tipo ix86, la selezione del tipo di microprocessore 386 garantisce la creazione di un kernel compatibile nella maggior parte delle situazioni, a discapito pero` delle prestazioni.

Sempre nel caso di architettura di tipo ix86, e` possibile abilitare l'emulazione per il

coprocessore matematico (i487), che in alcuni elaboratori molto vecchi non era incluso. Di solito, l'inclusione del codice di emulazione non crea problemi di conflitti, perche' viene individuata automaticamente la presenza dell'hardware relativo e l'emulazione non viene attivata se non quando necessario. In tal modo, includendo questa funzionalita` si genera un kernel piu` compatibile.

(386) Processor family
< > Toshiba Laptop support (NEW)
< > /dev/cpu/microcode - Intel IA32 CPU microcode support (NEW)
< > /dev/cpu/*/msr - Model-specific register support (NEW)
< > /dev/cpu/*/cpuid - CPU information support (NEW)
(off) High Memory Support
[*] Math emulation
[*] MTRR (Memory Type Range Register) support
[ ] Symmetric multi-processing support
[ ] APIC and IO-APIC support on uniprocessors (NEW)

General setup

Questa sezione raccoglie una serie di opzioni di importanza generale, che non hanno trovato una collocazione specifica in un'altra posizione della procedura di configurazione. Mano a mano che le funzionalita` del kernel Linux si estendono, aumentano le sezioni della configurazione, per cui capita che vi vengano spostate li` alcune di queste opzioni.

In particolare, all'interno di questa sezione dovrebbe essere possibile stabilire in modo preliminare:

- l'utilizzo della rete;
- il tipo di bus (ISA, EISA, MCA, PCI, ecc.);(4)
- IPC (Inter process communication) di System V (sezione 39.3);
- il tipo di programmi binari utilizzati nel sistema (ELF, a.out), oltre alla possibilita` di utilizzare interpreti opportuni per altri tipi di eseguibili.

[*] Networking support
[ ] SGI Visual Workstation support
[*] PCI support
(Any) PCI access mode
[*] PCI device name database (NEW)
[ ] EISA support (NEW)
[ ] MCA support
[*] Support for hot-pluggable devices (NEW)
PCMCIA/CardBus support --->
[*] System V IPC
[*] BSD Process Accounting
[*] Sysctl support
(ELF) Kernel core (/proc/kcore) format
<*> Kernel support for a.out binaries
<*> Kernel support for ELF binaries

<*> Kernel support for MISC binaries
[*] Power Management support (NEW)
[ ] ACPI support (NEW)
< > Advanced Power Management BIOS support

Parallel port support

La gestione della porta parallela non riguarda solo la stampa, dal momento che consente anche l'uso di altri tipi di unita` periferiche. In questo gruppo di opzioni e` possibile abilitare l'uso delle porte parallele, stabilendo eventualmente il grado di compatibilita` di queste.

<*> Parallel port support
<*> PC-style hardware
[ ] Use FIFO/DMA if available (EXPERIMENTAL) (NEW)
[ ] SuperIO chipset support (EXPERIMENTAL) (NEW)
[ ] Support for PCMCIA management for PC-style ports (NEW)
[*] Support foreign hardware
[ ] IEEE 1284 transfer modes (NEW)

Plug and Play configuration

La gestione del Plug & Play permette al kernel di configurare automaticamente alcuni dispositivi che aderiscono a queste specifiche.

<*> Plug and Play support
<*> ISA Plug and Play support (NEW)

Block devices

Un dispositivo a blocchi e` quello che utilizza una comunicazione a blocchi di byte di dimensione fissa, contrapponendosi al dispositivo a caratteri con cui la comunicazione avviene byte per byte. Il dispositivo a blocchi tipico e` un'unita` a disco.

Merita attenzione particolare anche il dispositivo definito loopback,(5) che rappresenta in pratica un file contenente l'immagine di un disco, che viene letto come se fosse un disco o una partizione reale. Questa possibilita`, tra le altre cose, consente di gestire direttamente i file che contengono la riproduzione esatta di dischetti, senza bisogno di trasferire questi file su dischetti reali.

Infine, e` qui che si puo` abilitare e configurare la gestione dei dischi RAM, ovvero di dischi che vengono rappresentati nella memoria RAM.

<*> Normal PC floppy disk support
< > XT hard disk support
< > Parallel port IDE device support
< > Compaq SMART2 support
< > Compaq Smart Array 5xxx support (NEW)
< > Mylex DAC960/DAC1100 PCI RAID Controller support

<*> Loopback device support
<*> Network block device support
<*> RAM disk support
(8192) Default RAM disk size (NEW)
[*] Initial RAM disk (initrd) support

Multi-device support (RAID and LVM)

La gestione di piu` unita` di memorizzazione in modo combinato richiede la selezione di un gruppo speciale di opzioni. Puo` trattarsi di dischi o partizioni ridondanti come forma di protezione dalle perdite di dati, oppure puo` essere un modo per fondere assieme piu` partizioni in una partizione logica piu` grande.

Networking options

La configurazione delle funzionalita` di rete e` importante anche se il proprio elaboratore e` isolato. Quando e` attiva la gestione della rete, il kernel fornisce implicitamente le funzionalita` di inoltro dei pacchetti, consentendo in pratica il funzionamento come router. Tuttavia, l'attivazione di cio` dipende dall'inclusione della gestione del file system /proc/ (29.2.14) e dell'interfaccia sysctl (29.2.4). Inoltre, durante il funzionamento del sistema e` necessario attivare espressamente l'inoltro attraverso un comando simile a quello seguente:

```
# echo '1' > /proc/sys/net/ipv4/ip_forward
```

<*> Packet socket
[ ] Packet socket: mmapped IO
[*] Kernel/User netlink socket
[*] Routing messages
<*> Netlink device emulation
[*] Network packet filtering (replaces ipchains)
[*] Socket Filtering
<*> Unix domain sockets
[*] TCP/IP networking
[*] IP: multicasting
[*] IP: advanced router
[*]   IP: policy routing
[*]   IP: fast network address translation
[*]   IP: equal cost multipath
[*]   IP: use TOS value as routing key
[*]   IP: verbose route monitoring
[*]   IP: large routing tables
[ ] IP: kernel level autoconfiguration
<*> IP: tunneling
<*> IP: GRE tunnels over IP
[ ]   IP: broadcast GRE over IP
[ ] IP: multicast routing
[ ] IP: ARP daemon support (EXPERIMENTAL)

[ ] IP: TCP Explicit Congestion Notification support
[*] IP: TCP syncookie support (disabled per default)
  IP: Netfilter Configuration --->
<*> The IPv6 protocol (EXPERIMENTAL)
[*]   IPv6: enable EUI-64 token format
[*]   IPv6: disable provider based addresses
< > Kernel httpd acceleration (EXPERIMENTAL)
[ ] Asynchronous Transfer Mode (ATM) (EXPERIMENTAL)

< > The IPX protocol
< > Appletalk protocol support
< > DECnet Support
< > 802.1d Ethernet Bridging
< > CCITT X.25 Packet Layer (EXPERIMENTAL)
< > LAPB Data Link Driver (EXPERIMENTAL)
[ ] 802.2 LLC (EXPERIMENTAL)
[ ] Frame Diverter (EXPERIMENTAL)
< > Acorn Econet/AUN protocols (EXPERIMENTAL)
< > WAN router
[ ] Fast switching (read help!)
[ ] Forwarding between high speed interfaces
QoS and/or fair queueing --->

La gestione relativa a IPTables, dopo aver attivato la voce

[*] Network packet filtering (replaces ipchains)

diventa accessibile in un menu` separato:

<*> Connection tracking (required for masq/NAT)
<*> FTP protocol support
<*> Userspace queueing via NETLINK (EXPERIMENTAL)
<*> IP tables support (required for filtering/masq/NAT)
<*> limit match support
<*> MAC address match support
<*> netfilter MARK match support
<*> Multiple port match support
<*> TOS match support
<*> Connection state match support
<*> Unclean match support (EXPERIMENTAL)
<*> Owner match support (EXPERIMENTAL)
<*> Packet filtering
<*>   REJECT target support
<*>   MIRROR target support (EXPERIMENTAL)
<*> Full NAT
<*>   MASQUERADE target support
<*>   REDIRECT target support
<*> Packet mangling

<*> TOS target support
<*> MARK target support
<*> LOG target support

ATA/IDE/MFM/RLL support

La gestione di unita` a blocchi tradizionali ed economiche, ATA/ATAPI, viene inserita in un menu` apposito.

Eventualmente, si puo` arrivare a specificare dettagliatamente il tipo di integrato della propria interfaccia ATA, per sfruttare al massimo le sue caratteristiche, anche se questo non e` indispensabile.(6)

<*> Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
--- Please see Documentation/ide.txt for help/info on IDE drives
[ ] Use old disk-only driver on primary interface
<*> Include IDE/ATA-2 DISK support
[ ] Use multi-mode by default
< > PCMCIA IDE support
<*> Include IDE/ATAPI CDROM support
< > Include IDE/ATAPI TAPE support
<*> Include IDE/ATAPI FLOPPY support
<*> SCSI emulation support
--- IDE chipset support/bugfixes
[*] CMD640 chipset bugfix/support
[ ] CMD640 enhanced support
[ ] ISA-PNP EIDE support
[*] RZ1000 chipset bugfix/support
[*] Generic PCI IDE chipset support
[*] Sharing PCI IDE interrupts support
[*] Generic PCI bus-master DMA support
[ ] Boot off-board chipsets first support
[ ] OPTi 82C621 chipset enhanced support (EXPERIMENTAL)
[ ] Other IDE chipset support

SCSI support

Questa sezione riguarda la gestione del kernel delle unita` SCSI. Le interfacce SCSI non hanno uno standard comune come avviene nel caso di quelle ATA e derivate, per cui e` indispensabile includere il codice specifico per tutte le interfacce che si intendono utilizzare.

In certe situazioni puo` essere necessario abilitare la gestione della ?gestione generica? SCSI. In particolare, questo serve nel caso si preveda l'uso di un masterizzatore SCSI.

<*> SCSI support
--- SCSI support type (disk, tape, CD-ROM)
<*> SCSI disk support

(40) Maximum number of SCSI disks that can be loaded as modules
< > SCSI tape support
< > SCSI OnStream SC-x0 tape support
<*> SCSI CD-ROM support
[ ] Enable vendor-specific extensions (for SCSI CDROM)
(2) Maximum number of CDROM devices that can be loaded as modules
<*> SCSI generic support
--- Some SCSI devices (e.g. CD jukebox) support multiple LUNs
[*] Enable extra checks in new queueing code
[*] Probe all LUNs on each SCSI device
[*] Verbose SCSI error reporting (kernel size +=12K)
[*] SCSI logging facility
SCSI low-level drivers --->
PCMCIA SCSI adapter support --->

Network device support

Questa sezione riguarda la definizione delle interfacce di rete che si utilizzano, includendo anche interfacce logiche, che non corrispondono a componenti fisici veri e propri, oltre che alcuni protocolli utilizzati come tunnel nell'ambito di hardware che di solito non viene usato per le reti (per esempio il PPP con le porte seriali e il PLIP con le porte parallele).

[*] Network device support
ARCnet devices --->
<*> Dummy net driver support
< > Bonding driver support
< > EQL (serial line load balancing) support
<*> Universal TUN/TAP device driver support
< > Ethertap network tap (OBSOLETE)
< > General Instruments Surfboard 1000
Ethernet (10 or 100Mbit) --->
Ethernet (1000 Mbit) --->
[ ] FDDI driver support
[ ] HIPPI driver support (EXPERIMENTAL)
<*> PLIP (parallel port) support
<*> PPP (point-to-point protocol) support
[ ] PPP multilink support (EXPERIMENTAL)
<*> PPP support for async serial ports
<*> PPP support for sync tty ports
<*> PPP Deflate compression
<*> PPP BSD-Compress compression
< > PPP over Ethernet (EXPERIMENTAL)
<*> SLIP (serial line) support
[*] CSLIP compressed headers
[*] Keepalive and linefill
[*] Six bit SLIP encapsulation
Wireless LAN (non-hamradio) --->

Token Ring devices --->
[ ] Fibre Channel driver support
< > Red Creek Hardware VPN (EXPERIMENTAL)
< > Traffic Shaper (EXPERIMENTAL)
Wan interfaces --->
PCMCIA network device support --->

Character devices

Un dispositivo a caratteri e` quello che utilizza una comunicazione byte per byte e si contrappone a quello a blocchi con cui la comunicazione avviene attraverso l'uso di blocchi di byte di dimensione fissa.

Questo gruppo di opzioni serve a definire l'uso del terminale (che potrebbe anche essere escluso nel caso di kernel con funzioni specifiche), inteso come complesso di schermo e tastiera, delle porte seriali, delle porte parallele, dei mouse e altri componenti simili.

E` importante sottolineare il fatto che non si deve definire esplicitamente l'uso di un mouse seriale, perche' per questo e` sufficiente configurare la porta seriale corrispondente, mentre nel caso di mouse differenti, occorre indicare espressamente di cosa si tratta.

In questo gruppo di opzioni appare anche un elenco di schede grafiche particolari.

[*] Virtual terminal
[*]   Support for console on virtual terminal
<*> Standard/generic (8250/16550 and compatible UARTs) serial support
[*]   Support for console on serial port
[ ] Extended dumb serial driver options
[ ] Non-standard serial port support
[*] Unix98 PTY support
(256) Maximum number of Unix98 PTYs in use (0-2048)
<*> Parallel printer support
[ ]   Support for console on line printer
< > Support for user-space parallel port device drivers
I2C support --->
Mice --->
Joysticks --->
< > QIC-02 tape support
Watchdog Cards --->
< > Intel i8x0 Random Number Generator support
<*> /dev/nvram support
< > Enhanced Real Time Clock Support
< > Double Talk PC internal speech card support
< > Siemens R3964 line discipline
< > Applicom intelligent fieldbus card support
Ftape, the floppy tape device driver --->
< > /dev/agpgart (AGP Support)

[ ] Direct Rendering Manager (XFree86 DRI support)
PCMCIA character device support --->

Filesystems

Attraverso questa sezione si definiscono i tipi di file system che si vogliono gestire. In particolare, anche i file system virtuali, come /proc/ e /dev/pty/, vengono definiti qui.

Il file system standard dei sistemi GNU/Linux e` il tipo Second-extended, ovvero Ext2 o Ext3. La gestione di questo tipo di file system deve essere inclusa nel kernel, a meno che si stia cercando di produrre del codice specifico per un'applicazione particolare.

In questo gruppo di opzioni trovano posto anche quelle necessarie alla condivisione attraverso la rete, per esempio con il protocollo NFS.

E` interessante osservare che e` necessario specificare anche i sistemi di partizionamento dei dischi. In generale e` indispensabile la gestione delle partizioni tipiche dei sistemi Dos.

Infine, e` importante anche tenere in considerazione il tipo di codifica che si vuole poter utilizzare nell'ambito del file system. La codifica in questione riguarda il modo di rappresentare i nomi dei file, che potrebbe richiedere estensioni particolari. In generale viene abilitata la codifica ISO 8859-1, che e` quella piu` frequente nel mondo occidentale.

[*] Quota support
<*> Kernel automounter support
<*> Kernel automounter version 4 support (also supports v3)
< > Reiserfs support
< > ADFS file system support
< > Amiga FFS file system support (EXPERIMENTAL)
< > Apple Macintosh file system support (EXPERIMENTAL)
< > BFS file system support (EXPERIMENTAL)
<*> DOS FAT fs support
<*>   MSDOS fs support
<*>   UMSDOS: Unix-like file system on top of standard MSDOS fs
<*>   VFAT (Windows-95) fs support
< > EFS file system support (read only) (EXPERIMENTAL)
< > Compressed ROM file system support
< > Simple RAM-based file system support
<*> ISO 9660 CDROM file system support
[*]   Microsoft Joliet CDROM extensions
<*> Minix fs support
< > NTFS file system support (read only)
< > OS/2 HPFS file system support
[*] /proc file system support
[ ] /dev file system support (EXPERIMENTAL)
[*] /dev/pts file system for Unix98 PTYs

< > QNX4 file system support (read only) (EXPERIMENTAL)
< > ROM file system support
<*> Second extended fs support
< > System V and Coherent file system support (read only)
< > UDF file system support (read only)
< > UFS file system support (read only)
Network File Systems --->
Partition Types --->
Native Language Support --->

Quello che segue e` il menu` specifico per i file system di rete (come NFS):

<*> Coda file system support (advanced network fs)
< > InterMezzo file system support (experimental, replicating fs)
<*> NFS file system support
[*]   Provide NFSv3 client support
<*> NFS server support
[*]   Provide NFSv3 server support
<*> SMB file system support (to mount Windows shares etc.)
[ ]   Use a default NLS
< > NCP file system support (to mount NetWare volumes)

Console drivers

Questa sezione permette di definire le caratteristiche della console. In generale si tratta di affermare l'uso di una console VGA, cosa praticamente obbligatoria, salva la possibilita` di compilare un kernel per un sistema senza console.

[*] VGA text console
[*] Video mode selection support
< > MDA text console (dual-headed) (EXPERIMENTAL)
Frame-buffer support --->

Sound

Questo gruppo di opzioni consente di gestire le funzionalita` audio, specificando l'uso di una scheda audio particolare. Un gruppo importante di schede audio e` gestito da un modulo speciale, definito ?OSS?.

USB support

Questo gruppo di opzioni consente la gestione di adattatori USB e delle unita` periferiche relative.

<*> Support for USB
[ ] USB verbose debug messages
--- Miscellaneous USB options
[*] Preliminary USB device filesystem
[ ] Enforce USB bandwidth allocation (EXPERIMENTAL)
--- USB Controllers
<*> UHCI (Intel PIIX4, VIA, ...) support
<*> OHCI (Compaq, iMacs, OPTi, SiS, ALi, ...) support
--- USB Device Class drivers
< > USB Audio support
< > USB Bluetooth support (EXPERIMENTAL)
<*> USB Mass Storage support
< > USB Modem (CDC ACM) support
< > USB Printer support
--- USB Human Interface Devices (HID)
--- Input core support is needed for USB HID
--- USB Imaging devices
< > USB Kodak DC-2xx Camera support
< > USB Mustek MDC800 Digital Camera support (EXPERIMENTAL)
<*> USB Scanner support
< > Microtek X6USB scanner support (EXPERIMENTAL)
--- USB Multimedia devices
< > DABUSB driver
--- USB Network adaptors
< > PLUSB Prolific USB-Network driver (EXPERIMENTAL)
< > USB ADMtek Pegasus-based ethernet device support (EXPERIMENTAL)
< > NetChip 1080-based USB Host-to-Host Link (EXPERIMENTAL)
--- USB port drivers
< > USS720 parport driver
USB Serial Converter support --->
--- USB misc drivers
< > USB Diamond Rio500 support (EXPERIMENTAL)

Come fare per configurare correttamente il kernel che si vuole compilare

Il kernel Linux e` molto dinamico e il suo sviluppo prende spesso delle strade imprevedibili. Questa vitalita` e` molto importante per il futuro del software libero; senza di essa non ci sarebbe modo di usare domani le nuove tecnologie che verranno proposte. In questo senso, diventa difficile dare delle indicazioni precise e durature sul modo corretto di configurare il kernel prima della compilazione.

L'unica documentazione sicura sotto questo aspetto e` quella che si puo` consultare in modo contestuale quando si utilizza il comando make menuconfig, oppure make xconfig. Eventualmente, puo` essere utile sapere che le informazioni che si leggono li` sono contenute nel file Documentation/Configure.help (nell'ambito dei sorgenti). Segue un estratto di questo file:

Support for USB
CONFIG_USB
  Universal Serial Bus (USB) is a specification for a serial bus
  subsystem which offers higher speeds and more features than the
  traditional PC serial port. The bus supplies power to peripherals
  and allows for hot swapping. Up to 127 USB peripherals can be
  connected to a single USB port in a tree structure. The USB port is
  the root of the tree, the peripherals are the leaves and the inner
  nodes are special USB devices called hubs. Many newer PC's have USB
  ports and newer peripherals such as scanners, keyboards, mice,
  modems, and printers support the USB protocol and can be connected
  to the PC via those ports.

  Say Y here if your computer has a USB port and you want to use USB
  devices. You then need to say Y to at least one of "UHCI support" or
  "OHCI support" below (the type of interface that the USB hardware in
  your computer provides to the operating system) and then choose from
  among the drivers for USB peripherals. You may want to check out the
  information provided in Documentation/usb/ and especially the links
  given in Documentation/usb/usb-help.txt.

  This code is also available as a module (= code which can be
  inserted in and removed from the running kernel whenever you want).
  The module will be called usbcore.o. If you want to compile it as a
  module, say M here and read Documentation/modules.txt.

Quando si parte da zero, e` sufficiente accertarsi di eliminare il file .config, che comunque viene eliminato con il comando make mrproper. In questo modo, il programma che guida alla configurazione del kernel offre gia` le risposte piu` ovvie alle domande che fa. Naturalmente e` sempre necessario leggere le prime volte il testo delle spiegazioni disponibili, fino a che si raggiunge una dimestichezza adeguata al tipo di esigenze che si hanno.

Come la documentazione interna suggerisce spesso, nella directory Documentation/ sono contenuti tanti file di testo contenenti spiegazioni particolareggiate rispetto a problemi specifici della configurazione. A questo punto dovrebbe essere evidente che non si puo` configurare e compilare un kernel se non si conosce minimamente la lingua inglese.

Questo tipo di lavoro passa poi necessariamente per una lunga serie di tentativi falliti (avendo cura di conservare i file .config, per poter ripartire almeno dall'ultima configurazione tentata). Tuttavia, il principiante non deve pensare di essersi messo involontariamente nei guai, perche' queste difficolta` riguardano tutti, anche gli esperti, proprio perche' la dinamicita` nello sviluppo del kernel Linux porta continue novita`.

cap 6) Moduli

Quando si ha una certa dimestichezza con la ricompilazione del kernel, si potrebbe considerare l'utilizzo dei moduli come una complicazione inutile. Tuttavia, ci sono situazioni in cui l'uso dei moduli e` una necessita`, prima tra tutte l'installazione di GNU/Linux attraverso le distribuzioni che fanno uso di moduli per ottenere un dischetto di avvio unico, oppure per ridurne la scelta a pochi. D'altro canto, la conoscenza dei meccanismi legati alla gestione dei moduli del kernel e` utile per sfruttare un sistema gia` ben organizzato dalla propria distribuzione GNU/Linux.

Questo capitolo, pur trovandosi in una posizione iniziale di questo documento, non e` rivolto ai principianti che potrebbero trovare alcuni punti particolarmente complessi (come il problema del disco RAM iniziale). Tuttavia, quando si installa GNU/Linux, potrebbe essere necessario conoscere l'uso dei parametri di alcuni moduli.

Tabella 31.1. Riepilogo dei programmi e dei file per la gestione dei moduli.
Nome Descrizione

insmod Carica manualmente i moduli del kernel.
rmmod Scarica manualmente i moduli del kernel.
lsmod Elenca i moduli caricati nel kernel.
depmod Rigenera il file delle dipendenze tra i moduli.
modprobe Carica un modulo rispettando le dipendenze.
/etc/conf.modules Configurazione dei moduli utilizzati.
kerneld Programma demone per il carico e lo scarico automatico dei moduli.

In questo capitolo vengono mostrati anche i parametri di alcuni tipi di moduli, mentre nel capitolo 32 ne sono riepilogati altri in modo sintetico.

Gestione dei moduli

I moduli del kernel sono porzioni di questo che possono essere caricate in memoria quando se ne presenta la necessita` e scaricate subito dopo. I moduli del kernel Linux sono quello che in altri sistemi viene definito driver. Nella sezione 29.1.2 e` gia` stato descritto in che modo possa essere compilato un kernel di questo tipo e anche come generare i moduli relativi.

Se si dispone di una distribuzione GNU/Linux organizzata con un kernel modulare, e` consigliabile sfruttare quel kernel gia` predisposto, assieme ai suoi moduli.

Funzionamento in breve

Il minimo indispensabile per attivare e disattivare i moduli e` costituito da due programmi di servizio specifici: insmod e rmmod. Il primo serve per caricare i moduli, il secondo per scaricarli.

L'operazione di caricamento dei moduli deve essere fatta tenendo presente le eventuali dipendenze che ci possono essere. Per esempio, se il modulo ?C? richiede la presenza del modulo ?B?, il quale a sua volta richiede la presenza del modulo ?A?, occorre caricare ordinatamente i moduli ?A?, ?B? e ?C?. Nello stesso modo, lo scarico dei moduli puo` essere fatto solo se si rispettano le dipendenze. Nel caso appena descritto, per scaricare il modulo ?A? occorre prima scaricare ?C? e ?B?.

Aspetto e collocazione

I moduli sono generalmente file che terminano con l'estensione .o e si collocano al di sotto della directory /lib/modules/versione/, dove la versione si riferisce al kernel per il quale sono stati predisposti. Per esempio, /lib/modules/2.4.2/, si riferisce ai moduli del kernel 2.4.2.

Per facilitare l'individuazione e il caricamento dei moduli, viene creato generalmente un file, modules.dep, nella directory iniziale di questi, attraverso il programma depmod.

```
# depmod -a
```

Generalmente questo comando viene inserito nella procedura di inizializzazione del

sistema, in modo da aggiornare sistematicamente questo file.

Il file contiene l'elenco dei moduli presenti, con l'indicazione precisa delle dipendenze. L'esempio seguente mostra il caso del modulo della scheda di rete NE2000, ne.o, il quale dipende dal modulo 8390.o.

```
/lib/modules/2.4.2/kernel/drivers/net/ne.o: (segue)
  /lib/modules/2.4.2/kernel/drivers/net/8390.o
```

Caricamento guidato

Invece di caricare i moduli con il programma insmod, che richiede attenzione nella sequenza di caricamento a causa delle dipendenze, si puo` utilizzare modprobe che si avvale del file modules.dep e si arrangia a caricare tutto quello che serve nel modo corretto. Per esempio, utilizzando il comando seguente,

```
# modprobe ne
```

si ottiene prima il caricamento del modulo 8390.o e successivamente di ne.o.

Parametri

Come accade gia` con i kernel monolitici, alcuni dispositivi possono essere individuati e gestiti correttamente solo se si forniscono delle informazioni aggiuntive. Per questo, alcuni moduli richiedono l'indicazione di parametri composti dalla sintassi

```
simbolo[=valore]
```

Quando si caricano i moduli di questo tipo con insmod e` necessario fornire anche i parametri, nella parte finale della riga di comando. La stessa cosa vale per modprobe, solo che in questo caso si puo` realizzare un file di configurazione, /etc/conf.modules, contenente le informazioni sui parametri dei moduli utilizzati e altre indicazioni eventuali.

Per esempio, attraverso la riga seguente del file /etc/conf.modules, si vuole specificare che l'indirizzo di I/O del dispositivo relativo al modulo ne.o e` 30016.

```
options ne io=0x0300
```

Gestione automatica

Gli strumenti e la configurazione descritti nelle sezioni precedenti, possono essere gestiti in modo automatico attraverso il demone, kerneld, oppure in modo integrato nei kernel 2.2.* e successivi (in questo caso si parla del thread kmod).

Quando e` utilizzato kerneld, questo viene avviato generalmente dalla procedura di inizializzazione del sistema, dopo che e` stato eseguito depmod per la rigenerazione del file delle dipendenze tra i moduli. Nel momento in cui il kernel ha bisogno di una

funzione che non trova al suo interno, prova a interrogare kerneld, il quale a sua volta di avvale di modprobe per il caricamento del modulo necessario. In questa situazione, il carico e lo scarico dei moduli in modo manuale non e` piu` necessario: e` kerneld che provvede. Ci possono essere comunque situazioni in cui il meccanismo non funziona, ma resta sempre la possibilita` di usare modprobe in quei casi.

L'automazione della gestione dei moduli richiede che il file /etc/conf.modules sia configurato correttamente per quei dispositivi che si intendono usare.

Come accennato, i kernel 2.2.* incorporano kmod che si sostituisce alle funzionalita` fondamentali di kerneld. In questo caso, si puo` osservare la presenza del file virtuale /proc/sys/kernel/modprobe, il cui scopo e` quello di informare il kernel sulla posizione in cui si trova il programma modprobe. Per questo, la procedura di inizializzazione del sistema dovrebbe provvedere a definirlo utilizzando un comando simile a quello seguente.

```
# echo "/sbin/modprobe" > /proc/sys/kernel/modprobe
```

Dal momento che kmod non e` efficiente quanto kerneld, occorre provvedere (ammesso che lo si voglia) a eliminare periodicamente i moduli che non sono piu` utilizzati. Per farlo si puo` usare il sistema Cron attraverso una direttiva simile a quella seguente che si riferisce al file crontab dell'utente root:

```
0-59/5 * * * * /sbin/rmmod -a
```

Nel caso si voglia utilizzare il file crontab di sistema, ovvero /etc/crontab, la cosa cambia leggermente, come nell'esempio seguente:

```
0-59/5 * * * * root /sbin/rmmod -a
```

 # insmod

insmod [opzioni] file_oggetto [simbolo=valore...]

insmod permette di caricare un modulo nel kernel. Il nome del modulo puo` essere indicato specificando il nome del file completo di estensione ed eventualmente di percorso (path), oppure specificando semplicemente il nome del file del modulo senza l'estensione: in questo ultimo caso, insmod cerca il file (con la sua estensione naturale) all'interno delle directory standard per i moduli.

Quando nel kernel e` attivato il supporto del kernel daemon e il demone kerneld e` in funzione, oppure si tratta di un kernel ? 2.2.* ed e` disponibile kmod, non dovrebbe essere necessario l'utilizzo di insmod per caricare i moduli.

Esempi

```
# insmod /lib/modules/2.0.30/net/plip.o
```

Attiva il modulo plip rappresentato dal file /lib/modules/2.0.30/net/plip.o.

insmod plip

Come nell'esempio precedente, ma si lascia a insmod il compito di cercare il file.

 # rmmod

rmmod [opzioni] modulo...

rmmod permette di scaricare uno o piu` moduli dal kernel, sempre che questi non siano in uso e non ci siano altri moduli caricati che vi fanno riferimento.

Nella riga di comando vengono indicati i nomi dei moduli e non i nomi dei file dei moduli. Se vengono indicati piu` moduli, questi vengono scaricati nell'ordine in cui appaiono.

Se viene usata l'opzione -a, vengono scaricati tutti i moduli che risultano essere inattivi, senza bisogno di specificarli nella riga di comando.

Quando nel kernel e` attivato il supporto del kernel daemon e il demone kerneld e` in funzione, non dovrebbe essere necessario l'utilizzo di rmmod per scaricare i moduli. Se invece si utilizza un kernel ? 2.2.* con il supporto per kmod, si deve provvedere periodicamente a eseguire il comando rmmod -a.

rmmod e` in realta` solo un collegamento a insmod che quindi cambia il suo comportamento quando viene avviato utilizzando quel nome.

Esempi

rmmod plip

Scarica il modulo plip.

rmmod -a

Scarica tutti i moduli inutilizzati.

$ lsmod

lsmod permette di visualizzare la situazione sull'utilizzo dei moduli. Le stesse informazioni ottenibili da lsmod si possono avere dal contenuto del file /proc/modules. Utilizzando lsmod si ottiene una tabellina di tre colonne:

- Module -- rappresenta il nome del modulo;
- Pages -- rappresenta il numero di pagine di memoria utilizzate (una pagina e` un blocco di 4 Kibyte);
- Used by -- rappresenta l'utilizzo da parte di altri moduli (lo zero indica che non e`

utilizzato).

Esempi

Supponendo di avere appena caricato il modulo plip si puo` ottenere quanto segue:

lsmod[Invio]

```
Module       Pages   Used by
plip         3       0
```

 # depmod

depmod [opzioni]

depmod serve a generare un file di dipendenze tra i moduli, che poi viene utilizzato da modprobe per caricarli rispettando le dipendenze. Precisamente, viene creato il file /lib/modules/versione/modules.dep.

Alcune opzioni

-a [versione] | --all [versione]

Scandisce tutti i moduli della versione del kernel in funzione. depmod viene utilizzato generalmente con questa opzione per creare il file delle dipendenze. Se si desidera creare il file delle dipendenze per i moduli di un'altra versione di kernel, si puo` specificare espressamente tale versione.

-s | --system-log

Invia le segnalazioni di errore al registro del sistema.

Esempi

depmod -a

Genera il file /lib/modules/versione/modules.dep.

depmod -a 2.1.99

Genera il file /lib/modules/2.1.99/modules.dep, riferito appunto ai moduli del kernel della versione 2.1.99.

```
if [ -x /sbin/depmod ]
then
   echo "Analisi delle dipendenze tra i moduli"
   /sbin/depmod -a
fi
```

Si tratta di un pezzo di uno degli script della procedura di inizializzazione del sistema, in cui si avvia la generazione del file delle dipendenze tra i moduli solo se il programma esiste.

 # modprobe

modprobe [opzioni] file_oggetto [simbolo=valore...]

modprobe e` un programma fatto per agevolare il caricamento dei moduli del kernel. Quando viene usato senza l'indicazione di alcuna opzione, cioe` solo con il nome del modulo e l'eventuale aggiunta dei parametri, modprobe carica prima i moduli necessari a soddisfare le dipendenze, quindi provvede al caricamento del modulo richiesto. Se l'operazione fallisce, tutti i moduli superflui vengono scaricati nuovamente.

Tra le altre cose, modprobe permette di tentare il caricamento del modulo ?giusto? a partire da un gruppo, quando non si conosce bene quale sia il modulo adatto a un certo tipo di dispositivo o di servizio. Per farlo e` necessario indicare il tipo di modulo e il modello. Il tipo e` rappresentato dalla directory che lo contiene (fs/, misc/, net/, scsi/, ecc.) e il modello si esprime utilizzando i consueti caratteri jolly (? e *).

modprobe fa uso di un file di configurazione, attraverso cui e` possibile modificare le sue impostazioni predefinite e in particolare si possono definire i parametri normali necessari ad alcuni tipi di moduli. Il file in questione e` /etc/conf.modules.

Alcune opzioni

-a | --all

Carica tutti i moduli (generalmente non viene utilizzata questa opzione).

-c | --show-conf

Emette la configurazione attuale per la gestione dei moduli; cio` comprende sia la parte predefinita che il contenuto del file di configurazione (/etc/conf.modules).

-l | --list

Elenca i moduli disponibili.

-r | --remove

Scarica i moduli dal kernel, eliminando anche quelli che erano stati caricati per soddisfare le dipendenze, sempre che cio` sia possibile.

-t tipo modello | --type tipo modello

Permette di definire il tipo di modulo, attraverso il nome usato per la directory che lo contiene (fs/, misc/, net/, scsi/,...) e attraverso un modello espresso con dei caratteri jolly.

Utilizzando questa opzione, occorre fare attenzione a proteggere i caratteri jolly dall'interpretazione da parte della shell, per esempio con l'uso di apici singoli o doppi. Esempi

modprobe -l

Elenca tutti i moduli disponibili.

modprobe -l -t net

Elenca tutti i moduli di tipo net, cioe` quelli contenuti nella directory omonima.

modprobe -l -t net '3c*'

Elenca i moduli il cui nome inizia per 3c, di tipo net; in pratica elenca i moduli delle schede di rete 3Com.

modprobe -c

Emette la configurazione attuale della gestione dei moduli di modprobe.

modprobe plip

Carica il modulo /lib/modules/versione/kernel/drivers/net/plip.o.

modprobe -t net 'p*'

Tenta di caricare un modulo che inizi con la lettera p, dalla directory /lib/modules/versione/kernel/drivers/net/.

 # kerneld

kerneld [debug] [keep] [delay=secondi] [type=numero_messaggio]

Nei kernel 2.0.*, kerneld e` il demone che si occupa di gestire automaticamente i moduli, sempre che il kernel sia stato compilato in modo da includere questa possibilita` di gestione automatizzata. kerneld viene attivato normalmente attraverso la procedura di inizializzazione del sistema, dopo che e` stato rigenerato il file delle dipendenze tra i moduli. In pratica, l'avvio potrebbe avvenire nel modo seguente:

```
if [ -x /sbin/depmod ]
then
   echo "Analisi delle dipendenze tra i moduli"
   /sbin/depmod -a
fi

#...
```

```
if [ -x /sbin/kerneld ]
then
   echo "Avvio del demone per la gestione dei moduli"
   /sbin/kerneld
fi
```

Vedere eventualmente la pagina di manuale kerneld(1).

Configurazione dei moduli

Il file /etc/conf.modules permette di configurare il comportamento di modprobe. Le righe vuote e quanto preceduto dal simbolo # viene ignorato. Le righe possono essere continuate utilizzando la barra obliqua inversa (\) alla fine, subito prima del codice di interruzione di riga.

Le righe di questo file vengono interpretate attraverso una shell, permettendo cosi` di utilizzare le tecniche di sostituzione fornite comunemente da queste, come i caratteri jolly e con la sostituzione di comando.

Questo file di configurazione puo` contenere diversi tipi di direttive; nelle sezioni seguenti se ne mostrano solo alcune. Per la descrizione completa si veda la pagina di manuale depmod(1).

In linea di massima, si possono accumulare piu` direttive dello stesso tipo.

alias

alias alias modulo_reale

La direttiva alias permette di indicare un nome alternativo a un nome di un modulo reale. Cio` puo` essere utile a vario titolo e in ogni caso sono stabiliti molti alias gia` in modo predefinito. Lo si puo` osservare con il comando seguente:

modprobe -l[Invio]

```
...
# Aliases
alias binfmt-2 binfmt_aout
alias binfmt-0107 binfmt_aout
...
alias block-major-2 floppy
alias block-major-3 ide-probe
...
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
...
alias dos msdos
```

```
...
alias iso9660 isofs
...
alias plip0 plip
alias plip1 plip
alias ppp0 ppp
alias ppp1 ppp
...
```

Per esempio, si puo` osservare che e` possibile fare riferimento al modulo isofs anche attraverso il nome iso9660. Tuttavia, gli alias non sono semplicemente di aiuto agli ?smemorati?, ma anche una necessita`. Si osservi la configurazione seguente tratta da un ipotetico file /etc/conf.modules.

```
alias eth0 ne
...
```

L'alias eth0 (ovvero la prima interfaccia Ethernet) permette di fare in modo che quando si configura l'interfaccia di rete con ifconfig, kerneld sappia quale modulo avviare: in questo caso ne.

Ogni modulo ha le sue particolarita`, quindi deve essere valutata caso per caso l'opportunita` di utilizzare un alias adatto a qualche scopo.

options

options nome simbolo=valore...

La direttiva options permette di definire i parametri di utilizzo di un modulo, identificato attraverso il suo nome reale, oppure un alias. Per esempio,

```
alias eth0 ne
options ne io=0x300 irq=11
```

definisce che il modulo ne (Ethernet NE2000) dovra` essere utilizzato per un dispositivo che si raggiunge con il canale di I/O 30016 e l'IRQ 11.

Attraverso questa direttiva si indicano solo le opzioni che non possono essere determinate altrimenti dal sistema. Questo significa che non e` necessaria una riga options per tutti i dispositivi che si intende utilizzare attraverso i moduli.

cap 7) Introduzione ai processi di elaborazione

Un programma singolo, nel momento in cui viene eseguito, e` un processo. La nascita di un processo, cioe` l'avvio di un programma, puo` avvenire solo tramite una richiesta da parte di un altro processo gia` esistente. Si forma quindi una sorta di gerarchia dei processi organizzata ad albero. Il processo principale (root) che genera tutti gli altri, e` quello dell'eseguibile init che a sua volta e` attivato direttamente dal kernel.

In linea di principio, il programma avviato dal kernel come processo principale, puo` essere qualunque cosa, anche una shell (tenendo conto, comunque, che il kernel predilige l'eseguibile /sbin/init), ma in tal caso si tratta di applicazioni specifiche e non di un sistema standard.

Qui si preferisce utilizzare il nome Init per identificare il processo principale, tenendo conto che questo si concretizza generalmente nell'eseguibile init.

Tabella dei processi

Il kernel gestisce una tabella dei processi che serve a tenere traccia del loro stato. In particolare sono registrati i valori seguenti:

- il nome dell'eseguibile in funzione;
- gli eventuali argomenti passati all'eseguibile al momento dell'avvio attraverso la riga di comando;
- il numero di identificazione del processo;
- il numero di identificazione del processo che ha generato quello a cui si fa riferimento;
- il nome del dispositivo di comunicazione se il processo e` controllato da un terminale;
- il numero di identificazione dell'utente;
- il numero di identificazione del gruppo;

/proc/

Il kernel Linux rende disponibile i dati della tabella dei processi attraverso un file system virtuale montato nella directory /proc/. Dalla presenza di questo file system virtuale

dipendono la maggior parte dei programmi che si occupano di gestire i processi.

In particolare, a partire da questa directory se ne diramano altre, tante quanti sono i processi in esecuzione, ognuna identificata dal numero del processo stesso. Per esempio, /proc/1/ contiene una serie di file virtuali che rappresentano lo stato del processo numero uno, ovvero Init che e` sempre il primo a essere messo in funzione.

Nascita e morte di un processo

Come e` gia` stato accennato, la nascita di un processo, cioe` l'avvio di un programma, puo` avvenire solo tramite una richiesta da parte di un altro processo gia` esistente, utilizzando la chiamata di sistema fork(). Per esempio, quando si avvia un programma attraverso il terminale, e` l'interprete dei comandi (la shell) che genera il processo corrispondente.

Quando un processo termina, lo fa attraverso la chiamata di sistema exit(), trasformandosi in un cosiddetto zombie. E` poi il processo che lo ha generato che si deve occupare di eliminarne le tracce.

Il processo genitore, per avviare l'eliminazione dei suoi processi zombie, deve essere avvisato che ne esiste la necessita` attraverso un segnale SIGCHLD. Questo segnale viene inviato proprio dalla funzione di sistema exit(), ma se il meccanismo non funziona come previsto, si puo` inviare manualmente un segnale SIGCHLD al processo genitore. In mancanza d'altro, si puo` far terminare l'esecuzione del processo genitore stesso.

Il processo che termina potrebbe avere avviato a sua volta altri processi (figli). In tal caso, questi vengono affidati al processo numero uno, cioe` Init.

Core dump

A volte, l'interruzione di un processo provoca il cosiddetto scarico della memoria o core dump. In pratica si ottiene un file nella directory corrente, contenente l'immagine del processo interrotto. Per tradizione, questo file e` denominato core, in onore dei primo tipo di memoria centrale che sia stato utilizzato: la memoria a nuclei magnetici, ovvero core memory.

Questi file servono a documentare un incidente di funzionamento e a permetterne l'analisi attraverso strumenti diagnostici opportuni. Solitamente possono essere cancellati tranquillamente.

La proliferazione di questi file va tenuta sotto controllo: di solito non ci si rende conto se un processo interrotto ha generato o meno lo scarico della memoria. Ogni tanto vale la pena di fare una ricerca all'interno del file system per rintracciare questi file, come nell'esempio seguente:

```
# find / -name core -type f -print
```

Cio` che conta e` di non confondere core con spazzatura: ci possono essere dei file chiamati core per qualche motivo, che nulla hanno a che fare con lo scarico della memoria.

Comunicazione tra processi

Nel momento in cui l'attivita` di un processo dipende da quella di un altro ci deve essere una forma di comunicazione tra i due. Cio` viene definito IPC, o Inter process communication, ma questa definizione viene confusa spesso con un tipo particolare di comunicazione definito IPC di System V.

I metodi utilizzati normalmente sono di tre tipi: invio di segnali, pipe e IPC di System V.

Segnali

I segnali sono dei messaggi elementari che possono essere inviati a un processo, permettendo a questo di essere informato di una condizione particolare che si e` manifestata e di potersi uniformare. I programmi possono essere progettati in modo da intercettare questi segnali, allo scopo di compiere alcune operazioni prima di adeguarsi agli ordini ricevuti. Nello stesso modo, un programma potrebbe anche ignorare completamente un segnale, o compiere operazioni diverse da quelle che sarebbero prevedibili per un tipo di segnale determinato. Segue un elenco dei segnali piu` importanti.

SIGINT
E` un segnale di interruzione intercettabile, inviato normalmente attraverso la tastiera del terminale, con la combinazione [Ctrl+c], al processo che si trova a funzionare in primo piano (foreground). Di solito, il processo che riceve questo segnale viene interrotto.

SIGQUIT
E` un segnale di interruzione intercettabile, inviato normalmente attraverso la tastiera del terminale, con la combinazione [Ctrl+\], al processo che si trova a funzionare in primo piano. Di solito, il processo che riceve questo segnale viene interrotto.

SIGTERM
E` un segnale di conclusione intercettabile, inviato normalmente da un altro processo. Di solito, provoca la conclusione del processo che ne e` il destinatario.

SIGKILL
E` un segnale di interruzione non intercettabile, che provoca la conclusione immediata del processo. Non c'e` modo per il processo destinatario di eseguire alcuna operazione di salvataggio o di scarico dei dati.

SIGHUP
E` un segnale di aggancio che rappresenta l'interruzione di una comunicazione. In particolare, quando un utente esegue un logout, i processi ancora attivi avviati eventualmente sullo sfondo (background) ricevono questo segnale. Puo` essere generato anche a causa della ?morte? del processo controllante.

L'utente ha a disposizione in particolare due mezzi per inviare segnali ai programmi:

la combinazione di tasti [Ctrl+c] che di solito genera l'invio di un segnale SIGINT al processo in esecuzione sul terminale o sulla console attiva;
l'uso di kill (programma o comando interno di shell) per inviare un segnale particolare a un processo stabilito.

Pipe

Attraverso la shell e` possibile collegare piu` processi tra loro in una pipeline, come nell'esempio seguente, in modo che lo standard output di uno sia collegato direttamente con lo standard input del successivo.

```
$ cat mio_file | sort | lpr
```

Ogni connessione tra un processo e il successivo, evidenziata dalla barra verticale (pipe), si comporta come un serbatoio provvisorio di dati ad accesso FIFO (First in first out -- il primo a entrare e` il primo a uscire).

E` possibile creare esplicitamente dei serbatoi FIFO di questo genere, in modo da poterli gestire senza dover fare ricorso alle funzionalita` della shell. Questi, sono dei file speciali definiti proprio ?FIFO? e vengono creati attraverso il programma mkfifo. Nell'esempio seguente viene mostrata una sequenza di comandi con i quali, creando due file FIFO, si puo` eseguire la stessa operazione indicata nella pipeline vista poco sopra.

```
$ mkfifo fifo1 fifo2
```

Crea due file FIFO: fifo1 e fifo2.

```
$ cat mio_file >> fifo1 &
```

Invia mio_file a fifo1 senza attendere (&).

```
$ sort < fifo1 >> fifo2 &
```

Esegue il riordino di quanto ottenuto da fifo1 e invia il risultato a fifo2 senza attendere (&).

```
$ lpr < fifo2
```

Accoda la stampa di quanto ottenuto da fifo2.

I file FIFO, data la loro affinita` di funzionamento con le pipeline gestite dalla shell, vengono anche chiamati pipe con nome, contrapponendosi a quelle normali che a volte vengono dette pipe anonime.

Quando un processo viene interrotto all'interno di una pipeline di qualunque tipo, il processo che inviava dati a quello interrotto riceve un segnale SIGPIPE e si interrompe a sua volta. Dall'altra parte, i processi che ricevevano dati da quello interrotto, vedono concludersi il flusso di questi dati e terminano la loro esecuzione in modo naturale. Quando questa situazione viene segnalata, si potrebbe ottenere il messaggio broken pipe.

IPC di System V

L'IPC di System V e` un sistema di comunicazione tra processi sofisticato che permette di gestire code di messaggi, semafori e memoria condivisa.

Scheduling e priorita`

La gestione simultanea dei processi e` ottenuta normalmente attraverso la suddivisione del tempo di CPU, in maniera tale che a turno ogni processo abbia a disposizione un breve intervallo di tempo di elaborazione. Il modo con cui vengono regolati questi turni e` lo scheduling, ovvero la pianificazione di questi processi.

La maggiore o minore percentuale di tempo di CPU che puo` avere un processo e` regolata dalla priorita` espressa da un numero. Il numero che rappresenta una priorita` deve essere visto al contrario di come si e` abituati di solito: un valore elevato rappresenta una priorita` bassa, cioe` meno tempo a disposizione, mentre un valore basso (o negativo) rappresenta una priorita` elevata, cioe` piu` tempo a disposizione.

Sotto questo aspetto diventa difficile esprimersi in modo chiaro: una bassa priorita` si riferisce al numero che ne esprime il valore o alle risorse disponibili? Si puo` solo fare attenzione al contesto per capire bene il significato di cio` che si intende.

La priorita` di esecuzione di un processo viene definita in modo autonomo da parte del sistema e puo` essere regolata da parte dell'utente sommandovi il cosiddetto valore nice. Di conseguenza, un valore nice positivo aumenta il valore della priorita`, mentre un valore negativo lo diminuisce.

Privilegi dei processi

Nei sistemi operativi Unix c'e` la necessita` di distinguere i privilegi concessi agli utenti, definendo un nominativo e un numero identificativo riferito all'utente e al gruppo (o ai gruppi) a cui questo appartiene. L'utente fisico e` rappresentato virtualmente dai processi che lui stesso mette in esecuzione; pertanto, un'informazione essenziale riferita ai processi e` quella che stabilisce l'appartenenza a un utente e a un gruppo. In altri termini, ogni processo porta con se' l'informazione del numero UID e del numero GID, in base ai quali ottiene i privilegi relativi e gli viene concesso o meno di compiere le operazioni per cui e` stato avviato.

cap 8) Procedura di inizializzazione del sistema (System V)

Quando GNU/Linux viene avviato, il kernel si prende cura di avviare il processo iniziale, Init, a partire dal quale vengono poi generati tutti gli altri. Di solito si utilizza un meccanismo di inizializzazione derivato dallo UNIX System V.

Init determina quali siano i processi da avviare successivamente, in base al contenuto di /etc/inittab il quale a sua volta fa riferimento a una serie di script contenuti normalmente all'interno della directory /etc/rc.d/ o in un'altra analoga.

All'interno di /etc/inittab si distinguono azioni diverse in funzione del livello di esecuzione (run level), di solito un numero da zero a sei. Per convenzione, il livello zero identifica le azioni necessarie per fermare l'attivita` del sistema, in modo da permetterne lo spegnimento; il livello sei riavvia il sistema; il livello uno mette il sistema in condizione di funzionare in modalita` monoutente.

Le distribuzioni GNU/Linux piu` sofisticate e confortevoli permettono di configurare il sistema attraverso dei programmi che guidano l'utente. Cio` significa che questi programmi sono in grado di produrre automaticamente script e file di configurazione tradizionali, ma per fare questo devono gestire un proprio sistema di file di configurazione che non appartiene allo standard generale.

L'organizzazione della procedura di inizializzazione del sistema e dei livelli di esecuzione costituisce il punto su cui si distinguono maggiormente le distribuzioni GNU/Linux. Benche' alla fine si tratti sempre della stessa cosa, il modo di strutturare e di collocare gli script e` molto diverso da una distribuzione all'altra. Quando si acquista piu` esperienza, ci si accorge che queste differenze non sono poi un grosso problema, ma all'inizio e` importante comprendere e accettare che cio` che si usa (la propria distribuzione GNU/Linux) mostra un'interpretazione della soluzione del problema e non il risultato definitivo.

Init

Init e` il processo principale che genera tutti gli altri. All'avvio del sistema legge il file /etc/inittab il quale contiene le informazioni per attivare gli altri processi necessari, compresa la gestione dei terminali. Per prima cosa viene determinato il livello di esecuzione iniziale, ottenendo l'informazione dalla direttiva initdefault di /etc/inittab. Quindi vengono attivati i processi essenziali al funzionamento del sistema e infine i processi che combaciano con il livello di esecuzione attivato.

 # init

init [opzioni]

L'eseguibile init puo` essere invocato dall'utente root durante il funzionamento del sistema, per cambiare il livello di esecuzione, oppure ottenere il riesame del suo file di configurazione (/etc/inittab).

Opzioni

-t secondi

Stabilisce il numero di secondi di attesa prima di cambiare il livello di esecuzione. In mancanza si intende 20 secondi.

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

Un numero da zero a sei stabilisce il livello di esecuzione a cui si vuole passare.

a | b | c

Una lettera a, b o c richiede di eseguire soltanto i processi indicati all'interno di /etc/inittab che hanno un livello di esecuzione pari alla lettera specificata. In pratica, una lettera non indica un livello di esecuzione vero e proprio, in quanto si tratta di una possibilita` di configurazione del file /etc/inittab per definire i cosiddetti livelli ?a richiesta? (on demand).

Q | q

Richiede di riesaminare il file /etc/inittab (dopo che questo e` stato modificato).

S | s

Richiede di passare alla modalita` monoutente, ma non e` pensato per essere utilizzato direttamente, in quanto per questo si preferisce selezionare il livello di esecuzione numero uno.
Esempi

init 1

Pone il sistema al livello di esecuzione uno: monoutente.

init 0

Pone il sistema al livello di esecuzione zero: arresto del sistema. Equivale (in linea di massima) all'esecuzione di shutdown -h now.

init 6

Pone il sistema al livello di esecuzione sei: riavvio. Equivale (in linea di massima) all'esecuzione di shutdown -r now.

/etc/inittab

Il file inittab descrive quali processi vengono avviati al momento dell'avvio del sistema e durante il funzionamento normale di questo. Init, il processo principale, distingue diversi livelli di esecuzione, per ognuno dei quali puo` essere stabilito un gruppo diverso di processi da avviare.

La struttura dei record che compongono le direttive di questo file puo` essere schematizzata nel modo seguente:

id:livelli_di_esecuzione:azione:processo

1. id -- e` una sequenza unica di due caratteri che identifica il record (la riga) all'interno di inittab;
2. livelli_di_esecuzione -- elenca i livelli di esecuzione con cui l'azione indicata deve essere eseguita;
3. azione -- indica l'azione da eseguire;
4. processo -- specifica il processo da eseguire.

Se il nome del processo inizia con un simbolo +, Init non eseguira` l'aggiornamento di /var/run/utmp e /var/log/wtmp per quel processo; cio` e` utile quando il processo stesso provvede da solo a questa operazione (la descrizione del significato e dell'importanza di questi due file si trova nel capitolo 53).

Il penultimo campo dei record di questo file, identifica l'azione da compiere. Questa viene rappresentata attraverso una parola chiave, come descritto dall'elenco seguente.

respawn
Quando il processo termina, viene riavviato.

wait
Il processo viene avviato una volta (sempre che il livello di esecuzione lo consenta) e Init attende che termini prima di eseguirne degli altri.

once
Il processo viene eseguito una volta quando il livello di esecuzione lo consente.

boot
Il processo viene eseguito al momento dell'avvio del sistema. Il campo del livello di esecuzione viene ignorato.

bootwait
Il processo viene eseguito al momento dell'avvio del sistema e Init attende la fine del processo prima di proseguire. Il campo del livello di esecuzione viene ignorato.

off
Non fa alcunche'.

ondemand
Si tratta dei record ?a richiesta? che vengono presi in considerazione quando viene richiamato Init seguito da una lettera a, b o c, che rappresentano appunto tre possibili livelli di esecuzione on demand. a, b o c non sono livelli di esecuzione, ma solo un modo per selezionare una serie di processi on demand indicati all'interno del file inittab.

initdefault
Permette di definire il livello di esecuzione predefinito per l'avvio del sistema. Se non viene specificato, Init richiede l'inserimento di questo valore attraverso la console.

sysinit
Il processo viene eseguito al momento dell'avvio del sistema, prima di quelli indicati come boot e bootwait. Il campo del livello di esecuzione viene ignorato.

powerwait
Il processo viene eseguito quando Init riceve il segnale SIGPWR che indica un problema con l'alimentazione elettrica. Init attende la fine del processo prima di proseguire.

powerfail
Il processo viene eseguito quando Init riceve il segnale SIGPWR che indica un problema con l'alimentazione elettrica. Init non attende la fine del processo.

powerokwait
Il processo viene eseguito quando Init ha ricevuto il segnale SIGPWR, che indica un problema con l'alimentazione elettrica, ed e` anche presente il file /etc/powerstatus contenente la parola OK. Cio` significa che l'alimentazione elettrica e` tornata allo stato di normalita`.

ctrlaltdel
Il processo viene eseguito quando Init riceve il segnale SIGINT. Cio` significa che e` stata premuta la combinazione di tasti [Ctrl+Alt+Canc] ([Ctrl+Alt+Del] nelle tastiere inglesi).

kbrequest
Il processo viene eseguito quando Init riceve un segnale dal gestore della tastiera che

sta a indicare la pressione di una combinazione speciale di tasti sulla tastiera della console.

Il secondo campo, quello dei livelli di esecuzione, puo` contenere diversi caratteri che stanno a indicare diversi livelli di esecuzione possibili. Per esempio, la stringa 123 indica che il processo specificato verra` eseguito indifferentemente per tutti i livelli di esecuzione da uno a tre. Questo campo puo` contenere anche una lettera dell'alfabeto: a, b o c che sta a indicare un livello a richiesta. Nel caso di azioni del tipo sysinit, boot e bootwait, il campo del livello di esecuzione viene ignorato.

Esempi

Negli esempi seguenti, si mostra prima un record del file /etc/inittab e quindi, sotto, la sua descrizione.

id:5:initdefault:

Definisce il livello di esecuzione iniziale: cinque.

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

Inizializzazione del sistema: e` la prima cosa a essere eseguita dopo l'avvio del sistema stesso. In pratica viene avviato lo script /etc/rc.d/rc.sysinit (Red Hat).

l1:1:wait:/etc/rc.d/rc 1

Indica di eseguire /etc/rc.d/rc, con l'argomento 1, nel caso in cui il livello di esecuzione sia pari a uno: singolo utente (Red Hat)

rc:123456:wait:/etc/rc.d/rc.M

Indica lo script (/etc/rc.d/rc.M) da eseguire per tutti i livelli di esecuzione da uno a sei (Slackware).

ca::ctrlaltdel:/sbin/shutdown -t5 -rfn now

Indica il programma da eseguire in caso di pressione della combinazione [Ctrl+Alt+Canc]. Il livello di esecuzione non viene indicato perche' e` indifferente (Slackware).

l0:0:wait:/etc/rc.d/rc 0

Indica di eseguire /etc/rc.d/rc, con l'argomento 0, nel caso in cui il livello di esecuzione sia pari a zero: arresto del sistema (Red Hat).

l6:6:wait:/etc/rc.d/rc 6

Indica di eseguire /etc/rc.d/rc, con l'argomento 6, nel caso in cui il livello di esecuzione

sia pari a sei: riavvio (Red Hat).

pf::powerfail:/sbin/shutdown -f +5 "THE POWER IS FAILING"

Indica il programma da eseguire quando si verifica un problema con l'alimentazione elettrica (Slackware).

pg:0123456:powerokwait:/sbin/shutdown -c "THE POWER IS BACK"

Indica il programma da eseguire se l'alimentazione elettrica torna normale prima del completamento del processo avviato quando si era verificato il problema (Slackware).

1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

Si tratta dell'elenco di console virtuali utilizzabili. La prima si attiva per tutti i livelli di esecuzione da uno a cinque, le altre solo per i livelli superiori a uno. In questo caso e` mingetty a essere responsabile dell'attivazione delle console virtuali (Red Hat).

s1:45:respawn:/sbin/agetty 19200 ttyS0 vt100

Indica l'attivazione di un terminale connesso sulla prima porta seriale. Si attiva solo con i livelli di esecuzione quattro o cinque (Slackware).

d2:45:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS1 vt100

Indica l'attivazione di un terminale remoto connesso via modem sulla seconda porta seriale. Si attiva solo con i livelli di esecuzione quattro o cinque (Slackware).

x:5:respawn:/usr/bin/X11/xdm -nodaemon

Nel caso il livello di esecuzione sia pari a cinque, esegue /usr/bin/X11/xdm che si occupa di avviare una procedura di accesso (login) all'interno dell'ambiente grafico X (Red Hat).

Script della procedura di inizializzazione del sistema

La prima differenza importante che distingue le varie distribuzioni GNU/Linux sta nell'organizzazione degli script della procedura di inizializzazione del sistema. Il punto di riferimento comune e` Init con il suo /etc/inittab, dal quale si intende quali siano il comandi avviati in presenza di un livello di esecuzione determinato; quello che c'e` dopo costituisce il problema piu` grosso.

Volendo semplificare molto le cose, si puo` pensare al fatto che ci dovrebbe essere una directory specifica, contenente un gruppetto di script utilizzato esclusivamente per questi

scopi. Volendo fare un esempio preciso, nel caso della distribuzione Red Hat, tale directory e` /etc/rc.d/; dagli esempi visti riguardo al file /etc/inittab, si puo` notare che all'interno di questa directory si trovano gli script rc.sysinit e rc.

Per una convenzione diffusa, lo script rc.local che dovrebbe essere contenuto in questa directory (/etc/rc.d/, o altra directory a seconda della propria distribuzione GNU/Linux), viene eseguito alla fine della procedura di inizializzazione del sistema e viene lasciato a disposizione dell'amministratore che puo` modificarlo come crede. Volendo fare un'associazione con il sistema Dos, si potrebbe paragonare questo script al file AUTOEXEC.BAT.(1)

Le motivazioni che spingono a un'impostazione differente di questi script della procedura di inizializzazione del sistema, possono essere varie. Anche la collocazione di tale directory e` controversa, a cominciare dal fatto che la directory /etc/ non dovrebbe contenere programmi e nemmeno script. Infatti, a questo proposito, la distribuzione SuSE colloca questi script nella directory /sbin/init.d/.

Procedura di attivazione e disattivazione dei servizi

Gli script della procedura di inizializzazione del sistema hanno il compito di avviare il sistema operativo e di fermarlo, attivando e disattivando tutti i servizi necessari, cioe` intervenendo nell'avvio e nella conclusione del funzionamento dei demoni relativi.

Si puo` intuire che non sia possibile realizzare uno o piu` script del genere per avviare tutti i tipi di demone che possono essere presenti nel proprio sistema, anche perche' ci possono essere dei servizi installati che pero` non si vogliono gestire. Di conseguenza, nella situazione piu` banale, quando si intende installare e gestire un nuovo servizio, occorre anche modificare la procedura di inizializzazione del sistema per attivare il demone relativo e per disattivarlo nel momento dell'arresto del sistema. Una cosa del genere puo` andare bene per una persona esperta, ma si tratta sempre di un'impostazione piuttosto scomoda.

Nel tempo si e` diffuso uno standard per risolvere questo problema, ed e` cio` che viene descritto nelle sezioni seguenti.

Script di avvio e interruzione di un servizio

Secondo una convenzione diffusa, per facilitare l'avvio e la conclusione dei servizi si definisce una directory specifica, che potrebbe essere /etc/rc.d/init.d/, o /etc/init.d/, o ancora /sbin/init.d/, all'interno della quale si possono inserire degli script che hanno una sintassi uniforme.

nome_servizio {start|stop}

In pratica, il nome dello script tende a corrispondere a quello del servizio che si intende controllare; l'argomento costituito dalla parola chiave start fa si` che lo script avvii il servizio, mentre la parola chiave stop serve a concluderlo.

Questi script possono essere piu` o meno raffinati, per esempio possono accettare anche altri tipi di ordini (come restart, allo scopo di riavviare un servizio), ma la cosa piu` importante e` che dovrebbero evitare di avviare dei doppioni, controllando prima di avviare qualcosa, se per caso questo risulta gia` attivo. Naturalmente, un servizio puo` essere ottenuto con l'avvio di diversi programmi demone e in questo e` molto comodo tale sistema di script specifici.

A titolo di esempio viene mostrato come potrebbe essere composto uno script del genere, per l'avvio del servizio ipotetico denominato pippo, che si avvale del programma omonimo per gestirlo. Per semplicita`, non vengono indicati accorgimenti particolari per controllare che il servizio sia gia` attivo o meno.

```
#!/bin/sh
#
# init.d/pippo {start|stop|restart}
#

# Analisi dell'argomento usato nella chiamata.
case "$1" in
  start)
        echo -n "Avvio del servizio Pippo: "
        /usr/sbin/pippo &
        echo
        ;;
  stop)
        echo -n "Disattivazione del servizio Pippo: "
        killall pippo
        echo
        ;;
  restart)
        killall -HUP pippo
        ;;
  *)
        echo "Utilizzo: pippo {start|stop|restart}"
        exit 1
esac

exit 0
```

Lo scopo e la vera utilita` di questi script sta nel facilitare una standardizzazione della procedura di inizializzazione del sistema; tuttavia si puo` intuire la possibilita` di sfruttarli anche per attivare e disattivare manualmente un servizio, senza intervenire direttamente sui programmi relativi.

Collegamenti simbolici per ogni livello di esecuzione

Procedendo intuitivamente, si potrebbe pensare di fare in modo che la procedura di inizializzazione del sistema, provveda a eseguire tutti gli script di controllo dei servizi, utilizzando l'argomento start all'avvio e l'argomento stop allo spegnimento. Una cosa del genere e` molto semplice da realizzare, ma si pongono due problemi: alcuni servizi potrebbero essere a disposizione, senza che la procedura di inizializzazione del sistema debba avviarli automaticamente; inoltre la sequenza di attivazione e di disattivazione dei

servizi potrebbe essere importante.

In pratica, si utilizza un meccanismo molto semplice: si predispongono tante directory quanti sono i livelli di esecuzione gestiti attraverso il file /etc/inittab. Queste directory hanno il nome rcn.d/, dove n rappresenta il numero del livello di esecuzione corrispondente. La loro collocazione effettiva potrebbe essere /etc/rcn.d/, /etc/rc.d/rcn.d/ o anche /sbin/init.d/rcn.d/. All'interno di queste directory si inseriscono dei collegamenti simbolici che puntano agli script descritti nella sezione precedente, in modo che siano presenti i riferimenti ai servizi desiderati per ogni livello di esecuzione (distinto in base alla directory rcn.d/ particolare).

I nomi di questi collegamenti iniziano in modo speciale: Knn e Snn. I collegamenti che iniziano con la lettera ?S? (Start) servono per individuare gli script da utilizzare per l'attivazione dei servizi, vengono avviati con l'argomento start, in ordine alfabetico, in base alla sequenza fissata con le due cifre numeriche successive che servono proprio a distinguerne la sequenza. I collegamenti che iniziano con la lettera ?K? (Kill) servono per individuare gli script da utilizzare per la disattivazione dei servizi, vengono avviati con l'argomento stop, anche questi in ordine alfabetico.

Ecco, a titolo di esempio, cosa potrebbe contenere una di queste directory.

```
lrwxrwxrwx 1 root root 13 13:39 K15gpm -> ../init.d/gpm
lrwxrwxrwx 1 root root 13 13:39 K60atd -> ../init.d/atd
lrwxrwxrwx 1 root root 15 13:39 K60crond -> ../init.d/crond
lrwxrwxrwx 1 root root 16 13:39 K96pcmcia -> ../init.d/pcmcia
lrwxrwxrwx 1 root root 17 13:39 S01kerneld -> ../init.d/kerneld
lrwxrwxrwx 1 root root 17 13:39 S10network -> ../init.d/network
lrwxrwxrwx 1 root root 15 13:39 S15nfsfs -> ../init.d/nfsfs
lrwxrwxrwx 1 root root 16 13:39 S20random -> ../init.d/random
lrwxrwxrwx 1 root root 16 13:39 S30syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root 14 13:39 S50inet -> ../init.d/inet
lrwxrwxrwx 1 root root 18 13:39 S75keytable -> ../init.d/keytable
```

Osservando i servizi syslog e inet, si puo` notare il numero attribuito per l'avvio, che serve a fare in modo che il servizio syslog sia avviato prima di inet.

Sempre a titolo di esempio, viene mostrato un pezzo di uno script, per una shell Bourne o derivata, fatto per scandire un elenco di collegamenti del genere, allo scopo di attivare e di disattivare i servizi, a partire dai collegamenti contenuti nella directory /etc/rc.d/rc3.d/. Per un lettore inesperto, questo potrebbe essere un po' difficile da leggere, ma l'esempio viene aggiunto per completare l'argomento.

```
#!/bin/sh

#...

# Attivazione dei servizi del livello di esecuzione 3.
```

```
for I in /etc/rc.d/rc3.d/K*;
do
   # Disattiva il servizio.
   $I stop
done

for I in /etc/rc.d/rc3.d/S*;
do
   # Attiva il servizio.
   $I start
done
```

In pratica, prima si disattivano i servizi corrispondenti ai collegamenti che iniziano con la lettera ?K?, quindi si attivano quelli che hanno la lettera ?S?. Si puo` intuire che le directory rc0.d/ e rc6.d/ contengano prevalentemente, o esclusivamente, riferimenti che iniziano con la lettera ?K?, dal momento che i livelli di esecuzione corrispondenti portano all'arresto del sistema o al suo riavvio.

cap 9) Situazione dei processi

Le informazioni sulla situazione dei processi vengono ottenute a partire dalla tabella dei processi messa a disposizione dal kernel. Dal momento che il meccanismo attraverso cui queste informazioni possono essere ottenute dal kernel non e` standardizzato per tutti i sistemi Unix, questi programmi che ne permettono la consultazione hanno raramente un funzionamento conforme.

Il meccanismo utilizzato in particolare dal kernel Linux e` quello del file system virtuale montato nella directory /proc/. A questo proposito, e` il caso di osservare che il pacchetto dei programmi di servizio che permettono di conoscere lo stato dei processi e` denominato Procps, in riferimento a questa particolarita` del kernel Linux.

Nome Descrizione
ps Elenca i processi in esecuzione.
pstree Elenca i processi in esecuzione in modo strutturato.
top Mostra l'utilizzo delle risorse da parte dei processi a intervalli regolari.
fuser Elenca i processi che utilizzano file determinati.

uptime Informa sul tempo di funzionamento e sul carico medio.
free Genera un rapporto stringato sull'uso della memoria.

Process status

Il controllo dello stato dei processi esistenti avviene fondamentalmente attraverso l'uso di ps, pstree e top. Il primo mostra un elenco di processi e delle loro caratteristiche, il secondo un albero che rappresenta la dipendenza gerarchica dei processi e il terzo l'evolversi dello stato di questi.

ps e pstree rappresentano la situazione di un istante: il primo si presta per eventuali rielaborazioni successive, mentre il secondo e` particolarmente adatto a seguire l'evoluzione di una catena di processi, specialmente quando a un certo punto si verifica una transizione nella proprieta` dello stesso (UID).

```
# ps[Invio]

  PID TTY STAT  TIME COMMAND
  374   1 S    0:01 /bin/login -- root
  375   2 S    0:00 /sbin/mingetty tty2
  376   3 S    0:00 /sbin/mingetty tty3
  377   4 S    0:00 /sbin/mingetty tty4
  380   5 S    0:00 /sbin/mingetty tty5
  382   1 S    0:00 -bash
  444  p0 S    0:00 su
  445  p0 S    0:00 bash
  588  p0 R    0:00 ps

$ pstree -u -p[Invio]

init(1)-+-crond(173)
        |-gpm(314)
        |-inetd(210)
        |-kerneld(23)
        |-kflushd(2)
        |-klogd(162)
        |-kswapd(3)
        |-login(374)---bash(382)
        |-login(381)---bash(404,daniele)---startx(415)---xinit(416)-...
        |-lpd(232)
        |-mingetty(380)
        |-mingetty(375)
        |-mingetty(376)
        |-mingetty(377)
        |-named(221)
        |-nfsiod(4)
        |-nfsiod(5)
        |-nfsiod(6)
```

```
|-nfsiod(7)
|-portmap(184)
|-rpc.mountd(246)
|-rpc.nfsd(255)
|-rxvt(433)---bash(434,daniele)---su(444,root)---bash(445)
|-rxvt(436)---bash(437,daniele)---pstree(608)
|-sendmail(302)
|-snmpd(198)
|-syslogd(153)
`-update(379)
```

top invece e` un programma che impegna un terminale (o una finestra di terminale all'interno del sistema grafico) per mostrare costantemente l'aggiornamento della situazione. Si tratta quindi di un controllo continuo, con l'aggiunta pero` della possibilita` di interferire con i processi inviandovi dei segnali o cambiandone il valore nice.

```
 10:13pm  up 58 min,  5 users,  load average: 0.09, 0.03, 0.01
67 processes: 65 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  5.9% user,  0.7% system,  0.0% nice, 93.5% idle
Mem:   62296K av,  60752K used,   1544K free,  36856K shrd,  22024K buff
Swap: 104416K av,      8K used, 104408K free                16656K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME
COMMAND
  588 root      16   0  6520 6520  1368 R       0  5.1 10.4   0:02 X
  613 daniele    6   0   736  736   560 R       0  1.3  1.1   0:00 top
  596 daniele    1   0  1108 1108   872 S       0  0.1  1.7   0:00 fvwm2
    1 root       0   0   388  388   336 S       0  0.0  0.6   0:08 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kswapd
   82 root       0   0   352  352   300 S       0  0.0  0.5   0:00 kerneld
  139 root       0   0   448  448   364 S       0  0.0  0.7   0:00 syslogd
  148 root       0   0   432  432   320 S       0  0.0  0.6   0:00 klogd
  159 daemon     0   0   416  416   340 S       0  0.0  0.6   0:00 atd
  170 root       0   0   484  484   400 S       0  0.0  0.7   0:00 crond
  181 bin        0   0   336  336   268 S       0  0.0  0.5   0:00 portmap
  204 root       0   0   404  404   336 S       0  0.0  0.6   0:00 inetd
```

Intestazioni

I programmi che visualizzano la situazione dei processi, utilizzano spesso delle sigle per identificare alcune caratteristiche. La tabella 41.2 ne descrive alcune.

Tabella 41.2. Elenco di alcune delle sigle utilizzate dai programmi che permettono di consultare lo stato dei processi in esecuzione.

Sigla Descrizione
UID Il numero di UID dell'utente proprietario del processo.

PID Il numero del processo, cioe` il PID.
PPID Il PID del processo genitore (quello da cui ha avuto origine).
USER Il nome dell'utente proprietario del processo.
PRI La priorita` del processo.
NI Il valore nice.
SIZE La dimensione dell'immagine del processo in memoria (virtuale).
RSS La dimensione della memoria RAM utilizzata effettivamente.
SWAP La dimensione della memoria virtuale utilizzata.
SHARE La quantita` di memoria condivisa utilizzata dal processo.
WCHAN L'evento per cui il processo e` in attesa.
STAT Lo stato del processo.
TT Il terminale, se il processo ne utilizza uno.
TIME Il tempo totale di utilizzo della CPU.
CTIME Il tempo di CPU sommando anche l'utilizzo da parte dei processi figli.
COMMAND Il comando utilizzato per avviare il processo.

In particolare, lo stato del processo rappresentato dalla sigla STAT, viene descritto da una o piu` lettere alfabetiche il cui significato viene riassunto nella tabella 41.3.

Tabella 41.3. Lo stato del processo espresso attraverso una o piu` lettere alfabetiche.
Lettera Stato
R In funzione (residente in memoria).
S In pausa o dormiente.
D In pausa non interrompibile.
T Sospeso.
Z Zombie.
W Non utilizza memoria (e` spostato completamente nella memoria virtuale).
N Ha un valore nice positivo (in pratica e` rallentato).

$ ps

ps [opzioni] [pid...]

Visualizza un elenco dei processi in corso di esecuzione. Se non viene specificato diversamente, si ottiene solo l'elenco dei processi che appartengono all'utente. Dopo le opzioni possono essere indicati esplicitamente i processi (in forma dei numeri PID) in modo da ridurre a loro l'elenco ottenuto.

Tabella 41.4. Elenco di alcune delle chiavi di ordinamento utilizzabili con l'opzione O, oppure --sort di ps.

Chiave Chiave Descrizione
c cmd Nome dell'eseguibile.
C cmdline Riga di comando completa.
o session Numero di sessione.
p pid PID.
P ppid PPID.
r rss RSS (memoria residente utilizzata).

t tty Terminale.
T start_time Orario di inizio del processo.
U uid UID.
u user Nominativo dell'utente
y priority Priorita`.

Alcune opzioni

Le opzioni rappresentate da un carattere singolo, possono iniziare eventualmente con un trattino, come avviene nella maggior parte dei comandi Unix, ma si tratta di un'eccezione, dal momento che il programma ps standard non le utilizza.

l

Emette un elenco lungo, composto in sostanza da piu` elementi informativi.

u

Formato utente: viene indicato in particolare l'utente a cui appartiene ogni processo e l'ora di inizio in cui il processo e` stato avviato.

f

Visualizza la dipendenza gerarchica tra i processi in modo semplificato.

a

Visualizza anche i processi appartenenti agli altri utenti.

r

Emette l'elenco dei soli processi in esecuzione effettivamente, escludendo cosi` quelli che per qualunque motivo sono in uno stato di pausa.

h

Elimina l'intestazione dall'elenco. Puo` essere utile quando si vuole elaborare in qualche modo l'elenco.

tx

Permette di ottenere l'elenco dei processi associati al terminale x. Per identificare un terminale, si puo` utilizzare il nome del file di dispositivo corrispondente, senza il percorso precedente (/dev/), oppure la sigla ottenuta dal nome eliminando il prefisso tty.

e

Mostra l'ambiente particolare del processo dopo la riga di comando.

w

Se la riga e` troppo lunga consente la visualizzazione di una riga in piu`: l'opzione puo` essere indicata piu` volte in modo da specificare quante righe aggiuntive possono essere utilizzate.

O[+|-]chiave[[+|-]chiave]...

--sort=[+|-]chiave[,[+|-]chiave]...

Permette di ottenere un risultato ordinato in base alle chiavi di ordinamento specificate. Le chiavi di ordinamento sono composte da una sola lettera nel caso si usi l'opzione O, mentre sono rappresentate da una parola nel caso dell'opzione --sort.

Il segno + (sottinteso) indica un ordinamento crescente, mentre il segno - indica un ordinamento decrescente. Le chiavi di ordinamento sono indicate simbolicamente in base all'elenco (parziale) visibile nella tabella 41.4.
Esempi

$ ps

Elenca i processi appartenenti all'utente che da` il comando.

$ ps a l

Elenca tutti i processi utilizzando un formato piu` ampio in modo da fornire piu` dettagli sui processi.

$ ps a r

Elenca tutti i processi in funzione escludendo quelli in pausa.

$ ps a l OUr

Elenca tutti i processi in formato allargato e riordinato per UID (numero utente) e quindi in base alla dimensione residente in memoria dei processi.

$ ps a l --sort=uid,rss

Equivalente all'esempio precedente.

$ pstree

pstree [opzioni] [PID | utente]

Visualizza uno schema ad albero dei processi in corso di esecuzione. E` possibile specificare un numero di processo (PID), oppure il nome di un utente per limitare

l'analisi. Di solito, quando da uno stesso genitore si diramano diversi processi con lo stesso nome, questi vengono raggruppati. Per cui:

init---4*[agetty]

rappresenta un gruppo di quattro processi agetty, tutti discendenti da Init.

Alcune opzioni

-a

Mostra tutta la riga di comando e non solo il nome del processo.

-c

Disabilita l'aggregazione dei processi con lo stesso nome derivanti dallo stesso genitore.

-h

Evidenzia il processo corrente e i suoi predecessori (antenati).

-l

Visualizza senza troncare le righe troppo lunghe.

-p

Mostra i PID.

-u

Mostra la transizione degli UID, quando da un genitore appartenente a un certo utente, viene generato un processo che appartiene a un altro.

$ top

top [opzioni]

Visualizza la situazione sull'utilizzo delle risorse di sistema attraverso una tabella dell'attivita` principale della CPU, cioe` dei processi che la impegnano maggiormente. Lo schema viene aggiornato a brevi intervalli, di conseguenza, impegna un terminale. Durante il suo funzionamento, top accetta dei comandi espressi con un carattere singolo.

Alcune opzioni

-d secondi_di_dilazione

Permette di specificare l'intervallo di tempo in secondi che viene lasciato trascorrere tra un aggiornamento e l'altro della tabella. Se non viene indicato questo argomento, l'intervallo di tempo tra gli aggiornamenti della tabella e` di cinque secondi.

-q

Permette all'utente root di richiedere un aggiornamento della tabella in modo continuo, senza intervalli di pausa.

-s

Disabilita la possibilita` di utilizzare alcuni comandi in modo interattivo. Puo` essere utile quando si vuole lasciare funzionare top in un terminale separato evitando incidenti.

-i

Permette di visualizzare anche i processi inattivi o zombie.

-c

Permette di visualizzare la riga di comando, invece del solo nome del programma.
Comandi interattivi

top accetta una serie di comandi interattivi, espressi da un carattere singolo.

h | ?

La lettera h o il simbolo ? fanno apparire un breve riassunto dei comandi e lo stato delle modalita` di funzionamento.

k

Permette di inviare un segnale a un processo che verra` indicato successivamente. Se il segnale non viene specificato, viene inviato SIGTERM.

i

Abilita o disabilita la visualizzazione dei processi inattivi e dei processi zombie.

n | #

Cambia la quantita` di processi da visualizzare. Il numero che esprime questa quantita` viene richiesto successivamente. Il valore predefinito di questa quantita` e` zero, che corrisponde al numero massimo in base alle righe a disposizione sullo schermo (o sulla finestra) del terminale.

q

Termina l'esecuzione di top.

r

Permette di modificare il valore nice di un processo determinato. Dopo l'inserimento della lettera r, viene richiesto il PID del processo su cui agire e il valore nice. Un valore nice positivo peggiora le prestazioni di esecuzione di un processo, mentre un valore negativo, che pero` puo` essere attribuito solo dall'utente root, migliora le prestazioni. Se non viene specificato il valore nice, si intende 10.

S

Attiva o disattiva la modalita` di visualizzazione cumulativa, con la quale, la statistica sull'utilizzo di risorse da parte di ogni processo, tiene conto anche di quello dei processi figli.

s

Cambia la durata, espressa in secondi, dell'intervallo tra un aggiornamento e l'altro dei valori visualizzati. L'utente root puo` attribuire il valore zero che implica un aggiornamento continuo. Il valore predefinito di questa durata e` di cinque secondi.

f | F

Permette di aggiungere o eliminare alcuni campi nella tabella dei processi.

Accesso ai file

A volte e` importante conoscere se un file e` utilizzato da qualche processo. Per questo si utilizza il programma fuser che e` in grado di dare qualche informazione aggiuntiva del modo in cui tale file viene utilizzato.

 # fuser

fuser [opzioni] file...

Il compito normale di fuser e` quello di elencare i processi che utilizzano i file indicati come argomento. In alternativa, fuser permette anche di inviare un segnale ai processi che utilizzano un gruppo di file determinato, con l'opzione -k.

fuser si trova normalmente nella directory /usr/sbin/, ma puo` essere utilizzato anche dagli utenti comuni per buona parte delle sue funzionalita`.

Quando si utilizza fuser per ottenere l'elenco dei processi che accedono a file determinati, i numeri di questi processi sono abbinati a una lettera che indica il modo in cui accedono:

- c directory corrente;
- e eseguibile in esecuzione;

- f file aperto (spesso questa lettera non viene mostrata affatto);
- r directory radice;
- m file mappato in memoria o libreria condivisa.

fuser restituisce il valore zero quando tra i file indicati come argomento ne esiste almeno uno che risulta utilizzato da un processo.

Alcune opzioni

-a

Mostra tutti i file indicati nell'argomento, anche se non sono utilizzati da alcun processo. Normalmente, fuser mostra solo i file in uso.

-k

Invia un segnale ai processi. Se non viene specificato diversamente attraverso l'opzione -segnale, si utilizza il segnale SIGKILL.

-segnale

Permette di specificare il segnale da inviare con l'opzione -k. In pratica, si tratta di un trattino seguito dal segnale espresso in forma numerica o in forma simbolica (per esempio -TERM).

-l

Elenca i nomi dei segnali conosciuti.

-m

Utilizzando questa opzione puo` essere indicato solo un nome di file, il quale puo` essere un file di dispositivo, riferito a un'unita` di memorizzazione montata nel file system, o una directory che costituisce il punto di innesto della stessa. Quello che si ottiene e` l'indicazione di tutti i processi che accedono a quella unita` di memorizzazione.

-u

Viene aggiunta l'indicazione dell'utente proprietario di ogni processo.

-v

Mostra una tabellina dei processi abbinati ai file, in forma piu` chiara rispetto alla visualizzazione normale.

-s

Disabilita qualunque emissione di informazioni. Viene utilizzato quando tutto cio` che

conta e` il solo valore restituito dal programma.
Esempi

```
# fuser *
```

Mostra i processi che accedono ai file della directory corrente.

```
# fuser -k /usr/games/*
```

Elimina tutti i processi che utilizzano file nella directory /usr/games/.

Uno script puo` utilizzare fuser nel modo seguente per verificare che un file non sia utilizzato da alcun processo prima di eseguire una qualche azione su di esso.

```
#!/bin/bash

MIO_FILE=./mio_file

if fuser -s $MIO_FILE
then
   echo "Il file $MIO_FILE e` in uso";
else
   # esegue qualche azione sullo stesso
   ...
fi
```

Informazioni riepilogative

Oltre alle informazioni dettagliate sui processi possono essere interessanti delle informazioni riassuntive dell'uso delle risorse di sistema. In particolare si usano uptime e free. Il primo permette di conoscere da quanto tempo e` in funzione il sistema senza interruzioni, il secondo mostra l'utilizzo della memoria.

```
$ uptime[Invio]

 5:10pm  up  2:21,  6 users,  load average: 0.45, 0.48, 0.41

$ free[Invio]

             total       used       free     shared    buffers     cached
Mem:         22724      22340        384      13884       3664       5600
-/+ buffers:            13076       9648
Swap:        16628       6248      10380
```

$ uptime

uptime [opzioni]

Emette una sola riga contenente:

- l'orario attuale;
- da quanto tempo e` in funzione il sistema;
- il carico medio di sistema dell'ultimo minuto, degli ultimi cinque minuti e degli ultimi 15 minuti.

$ free

free [opzioni]

free emette attraverso lo standard output una serie di informazioni relative alla memoria reale e virtuale (swap).

Alcune opzioni

-b

I valori vengono espressi in byte.

-k

I valori vengono espressi in kibibyte (simbolo: Kibyte) e si tratta della modalita` predefinita.

-t

Visualizza anche una riga contenente i totali.

-o

Disabilita il cosiddetto aggiustamento dei buffer. Normalmente, senza questa opzione, la memoria tampone, ovvero quella destinata ai buffer, viene considerata libera.

-s secondi_di_dilazione

Permette di ottenere un aggiornamento continuo a intervalli regolari stabiliti dal numero di secondi indicato come argomento. Questo numero puo` essere anche decimale.

Processi e shell

La shell e` l'intermediario tra l'utente e il sistema, pertanto e` il mezzo normale attraverso cui si puo` avviare e controllare un processo. Un comando impartito attraverso una shell puo` generare piu` di un processo, per esempio quando viene avviato un programma o uno script che avvia a sua volta diversi programmi, oppure quando si realizzano delle pipeline. Per questo motivo, quando si vuole fare riferimento all'attivita` derivata da un comando dato attraverso una shell, si parla di job e non di singoli processi.

Controllo dei job di shell

Attraverso alcune shell e` possibile gestire i job che in questo caso rappresentano raggruppamenti di processi generati da un solo comando.

La shell Bash e in generale le shell POSIX, oltre alla shell Korn e alla shell C, gestiscono i job. Nelle sezioni seguenti si fa riferimento al comportamento di Bash (in qualita` di shell POSIX), ma la maggior parte di quanto spiegato in queste sezioni vale anche per le shell Korn e C (ksh e csh).

Non si deve confondere un job di shell con un processo. Un processo e` un singolo eseguibile messo in funzione: se questo a sua volta avvia un altro eseguibile, viene generato un nuovo processo a esso associato. Un job di shell rappresenta tutti i processi che vengono generati da un comando impartito tramite la shell stessa. Basta immaginare cosa succede quando si utilizza una canalizzazione di programmi (pipe), dove l'output di un programma e` l'input del successivo.

Processi in primo piano e processi sullo sfondo

L'attivita` di un job puo` avvenire in primo piano (foreground) o sullo sfondo (background). Nel primo caso, il job impegna la shell e quindi anche il terminale, mentre nel secondo la shell e` libera da impegni e cosi` anche il terminale. Di conseguenza, non ha senso pretendere da un programma che richiede l'interazione continua con l'utente che possa anche funzionare sullo sfondo.

Se un programma richiede dati dallo standard input o ha la necessita` di emettere dati attraverso lo standard output o lo standard error, per poterlo avviare come job sullo sfondo, bisogna almeno provvedere a ridirigere l'input e l'output.

Avvio di un job sullo sfondo

Un programma e` avviato esplicitamente come job sullo sfondo quando alla fine della riga di comando viene aggiunto il simbolo &. Per esempio:

```
# make zImage > ~/make.msg &
```

avvia sullo sfondo il comando make zImage, per generare un kernel, dirigendo lo standard output verso un file per consentire un controllo successivo dell'esito della compilazione.

Dopo l'avvio di un programma come job sullo sfondo, la shell restituisce una riga contenente il numero del job e il numero del processo terminale generato da questo job

(PID). Per esempio:

[1] 173

rappresenta il job numero uno che termina con il processo 173.

Se viene avviato un job sullo sfondo, quando a un certo punto ha la necessita` di emettere dati attraverso lo standard output o lo standard error e questi non sono stati ridiretti, si ottiene una segnalazione simile a quella seguente:

[1]+ Stopped (tty output) pippo

Nell'esempio, il job avviato con il comando pippo si e` bloccato in attesa di poter emettere dell'output. Nello stesso modo, se viene avviato un job sullo sfondo che a un certo punto ha la necessita` di ricevere dati dallo standard input e questo non e` stato ridiretto, si ottiene una segnalazione simile alla seguente:

[1]+ Stopped (tty input) pippo

Sospensione di un job in primo piano

Se e` stato avviato un job in primo piano e si desidera sospenderne l'esecuzione, si puo` inviare attraverso la tastiera il carattere susp, che di solito si ottiene con la combinazione [Ctrl+z]. Il job viene sospeso e posto sullo sfondo. Quando un job viene sospeso, la shell genera una riga come nell'esempio seguente:

[1]+ Stopped pippo

dove il job pippo e` stato sospeso.

jobs

jobs [opzioni] [job]

Il comando di shell jobs, permette di conoscere l'elenco dei job esistenti e il loro stato. Per poter utilizzare il comando jobs occorre che non ci siano altri job in esecuzione in primo piano, di conseguenza, quello che si ottiene e` solo l'elenco dei job sullo sfondo.

Alcune opzioni

-l

Permette di conoscere anche i numeri PID dei processi di ogni job.

-p

Emette solo il numero PID del processo iniziale di ogni job.

Esempi

```
$ jobs
```

Si ottiene l'elenco normale dei job sullo sfondo. Nel caso dell'esempio seguente, il primo job e` in esecuzione, il secondo e` sospeso in attesa di poter emettere l'output, l'ultimo e` sospeso in attesa di poter ricevere l'input.

```
[1]   Running                 yes >/dev/null &
[2]-  Stopped (tty output)    mc
[3]+  Stopped (tty input)     unix2dos
```

Per comprendere l'utilizzo dell'opzione -l e dell'opzione -p , occorre avviare sullo sfondo qualche comando un po' articolato.

```
$ yes | cat | sort > /dev/null &[Invio]

[1] 594

$ yes | cat > /dev/null &[Invio]

[2] 596

$ jobs -l[Invio]

[1]-   592 Running                 yes
       593                 | cat
       594                 | sort >/dev/null &
[2]+   595 Running                 yes
       596                 | cat >/dev/null &
```

Come si puo` osservare, l'opzione -l permette di avere informazioni piu` dettagliate su tutti i processi che dipendono dai vari job presenti.

```
$ jobs -p
```

Si ottiene soltanto l'elenco dei numeri PID del processo iniziale di ogni job.

```
592
595
```

Riferimenti ai job

L'elenco di job ottenuto attraverso il comando jobs, mostra in particolare il simbolo + a fianco del numero del job attuale ed eventualmente il simbolo - a fianco di quello che diventerebbe il job attuale se il primo termina o viene comunque eliminato.

Il job attuale e` quello a cui si fa riferimento in modo predefinito tutte le volte che un comando richiede l'indicazione di un job e questo non viene fornito.

Di norma si indica un job con il suo numero preceduto dal simbolo %, ma si possono anche utilizzare altri metodi elencati nella tabella 43.1.

Tabella 43.1. Elenco dei parametri utilizzabili come riferimento ai job di shell.

Simbolo Descrizione
%n Il job con il numero indicato dalla lettera n.
%stringa Il job il cui comando inizia con la stringa indicata.
%?stringa Il job il cui comando contiene la stringa indicata.
%% Il job attuale.
%+ Il job attuale.
%- Il job precedente a quello attuale.

fg

fg [job]

Il comando fg porta in primo piano un job che prima era sullo sfondo. Se non viene specificato il job su cui agire, si intende quello attuale.

bg

bg [job]

Il comando bg permette di fare riprendere (sullo sfondo) l'esecuzione di un job sospeso. Cio` e` possibile solo se il job in questione non e` in attesa di un input o di poter emettere l'output. Se non si specifica il job, si intende quello attuale.

Quando si utilizza la combinazione [Ctrl+z] per sospendere l'esecuzione di un job, questo viene messo sullo sfondo e diviene il job attuale. Di conseguenza, e` normale utilizzare il comando bg subito dopo, senza argomenti, in modo da fare riprendere il job appena sospeso.

kill

kill [-s segnale | -segnale] [job]

Il comando kill funziona quasi nello stesso modo del programma omonimo. Di solito, non ci si rende conto che si utilizza il comando e non il programma. Il comando kill in particolare, rispetto al programma, permette di inviare un segnale ai processi di un job, indicando direttamente il job.

Quando si vuole eliminare tutto un job, a volte non e` sufficiente un segnale SIGTERM. Se necessario si puo` utilizzare il segnale SIGKILL (con prudenza pero`).

Esempi

```
$ kill -KILL %1
```

Elimina i processi abbinati al job numero uno, inviando il segnale SIGKILL.

```
$ kill -9 %1
```

Elimina i processi abbinati al job numero uno, inviando il segnale SIGKILL, espresso in forma numerica.

Cattura dei segnali

Attraverso il comando interno trap e` possibile catturare ed eventualmente attribuire un comando (comando interno, funzione o programma) a un segnale particolare. In questo modo uno script puo` gestire i segnali. L'esempio seguente ne mostra uno (trappola) in grado di reagire ai segnali SIGUSR1 e SIGUSR2 emettendo semplicemente un messaggio.

```
#!/bin/bash

   trap 'echo "Ho catturato il segnale SIGUSR1"' SIGUSR1
   trap 'echo "Ho catturato il segnale SIGUSR2"' SIGUSR2

   while [ 0 ]         # ripete continuamente
   do
      NULLA="ciao"    # esegue un'operazione inutile
   done
```

Supponendo di avere avviato lo script nel modo seguente,

```
$ trappola &[Invio]
```

e che il suo numero PID sia 1 234...

```
$ kill -s SIGUSR1 1234[Invio]

Ho catturato il segnale SIGUSR1

$ kill -s SIGUSR2 1234[Invio]

Ho catturato il segnale SIGUSR2
```

trap

trap [-l] [comando] [segnale]

Il comando espresso come argomento di trap viene eseguito quando la shell riceve il segnale o i segnali indicati. Se non viene fornito il comando, o se al suo posto si mette un trattino (-), tutti i segnali specificati sono riportati al loro valore originale (i valori che avevano al momento dell'ingresso nella shell), cioe` riprendono il loro significato normale. Se il comando fornito corrisponde a una stringa nulla, il segnale relativo viene ignorato dalla shell e dai comandi che questo avvia. Il segnale puo` essere espresso in forma verbale (per nome) o con il suo numero. Se il segnale e` EXIT, pari a zero, il comando viene eseguito all'uscita della shell.

Se viene utilizzato senza argomenti, trap emette la lista di comandi associati con ciascun numero di segnale.

Esempi

$ trap 'ls -l' SIGUSR1

Se la shell riceve un segnale SIGUSR1 esegue ls -l.

$ trap '' SIGUSR1

La shell e tutti i processi figli ignorano il segnale SIGUSR1.

cap 10) VI: cenni sull'utilizzo

Il programma piu` importante che e` necessario conoscere quando ci si avvicina a un nuovo sistema operativo e` quello che permette di creare e modificare i file di testo normale. Nel caso dei sistemi Unix, e` indispensabile conoscere VI, oltre ad altri applicativi simili che possono risultare piu` comodi da usare.

L'azione con la quale si indica l'intervento su un file del genere, viene espressa usualmente con la parola inglese editing, per cui alle volte questi programmi applicativi vengono identificati come degli editor.

VI

Nei sistemi Unix, l'applicativo piu` importante per la creazione e la modifica dei file di testo e` VI. E` piu` importante perche' onnipresente, soprattutto nei dischetti di emergenza, anche se non si tratta di un programma comodo da utilizzare.

VI ha una logica di funzionamento tutta sua che ne impedisce l'utilizzo a chi non abbia letto qualcosa al riguardo. L'intento e` quello di chiarire questa logica, almeno in parte, in modo da facilitarne l'utilizzo in caso di necessita`.

Per GNU/Linux, non esiste in circolazione una versione originale di VI, ma tante interpretazioni di questo, con potenzialita` piu` o meno ampliate. Per tale motivo, /bin/vi e` solitamente un collegamento (simbolico o meno) al programma che si utilizza effettivamente.

La realizzazione di VI piu` importante e` quella del programma Vim che e` stato portato su diversi sistemi. In particolare, la versione Dos e` in grado di gestire file di dimensioni molto grandi (diversi milioni di byte) anche su sistemi i286 che dispongono soltanto dei classici 640 Kibyte di memoria RAM.

Avvio

vi [opzioni] [file...]

L'eseguibile vi puo` essere avviato o meno con l'indicazione di un file sul quale intervenire. Se questo file esiste, viene aperto e si ottiene la visualizzazione del suo contenuto per permetterne la modifica. Se non esiste, viene creato.

E` anche possibile l'indicazione di alcune opzioni, tra cui, le piu` importanti sono elencate di seguito:

-R
i file vengono aperti (inizialmente) in sola lettura;

-c comando
subito dopo aver caricato il primo dei file degli argomenti, viene eseguito il comando indicato;

-s file_di_comandi
subito dopo aver caricato il primo dei file degli argomenti, vengono eseguiti i comandi contenuti nel file di comandi indicato.

Normalmente, l'eseguibile vi puo` essere avviato utilizzando nomi diversi: per rispettare

le tradizioni o per definire implicitamente degli attributi.

view [opzioni] [file...]

Di solito, view e` un collegamento alla realizzazione di VI che si ha a disposizione. Di norma, quando l'eseguibile di VI viene avviato con questo nome, si comporta come se gli fosse stata data l'opzione -R: sola lettura.

ex [opzioni] [file...]

Come gia` accennato, EX (precisamente l'eseguibile ex) e` il programma da cui e` derivato VI. Generalmente, nelle distribuzioni GNU/Linux si trova un collegamento (simbolico o meno) con questo nome che punta a vi. Alcune realizzazioni di VI, quando sono avviati con questo nome, tendono a comportarsi in maniera leggermente differente.

Modalita` di funzionamento

VI distingue diverse modalita` di funzionamento, altrimenti definibili come stati o contesti. Quando si avvia VI, questo si trova di solito nella modalita` di comando che permette di usare tasti determinati, attribuendogli significati speciali (di comando). Quando si vuole agire per inserire o modificare del testo, occorre utilizzare un comando con il quale VI passa alla modalita` di inserimento e modifica.

Per complicare ulteriormente le cose, c'e` da aggiungere che esistono in realta` due tipi di comandi: ?visuali? (visual) e ?due punti? (colon). I comandi visuali sono i piu` semplici e si compongono di sequenze di uno o piu` tasti il cui inserimento non appare in alcuna parte dello schermo e si concludono senza la pressione del tasto [Invio]; i comandi ?due punti? iniziano tutti con il simbolo : (da cui il nome), terminano con [Invio] e durante la digitazione appaiono sulla riga inferiore dello schermo. In particolare, questi ultimi sono quelli derivati da EX.

La modalita` di inserimento si riferisce al momento in cui e` possibile modificare il testo. Per passare dalla modalita` di comando a quella di inserimento, si preme il tasto corrispondente alla lettera ?i? (inserimento prima del cursore) o alla lettera ?a? (inserimento dopo il cursore).

Per tornare alla modalita` di comando, da quella di inserimento, e` sufficiente premere il tasto [Esc]. Quando ci si trova gia` nella modalita` di comando, la pressione del tasto [Esc] non produce alcunche' o al massimo interrompe l'introduzione di un comando, di conseguenza, se lo si usa inavvertitamente o troppo, non ne derivano inconvenienti.

Lo svantaggio principale di questo tipo di approccio e` quello di dover passare alla modalita` di comando per qualunque operazione diversa dal puro inserimento di testo. Anche lo spostamento del cursore avviene attraverso dei comandi, obbligando l'utente a premere il tasto [Esc] prima di poter utilizzare i tasti per il suo spostamento.

Tuttavia, le realizzazioni piu` diffuse di VI addolciscono un po' il suo funzionamento introducendo l'uso dei tasti freccia nel modo consueto dei programmi di scrittura piu`

recenti.

Posizione attiva

Per la descrizione del funzionamento di VI e` importante definire il concetto di posizione attiva che si riferisce al punto in cui si trova il cursore. Estendendo il significato, si puo` parlare di riga attiva, colonna attiva e parola attiva, intendendo quelle su cui si trova il cursore.

Moltiplicatori

Prima di affrontare i comandi di VI e` importante comprendere che l'effetto di molti di questi puo` essere moltiplicato utilizzando un numero. Il concetto e` molto semplice e si richiama alla matematica: 2a = a+a.

Inserimento

Come gia` accennato, si puo` inserire o modificare del testo solo quando si passa alla modalita` di inserimento attraverso il comando i (insert) oppure a (append). Durante questa fase, tutti i simboli della tastiera servono per inserire del testo. Con il VI standard si puo` usare:

- [Invio] per terminare una riga e passare alla successiva;
- [Backspace] per tornare indietro (nella maggior parte dei casi si ottiene anche la cancellazione del testo);
- [Esc] per terminare la modalita` di inserimento e passare a quella di comando.

Con le realizzazioni di VI piu` sofisticate, e` concesso normalmente l'uso dei tasti freccia e in alcuni casi anche del tasto [Canc].

Per tutte le altre operazioni di modifica del testo si deve passare alla modalita` di comando.

I comandi a disposizione per passare alla modalita` di inserimento sono molti e non si limitano quindi ai due modi appena descritti. La tabella 84.1 ne elenca alcuni.

Tabella 84.1. Elenco dei comandi utilizzabili per passare alla modalita` di inserimento.
Comando Descrizione
I Inserisce all'inizio della riga attiva.
i Inserisce prima della posizione attiva.
A Aggiunge alla fine della riga attiva.
a Aggiunge dopo la posizione attiva.
O Inserisce prima della riga attiva (inserendo una riga).
o Aggiunge dopo la riga attiva (inserendo una riga).

Navigazione

Come gia` accennato, lo spostamento del cursore e di conseguenza della posizione attiva, avviene per mezzo di comandi che generalmente obbligano a terminare la fase di inserimento. Le realizzazioni di VI piu` recenti permettono l'uso dei tasti freccia durante la modalita` di inserimento (oltre che durante la modalita` di comando), ma questo e` solo un aiuto minimo: in generale e` necessario tornare alla modalita` di comando.

I comandi normali per lo spostamento del cursore sono le lettere h, j, k e l, che rispettivamente spostano il cursore a sinistra, in basso, in alto e a destra. La ragione della scelta sta nella vicinanza di queste lettere nella maggior parte delle tastiere.

Figura 84.3. Promemoria visuale per i comandi che permettono lo spostamento del cursore.

Salvo casi particolari e situazioni in cui questo concetto non e` ragionevolmente applicabile, i comandi di spostamento, preceduti da un numero, vengono ripetuti tante volte quante ne rappresenta quel numero. Per esempio, il comando 2h sposta il cursore a sinistra di due posizioni.

Per raggiungere una riga determinata e` possibile utilizzare il comando nG o :n (in questo ultimo caso, seguito da [Invio])

Per esempio, per raggiungere la decima riga di un documento ipotetico, si puo` utilizzare indifferentemente uno dei due comandi seguenti:

10G

:10[Invio]

Per fare scorrere il testo di una schermata alla volta si utilizzano le combinazioni di tasti [Ctrl+B] e [Ctrl+F] che rispettivamente spostano il testo all'indietro e in avanti (back e forward).

I comandi a disposizione per lo spostamento sono ovviamente numerosi, la tabella 84.2 ne elenca alcuni.

Tabella 84.2. Elenco dei comandi utilizzabili per la navigazione all'interno del testo.

Comando	Descrizione
h	Sposta il cursore a sinistra di un carattere.
j	Sposta il cursore in basso nella riga successiva.
k	Sposta il cursore in alto nella riga precedente.
l	Sposta il cursore a destra di un carattere.
-	Sposta il cursore all'inizio della riga precedente.
+	Sposta il cursore all'inizio della riga successiva.

w Sposta il cursore all'inizio della parola successiva.
e Sposta il cursore alla fine della parola successiva.
b Sposta il cursore all'inizio della parola precedente.
^ Sposta il cursore all'inizio della prima parola della riga.
0 Sposta il cursore all'inizio della riga.
$ Sposta il cursore alla fine della riga.
H Sposta il cursore sulla prima riga che appare sullo schermo.
M Sposta il cursore sulla riga centrale dello schermo.
L Sposta il cursore sull'ultima riga che appare sullo schermo.
G Sposta il cursore sull'ultima riga del file.
nG Sposta il cursore sulla riga identificata dal numero n.
| Sposta il cursore sulla prima colonna (all'inizio della riga).
n| Sposta il cursore sulla colonna identificata dal numero n.
:n Sposta il cursore sulla riga identificata dal numero n.
Ctrl+B Fa scorrere il testo all'indietro di una schermata.
Ctrl+F Fa scorrere il testo in avanti di una schermata.
Ctrl+U Fa scorrere il testo all'indietro di mezza schermata.
Ctrl+D Fa scorrere il testo in avanti di mezza schermata.

Esempi

5w

Sposta il cursore all'inizio della quinta parola successiva.

2b

Sposta il cursore all'inizio della seconda parola precedente.

5G

Sposta il cursore all'inizio della quinta riga.

4|

Sposta il cursore sulla quarta colonna.

Modificatori

I comandi di spostamento, esclusi quelli che iniziano con i due punti (:) e quelli che si ottengono per combinazione ([Ctrl+...]), possono essere utilizzati come modificatori di altri comandi.

All'interno di VI manca il concetto di zona di intervento. Per definire l'estensione di un comando lo si puo` far precedere da un moltiplicatore (un numero) e in piu`, o in alternativa, lo si puo` fare seguire da un comando di spostamento. Il comando di spostamento viene utilizzato in questo caso per definire una zona che va dalla posizione attiva a quella di destinazione del comando, in modo che su questa zona intervenga il

comando precedente.

Tuttavia, per poter applicare questo concetto, e` necessario che i comandi da utilizzare in associazione con i modificatori (di spostamento), siano stati previsti per questo. Deve trattarsi cioe` di comandi che richiedono questa ulteriore aggiunta.

Come si vedra` in seguito, il comando x permette di cancellare quello che appare in corrispondenza del cursore. Quando viene premuto il tasto [x] si ottiene subito la cancellazione del carattere, pertanto, a questo genere di comandi non si puo` far seguire alcun modificatore. Questo tipo di comandi puo` solo essere preceduto da un moltiplicatore.

Si comporta diversamente il comando d che invece deve essere seguito da un modificatore e con questo definisce una zona da cancellare. Per esempio, dw cancella dalla posizione attiva fino all'inizio della prossima parola e d$ cancella dalla posizione attiva fino alla fine della riga.

Cancellazione

Durante la fase di inserimento e` possibile cancellare solo il carattere appena scritto utilizzando il tasto [Backspace], sempre che la realizzazione di VI a disposizione lo consenta, altrimenti si ottiene solo l'arretramento del cursore. Per qualunque altro tipo di cancellazione occorre passare alla modalita` di comando.

I comandi di cancellazione piu` importanti sono x, d seguito da un modificatore e dd. Il primo cancella il carattere che si trova in corrispondenza della posizione attiva, cioe` del cursore, il secondo cancella dalla posizione attiva fino all'estensione indicata dal modificatore e il terzo cancella tutta la riga attiva. Con VI non e` possibile cancellare il carattere che conclude una riga (il codice di interruzione di riga), di conseguenza, per unire due righe insieme si utilizza il comando J oppure j (bisogna provare).

Tabella 84.3. Elenco dei comandi utilizzabili per cancellare.
Comando Descrizione
x Cancella il carattere che si trova sulla posizione attiva.
J oppure j Unisce la riga attiva con quella successiva.
dd Cancella la riga attiva.
dmod Cancella dalla posizione attiva fino all'estensione indicata dal modificatore.
D agisce come d$.

Esempi

5x

Ripete cinque volte la cancellazione di un carattere. In pratica, cancella cinque caratteri.

2dd

Ripete due volte la cancellazione di una riga. In pratica, cancella la riga attiva e quella seguente.

dw

Cancella a partire dalla posizione attiva, fino al raggiungimento della prossima parola.

2dw

Ripete per due volte il tipo di cancellazione dell'esempio precedente. In pratica cancella fino all'inizio della seconda parola.

d2w

Cancella a partire dalla posizione attiva, fino al raggiungimento della seconda parola successiva. In pratica, esegue la stessa operazione del comando 2dw.

db

Cancella a ritroso, dalla posizione corrente fino all'inizio della prima parola che viene incontrata.

d$

Cancella a partire dalla posizione attiva fino alla fine della riga.

d5G

Cancella dalla posizione attiva fino all'inizio della riga numero cinque.

Sostituzione

La modifica del testo inserito puo` avvenire attraverso i comandi di cancellazione gia` visti, oppure attraverso comandi di sostituzione. Generalmente si tratta di comandi che prima cancellano parte del testo e subito dopo attivano l'inserimento.

I comandi di sostituzione piu` importanti sono c seguito da un modificatore e cc. Il primo sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore e il secondo sostituisce tutta la riga attiva.

A fianco di questi se ne aggiungono un paio che possono essere utili proprio per il fatto che non passano alla modalita` di inserimento: rx e ~. Il primo sostituisce il carattere in corrispondenza del cursore con quello rappresentato da x e il secondo inverte le lettere minuscole in maiuscole e viceversa.

Tabella 84.4. Elenco dei comandi di sostituzione e rimpiazzo.
Comando Descrizione

C Sostituisce dalla posizione attiva alla fine della riga.
cc Sostituisce la riga attiva a partire dall'inizio.
cmod Sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore.
rx Rimpiazza quanto contenuto nella posizione attiva con x.
~ Inverte maiuscole e minuscole.

Esempi

cc

Sostituisce la riga attiva.

c$

Sostituisce a partire dalla posizione attiva fino alla fine della riga.

rb

Rimpiazza il carattere che si trova nella posizione attiva con la lettera ?b?.

10~

Inverte le lettere maiuscole e minuscole a partire dalla posizione attiva, per 10 caratteri.

Copia e spostamento di porzioni di testo

La gestione della copia e dello spostamento di testo attraverso VI e` un po' complicata. Per questa attivita` si utilizzano delle aree temporanee di memoria alle quali si possono accodare diverse parti di testo.

L'operazione con la quale si copia una porzione di testo in un'area temporanea di memoria viene detta yanking, ovvero estrazione, giustificando cosi` l'uso della lettera ?y? nei comandi che compiono questa funzione.

Le aree temporanee di memoria per lo spostamento o la copia di testo possono essere 27: una per ogni lettera dell'alfabeto e una aggiuntiva senza nome.

Il modo piu` semplice di gestire questo meccanismo e` quello di usare l'area temporanea senza nome. Per copiare una porzione di testo si puo` utilizzare il comando y seguito da un modificatore, oppure il comando yy che invece si riferisce a tutta la riga attiva. Per incollare il testo copiato, dopo aver posizionato il cursore nella posizione di destinazione, si puo` utilizzare il comando p oppure P, a seconda che si intenda incollare prima o dopo la posizione del cursore.

Il comandi p e P non cancellano il contenuto dell'area temporanea, di conseguenza, se

serve si puo` ripetere l'operazione di inserimento riutilizzando questi comandi.

Se invece di copiare si vuole spostare il testo, al posto dei comandi di estrazione si possono usare quelli di cancellazione, che, anche se non era stato chiarito precedentemente, prima di cancellare il testo fanno una copia nell'area temporanea.

Tabella 84.5. Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso dell'area temporanea di memoria senza nome.
Comando Descrizione
yy Copia la riga attiva nell'area temporanea.
ymod Copia nell'area temporanea il testo fino all'estensione indicata dal modificatore.
dd Trasferisce la riga attiva nell'area temporanea.
dmod Trasferisce nell'area temporanea il testo fino all'estensione indicata dal modificatore.
p Incolla prima della posizione del cursore.
P Incolla dopo la posizione del cursore.

Esempi

5yy

Copia nell'area temporanea cinque righe a partire da quella attiva.

yw

Copia nell'area temporanea il testo che parte dalla posizione attiva fino all'inizio della prossima parola.

y$

Copia nell'area temporanea il testo che parte dalla posizione attiva fino alla fine della riga.

3dd

Sposta nell'area temporanea tre righe a partire da quella attiva.

2P

Incolla due copie del testo contenuto nell'area temporanea a partire dalla posizione a sinistra del cursore.

Copia e spostamento con nome

Quando si vogliono utilizzare delle aree temporanee di memoria specifiche, cioe` identificate attraverso le lettere dell'alfabeto, si procede nei modi gia` visti nel caso dell'uso dell'area temporanea senza nome, con la differenza che i comandi sono preceduti da "x, dove x e` la lettera che si vuole usare.

Si introduce pero` una novita` importante: e` possibile aggiungere del testo a un'area temporanea: basta indicarla attraverso una lettera maiuscola.

Tabella 84.6. Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso delle aree temporanee con nome.
Comando Descrizione
"xyy Copia la riga attiva nell'area temporanea x
"xymod Copia nell'area temporanea x il testo fino all'indicazione dal modificatore.
"xdd Trasferisce la riga attiva nell'area temporanea x.
"xdmod Trasferisce nell'area temporanea x il testo fino all'indicazione dal modificatore.
"xp Incolla il contenuto dell'area temporanea x prima del cursore.
"xP Incolla il contenuto dell'area temporanea x dopo il cursore.

Esempi

"adw

Sposta il testo nell'area temporanea a, a partire dalla posizione attiva fino all'inizio della prossima parola.

"a5yy

Copia cinque righe nell'area temporanea a, a partire dalla posizione attiva (inclusa).

"A3yy

Aggiunge tre righe nell'area temporanea a, a partire dalla posizione attiva (inclusa).

"a2P

Incolla due copie del contenuto dell'area temporanea a, a partire dalla posizione precedente a quella su cui si trova il cursore.

Ricerche

Per effettuare delle ricerche all'interno del documento aperto con VI si possono utilizzare le espressioni regolari (capitolo 316) attraverso due comandi un po' strani: / e ?. La sintassi per la ricerca in avanti e`:

/modello

mentre per la ricerca a ritroso e` la seguente:

?modello

Nel momento in cui si preme il tasto della barra obliqua o del punto interrogativo, VI visualizza il comando nella riga inferiore dello schermo permettendone il controllo e la

correzione come avviene per i comandi che iniziano con i due punti. Al termine, l'inserimento di questo tipo di comando deve essere concluso con un [Invio].

Tabella 84.7. I comandi di ricerca attraverso espressioni regolari.
Comando Descrizione
/modello Cerca in avanti una corrispondenza con il modello indicato.
?modello Cerca all'indietro una corrispondenza con il modello indicato.
n Ripete l'ultimo comando / o ?.
N Ripete l'ultimo comando / o ? in modo inverso .

Il tipo di espressione regolare che puo` essere utilizzato con VI e` solo quello elementare e valgono in particolare le regole indicate nella tabella 84.8.

Tabella 84.8. Le espressioni regolari che dovrebbero essere disponibili con la maggior parte delle realizzazioni di VI.
Simbolo Descrizione
. Corrisponde a un carattere qualsiasi.
\ Fa perdere il significato speciale che puo` avere il carattere seguente.
^ Corrisponde all'inizio di una riga.
$ Corrisponde alla fine di una riga.
[abc] Corrisponde a un carattere qualsiasi tra quelli tra parentesi quadre.
[^abc] Corrisponde a un carattere qualsiasi diverso da quelli tra parentesi quadre.
[a-z] Un carattere qualsiasi nell'intervallo compreso tra a e z.
[^a-z] Un carattere qualsiasi diverso dall'intervallo compreso tra a e z.

Esempi

/[Ll]inux

Cerca in avanti tutte le stringhe corrispondenti a ?Linux? oppure ?linux?.

/\.$

Cerca in avanti il punto finale di una riga (si osservi l'uso della barra obliqua inversa per togliere il significato speciale che ha il punto in un'espressione regolare).

Ricerche e sostituzioni

La ricerca e sostituzione sistematica avviene attraverso un comando particolare che inizia con i due punti. La sua sintassi e` la seguente:

:inizio,fines/modello_da_cercare/sostituzione/[g][c]

L'indicazione inizio e fine fa riferimento alle righe su cui intervenire. Si possono indicare dei numeri, oppure dei simboli con funzioni simili. Il simbolo $ puo` essere usato per indicare l'ultima riga del file. Un punto singolo (.) rappresenta la riga attiva. Il simbolo % viene invece utilizzato da solo per indicare tutte le righe del file.

Il modello utilizzato per la ricerca viene espresso secondo la forma delle espressioni regolari.

Normalmente, il valore da sostituire al modello cercato e` fisso, ma puo` contenere un riferimento alla stringa trovata, attraverso il simbolo &.

La direttiva g specifica che si deve intervenire su tutte le occorrenze della corrispondenza con il modello, altrimenti la sostituzione riguarda solo la prima di queste.

La direttiva c specifica che ogni sostituzione deve essere confermata espressamente.

Esempi

:1,$s/pippo/pappa/g

Sostituisce ogni occorrenza della parola ?pippo? con la parola ?pappa?. La ricerca viene effettuata a partire dalla prima riga fino all'ultima.

:%s/pippo/pappa/g

Questo e` un modo alternativo per eseguire la stessa operazione dell'esempio precedente: il simbolo % rappresenta da solo tutte le righe del file.

:.,10s/^..//g

Elimina i primi due caratteri (^..) da 10 righe a partire da quella attiva.

:%s/^..//gc

Esegue la stessa operazione dell'esempio precedente, applicando la sostituzione su tutto il file, richiedendo pero` conferma per ogni sostituzione.

:.,10s/^/xxxx/g

Inserisce la stringa ?xxxx? all'inizio di 10 righe a partire da quella attiva.

Annullamento dell'ultimo comando

VI permette di annullare l'ultimo comando inserito attraverso il comando u. A seconda della realizzazione di VI utilizzata, richiamando nuovamente il comando u si riottengono le modifiche annullate precedentemente, oppure si continuano ad annullare gli effetti dei comandi precedenti.

Tabella 84.9. Annullamento di un comando.
Comando Descrizione
u Annulla l'ultimo comando.
U Annulla le modifiche sulla riga attiva.

Caricamento, salvataggio e conclusione

Il file o i file utilizzati per la modifica (compresi quelli che si creano) vengono aperti normalmente attraverso l'indicazione nella riga di comando, al momento dell'avvio dell'eseguibile vi.

Il salvataggio di un file puo` essere fatto per mezzo del comando :w (write) seguito eventualmente dal nome del file (quando si vuole salvare con un nome diverso oppure quando si sta creando un file nuovo). La conclusione dell'attivita` di VI si ottiene con il comando :q (quit). I comandi possono essere combinati tra loro, per esempio quando si vuole salvare e concludere l'attivita` simultaneamente con il comando :wq. Il punto esclamativo (!) puo` essere usato alla fine di questi comandi per forzare le situazioni, come quando si vuole concludere l'attivita` senza salvare con il comando :q!.

Dal momento che VI non mostra normalmente alcuna informazione riferita al file su cui si opera (compreso il nome), il comando :f (oppure la combinazione [Ctrl+g]) mostra sulla riga inferiore dello schermo: il nome del file aperto, le dimensioni e il numero della riga attiva.

Tabella 84.10. I comandi per il caricamento dei file e il loro salvataggio.
Comando Descrizione
:e nome_file Carica il file indicato per poterlo modificare.
:e! Ricarica il file annullando le modifiche fatte nel frattempo.
:r nome_file Legge il file indicato e ne inserisce il contenuto dopo la riga attiva.
:f Mostra il nome e le caratteristiche del file aperto.
:w Salva.
:w nome_file Salva una copia con il nome indicato.
:wq Salva e termina l'esecuzione.
:q Fine lavoro.
:q! Fine lavoro forzato.

Variabili

VI ha ereditato da EX delle variabili di configurazione. Non si tratta di variabili di ambiente, ma di variabili interne a VI. Per attivare una variabile si utilizza il comando seguente:

:set [no]nome_della_variabile

Il prefisso no, prima del nome della variabile, serve per disattivarla. La tabella 84.11 mostra l'elenco di alcune di queste variabili.

Tabella 84.11. Variabili utilizzate da VI attraverso il comando :set.
Variabile Descrizione
autoindent Mantiene i livelli di rientro nelle righe nuove.
beautify Elimina i caratteri speciali non stampabili.
ignorecase Nelle ricerche, ignora la differenza tra maiuscole e minuscole.
list Mostra i caratteri di tabulazione e di interruzione di riga.

number Visualizza i numeri delle righe.
ruler Visualizza le coordinate del cursore alla base dello schermo.

Esempi

:set nolist

Disabilita la visualizzazione dei caratteri di tabulazione e di fine riga.

:set number

Visualizza i numeri di riga.

Comandi particolari

Di seguito sono elencati una serie di comandi particolari che non sono stati inclusi nelle categorie precedenti.

mx
Etichetta la posizione corrente con la lettera rappresentata da x. Valgono solo le lettere minuscole. Il testo non viene modificato.

'x
Sposta il cursore all'inizio della riga che contiene l'etichetta rappresentata da x.

[Ctrl+L]
Riscrive la schermata: se sono apparsi messaggi che creano difficolta` alla visualizzazione del testo su cui si sta lavorando, questo comando permette di farli scomparire mostrando il testo effettivo del file.

:!comando
Esegue il comando di shell indicato.

:ab abbreviazione testo_da_sostituire
Permette di stabilire un'abbreviazione che verra` sostituita sistematicamente con tutto quello che segue il comando. Se si usa :ab da solo, si ottiene un elenco delle abbreviazioni disponibili.

File di configurazione

Puo` essere utilizzato il file ~/.exrc per personalizzare la configurazione di VI attraverso comandi colon (quelli tipici di EX). Le cose piu` comuni che appariranno all'interno di questo file saranno la definizione di abbreviazioni e la definizione di alcune variabili. Segue un esempio.

:ab lx Linux
:ab xwin X Window System

:set autoindent
:set number

Problemi di portabilita`

Uno dei vantaggi importanti nell'uso di VI sta nella disponibilita` di realizzazioni di questo programma per qualsiasi piattaforma. All'inizio di questo gruppo di sezioni su VI si accennava al fatto che esiste un'ottima versione Dos in grado di funzionare molto bene anche con i vecchi elaboratori dotati di poca memoria.

Quando si trasferiscono testi da un sistema GNU/Linux a Dos e viceversa si pone il problema dell'insieme di caratteri a disposizione: su GNU/Linux si utilizza presumibilmente la codifica Latin 1, mentre con il Dos no.

La soluzione piu` semplice a questo problema e` probabilmente quella di usare Latin 1 in generale e di convertire le lettere accentate Dos in Latin 1 quando possibile. Per questo si puo` realizzare un semplice file di comandi da eseguire automaticamente utilizzando l'opzione -s.(4)

:1,$s/~E/a`/g
:1,$s/~J/e`/g
:1,$s/~B/e'/g
:1,$s/~M/i`/g
:1,$s/~U/o`/g
:1,$s/~W/u`/g

In pratica, la sintassi da usare all'avvio di VI dovrebbe essere la seguente:

vi -s file_comandi file_da_elaborare

In generale, i problemi legati alla conversione di un file da un insieme di caratteri all'altro, si risolvono attraverso l'uso di recode, descritto nel capitolo 342.

cap 11) Applicativi distribuiti in forma sorgente o compilata

La maggior parte dei programmi per Unix il cui utilizzo viene concesso gratuitamente, viene distribuita in forma sorgente. Cio` significa che per poterli utilizzare, questi programmi devono essere compilati. Fortunatamente, nella maggior parte dei sistemi Unix e` disponibile il compilatore ANSI C GNU che permette di uniformare questo procedimento di compilazione.

Alcuni programmi vengono distribuiti in forma gia` compilata (e senza sorgenti) soprattutto quando si tratta di prodotti proprietari. Anche in questi casi si possono incontrare problemi nell'installazione.

In generale, i problemi che derivano dall'installazione di questi applicativi nasce dalla mancanza di sostegno da parte del sistema di gestione dei pacchetti della propria distribuzione GNU/Linux.

Struttura tipica di un pacchetto sorgente

Un programma distribuito in forma sorgente si trova di solito confezionato in un archivio il cui nome ha un'estensione .tar.gz o .tgz (ottenuto attraverso tar e gzip), oppure ancora con un'estensione .tar.bz2 (tar e bzip2). Prima di poter procedere con la sua compilazione deve essere estratto il suo contenuto. Solitamente si fa questo in una directory di lavoro. Nell'esempio che segue, si fa riferimento a un pacchetto ipotetico archiviato nel file pacch.tar.gz:

```
$ cd ~/tmp

$ tar xzvf pacch.tar.gz
```

Se invece si trattasse dell'archivio pacch.tar.bz2, sarebbe stato necessario decomprimerlo attraverso un comando un po' piu` complesso:

```
$ cd ~/tmp

$ cat pacch.tar.bz2 | bunzip2 | tar xvf -
```

Di solito si ottiene una struttura ad albero piu` o meno articolata in sottodirectory, dove nella directory principale di questa struttura si trovano:

- uno o piu` file di documentazione (README, INSTALL, ecc.) che servono per ricordare il procedimento corretto per ottenere la compilazione;
- uno o piu` script preparati per facilitare questo procedimento;
- il file-make (o makefile).

Seguendo l'esempio visto poco prima, dovrebbe essere stata creata la directory pacch/.

$ cd pacch

$ ls

Documentazione necessaria alla compilazione

I file di testo che si trovano nella directory principale del pacchetto contenente il programma sorgente, servono per presentare brevemente il programma e per riassumere le istruzioni necessarie alla sua compilazione. Di solito, queste ultime sono contenute nel file INSTALL. In ogni caso, tutti questi file vanno letti, in particolare quello che spiega il procedimento per la compilazione e l'installazione.

Il modo piu` semplice per leggere un file e` l'utilizzo del programma less, o in sua mancanza di more.

$ less README

$ less INSTALL

./configure

La composizione classica di un pacchetto distribuito in forma sorgente, prevede la presenza di uno script il cui scopo e` quello di costruire un file-make adatto all'ambiente in cui si vuole compilare il programma. Cio` e` necessario perche' i vari sistemi Unix sono diversi tra loro per tanti piccoli dettagli.

Spesso questo script e` in grado di accettare argomenti. Cio` puo` permettere, per esempio, di definire una directory di destinazione del programma, diversa da quella predefinita.

Quando questo script di preparazione manca, occorre modificare manualmente il file-make in modo che sia predisposto correttamente per la compilazione nel proprio sistema.

Per evitare ambiguita`, questo script viene sempre avviato indicando un percorso preciso: ./configure.

File-make

Il file-make, o makefile, e` quel file che viene letto da Make, precisamente dall'eseguibile make, allo scopo di coordinare le varie fasi della compilazione ed eventualmente anche

per l'installazione del programma compilato. Il nome di questo file puo` essere diverso, generalmente si tratta di Makefile, oppure di makefile.

Questo file viene generato frequentemente da uno script, di solito si tratta di ./configure, ma se manca deve essere controllato e, se necessario, modificato prima della compilazione.

Spesso e` bene controllare il contenuto del file-make anche quando questo e` stato generato automaticamente.

Fasi tipiche di una compilazione e installazione

I pacchetti piu` comuni si compilano e si installano con tre semplici operazioni.

$ ./configure

Genera automaticamente il file-make.

$ make

Esegue la compilazione generando i file eseguibili.

make install

Installa gli eseguibili e gli altri file necessari nella loro destinazione prevista per il funzionamento: l'ultima fase deve essere eseguita con i privilegi dell'utente root.

Problemi

Le note di questo capitolo valgono solo in linea di massima: e` sempre indispensabile leggere le istruzioni che si trovano nei file di testo distribuiti insieme ai sorgenti dei programmi.

I problemi maggiori si hanno quando non e` stato predisposto uno script ./configure o simile e si e` costretti a modificare il file-make.

In altri casi, il file-make potrebbe non prevedere la fase di installazione (make install), per cui si deve installare il programma copiando pezzo per pezzo nella destinazione giusta.

Spesso l'installazione non rispetta la struttura standard del proprio file system. Cio` nel senso che magari vengono piazzati file che devono poter essere modificati, all'interno di una zona che si voleva riservare in sola lettura.

Quando si utilizza una distribuzione GNU/Linux ben organizzata, si trova una gestione dei pacchetti installati che permette l'eliminazione e l'aggiornamento di questi senza rischiare di lasciare file inutilizzati in giro. Quando si installa un programma distribuito in forma originale, viene a mancare questo supporto della gestione dei pacchetti. In questi

casi si cerca di installare tali pacchetti al di sotto della directory /opt/, o almeno al di sotto di /usr/local/.

Installazione di programmi gia` compilati

L'installazione di programmi gia` compilati per GNU/Linux, anche se potrebbe sembrare piu` semplice rispetto a un procedimento che richiede la compilazione, potrebbe creare qualche problema a chi non conosce perfettamente l'interdipendenza che c'e` tra le varie parti del sistema operativo.

I problemi e le soluzioni che si descrivono nelle sezioni seguenti, riguardano a volte anche i programmi distribuiti in forma sorgente. Infatti, alcune volte, i programmi distribuiti in questo modo non sono stati preparati per un'installazione soddisfacente, di conseguenza bisogna provvedere da soli a collocare i file nelle posizioni corrette e a sistemare tutto quello che serve.

Scelta della piattaforma

Quando si cerca del software per il proprio sistema che puo` essere ottenuto solo in forma gia` compilata, occorre fare attenzione alla piattaforma. Infatti, non basta che si tratti di programmi compilati per GNU/Linux, occorre che gli eseguibili siano adatti al tipo di elaboratore su cui GNU/Linux e` in funzione.

Normalmente, per identificare l'architettura dei ?PC?, si utilizza la sigla i386 nel nome dei file degli archivi.

Eseguibili e variabili di ambiente

Un programma distribuito in forma binaria, deve essere estratto normalmente dall'archivio compresso che lo contiene. A volte e` disponibile uno script o un programma di installazione, altre volte e` necessario copiare manualmente i file nelle varie destinazioni finali. Quando si puo` scegliere, e` preferibile collocare tutto quanto a partire da un'unica directory discendente da /opt/.

A volte, perche' il programma possa funzionare e` necessario predisporre o modificare il contenuto di alcune variabili di ambiente. Il caso piu` comune e` costituito da PATH che deve (o dovrebbe) contenere anche il percorso necessario ad avviare il nuovo programma. Spesso, i file di documentazione che accompagnano il software indicano chiaramente tutte le variabili che devono essere presenti durante il loro funzionamento.

La dichiarazione di queste variabili puo` essere collocata direttamente in uno dei file di configurazione della shell utilizzata (per esempio /etc/profile, oppure ~/.bash_profile o altri ancora a seconda di come e` organizzato il proprio sistema).

Librerie dinamiche

Alcuni programmi utilizzano delle librerie non standard, che spesso vengono collocate al

di fuori delle directory predisposte per contenerle. Per fare in modo che queste librerie risultino disponibili, ci sono due modi possibili:

1. modificare la configurazione di /etc/ld.so.cache;
2. utilizzare la variabile di ambiente LD_LIBRARY_PATH.

Per agire secondo la prima possibilita`, occorre prima comprendere come sia organizzato questo sistema. Il file /etc/ld.so.cache viene creato a partire da /etc/ld.so.conf che contiene semplicemente un elenco di directory destinate a contenere librerie. Il programma ldconfig serve proprio a ricreare il file /etc/ld.so.cache leggendo /etc/ld.so.conf, pertanto viene avviato solitamente dalla stessa procedura di inizializzazione del sistema, allo scopo di garantire che questo file sia sempre aggiornato.

Dovrebbe essere chiaro, ormai, il modo giusto di includere nuovi percorsi di librerie nel file /etc/ld.so.cache: occorre indicare questa o queste directory nel file /etc/ld.so.conf e quindi basta avviare il programma ldconfig.

L'utilizzo della variabile di ambiente LD_LIBRARY_PATH e` meno impegnativo, ma soprattutto, in questo modo si puo` intervenire facilmente attraverso dei semplici script. Cio` permette, per esempio, di fare in modo che solo un certo programma ?veda? certe librerie. In ogni caso, quando si intende usare questa variabile di ambiente, e` importante ricordare di includere tra i vari percorsi anche quelli standard: /lib/, /usr/lib/ e /usr/local/lib/. L'esempio seguente rappresenta un pezzo di uno script (potrebbe trattarsi di /etc/profile) in cui viene assegnata la variabile di ambiente in questione.

```
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/opt/mio_prog/lib:/opt/tuo_prog/lib
export LD_LIBRARY_PATH
```

Se un certo programma richiede determinate librerie che potrebbero entrare in conflitto con altri programmi, e` indispensabile l'utilizzo della variabile di ambiente LD_LIBRARY_PATH, configurandola esclusivamente nell'ambito del processo di quel programma. In pratica, si tratta di avviare il programma attraverso uno script che genera l'ambiente adatto, in modo che non si rifletta negli altri processi, come mostrato nell'esempio seguente:

```
#! /bin/sh

# modifica il percorso di ricerca delle librerie
LD_LIBRARY_PATH="/opt/mio_programma/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH

# avvia il programma
mio_programma
```

```
# al termine dello script non resta traccia
```

Avviando lo script, viene modificata la variabile di ambiente LD_LIBRARY_PATH per quel processo e per i suoi discendenti (viene esportata), quindi, al termine del programma termina lo script e con lui anche gli effetti di queste modifiche.

Si osservi in particolare il fatto che nella nuova definizione del percorso delle librerie, viene posto all'inizio quello per le librerie specifiche del programma, in modo che venga utilizzato per primo; subito dopo, viene inserito l'elenco dei percorsi eventualmente gia` esistente.

$ ldd

ldd [opzioni] programma...

ldd emette l'elenco delle librerie condivise richieste dai programmi indicati come argomenti della riga di comando. Si utilizza ldd per determinare le dipendenze di uno o piu` programmi dalle librerie.

Esempi

```
$ ldd /bin/bash[Invio]

    libncurses.so.4 => /usr/lib/libncurses.so.4 (0x40000000)
    libdl.so.1 => /lib/libdl.so.1.7.14 (0x40045000)
    libc.so.5 => /lib/libc.so.5.4.38 (0x40048000)
```

L'esempio mostra le dipendenze dalle librerie di /bin/bash. Il risultato ottenuto indica il nome delle librerie e la collocazione effettiva nel sistema, risolvendo anche eventuali collegamenti simbolici.

File di differenze, o patch

Quando si ha a che fare con programmi il cui aggiornamento e` frequente, come avviene nel caso del kernel, si possono anche trovare aggiornamenti in forma di file di differenze (patch), cioe` di file che contengono solo le variazioni da una versione all'altra. Queste variazioni si applicano ai file di una versione per ottenerne un'altra.

Se la versione da aggiornare e` stata espansa a partire dall'ipotetica directory ~/tmp/, per applicarvi una modifica, sara` necessario posizionarsi sulla stessa directory e poi eseguire il comando seguente:

```
patch < file_di_differenze
```

Puo` darsi che la posizione in cui ci si deve trovare sia diversa o che i sorgenti da aggiornare debbano trovarsi in una posizione precisa. Per capirlo, dovrebbe bastare l'osservazione diretta del contenuto del file di differenze.

L'applicazione delle variazioni, puo` fallire. Se non si vuole perdere il rapporto degli errori, questi possono essere ridiretti in un file specifico.

patch < file_di_differenze 2> file_degli_errori

Se gli aggiornamenti sono piu` d'uno, occorre applicare le modifiche in sequenza.

Il capitolo 82 tratta meglio questo problema.

Aggiornamento delle librerie standard

Oltre al problema delle librerie specifiche di un programma particolare, cosa gia` descritta in questo capitolo, ci puo` essere la necessita` di aggiornare le librerie standard di GNU/Linux a seguito di qualche aggiornamento di altro software.

Normalmente, la propria distribuzione GNU/Linux dovrebbe offrire questi aggiornamenti in forma di archivi gia` pronti, installabili attraverso il proprio sistema di gestione dei pacchetti. Cio` garantendo la sistemazione di una serie di dettagli importanti.

Quando si e` costretti a fare da soli e` importante fare attenzione. In particolare, quando si interviene in cio` che risiede nella directory /lib/, se si commette un errore, si rischia di bloccare il sistema senza possibilita` di rimedio.

I file delle librerie sono organizzati normalmente con un numero di versione piuttosto articolato. Quando cio` accade, e` normale che a questi file siano affiancati una serie di collegamenti simbolici strutturati in modo che si possa accedere a quelle librerie anche attraverso l'indicazione di versioni meno dettagliate, oppure semplicemente attraverso nomi differenti. Questi collegamenti sono molto importanti perche' ci sono dei programmi che dipendono da questi; quando si aggiorna una libreria occorre affiancare la nuova versione a quella vecchia, quindi si devono modificare questi collegamenti. Solo alla fine, sperando che tutto sia andato bene, si puo` eliminare eventualmente il vecchio file di libreria.

Per fare un esempio pratico, si suppone di disporre della libreria libc.so.9.8.7, articolata nel modo seguente:

libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7

Volendo sostituire questa libreria con la versione 9.8.10, il cui file ha il nome libc.so.9.8.10, occorre procedere come segue:

ln -s -f libc.so.9.8.10 libc.so.9

Come si vede, per generare il collegamento, e` stato necessario utilizzare l'opzione -f che permette di sovrascrivere il collegamento preesistente. Infatti, non sarebbe stato possibile

eliminare prima il collegamento vecchio, perche' cosi` si sarebbe rischiato il blocco del sistema.

libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.10
libc.so.9.8.7
libc.so.9.8.10

In generale, per sicurezza, e` meglio lasciare le librerie vecchie, perche' ci potrebbero essere ugualmente dei programmi che ne hanno ancora bisogno.

Quando la libreria da aggiornare ha subito un aggiornamento molto importante, per cui i numeri delle versioni sono molto distanti rispetto a quanto si utilizzava prima, conviene evitare di sostituirle, mentre e` solo il caso di affiancarle. Volendo ritornate all'esempio precedente, si puo` supporre che la libreria da aggiornare sia arrivata alla versione 10.1.1, con il file libc.so.10.1.1. Intuitivamente si comprende che il collegamento simbolico libc.so.9 non puo` puntare a questa nuova libreria, mentre resta il dubbio per libc.so.

In generale e` meglio lasciare stare le cose come sono, a meno di scoprire che qualche programma cerca proprio la libreria libc.so e si lamenta perche' non si tratta della versione adatta a lui. Comunque, sempre seguendo l'esempio, sarebbe il caso di riprodurre un collegamento equivalente a libc.so.9, denominato ovviamente libc.so.10.

libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7
libc.so.10 -> libc.so.10.1.1
libc.so.10.1.1

cap 12) Pacchetti Debian

Gli archivi della distribuzione GNU/Linux Debian hanno un formato particolare e l'estensione .deb (quelli binari). La distribuzione Debian e poche altre derivate utilizzano questo formato; tuttavia, la maggior parte del software per GNU/Linux e` disponibile sotto forma di archivi Debian; spesso anche molto di piu` di quello che si puo` ottenere

dalle altre distribuzioni.

Per poter gestire tale formato in un sistema GNU/Linux diverso dalla distribuzione Debian, occorre il programma dpkg. Con l'aiuto di FTPSearch, <http://www.alltheweb.com/?c=ftp>, si puo` cercare un archivio che assomigli a dpkg*.tar.gz.

Valgono le stesse considerazioni fatte a proposito degli archivi RPM: se la propria distribuzione GNU/Linux non e` fatta per gestire i pacchetti archiviati in formato Debian, l'unica motivazione ragionevole per procurarsi il programma di gestione di questi e` quella di poterli convertire nel formato a cui si e` abituati.

Caratteristiche dei pacchetti Debian

I pacchetti della distribuzione sono archiviati in modo differente, a seconda che si tratti di pacchetti binari o di pacchetti sorgenti. I pacchetti binari, cioe` tutto quello che puo` essere usato cosi` come si trova (compresa la documentazione), e` archiviato nel formato Debian (.deb), mentre i sorgenti sono composti da terne di file: una descrizione contenuta in un file con estensione .dsc, l'archivio dei sorgenti originali in un file con estensione .orig.tar.gz e un file di differenze da applicare ai sorgenti originali, in questo caso con l'estensione .diff.gz.

Il nome di un archivio contenente un pacchetto binario Debian ha una struttura riassumibile nello schema seguente:

nome_pacchetto_versione-revisione.deb

Un pacchetto Debian binario, oltre ai file che compongono il pacchetto e che vengono installati, contiene:

- un file di ?controllo? (control), con quasi tutte le informazioni relative al pacchetto, in particolare le sue dipendenze;
- un file contenente l'elenco dei file di configurazione, per i quali occorre avere un occhio di riguardo (conffiles);
- due coppie di script che controllano la fase di installazione e quella di disinstallazione del pacchetto (preinst, postinst, prerm, postrm).

Priorita` di un pacchetto

A ogni pacchetto Debian viene attribuita una priorita`, rappresentata da una definizione standard. Questa permette di facilitare la scelta dei pacchetti da installare. Si usano le parole chiave seguenti:

required
indica un pacchetto necessario per il funzionamento elementare del sistema;

important
indica un pacchetto importante per il buon funzionamento del sistema;

standard
indica un pacchetto che fa parte di software comune in un sistema GNU/Linux, senza richiedere la presenza del sistema grafico X;

optional
indica un pacchetto opzionale;

extra
indica un pacchetto destinato a un uso specializzato, o che va in conflitto con altri pacchetti di livello precedente.

E` interessante osservare che il livello di priorita` e` un'informazione che normalmente non e` contenuta nei pacchetti, ma viene attribuita all'esterno di questi. La si ritrova nei file Packages e Packages.cd che verranno descritti in seguito.

Dipendenze secondo i pacchetti Debian

Come accennato, il file di controllo contiene in particolare le informazioni sulle dipendenze del pacchetto. Secondo la logica di Debian, le dipendenze avvengono sempre in relazione a ?pacchetti?: quando si tratta di una dipendenza da una funzionalita`, questa viene identificata attraverso un ?pacchetto virtuale?. L'esempio seguente e` un estratto delle informazioni relative al pacchetto apache-ssl, dove si vede l'uso delle definizioni di quasi tutti i tipi di dipendenza e di incompatibilita`:

Depends: libc6 (>= 2.0.7u-6), libssl09, mime-support, perl, ...
Suggests: apache-doc, lynx
Conflicts: apache-modules, php3 (<= 3.0.3-1), libapache-mod-perl (<= 1.15-2.1)
Replaces: apache-modules
Provides: httpd

Le varie parole chiave hanno il significato seguente:

depends
indica un elenco di pacchetti indispensabili, eventualmente con il livello di versione richiesto, che devono essere presenti perche' questo possa funzionare;

recommends
indica un elenco di pacchetti raccomandati, anche se non indispensabili, perche' il

pacchetto che si installa possa essere utilizzato opportunamente;

suggests
indica un elenco di pacchetti suggeriti, che starebbero bene assieme a quello che si installa;

conflicts
indica un elenco di pacchetti che non possono convivere assieme a questo;

replaces
indica un elenco di pacchetti che vengono rimpiazzati da questo;

provides
indica un elenco di pacchetti che rappresentano le funzionalita` offerte da questo.

Le parole chiave utilizzate sono verbi posti alla terza persona singolare, come dire che ?il pacchetto A: dipende da... raccomanda... suggerisce... va in conflitto con... sostituisce... fornisce le funzionalita`...?. Osservando l'esempio, il pacchetto in questione fornisce le funzionalita` httpd (in questo caso un pacchetto virtuale), ovvero quelle di un servente HTTP, e` incompatibile con il pacchetto apache-modules e che con altri se hanno una versione troppo vecchia, inoltre va a sostituire lo stesso pacchetto apache-modules.

Stato di un pacchetto

Secondo la logica del sistema di gestione dei pacchetti Debian, lo stato di un pacchetto puo` avere tre gruppi di caratteristiche: lo stato in relazione a cio` che e` installato nel sistema, lo stato di selezione e alcune caratteristiche speciali. Rispetto al sistema, un pacchetto puo` essere:

installed -- installato
il pacchetto risulta installato correttamente nel sistema e anche la configurazione e` stata completata;

half-installed -- semi-installato
l'installazione del pacchetto non e` stata completata per qualche ragione;

not-installed -- non installato
il pacchetto non risulta installato;

unpacked -- estratto
il pacchetto risulta estratto dall'archivio, ma non e` stato configurato;

half-configured -- semi-configurato
il pacchetto risulta estratto dall'archivio e la configurazione non e` stata completata per qualche ragione;

config-files -- file di configurazione

del pacchetto sono presenti solo i file di configurazione.

Il sistema di installazione e disinstallazione dei pacchetti Debian e` in realta` una procedura, con la quale si ?prenotano? delle operazioni che poi vengono eseguite in sequenza. Sotto questo aspetto, un pacchetto che in qualche modo sia ?conosciuto? da questa procedura ha anche uno stato di selezione (come gia` accennato), che puo` essere:

unknown -- sconosciuto
quando non e` mai stata richiesta la sua installazione (e di conseguenza non e` nemmeno installato);

install -- da installare
quando e` stata richiesta la sua installazione, o il suo aggiornamento;

remove, deinstall -- da togliere
quando e` stata richiesta la sua disinstallazione normale, cioe` senza cancellare i file di configurazione;

purge -- da eliminare completamente
quando e` stata richiesta la sua eliminazione totale, compresi i file di configurazione;

Infine, un pacchetto puo` essere stato marcato in modo che non venga aggiornato o sostituito con un'altra versione, hold, oppure puo` essere stato marcato dal sistema di gestione dei pacchetti perche' risulta danneggiato in qualche modo e in tal senso viene indicato come candidato alla reinstallazione, reinst-required.

Per conoscere lo stato di un pacchetto si puo` usare dpkg richiedendo l'azione -l. L'esempio seguente mostra l'elenco di alcuni pacchetti con l'indicazione del loro stato:

```
Desired=Unknown/Install/Remove/Purge
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version       Description
+++-==============-==============-
=============================================
ii  gnome-admin     1.0.1-1       Gnome Admin Utilities (gulp and logview)
ii  gnome-bin       1.0.3-1       Miscellaneous binaries used by Gnome
rc  gnome-card-game 1.0.1-4       Gnome card games - Solitaire games (FreeCell
rc  gnome-control-c 0.30-2        The Gnome Control Center
ii  gnome-core      1.0.1-0.3     Common files for Gnome core apps and help br
ii  gnome-dev-doc   1.0.3-1       Gnome developers documentation
pn  gnome-games     <none>        (no description available)
ii  gnome-games-loc 1.0.1-4       The locale databases for the gnome-games pa
rc  gnome-gnibbles  1.0.1-4       A cute little game that has no description
rc  gnome-gnobots   1.0.1-4       Gnome version of text based robots game for
pn  gnome-guile     <none>        (no description available)
un  gnome-gxnsnmp   <none>        (no description available)
pn  gnome-gxsnmp    <none>        (no description available)
```

ii gnome-terminal 1.0.1-0.3 The Gnome terminal emulator application

Tabella 24.1. Significato delle lettere utilizzate nelle prime tre colonne del rapporto generato con dpkg -l.

Colonna	Sigla	Significato
1	u	Pacchetto sconosciuto.
1	i	Pacchetto da installare.
1	r	Pacchetto da rimuovere (lasciando la configurazione).
1	p	Pacchetto da eliminare completamente.
2	n	Pacchetto non installato.
2	i	Pacchetto installato.
2	c	Sono presenti solo i file di configurazione.
2	u	Pacchetto estratto dall'archivio, ma non configurato.
2	f	Configurazione interrotta.
2	h	Installazione interrotta.
3		Nessuno stato particolare.
3	h	Segnato per la conservazione alla versione attuale.
3	r	Si richiede la reinstallazione.
3	x	Equivalente a ?h? e a ?r? messi assieme.

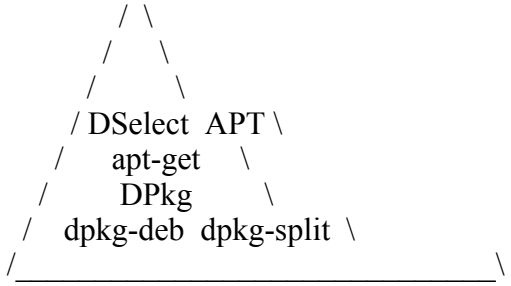
Disponibilita` di un pacchetto

La gestione dei pacchetti Debian, richiede che i programmi che ne consentono l'installazione, siano in grado di sapere quali sono i pacchetti effettivamente disponibili. I pacchetti disponibili sono quelli che si trovano in una distribuzione su CD-ROM, in una copia locale nel file system (eventualmente condiviso in rete), in un sito Internet,... In ogni caso, tali informazioni sono contenute in un file che accompagna i pacchetti (uno per ogni raggruppamento principale) e si tratta di Packages, o Packages.cd nel caso di distribuzioni suddivise su piu` dischi (CD-ROM, o dischi di altro tipo che possono essere sostituiti durante l'installazione).

Stratificazione degli strumenti di gestione dei pacchetti Debian

Gli strumenti per la gestione dei pacchetti Debian sono molti e sono fatti per intervenire a livelli diversi. La figura 24.1 mostra la piramide, alla base della quale si trovano gli strumenti fondamentali.

Figura 24.1. Gerarchia semplificata tra gli strumenti di gestione dei pacchetti Debian.



In breve:

dpkg-deb interviene solo al livello di archivi Debian, consentendo l'estrazione e l'archiviazione in questo formato;

dpkg-split e` uno strumento aggiuntivo in grado di suddividere e riassemblare assieme gli archivi Debian, in modo da poterli gestire in file piu` piccoli, soprattutto quando questi devono essere trasportati su dischetti;

DPkg (l'eseguibile dpkg) interviene nei pacchetti Debian, a livello elementare, consentendone l'installazione e la loro disinstallazione, avvalendosi eventualmente di dpkg-deb quando necessario;

apt-get interviene nei pacchetti Debian, a livello piu` evoluto di DPkg, essendo in grado di risolvere da solo molti problemi di dipendenze;

DSelect si trova al livello piu` alto per la gestione dei pacchetti (assieme a APT) e si avvale di tutti gli strumenti inferiori;

APT e` un sistema di strumenti paralleli a DSelect, composto da diversi programmi frontali alternativi che poi si avvalgono di apt-get per lo svolgimento dei loro compiti.

Gestione elementare attraverso gli strumenti fondamentali

Per l'installazione e la disinstallazione dei pacchetti Debian, lo strumento piu` banale e` dato da DPkg, composto dall'eseguibile dpkg. Questo e` in grado di utilizzare a sua volta anche dpkg-deb, quando si vuole intervenire a livello di archivio Debian.

dpkg [opzioni] azione

Alla fine della riga di comando dell'eseguibile dpkg deve apparire l'indicazione di un'azione, che puo` essere preceduta da alcune opzioni. Come si intuisce, l'azione stabilisce cosa debba essere fatto, mentre le opzioni ne alterano l'ambito. Qui viene mostrato solo l'uso di alcune azioni, perche' per le altre conviene agire attraverso strumenti di livello superiore. Non viene mostrato l'uso delle opzioni, comunque si puo` approfondire l'uso di DPkg consultando la pagina di manuale dpkg(8).

Alcune azioni

-c archivio_debian... | --contents archivio_debian...

Questa azione richiama l'utilizzo di dpkg-deb allo scopo di mostrare l'elenco dei file contenuti nell'archivio Debian indicato come argomento.

-f archivio_debian... | --field archivio_debian...

Questa azione richiama l'utilizzo di dpkg-deb allo scopo di mostrare le informazioni di

controllo sui pacchetti contenuti negli archivi elencati.

-I archivio_debian... | --info archivio_debian...

Questa azione richiama l'utilizzo di dpkg-deb allo scopo di mostrare tutte le informazioni disponibili sui pacchetti contenuti negli archivi elencati.

-i archivio_debian... | --install archivio_debian...

Installa o aggiorna i pacchetti contenuti negli archivi indicati come argomento. Questa azione puo` essere abbinata all'opzione -R, o --recursive, allo scopo di installare tutti i pacchetti i cui archivi si trovano in una directory, che diventerebbe quindi l'argomento di questa azione.

L'installazione del pacchetto include anche la configurazione dello stesso.

--configure pacchetto...

Richiede espressamente di eseguire la configurazione dei pacchetti indicati (ammesso che questi non risultino gia` installati e configurati correttamente).

-r pacchetto... | --remove pacchetto...

Rimuove i pacchetti indicati, senza eliminare i file di configurazione.

--purge pacchetto...

Elimina completamente i pacchetti indicati, compresi i file di configurazione.

-l [modello_pacchetti...] | --list [modello_pacchetti...]

Mostra l'elenco dei pacchetti corrispondenti ai modelli indicati (si usano i caratteri jolly comuni, proteggendoli in modo che la shell non li interpreti direttamente). Se vengono indicati dei modelli, vengono fornite informazioni su tutti i pacchetti conosciuti, non solo quelli installati, mentre utilizzando l'azione senza argomenti, si ottengono informazioni solo sui pacchetti installati, o che comunque hanno mantenuto i file di configurazione.

-s pacchetto... | --status pacchetto...

Mostra le informazioni sullo stato dei pacchetti indicati, aggiungendo anche la descrizione se il pacchetto risulta installato. In questo caso non si possono piu` utilizzare i caratteri jolly.

-L pacchetto... | --listfiles pacchetto...

Elenca i file che appartengono a un pacchetto, in base a quanto contenuto nell'archivio originale. Tuttavia, non e` possibile conoscere in questo modo quali file sono stati creati dagli script di installazione del pacchetto stesso.

-S modello_file... | --search modello_file...

Permette di fare una ricerca per trovare a quali pacchetti appartengono i file indicati come argomento, con o senza l'ausilio di caratteri jolly. In particolare, se si indica un nome o un modello senza l'indicazione di un percorso, si ottengono tutti i pacchetti che hanno file o directory con quel nome.

-C | --audit

Controlla i pacchetti installati per determinare quali sono stati installati solo parzialmente.

--compare-versions versione_1 operatore versione_2

Si tratta di una richiesta particolare con la quale si ottiene il confronto tra due stringhe che rappresentano una versione (completa di revisione). Cio` puo` essere molto utile nella realizzazione di script. Se il confronto da un risultato Vero, l'eseguibile dpkg restituisce zero, mentre negli altri casi restituisce un valore maggiore. Gli operatori che possono essere utilizzati sono elencati nella tabella 24.2

Tabella 24.2. Espressioni per il confronto tra le versioni attraverso l'uso dell'azione --compare-versions con dpkg.

Espressione	Descrizione
ver1 eq ver2	Vero se le versioni sono uguali.
ver1 ne ver2	Vero se le versioni sono differenti.
ver1 lt ver2	Vero se la prima versione e` minore della seconda.
ver1 le ver2	Vero se la prima versione e` minore o uguale alla seconda.
ver1 gt ver2	Vero se la prima versione e` maggiore della seconda.
ver1 ge ver2	Vero se la prima versione e` maggiore o uguale alla seconda.

Esempi

$ dpkg -c zsh_3.1.2-10.deb

Mostra l'elenco dei file che compongono il pacchetto zsh, contenuti nell'archivio indicato, esclusi i file che vengono creati dagli script del pacchetto stesso.

$ dpkg -I zsh_3.1.2-10.deb

Mostra tutte le informazioni disponibili sull'archivio indicato.

dpkg -i zsh_3.1.2-10.deb

Installa, o aggiorna, il pacchetto contenuto nell'archivio indicato, ammesso che cio` sia possibile in relazione alle dipendenze di questo.

dpkg -r zsh

Rimuove il pacchetto indicato, senza eliminare i file di configurazione.

dpkg --purge zsh

Elimina completamente il pacchetto indicato, compresi i file di configurazione.

$ dpkg -l

Elenca lo stato di tutti i pacchetti installati, o dei quali rimangono i file di configurazione.

$ dpkg -l z*

Elenca lo stato di tutti i pacchetti conosciuti che iniziano con la lettera ?z?. Si osservi l'uso della barra obliqua inversa per proteggere l'asterisco contro l'interpretazione da parte della shell.

$ dpkg -s zsh

Mostra le informazioni sullo stato del pacchetto indicato, in modo piu` dettagliato.

$ dpkg -L zsh

Elenca i file che appartengono al pacchetto zsh.

$ dpkg -S /bin/cat

Cerca di scoprire a chi appartiene il file /bin/cat (scoprendo che appartiene al pacchetto textutils).

Gestione piu` evoluta dei pacchetti: organizzazione di una copia della distribuzione

Per una gestione piu` evoluta dei pacchetti, occorre definire la fonte di questi, dalla quale deve essere ottenibile il file Packages. Per comprendere la cosa, e` necessario prima di tutto conoscere in che modo dovrebbe essere organizzato un CD-ROM o una copia nel file system della distribuzione. Lo schema seguente rappresenta l'essenziale (la metavariabile arch viene sostituita dalla sigla dell'architettura per cui sono fatti i pacchetti):

```
debian/
 |-- .disk/
 |    `-- info
 |
 `-- dists/
      |-- stable/
      |    |-- main/
      |    |    `-- binary-arch/
      |    |         |-- Packages
```

```
|   |       |-- Packages.gz
|   |       |-- Packages.cd
|   |       |-- Packages.cd.gz
|   |       `-- *.deb
|   |
|   |-- contrib/
|   |   `-- binary-arch/
|   |       |-- Packages
|   |       |-- Packages.gz
|   |       |-- Packages.cd
|   |       |-- Packages.cd.gz
|   |       `-- *.deb
|   |
|   |-- non-free/
|   |   `-- binary-arch/
|   |       |-- Packages
|   |       |-- Packages.gz
|   |       |-- Packages.cd
|   |       |-- Packages.cd.gz
|   |       `-- *.deb
|   |
|   |-- non-US/
|   |   `-- binary-arch/
|   |       |-- Packages
|   |       |-- Packages.gz
|   |       |-- Packages.cd
|   |       |-- Packages.cd.gz
|   |       `-- *.deb
|   |
|   `-- local/
|       `-- binary-arch/
|           |-- Packages
|           |-- Packages.gz
|           |-- Packages.cd
|           |-- Packages.cd.gz
|           `-- *.deb
|
`--local/
   `-- local --> ../stable/local
```

Il file debian/.disk/info e` indispensabile quando la distribuzione e` suddivisa su piu` dischi. Questo file contiene una riga che serve a identificare il supporto. I file debian/dists/stable/*/binary-arch/Packages, assieme alle loro versioni compresse con gzip, contengono le informazioni sui pacchetti contenuti negli archivi che si trovano nella stessa directory, o in quelle successive, mentre i file debian/dists/stable/*/binary-arch/Packages.cd contengono le informazioni di tutti i file Packages delle stesse directory per tutti i dischi in cui si articola la distribuzione.

Di tutta questa struttura, la directory debian/ e` la radice, o l'inizio, e questa e` la posizione che viene richiesta dai programmi per la gestione dei pacchetti quando devono attingere le informazioni sulla disponibilita` dei pacchetti.

Come si vede dalla struttura mostrata, una distribuzione Debian si articola anche in componenti, a causa di possibili limitazioni nell'uso e nella distribuzione del software relativo. In generale, gli archivi che si trovano a partire da debian/dists/stable/main/ sono quelli principali che non contengono limitazioni particolari, mentre per gli altri valgono considerazioni differenti. Le varie componenti in cui si articola una distribuzione sono identificate dai nomi delle directory che si diramano da debian/dists/stable/.

APT a livello essenziale

Il sistema APT si basa sul programma di servizio apt-get, che a sua volta si avvale di dpkg.

Per funzionare, apt-get richiede la presenza di un file di configurazione, /etc/apt/sources.list, all'interno del quale vanno elencate le fonti da cui si possono ottenere delle distribuzioni Debian. Questo file puo` contenere commenti, preceduti dal simbolo #, righe vuote che vengono ignorate come i commenti e righe contenenti ognuna l'indicazione di un'origine, espressa secondo la sintassi seguente:

deb uri_inizio_distribuzione distribuzione componente...

Per esempio, per indicare l'utilizzo della distribuzione stable (come si e` visto fino a questo punto negli esempi) contenuta a partire da /home/pippo/debian/, della quale si vogliono tutte le componenti normali, si puo` utilizzare la direttiva seguente:

deb file:/home/pippo/debian/ stable main contrib non-free non-US local

Nello stesso modo si potrebbero indicare degli URI riferiti a siti FTP o HTTP. Inoltre, e` importante tenere presente che si possono indicare molte fonti differenti, come si vede dall'esempio seguente:

deb file:/home/pippo/1/debian/ stable main contrib non-free non-US local
deb file:/home/pippo/2/debian/ stable main contrib non-free non-US local
deb http://ftp.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable non-US

Dopo la configurazione, apt-get richiede espressamente che gli sia ordinato di leggere le sorgenti per aggiornare l'elenco locale dei pacchetti disponibili, in modo da disporre di una visione di cio` che esiste e delle dipendenze relative. Si osservi lo schema sintattico per l'utilizzo di apt-get:

apt-get [opzioni] [comando] [pacchetto...]

Da quello che si vede, nella riga di comando di apt-get non si fa mai riferimento direttamente ad archivi Debian, ma sempre solo a pacchetti.

Per comprendere il funzionamento di apt-get e` bene cominciare da alcuni esempi. Normalmente si inizia aggiornando l'elenco locale dei pacchetti disponibili;

```
# apt-get update
```

quindi potrebbe essere conveniente chiedere l'aggiornamento dei pacchetti, riferiti alla stessa versione della distribuzione che si sta utilizzando.

```
# apt-get upgrade
```

Per installare o aggiornare un pacchetto specifico, soddisfacendo le dipendenze necessarie, si puo` intervenire come nell'esempio seguente,

```
# apt-get install zsh
```

dove si mostra in che modo installare o aggiornare il pacchetto zsh, rispettando le dipendenze. Infine, per aggiornare il proprio sistema a una nuova versione della distribuzione, si utilizza il comando

```
# apt-get -f dist-upgrade
```

seguito generalmente da una richiesta esplicita di configurazione dei pacchetti che ne avessero bisogno, per mezzo di dpkg:

```
# dpkg --configure --pending
```

Alcuni comandi

update

Risincronizza l'elenco locale dei pacchetti rispetto alle origini dichiarate nel file di configurazione. In generale, prima di un comando upgrade o dist-upgrade, occorrerebbe eseguire un comando update.

upgrade

Aggiorna tutti i pacchetti che risultano gia` installati, per i quali e` disponibile un aggiornamento. L'aggiornamento non viene fatto se questo puo` provocare degli inconvenienti.

dist-upgrade

Aggiorna tutti i pacchetti che risultano gia` installati, per i quali e` disponibile un aggiornamento. L'aggiornamento viene fatto tenendo conto delle dipendenze, che nel frattempo potrebbero essere cambiate (di solito quando si tratta di un aggiornamento che coinvolge l'intera distribuzione).

Di solito, dopo questo tipo di operazione, si avvia il comando dpkg --configure --pending allo scopo di procedere con la configurazione di cio` che richiede tale passaggio.

install pacchetto...

Installa i pacchetti indicati come argomento, provvedendo a sistemare anche le dipendenze. E` bene sottolineare che vanno indicati i nomi dei pacchetti e non i nomi degli archivi che li contengono.

check

Esegue un controllo, anche sui pacchetti che sono stati installati in modo errato.

clean

APT utilizza un deposito transitorio per gli archivi utilizzati per l'installazione o l'aggiornamento. Si tratta della directory /var/cache/apt/archives/, che va ripulita periodicamente attraverso il comando clean.
Alcune opzioni

-f | --fix-broken

Con l'uso di questa opzione si fa in modo che apt-get cerchi di sistemare i problemi legati alle dipendenze. Questa opzione puo` essere usata da sola, senza l'indicazione di un comando, oppure con un comando, di solito dist-upgrade, per richiedere espressamente la sistemazione dei problemi che si possono generare con lo stesso.

-s | --simulate

Simula l'esecuzione del comando, in modo da mostrare cosa si otterrebbe.

Ricerca dei file che apparentemente non appartengono ad alcun pacchetto

Con l'aiuto di DPkg e` possibile cercare di individuare a quale pacchetto appartiene questo o quel file. Precisamente si usa l'opzione -S, come nell'esempio seguente:

$ dpkg -S /etc/timezone

In questo caso si fa riferimento al file /etc/timezone e si dovrebbe ottenere una segnalazione simile a quella seguente, da cui si comprende che il file e` abbinato al pacchetto timezones:

timezones: /etc/timezone

Per avere una visione di insieme dei file che potrebbero essere stati abbandonati inavvertitamente, si puo` usare Cruft, che scandisce il file system e genera un rapporto. Naturalmente, non tutto cio` che viene indicato e` necessariamente un file superfluo, ma e` comunque il punto di partenza per la propria ricerca.

cruft [opzioni]

L'eseguibile cruft puo` essere usato solo dall'utente root e l'elenco che genera, di file sospetti, viene emesso attraverso lo standard output, a meno che siano usate delle opzioni particolari.

Pacchetti Debian sorgenti

I pacchetti sorgenti messi a disposizione dalla distribuzione GNU/Linux Debian sono composti generalmente da tre file:

nome_pacchetto_versione-revisione.dsc

nome_pacchetto_versione-revisione.orig.tar.gz

nome_pacchetto_versione-revisione.diff.gz

Il file con estensione .dsc contiene informazioni essenziali sul pacchetto, con le firme eventuali, per garantire la sua integrita`. Il file con estensione .orig.tar.gz contiene i sorgenti originali, archiviati attraverso tar+gz. Il file con estensione .diff.gz e` un file di differenze da applicare per l'adattamento alla distribuzione; eventualmente questo file potrebbe mancare se il pacchetto nasce espressamente per la distribuzione Debian.

Per compilare un pacchetto occorre prima estrarlo, applicandogli le modifiche previste. Per farlo nel modo corretto, si usa il comando seguente, dove si suppone che nella directory corrente siano disponibili i tre file del pacchetto che interessa:

$ dpkg-source -x nome_pacchetto_versione-revisione.dsc

In questo modo si ottiene la directory nome_pacchetto-versione/, all'interno della quale bisogna entrare per avviare uno script che viene creato proprio con l'applicazione delle modifiche:

$ cd nome_pacchetto-versione

$ su

debian/rules binary

Come si vede, e` stato necessario acquisire i privilegi dell'utente root per procedere alla compilazione.

In alternativa, se manca il file con estensione .dsc, si puo` rimediare nel modo seguente:

$ tar xzvf nome_pacchetto_versione-revisione.orig.tar.gz

$ cd nome_pacchetto-versione

```
$ cat ../nome_pacchetto_versione-revisione.diff.gz | gunzip | patch -p1

$ chmod a+x debian/rules

$ su

# debian/rules binary
```

Come si vede, si devono applicare le modifiche manualmente, quindi occorre attribuire al file debian/rules i permessi di esecuzione. Il resto funziona regolarmente.

Al termine si ottiene il file nome_pacchetto_versione-revisione_arch.deb, contenente il pacchetto binario pronto per l'installazione.

E` evidente che, se si vogliono apportare delle modifiche ulteriori al sorgente, queste vanno fatte dopo l'estrazione e dopo l'applicazione delle modifiche gia` previste per la distribuzione Debian. Volendo ricostruire un pacchetto sorgente corretto, si interviene secondo la sequenza seguente.

1. Si estrae l'archivio originale e si applicano le modifiche gia` previste:

   ```
   $ tar xzvf nome_pacchetto_versione-revisione.orig.tar.gz

   $ cd nome_pacchetto-versione

   $ cat ../nome_pacchetto_versione-revisione.diff.gz | gunzip | patch -p1

   $ cd ..
   ```
2. Si fanno le modifiche aggiuntive che si ritengono necessarie.
3. Si cancella il file con estensione .diff.gz e .dsc:

   ```
   $ rm nome_pacchetto_versione-revisione.diff.gz

   $ rm nome_pacchetto_versione-revisione.dsc
   ```
4. Si ricostruisce tutto con dpkg-source:

   ```
   $ dpkg-source -b nome_pacchetto-versione
   ```

   In pratica, l'argomento dell'opzione -b e` il nome della directory contenente i sorgenti

modificati.

Al termine si ottiene un file nome_pacchetto_versione-revisione.diff.gz nuovo, assieme al file nome_pacchetto_versione-revisione.dsc appropriato.

cap 13) IPv4: configurazione delle interfacce di rete

La connessione in una rete basata su IP necessita inizialmente dell'assegnazione di indirizzi IP e quindi di un instradamento per determinare quale strada, o itinerario, devono prendere i pacchetti per raggiungere la destinazione.

Generalmente, ma non necessariamente, valgono queste regole:

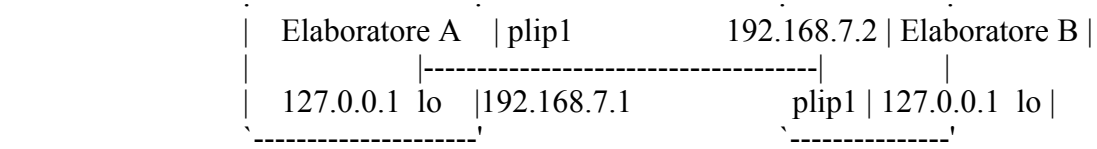
- ogni interfaccia di rete ha un proprio indirizzo IP;
- un'interfaccia di rete di un elaboratore puo` comunicare con un'interfaccia di un altro elaboratore solo se queste sono fisicamente connesse alla stessa rete;
- un'interfaccia di rete di un elaboratore puo` comunicare con un'interfaccia di un altro elaboratore solo se gli indirizzi di queste interfacce appartengono alla stessa rete.

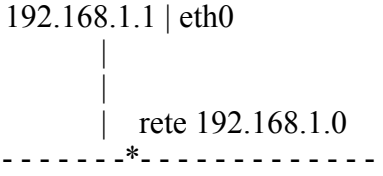
In un sistema GNU/Linux, per poter gestire una connessione in rete di qualunque tipo, occorre un kernel predisposto in modo da attivarne la gestione.

E` necessario anche provvedere alla gestione delle interfacce di rete particolari che si utilizzano. Cio` puo` essere fatto sia attraverso la realizzazione di un kernel monolitico, sia modulare. Per quanto riguarda la gestione specifica di ogni singola scheda, la tendenza e` quella di usare preferibilmente i moduli.

Configurazione delle interfacce di rete

La configurazione di un'interfaccia implica essenzialmente l'attribuzione di un indirizzo IP. Un indirizzo IP di un'interfaccia vale in quanto inserito in una rete logica, identificata anche questa da un proprio indirizzo IP. Pertanto, quando si assegna un indirizzo a un'interfaccia, occorre anche stabilire la rete a cui questo appartiene, attraverso la maschera di rete, con la quale, il risultato di indirizzo_di_interfaccia AND maschera_di_rete genera l'indirizzo della rete.





Lo schema mostra la situazione di due elaboratori, che si potrebbe riassumere sinteticamente nelle due tabelle seguenti, riferite rispettivamente all'elaboratore ?A? e all'elaboratore ?B?:

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
lo virtuale 127.0.0.1 255.0.0.0 127.255.255.255 --
plip1 porta parallela 192.168.7.1 255.255.255.255 -- 192.168.7.2
eth0 Ethernet 192.168.1.1 255.255.255.0 192.168.1.255 --

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
lo virtuale 127.0.0.1 255.0.0.0 127.255.255.255 --
plip1 porta parallela 192.168.7.2 255.255.255.255 -- 192.168.7.1

Per la spiegazione di questa configurazione vengono mostrati nelle sezioni seguenti degli esempi ottenuti con un sistema GNU/Linux, attraverso il programma Ifconfig (1) (Interface configuration), a cui corrisponde l'eseguibile ifconfig. Tuttavia, il concetto rimane tale per gli altri sistemi operativi, anche se il comando che si usa per impostare le interfacce di rete puo` avere un nome e un funzionamento differente.

Loopback

Un elaboratore connesso o meno a una rete fisica vera e propria, deve avere una connessione virtuale a una rete immaginaria interna allo stesso elaboratore. A questa rete virtuale inesistente si accede per mezzo di un'interfaccia immaginaria, che in un sistema GNU/Linux e` denominata lo, e l'indirizzo utilizzato e` sempre lo stesso, 127.0.0.1, ma ugualmente deve essere indicato esplicitamente.

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
lo virtuale 127.0.0.1 255.0.0.0 127.255.255.255 --

Come si vede dallo schema, la maschera di rete e` quella di una classe A e, di solito, il comando che si usa per associare l'indirizzo all'interfaccia locale determina da solo questa maschera. In un sistema GNU/Linux si puo` definire il nodo locale in modo molto semplice:

ifconfig lo 127.0.0.1

Quindi, si puo` controllare la configurazione:

$ ifconfig lo[Invio]

lo Link encap:Local Loopback

```
        inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
        UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
        ...
```

E` indispensabile che sia presente l'interfaccia locale virtuale per il buon funzionamento del sistema, soprattutto quando l'elaboratore ha gia` una connessione a una rete reale. Infatti, si potrebbe essere tentati di non definire tale interfaccia, oppure di non attivare l'instradamento relativo, quando sono presenti altre interfacce fisiche reali, ma cio` potrebbe provocare un malfunzionamento intermittente della rete.

Ethernet

La configurazione degli indirizzi di una scheda di rete Ethernet e` la cosa piu` comune: si tratta semplicemente di abbinare all'interfaccia il suo indirizzo stabilendo il proprio ambito di competenza, attraverso la maschera di rete. In precedenza era stato mostrato un esempio di configurazione schematizzato nel modo seguente:

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
eth0 Ethernet 192.168.1.1 255.255.255.0 192.168.1.255 --

In questo modo, l'indirizzo 192.168.1.1 risulta assegnato all'interfaccia eth0, che in un sistema GNU/Linux rappresenta la prima scheda Ethernet. La maschera di rete, 255.255.255.0, fa si` che l'indirizzo di rete sia 192.168.1.0; infatti, 192.168.1.1 AND 255.255.255.0 = 192.168.1.0.

In un sistema GNU/Linux, si definisce questo abbinamento con il comando seguente:

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
```

In questo caso, tuttavia, dal momento che l'indirizzo 192.168.1.1 appartiene alla classe C, la maschera di rete predefinita sarebbe stata la stessa di quella che e` stata indicata esplicitamente.

La verifica della configurazione potrebbe dare l'esito seguente:

```
$ ifconfig eth0[Invio]

eth0      Link encap:10Mbps Ethernet  HWaddr 00:4F:56:00:11:87
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...
```

Connessioni punto-punto

Le connessioni di tipo punto-punto, ovvero quelle in cui si possono collegare solo due punti alla volta, hanno caratteristiche diverse da quelle di tipo a bus, come nel caso della tecnologia Ethernet. In linea di massima si puo` dire che questo tipo di connessione implichi la specificazione di entrambi gli indirizzi dei due punti collegati, cioe` delle

rispettive interfacce. Tuttavia, la configurazione effettiva dipende anche dalle strategie che si vogliono adottare. A titolo di esempio si fa riferimento a una connessione PLIP, che si ottiene collegando due elaboratori con un cavo apposito attraverso le porte parallele.

Il modo piu` semplice, da un punto di vista intuitivo, per configurare una connessione punto-punto, e` quello di trattarla come se fosse una connessione a bus. Per esempio, i due lati della connessione potrebbero essere definiti rispettivamente nel modo seguente:

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
plip1 porta parallela 192.168.7.1 255.255.255.0 -- 192.168.7.2

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
plip1 porta parallela 192.168.7.2 255.255.255.0 -- 192.168.7.1

Come si vede, si dichiara una maschera di rete che impegna un ottetto completo per connettere i due nodi. Segue il comando corrispondente, da utilizzare in un sistema GNU/Linux dal lato del primo dei due nodi:

```
# ifconfig plip1 192.168.7.1 pointopoint 192.168.7.2 <-'
`->netmask 255.255.255.0
```

Come si comprende intuitivamente, si assegna l'indirizzo 192.168.7.1 all'interfaccia parallela plip1 locale e si stabilisce l'indirizzo 192.168.7.2 per l'altro capo della comunicazione. Il risultato e` che si dovrebbe generare la configurazione seguente:

```
$ ifconfig plip1[Invio]

plip1     Link encap:Ethernet  HWaddr FC:FC:C0:A8:64:84
          inet addr:192.168.7.1  P-t-P:192.168.7.2  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
          ...
```

Dall'altro capo della connessione si deve eseguire la configurazione opposta. Per seguire l'esempio mostrato, si deve usare il comando seguente:

```
# ifconfig plip1 192.168.7.2 pointopoint 192.168.7.1 <-'
`->netmask 255.255.255.0
```

In alternativa, dal momento che si tratta di una connessione di due soli punti, non e` sempre indispensabile indicare precisamente l'indirizzo all'altro capo: di solito si puo` fare in modo che venga accettato qualunque indirizzo, facilitando la configurazione.

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
plip1 porta parallela 192.168.7.1 255.255.255.0 -- 0.0.0.0

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
plip1 porta parallela 192.168.7.2 255.255.255.0 -- 0.0.0.0

Sempre con un sistema GNU/Linux, la configurazione del primo nodo puo` essere ottenuta in questo modo alternativo:

```
# ifconfig plip1 192.168.7.1 pointopoint 0.0.0.0 netmask 255.255.255.0
```

L'esempio che si vede sopra e` lo stesso gia` proposto con la variante dell'indicazione dell'indirizzo all'altro capo. In questo caso, 0.0.0.0 fa in modo che venga accettata la connessione con qualunque indirizzo.

```
$ ifconfig plip1[Invio]

plip1     Link encap:Ethernet  HWaddr FC:FC:C0:A8:64:84
          inet addr:192.168.7.1  P-t-P:0.0.0.0  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
          ...
```

Dall'altro capo della connessione ci si puo` comportare in modo analogo, come nell'esempio seguente:

```
# ifconfig plip1 192.168.7.2 pointopoint 0.0.0.0 netmask 255.255.255.0
```

Tuttavia, e` bene trattare le connessioni punto-punto per quello che sono, pertanto e` bene specificare una maschera di rete che non impegni altri indirizzi se non quelli indicati. In pratica, si tratta si usare la maschera 255.255.255.255, che tra l'altro e` quella predefinita in questo tipo di connessione.

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
plip1 porta parallela 192.168.7.1 255.255.255.255 -- 192.168.7.2

Interfaccia Tipo Indirizzo IP Maschera di rete Indirizzo broadcast Indirizzo punto-punto
plip1 porta parallela 192.168.7.2 255.255.255.255 -- 192.168.7.1

Ecco il comando corrispondente per GNU/Linux:

```
# ifconfig plip1 192.168.7.1 pointopoint 192.168.7.2 <-'
`->netmask 255.255.255.255
```

L'esempio mostra una configurazione in cui si specificano gli indirizzi IP di entrambi i punti. In alternativa, anche in questo caso, si puo` fare a meno di indicare espressamente l'indirizzo dell'altro capo, come nell'esempio seguente:

```
# ifconfig plip1 192.168.7.1 pointopoint 0.0.0.0 <-'
`->netmask 255.255.255.255
```

Il vantaggio di usare questo tipo di configurazione sta nel risparmio di indirizzi; lo svantaggio sta nella necessita` di stabilire instradamenti specifici per ognuno dei due punti (questo particolare verra` chiarito in seguito).

Configurazione delle interfacce di rete con un sistema GNU/Linux

In un sistema GNU/Linux, le interfacce di rete vengono identificate attraverso un nome, assegnato dal kernel nel momento della loro identificazione. Alcuni nomi di interfaccia di rete sono elencati nella tabella 112.1.

La configurazione delle interfacce di rete avviene attraverso Ifconfig (l'eseguibile ifconfig), che consente impostazioni differenti a seconda della famiglia di protocolli a cui si intende fare riferimento. In particolare, il riferimento a IPv4 e` implicito, ma si puo` indicare esplicitamente attraverso la parola chiave inet (mentre inet6 fa riferimento a IPv6).

 # ifconfig

ifconfig [interfaccia]

ifconfig [interfaccia... [famiglia_indirizzamento] [indirizzo] opzioni]

ifconfig viene utilizzato per attivare e mantenere il sistema delle interfacce di rete residente nel kernel. Viene utilizzato al momento dell'avvio per configurare la maggior parte di questo sistema in modo da portarlo a un livello di funzionamento. Dopo, viene utilizzato di solito solo a scopo diagnostico o quando sono necessarie delle regolazioni. Se non vengono forniti argomenti, oppure se vengono indicate solo delle interfacce, ifconfig visualizza semplicemente lo stato delle interfacce specificate, oppure di tutte se non sono state indicate.

Il primo argomento successivo al nome di interfaccia puo` essere la sigla identificativa di una famiglia di indirizzamento, ovvero di un sistema di protocolli di comunicazione particolare. A seconda del tipo di questo, cambia il modo di definire gli indirizzi che si attribuiscono alle interfacce. Se questo non viene specificato, come si fa di solito, si intende fare riferimento al sistema di protocolli che si basano su IPv4.

L'indirizzo e` il modo con cui l'interfaccia viene riconosciuta all'interno del tipo di protocollo particolare che si utilizza. Nel caso di IP, puo` essere indicato l'indirizzo IP numerico o il nome di dominio, che in questo caso sara` convertito automaticamente (sempre che cio` sia possibile) nell'indirizzo numerico corretto.

Alcune opzioni

up | down

L'opzione up attiva l'interfaccia. Quando all'interfaccia viene attribuito un nuovo indirizzo, questa viene attivata implicitamente. L'opzione down disattiva l'interfaccia.

arp | -arp

Abilita o disabilita l'uso del protocollo ARP per questa interfaccia.

allmulti | -allmulti

Abilita o disabilita la modalita` promiscua dell'interfaccia. Cio` permette di fare un monitoraggio della rete attraverso applicazioni specifiche che cosi` possono analizzare ogni pacchetto che vi transita, anche se non e` diretto a quella interfaccia.

mtu n

Permette di specificare l'unita` massima di trasferimento (MTU o Max transfer unit) dell'interfaccia. Per le schede Ethernet, questo valore puo` variare in un intervallo di 1 000-2 000 (il valore predefinito e` 1 500). Per il protocollo SLIP si possono utilizzare valori compresi tra 200 e 4 096. E` da notare pero` che attualmente non e` possibile gestire la frammentazione IP, di conseguenza, e` meglio utilizzare un MTU sufficientemente grande.

pointopoint [indirizzo_di_destinazione] | -pointopoint

Abilita o disabilita la modalita` punto-punto per questa interfaccia. La connessione punto-punto e` quella che avviene tra due elaboratori soltanto. Se viene indicato l'indirizzo, si tratta di quello dell'altro nodo.

netmask indirizzo_di_netmask

Stabilsce la maschera di rete per questa interfaccia. L'indicazione della maschera di rete puo` essere omessa, in tal caso, viene utilizzato il valore predefinito che e` determinato in base alla classe a cui appartiene l'indirizzo (A, B o C). Naturalmente, se si usa una sottorete, il valore della maschera di rete non puo` coincidere con quello predefinito.

irq numero_di_irq

Alcune interfacce permettono di definire il numero di IRQ in questo modo. Nella maggior parte dei casi, cio` non e` possibile.

broadcast [indirizzo] | -broadcast

Abilita o disabilita la modalita` broadcast per questa interfaccia. Se abilitandola, viene indicato l'indirizzo, si specifica l'indirizzo broadcast di questa interfaccia.

multicast

Questa opzione permette di attivare esplicitamente la modalita` multicast, anche se normalmente cio` viene determinato automaticamente in base al tipo di interfaccia utilizzato.
Esempi

```
# ifconfig lo 127.0.0.1
```

Attiva l'interfaccia lo corrispondente al loopback con il noto indirizzo IP 127.0.0.1.

ifconfig eth0 192.168.1.1 netmask 255.255.255.0

Attiva l'interfaccia eth0 corrispondente alla prima scheda Ethernet, con l'indirizzo IP 192.168.1.1 e la maschera di rete 255.255.255.0.

$ ifconfig eth0

Emette la situazione dell'interfaccia eth0 corrispondente alla prima scheda Ethernet.

$ ifconfig

Emette la situazione di tutte le interfacce di rete attivate.

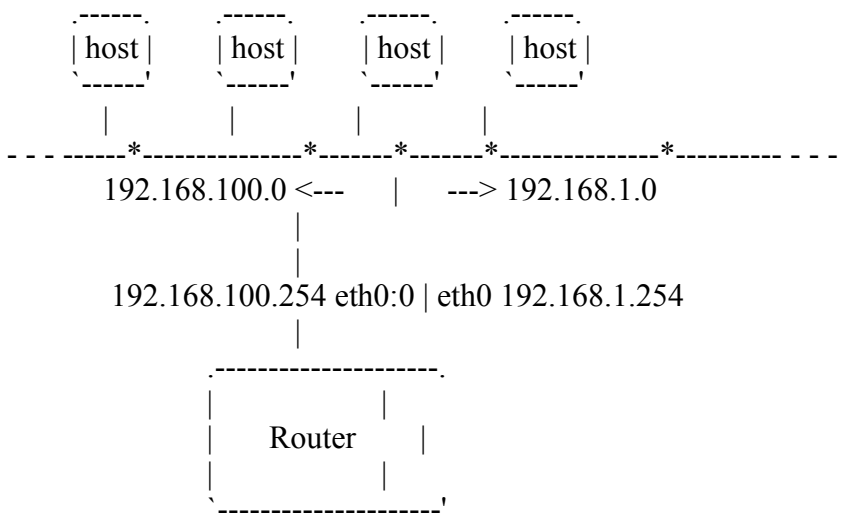
Alias IP

E` possibile attribuire a ogni interfaccia di rete piu` di un indirizzo IP. Cio` si ottiene definendo delle interfacce virtuali, riferite a quelle reali, a cui poi si attribuiscono degli indirizzi IP differenti. Il nome di un'interfaccia virtuale ha l'aspetto seguente:

interfaccia_reale:n_interfaccia_virtuale

Per esempio, eth0 e` il nome reale di un'interfaccia di rete Ethernet, mentre eth0:0, eth0:1,... sono una serie di interfacce virtuali riferite sempre all'interfaccia reale eth0. Naturalmente, lo stesso vale per gli altri tipi di interfaccia di rete: ppp0:n, plip0:n,...

Figura 114.2. Utilizzo ipotetico degli alias IP.



Naturalmente, per ottenere la definizione di alias IP, potrebbe essere necessario predisporre un kernel adatto (sezione 29.2.9).

Nel momento in cui si configura un'interfaccia virtuale, questa viene definita implicitamente. Si interviene nel modo solito attraverso ifconfig. L'esempio seguente si riferisce a quanto mostrato nella figura 114.2, in cui, su una sola rete fisica si distinguono gli indirizzi di due sottoreti differenti: 192.168.1.0 e 192.168.100.0.

```
# ifconfig eth0 192.168.1.254 netmask 255.255.255.0

# ifconfig eth0:0 192.168.100.254 netmask 255.255.255.0
```

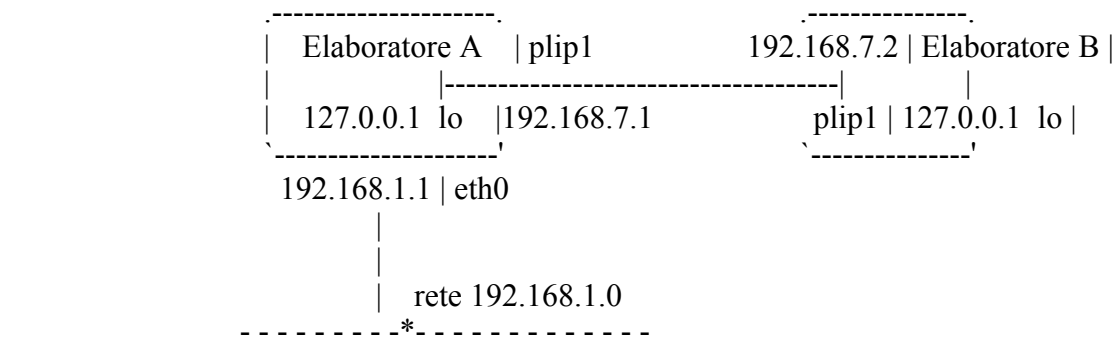
IPv4: instradamento locale

L'instradamento definisce la strada che devono prendere i pacchetti di livello 3 (rete), secondo il modello ISO-OSI, a partire dal nodo a cui si fa riferimento. In questo capitolo, viene preso in considerazione l'instradamento locale, inteso come quello che non si serve di router.

Rete locale

In una rete elementare, in cui ogni elaboratore ha una sola scheda di rete e tutte le schede sono connesse con lo stesso cavo, potrebbe sembrare strana la necessita` di dover stabilire un percorso per l'instradamento dei dati sulla rete. Ma in una rete IPv4 non e` cosi`: per qualunque connessione possibile e` necessario stabilire il percorso, anche quando si tratta di connettersi con l'interfaccia locale immaginaria (loopback).

Ogni elaboratore che utilizza la rete ha una sola necessita`: quella di sapere quali percorsi di partenza siano possibili, in funzione degli indirizzi utilizzati. Gli eventuali percorsi successivi, vengono definiti da altri elaboratori nella rete. Si tratta di costruire la cosiddetta tabella di instradamento, attraverso la quale, ogni elaboratore sa quale strada deve prendere un pacchetto a partire da quella posizione.



Riprendendo l'esempio gia` mostrato a proposito della configurazione delle interfacce di rete, si potrebbero definire le tabelle di instradamento seguenti, che si riferiscono rispettivamente al nodo A e al nodo B dello schema:

Destinazione Maschera di rete Router Interfaccia di rete
192.168.1.0 255.255.255.0 -- eth0

192.168.7.1 255.255.255.255 -- plip1
192.168.7.2 255.255.255.255 -- plip1
127.0.0.0 255.0.0.0 -- lo

Destinazione Maschera di rete Router Interfaccia di rete
192.168.7.1 255.255.255.255 -- plip1
192.168.7.2 255.255.255.255 -- plip1
127.0.0.0 255.0.0.0 -- lo

Quando si configura un'interfaccia di rete e gli si attribuisce l'indirizzo IP, dal momento che esiste una maschera di rete indicata espressamente o predefinita, potrebbe essere lo stesso programma di configurazione dell'interfaccia che si occupa di definirne l'instradamento nella rete locale; a ogni modo, rimane la necessita` di definirlo (in un modo o nell'altro).

Per la spiegazione di questi instradamenti vengono mostrati nelle sezioni seguenti degli esempi ottenuti con un sistema GNU/Linux, attraverso il programma Route, (1) a cui corrisponde l'eseguibile route. Tuttavia, il concetto rimane tale per gli altri sistemi operativi, anche se la modalita` per definire gli instradamenti puo` essere differente.

Loopback

La definizione dell'instradamento per gli indirizzi locali di loopback e` obbligatoria:

Destinazione Maschera di rete Router Interfaccia di rete
127.0.0.0 255.0.0.0 -- lo

Con un sistema GNU/Linux dovrebbe essere lo stesso programma Ifconfig che prepara l'instradamento corretto all'atto dell'impostazione dell'interfaccia lo; tuttavia, usando Route si potrebbe intervenire nel modo seguente:

route add -net 127.0.0.0 netmask 255.0.0.0 dev lo(2)

La tabella di instradamento che si ottiene viene descritta di seguito.

$ route -n[Invio]

Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
127.0.0.0 0.0.0.0 255.0.0.0 U 0 0 2 lo

Di solito la rete 127.0.0.0 serve a raggiungere solo l'indirizzo 127.0.0.1, quindi, spesso si preferisce inserire solo questo nella tabella di instradamento. In pratica si utilizza il comando:

route add -host 127.0.0.1 dev lo

In questo caso non si indica la maschera di rete perche' deve essere necessariamente

255.255.255.255, essendo riferita a un nodo singolo.

La verifica dell'instradamento e` semplice, basta provare a richiedere un eco all'interfaccia lo.

$ ping 127.0.0.1[Invio]

PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.3 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.3 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.3 ms

[Ctrl+c]

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.4 ms

Ethernet

Le schede di rete Ethernet sono usate per la connessione a una rete locale e per questo sono potenzialmente in grado di offrire un collegamento con tutti gli indirizzi che ricadono all'interno della rete logica di cui fanno parte. Quando si stabilisce un instradamento che utilizza questo tipo di interfaccia, e` preferibile l'indicazione dell'intera rete logica a cui appartiene.

Seguendo l'esempio visto in precedenza nella sezione che riguardava la configurazione di una scheda Ethernet, dal momento che questa si trovava a operare nella rete 192.168.1.0, l'instradamento corretto corrisponde allo schema seguente:

Destinazione Maschera di rete Router Interfaccia di rete
192.168.1.0 255.255.255.0 -- eth0

Con un sistema GNU/Linux, se Ifconfig non ha gia` provveduto da solo, si puo` usare Route nel modo seguente:

route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0

La tabella di instradamento che ne deriva viene descritta di seguito.

$ route -n[Invio]

Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 1 eth0

Volendo e` possibile indicare un instradamento specifico per ogni destinazione.

Nell'esempio seguente si aggiunge l'instradamento per alcuni elaboratori: si deve utilizzare route piu` volte.

```
# route add -host 192.168.1.1 dev eth0

# route add -host 192.168.1.2 dev eth0

# route add -host 192.168.1.3 dev eth0

# route add -host 192.168.1.4 dev eth0
```

Si ottiene una tabella di instradamento simile a quella seguente:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.1     0.0.0.0         255.255.255.255 UH    0      0        0 eth0
192.168.1.2     0.0.0.0         255.255.255.255 UH    0      0        0 eth0
192.168.1.3     0.0.0.0         255.255.255.255 UH    0      0        0 eth0
192.168.1.4     0.0.0.0         255.255.255.255 UH    0      0        0 eth0
```

Anche l'indirizzo dell'interfaccia locale, quella del proprio elaboratore, e` raggiungibile solo se e` stato specificato un instradamento. Quando si indicava un instradamento della rete, questa veniva inclusa automaticamente nel gruppo; nel caso si voglia indicare dettagliatamente ogni indirizzo da raggiungere, se si vuole accedere anche alla propria interfaccia, occorre inserirla nella tabella di instradamento. Nell'esempio visto sopra, viene aggiunto anche l'indirizzo 192.168.1.1 per questo scopo.

La verifica dell'instradamento deve essere fatta inizialmente controllando l'interfaccia locale, quindi tentando di raggiungere l'indirizzo di un altro elaboratore sulla rete. Naturalmente, occorre che quell'elaboratore abbia una tabella di instradamento corretta.

```
$ ping 192.168.1.1[Invio]

PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.5 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.4 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.4 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.4 ms

[Ctrl+c]

--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.5 ms

$ ping 192.168.1.2[Invio]

PING 192.168.1.2 (192.168.1.2): 56 data bytes
```

```
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=1.1 ms

[Ctrl+c]

--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.1/1.1/1.1 ms
```

 # route

route [opzioni]

La sintassi di route puo` articolarsi in diversi modi a seconda del tipo di azione da compiere. In particolare, conviene distinguere tra: l'analisi della tabella di instradamento;

route [-v] [-n] [-e | -ee]

l'aggiunta di un nuovo instradamento;

route [-v] add [-net|-host] destinazione [netmask maschera_di_rete] [gw router] <-'
`->[altre_opzioni] [[dev] interfaccia]

l'eliminazione di un instradamento preesistente.

route [-v] del [-net|-host] destinazione [netmask maschera_di_rete] [gw router] <-'
`->[altre_opzioni] [[dev] interfaccia]

In pratica, nel primo caso e` possibile visualizzare (attraverso lo standard output) la tabella di instradamento. Generalmente, per questo scopo, l'uso normale e` proprio quello di route senza argomenti.

Nel secondo caso, l'inserimento di una nuova voce nella tabella di instradamento avviene per mezzo dell'opzione add e dell'indicazione della destinazione da raggiungere. L'indicazione dell'interfaccia e` facoltativa, se puo` essere determinata in modo predefinito.

Nel terzo caso, l'eliminazione di una voce della tabella di instradamento avviene per mezzo dell'opzione del e dell'indicazione della destinazione che prima veniva raggiunta. Anche in questo caso, l'indicazione dell'interfaccia e` facoltativa, se puo` essere determinata in modo predefinito.

Quando si visualizza la tabella degli instradamenti, il programma tenta di risolvere gli indirizzi in nomi. Spesso, questo fatto puo` essere inopportuno, pertanto e` comune l'uso dell'opzione -n con cui si evita tale conversione e non si perde tempo nel tentativo di

risolvere indirizzi che non hanno un nome.

Si osservi che, solitamente, la risoluzione di un indirizzo relativo a una rete, non ha un nome offerto dal servizio DNS, pertanto occorre predisporre il file /etc/networks, per consentire tale trasformazione.

Alcune opzioni

-n

Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi e delle reti. Questo tipo di approccio potrebbe essere utile specialmente quando si hanno difficolta` ad accedere a un servizio di risoluzione dei nomi, o comunque quando si vuole avere la situazione completamente sotto controllo.

-net destinazione

L'indirizzo indicato nella destinazione fa riferimento a una rete. L'indirizzo puo` essere indicato in forma numerica o attraverso un nome di dominio; in questo ultimo caso, la traduzione avviene in base al contenuto del file /etc/networks.

-host destinazione

L'indirizzo indicato nella destinazione fa riferimento a un nodo. L'indirizzo puo` essere indicato in forma numerica o attraverso un nome di dominio.

netmask maschera_di_rete

Permette di specificare la maschera di rete quando si sta facendo riferimento a un indirizzo di rete. Quando si inserisce una voce riferita a un nodo singolo, questa indicazione non ha senso. Quando la maschera di rete e` un dato richiesto, se non viene inserito si assume il valore predefinito che dipende dalla classe a cui appartiene l'indirizzo indicato.

gw router

Fa in modo che i pacchetti destinati alla rete o al nodo per il quale si sta indicando l'instradamento, passino per il router specificato. Per questo, occorre che l'instradamento verso l'elaboratore che funge da router sia gia` stato definito precedentemente e in modo statico.

Normalmente, l'indirizzo utilizzato come router riguarda un'interfaccia collocata in un altro nodo. Eventualmente, per mantenere la compatibilita` con Unix BSD, e` possibile specificare un'interfaccia locale, intendendo cosi` che il traffico per l'indirizzo di destinazione deve avvenire utilizzando quella interfaccia.

metric valore_metrico

Permette di definire il valore metrico dell'instradamento e viene utilizzato dai demoni che si occupano dell'instradamento dinamico per determinare il costo di una strada, o meglio per poter decidere il percorso migliore.

reject

Permette di impedire l'utilizzo di un instradamento.

[dev] interfaccia

Permette di definire esplicitamente l'interfaccia da utilizzare per un certo instradamento. Solitamente, questa informazione non e` necessaria perche' il kernel riesce a determinare l'interfaccia in base alla configurazione delle stesse.

E` importante che questa indicazione appaia alla fine della riga di comando, in questo modo, il parametro dev, che precede il nome dell'interfaccia, e` solo facoltativo.

Utilizzo di Route

Quando si interroga la tabella degli instradamenti, si ottiene una struttura composta da diverse colonne, in cui, quelle principali sono descritte nella tabella 115.7.

Tabella 115.7. Intestazioni della tabella di instradamento.

Nome	Descrizione
Destination	La rete o il nodo di destinazione.
Gateway	Il router. Se appare un asterisco (*) o l'indirizzo 0.0.0.0 significa che non si tratta di un instradamento attraverso un router.
Genmask	In linea di massima corrisponde alla maschera di rete; in particolare, se e` un instradamento verso un nodo appare 255.255.255.255, se invece e` l'instradamento predefinito appare 0.0.0.0 (default).
Flags	Indica diversi tipi di informazioni utilizzando lettere o simboli.
Metric	La distanza o il costo della strada. Rappresenta la distanza (espressa solitamente in hop o salti) per raggiungere la destinazione.
Ref	Il numero di riferimenti all'instradamento. Questa informazione non viene utilizzata dal kernel Linux e, di conseguenza, l'informazione appare sempre azzerata.
Use	Conteggio del numero di volte in cui la voce e` stata visionata.
Iface	Il nome dell'interfaccia da cui partono i pacchetti IP.

I tipi di informazioni che possono essere rappresentati nella colonna Flags sono elencati nella tabella 115.8.

Tabella 115.8. Significato delle lettere e dei simboli utilizzati nella colonna Flags della tabella di instradamento.

Simbolo	Descrizione
U	L'instradamento e` attivo.
H	L'indirizzo indicato fa riferimento a un nodo.
G	Viene utilizzato un router.
R	Instradamento reintegrato (instradamento dinamico).

D Instradamento installato dinamicamente da un demone o attraverso ridirezione.
M Instradamento modificato da un demone o attraverso ridirezione.
! Instradamento impedito (opzione reject).

Seguono alcuni esempi di utilizzo.

route add -host 127.0.0.1 dev lo
Attiva l'instradamento verso l'interfaccia locale loopback.

route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
Attiva l'instradamento della rete 192.168.1.0 che utilizza la maschera di rete 255.255.255.0, specificando che riguarda l'interfaccia di rete eth0.

route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.254
Attiva l'instradamento della rete 192.168.2.0 che utilizza la maschera di rete 255.255.255.0, attraverso il router 192.168.1.254 per il quale era gia` stato definito un instradamento precedentemente.

route add default gw 192.168.1.254
Attiva l'instradamento predefinito (nel caso che non siano disponibili altre possibilita`) attraverso il router 192.168.1.254. La parola default fa automaticamente riferimento all'indirizzo IP 0.0.0.0.

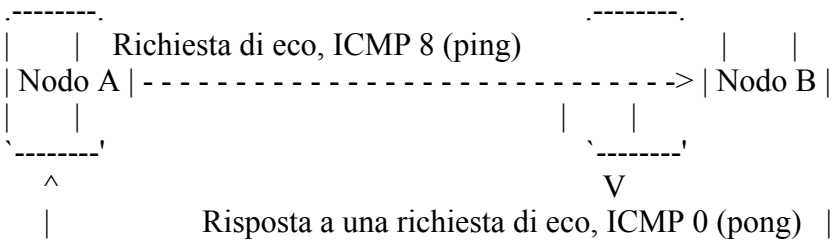
route add 10.0.0.0 netmask 255.0.0.0 reject
Definisce un instradamento il cui accesso deve essere impedito.

$ route
Mostra la tabella di instradamento attuale.

Verifica di un instradamento

La definizione degli instradamenti, serve per stabilire un collegamento con le interfacce di altri elaboratori. Quando anche le tabelle di instradamento degli altri elaboratori sono corrette, si puo` verificare che le comunicazioni sono possibili attraverso il programma ping.

ping permette di inviare una richiesta di eco a un indirizzo determinato, ovvero, a un'interfaccia determinata. Si riesce a ottenere l'eco solo se l'instradamento verso quell'indirizzo e` funzionante e, nello stesso modo, se e` attivo quello di ritorno gestito a partire dall'indirizzo di destinazione.



```
`- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - '
```

Normalmente si procede controllando prima l'indirizzo della propria interfaccia locale, quindi, via via si tenta di raggiungere indirizzi piu` lontani.

$ ping

ping [opzioni] indirizzo

ping permette di inviare una richiesta di eco a un indirizzo, utilizzando il protocollo ICMP, verificando di ricevere tale eco in modo corretto. ping viene usato quasi sempre senza opzioni, in modo da ottenere una richiesta di eco continuo, a intervalli di un secondo, che puo` essere interrotta attraverso la tastiera con la combinazione [Ctrl+c]. Tuttavia, dal momento che ping serve a scoprire dei problemi negli instradamenti e nel sistema di trasporto generale, puo` essere conveniente intervenire sulla dimensione dei pacchetti trasmessi e sul loro contenuto.

Alcune opzioni

-c quantita`

Conclude il funzionamento di ping dopo aver ricevuto il numero indicato di risposte.

-f

Invia la maggior quantita` possibile di pacchetti di richiesta, limitandosi a segnalare graficamente la quantita` di quelli che risultano persi, cioe` per i quali non si ottiene l'eco di risposta. Serve per analizzare pesantemente un tratto di rete, tenendo conto che questa possibilita` va usata con prudenza. Proprio a causa della pericolosita` di tale opzione, questa puo` essere richiesta solo dall'utente root.

-i n_secondi_pausa

Permette di stabilire una pausa, espressa in secondi, tra l'invio di una richiesta di eco e la successiva. Se non viene utilizzata l'opzione -f, il valore predefinito di questa e` di un secondo.

-p stringa_di_riempimento

Permette di aggiungere un massimo di 16 byte ai pacchetti utilizzati da ping, specificandone il contenuto in esadecimale. Cio` puo` essere utile per verificare il passaggio di pacchetti che hanno contenuti particolari e che per qualche ragione possono avere delle difficolta`.

-s dimensione

Permette di definire la dimensione dei pacchetti utilizzati, a cui si aggiunge l'intestazione ICMP. Il valore predefinito e` di 56 byte a cui si aggiungono 8 byte di

intestazione (64 in tutto).
Esempi

$ ping 192.168.1.1

Invia una richiesta di eco all'indirizzo 192.168.1.1, a intervalli regolari di un secondo, fino a che riceve un segnale di interruzione.

$ ping -c 1 192.168.1.1

Invia una richiesta di eco all'indirizzo 192.168.1.1 e termina di funzionare quando riceve la prima risposta di eco.

$ ping -p ffffffff 192.168.1.1

Invia una richiesta di eco all'indirizzo 192.168.1.1, utilizzando pacchetti contenenti una serie di 32 bit a uno (FFFFFFFF16).

$ ping -s 30000 192.168.1.1

Invia una richiesta di eco all'indirizzo 192.168.1.1, utilizzando pacchetti lunghi 30 000 byte, oltre all'intestazione ICMP.

ARP

Nel capitolo introduttivo alle reti TCP/IP, si e` accennato al protocollo ARP, con il quale si ottengono le corrispondenze tra indirizzi di livello 2 (collegamento dati) e indirizzi di livello 3 (rete), ovvero IP nel nostro caso. In particolare e` stato fatto riferimento a una tabella ARP che viene mantenuta automaticamente da ogni nodo durante il suo funzionamento.

Potrebbe essere interessante ispezionare ed eventualmente modificare il contenuto di questa tabella ARP. Questo si fa con il programma arp. (7)

Ci sono situazioni in cui il protocollo ARP non puo` funzionare e in quei casi e` possibile predisporre una tabella ARP preconfezionata attraverso la configurazione di un file: /etc/ethers.

 # arp

arp opzioni

arp permette di ispezionare e di modificare la tabella ARP del sistema.

Alcune opzioni

-n | --numeric

Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi.

-a [host] | --display [host]

Mostra le voci corrispondenti a un nodo particolare, oppure tutti gli abbinamenti conosciuti.

-d host | --delete host

Elimina le voci riferite al nodo indicato.

-s host indirizzo_fisico

Crea una voce nella tabella ARP, abbinando l'indirizzo di un nodo a un indirizzo fisico (generalmente si tratta di un indirizzo Ethernet).

-f file | --file file

Indica un file da utilizzare per caricare delle voci nella tabella ARP. Generalmente, quando le interfacce sono di tipo Ethernet, questo file e` rappresentato da /etc/ethers. Esempi

arp -a

Elenca tutte le voci accumulate nella tabella ARP.

arp -a 192.168.1.2

Mostra le voci riferite esclusivamente al nodo 192.168.1.2.

arp -n -a 192.168.1.2

Come nell'esempio precedente, mostrando solo indirizzi numerici.

arp -d 192.168.1.2

Cancella le voci riferite al nodo 192.168.1.2 contenute nella tabella ARP.

arp -s 192.168.1.2 00:01:02:03:04:05

Assegna permanentemente (per la durata del funzionamento del sistema) l'indirizzo Ethernet 00:01:02:03:04:05 all'indirizzo IP 192.168.1.2.

arp -f /etc/ethers

Legge il file /etc/ethers e utilizza il contenuto per definire delle voci permanenti nella tabella ARP.

cap 14) Indirizzi e nomi

La gestione diretta degli indirizzi IP in forma numerica puo` essere utile in fase di progetto di una rete, ma a livello di utente e` una pretesa praticamente inaccettabile. Per questo, agli indirizzi IP numerici si affiancano quasi sempre dei nomi che teoricamente potrebbero anche essere puramente fantastici e senza alcuna logica. Ogni volta che si fa riferimento a un nome, il sistema e` (o dovrebbe essere) in grado di convertirlo nel numero IP corrispondente.

In pratica, si usa di solito la convenzione dei nomi di dominio, come gia` descritto in precedenza.

Ci sono due metodi per trasformare un nome in un indirizzo IP e viceversa: un elenco contenuto nel file /etc/hosts oppure l'uso di un servente DNS.

In questo capitolo si analizza /etc/hosts e gli altri file di configurazione legati alla traduzione dei nomi; nel prossimo verra` trattata la gestione di un servente DNS con il quale si ottiene un servizio di risoluzione dei nomi (name server).

Configurazione del tipo di conversione

Prima di procedere con la trasformazione di un nome in un indirizzo IP, occorre definire in che modo si vuole che il sistema esegua questa operazione. Il file di configurazione attraverso il quale si definisce cio` e` /etc/host.conf, ma anche attraverso l'uso di variabili di ambiente si puo` intervenire in questa configurazione.

/etc/host.conf

Viene usato per determinare quali servizi usare per risolvere i nomi di dominio. Ogni riga

rappresenta un'opzione di funzionamento, inoltre il simbolo # rappresenta l'inizio di un commento. Solitamente vengono specificate solo due direttive: order e multi, come nell'esempio seguente:

```
order hosts,bind
multi on
```

Nella prima riga, l'opzione order indica l'ordine dei servizi. In questo caso si utilizza prima il file /etc/hosts e quindi si interpella il servizio di risoluzione dei nomi. Nella seconda riga, multi on, abilita la possibilita` di trovare all'interno del file /etc/hosts l'indicazione di piu` indirizzi IP per lo stesso nome. Un evento del genere puo` verificarsi quando uno stesso elaboratore ha due o piu` connessioni per la rete e per ognuna di queste ha un indirizzo IP diverso.

```
order {hosts|bind|nis}[,...[,...]]
```

L'opzione order richiede uno o piu` argomenti (separati da spazio, virgola, punto e virgola o due punti) indicanti la sequenza di servizi attraverso cui si deve tentare di risolvere un nome.

```
multi {on|off}
```

L'opzione multi attiva o disattiva la possibilita` di trovare all'interno del file /etc/hosts l'indicazione di piu` indirizzi IP per lo stesso nome.

Variabili di ambiente

Attraverso l'uso di variabili di ambiente e` possibile interferire con la configurazione del file /etc/host.conf.

RESOLV_HOST_CONF
Se esiste e non e` vuota, definisce il nome di un file alternativo a /etc/host.conf.

RESOLV_SERV_ORDER
Definisce l'ordine dei servizi di risoluzione dei nomi, senza tenere conto di quanto eventualmente gia` definito attraverso l'opzione order nel file /etc/host.conf.

RESOLV_SERV_MULTI
Puo` contenere la stringa on oppure off, con lo stesso significato dell'opzione multi del file /etc/host.conf e serve a sostituirsi all'eventuale dichiarazione fatta nel file stesso.

File per la conversione

Prima che esistessero i serventi DNS si dovevano risolvere i nomi attraverso l'uso di un file unico, contenente un elenco di indirizzi IP associato ai nomi rispettivi. Teoricamente, utilizzando un servente DNS questo file potrebbe non essere piu` necessario. In pratica conviene utilizzare ugualmente questo vecchio metodo per garantirsi l'accessibilita` alla rete locale anche quando l'eventuale servente DNS non dovesse funzionare.

/etc/hosts

Il file /etc/hosts viene usato per convertire i nomi degli elaboratori in numeri IP e viceversa. E` particolarmente utile la sua compilazione all'interno di piccole reti che non dispongono di un servente DNS. All'interno di una rete locale puo` essere predisposto uguale per tutti gli elaboratori connessi, cosi` da facilitare per quanto possibile l'aggiornamento all'interno di questi. Segue un estratto di esempio di questo file.

```
# necessario per il loopback IPv4
127.0.0.1                  localhost.localdomain   localhost

# indirizzi IPv4
192.168.1.1                dinkel.brot.dg         dinkel
192.168.1.2                roggen.brot.dg          roggen

192.168.2.1                weizen.mehl.dg          weizen

#necessario per il loopback IPv6
::1                        ip6-localhost          ip6-loopback

# necessari per il multicast IPv6
fe00::0                    ip6-localnet
ff00::0                    ip6-mcastprefix
ff02::1                    ip6-allnodes
ff02::2                    ip6-allrouters
ff02::3                    ip6-allhosts

# indirizzi IPv6
fec0::1:2a0:24ff:fe77:4997     dinkel.brot.dg         dinkel
fec0::1:280:5fff:fea6:6d3d     roggen.brot.dg         roggen

fec0::2:280:adff:fec8:a981     weizen.mehl.dg          weizen
```

In pratica, il file puo` contenere righe vuote o commenti (le righe che iniziano con il simbolo #) e righe che iniziano con un indirizzo IP (sia IPv4 che IPv6). Dopo l'indirizzo IP, separato da spazi o caratteri di tabulazione, inizia l'elenco dei nomi a esso abbinati, anche questo puo` essere separato da spazi o da caratteri di tabulazione.

Di solito, si indica il nome di dominio completo (FQDN o Fully qualified domain name), seguito eventualmente da possibili abbreviazioni o soprannomi.

Poco sopra era stata accennata la possibilita` di creare un file identico /etc/hosts per tutti gli elaboratori della propria rete locale. Ma se la rete locale si articola in sottoreti, e` normale che il dominio di appartenenza di ogni sottorete cambi. Nell'esempio visto, si fa riferimento a due sottoreti IPv4 e IPv6: 192.168.1.0 e fec0::1::/64 denominata brot.dg; 192.168.2.0 e fec0::2::/64 denominata mehl.dg. In questa situazione, potrebbe capitare che un elaboratore nella rete mehl.dg abbia lo stesso nome locale di un altro collocato

nelle rete brot.dg.

Per questo, l'attribuzione di soprannomi, o semplicemente di abbreviazioni, deve essere limitata alla sottorete di appartenenza, oppure deve essere evitata. A questo fa eccezione il caso dell'indirizzo di loopback: ogni elaboratore e` bene che si chiami localhost.

Se si decide di fare il lavoro in serie, l'esempio visto sopra deve essere trasformato in quello seguente:

```
# necessario per il loopback IPv4
127.0.0.1                  localhost.localdomain   localhost

# indirizzi IPv4
192.168.1.1                dinkel.brot.dg
192.168.1.2                roggen.brot.dg

192.168.2.1                weizen.mehl.dg

#necessario per il loopback IPv6
::1                        ip6-localhost          ip6-loopback

# necessari per il multicast IPv6
fe00::0                    ip6-localnet
ff00::0                    ip6-mcastprefix
ff02::1                    ip6-allnodes
ff02::2                    ip6-allrouters
ff02::3                    ip6-allhosts

# indirizzi IPv6
fec0::1:2a0:24ff:fe77:4997     dinkel.brot.dg
fec0::1:280:5fff:fea6:6d3d     roggen.brot.dg

fec0::2:280:adff:fec8:a981     weizen.mehl.dg
```

/etc/resolv.conf

Quando il file /etc/hosts non basta, si deve poter accedere a un servizio di risoluzione dei nomi, ovvero a un servente DNS. Viene usato il file /etc/resolv.conf per conoscere l'indirizzo o gli indirizzi dei servizi di risoluzione dei nomi di competenza della rete cui si appartiene. Se non si intende utilizzare il sistema DNS per risolvere i nomi della propria rete, oppure si dispone di un solo elaboratore, ma si vuole accedere alla rete Internet, dovranno essere indicati gli indirizzi dei servizi di risoluzione dei nomi forniti dall'ISP (Internet service provider), ovvero dal fornitore di accesso a Internet.

Questo file puo` contenere righe vuote o commenti (le righe che iniziano con il simbolo #) e righe che iniziano con un nome di opzione seguite normalmente da un argomento. Le opzioni utilizzabili sono descritte qui di seguito.

nameserver indirizzo_ip_servente_DNS

L'opzione nameserver e` la piu` importante e permette di definire l'indirizzo IP di un servizio di risoluzione dei nomi. Se questa opzione non viene utilizzata, si fa riferimento a un servizio locale, raggiungibile precisamente all'indirizzo 127.0.0.1. Il file /etc/resolv.conf puo` contenere piu` righe con questa opzione, in modo da poter fare riferimento a servizi di risoluzione dei nomi alternativi quando quello principale non risponde.

domain nome_di_dominio

Stabilisce il dominio predefinito per le interrogazioni del servizio di risoluzione dei nomi.

search nome_di_dominio...

Definisce un elenco di domini possibili (l'elenco e` separato da spazi o caratteri di tabulazione) per le interrogazioni del servizio di risoluzione dei nomi.

Una configurazione normale non ha bisogno dell'indicazione delle opzioni domain e search. Se il file /etc/resolv.conf si limita a contenere opzioni nameserver, questo puo` essere standardizzato su tutta la rete locale.

Segue un esempio in cui si utilizza il servizio di risoluzione dei nomi offerto dall'indirizzo IP 192.168.1.1 ed eventualmente, in sua mancanza, dall'indirizzo 192.168.2.15.

```
nameserver 192.168.1.1
nameserver 192.168.2.15
```

cap 15) Servente HTTP: Apache

Apache e` un servente HTTP derivato da quello di NCSA, che costituisce lo standard di fatto per i sistemi GNU e molte altre piattaforme.

Le funzionalita` che Apache mette a disposizione sono molte e di conseguenza la sua configurazione puo` anche essere complicata. Eventualmente, quando non ci sono esigenze particolari, si puo` preferire l'installazione di un servente HTTP meno sofisticato, come Boa, descritto nel capitolo 161.

Visione generale

Apache e` costituito essenzialmente dall'eseguibile httpd, che si avvia di solito come demone autonomo dal supervisore dei servizi di rete:

httpd [opzioni]

Nelle sezioni seguenti si fa sempre riferimento a un'installazione in cui il servizio viene avviato in modo indipendente dal supervisore dei servizi di rete. Eventualmente si puo` consultare la documentazione originale per un'impostazione differente.

Opzione Descrizione

-d directory_radice_del_servente
Permette di definire la directory che funge come punto di partenza per il servizio che viene offerto. Questa e` gia` stabilita in modo predefinito in fase di compilazione del programma e cio` dipende dalla scelta di chi ha compiuto questa operazione. Attraverso questa opzione, si puo` indicare in modo esplicito una posizione diversa, che pero` puo` essere scavalcata dalla direttiva ServerRoot del file di configurazione httpd.conf.

-f file_di_configurazione
Permette di indicare in modo esplicito il file di configurazione che httpd deve leggere ed eseguire prima di iniziare a gestire il suo servizio. Se il file viene indicato utilizzando un percorso relativo, se cioe` manca la prima barra obliqua che identifica la directory radice, si fa riferimento a una posizione relativa che parte dalla directory ServerRoot, ovvero quella definibile con l'opzione -d.
Il valore predefinito di questa opzione, dipende dal modo in cui e` stato compilato il programma. In un sistema GNU dovrebbe trattarsi di /etc/apache/httpd.conf.

Di solito, non occorre configurare nulla per vedere funzionare il servente in modo ?normale?, per la pubblicazione di qualche pagina senza esigenze particolari, ma la gestione di un sito vero e proprio richiede quasi sempre un intervento nella configurazione.

Purtroppo, Apache gestisce piu` di un file configurazione e questo puo` creare un po' di confusione. In generale, questi file potrebbero trovarsi nella directory /etc/apache/ e si tratta almeno di: httpd.conf, srm.conf e access.conf.

Prima di vedere i dettagli dell'impostazione del servente Apache, e` il caso di descrivere alcune caratteristiche che lo riguardano.

1. L'accesso al servizio HTTP avviene a partire da una parte del file system, che inizia dal cosiddetto document root.
2. Il programma servente non esegue la funzione chroot() in questa directory, pertanto e` possibile articolare le directory successive anche attraverso l'uso di collegamenti simbolici in posizioni precedenti alla directory document root.
3. In linea di massima, ogni utente puo` realizzare una struttura personalizzata di documenti HTML, a partire dalla propria directory personale (home).
4. Il servente e` in grado di mettere in funzione dei programmi, detti CGI, per la gestione interattiva di pagine HTML contenenti dei moduli.

Struttura di directory

Nella configurazione di Apache si distinguono due directory che vengono definite attraverso un nome particolare; si tratta di ServerRoot e DocumentRoot. A queste se ne affiancano altre che derivano dalla configurazione consueta di questo programma servente.

server root
La directory nota come server root e` il punto di origine dei file amministrativi di Apache. Viene dichiarata nel file httpd.conf e gli altri file dichiarati all'interno di questo sono intesi essere collocati in posizione relativa a tale directory.

document root
La directory nota come document root e` il punto di origine dei documenti HTML.

Programmi CGI
Convenzionalmente, e` opportuno collocare i programmi CGI in una posizione estranea alla gerarchia che si articola a partire dalla directory document root, per facilitare

la configurazione della sua accessibilita`.

Icone di sistema

Il servente HTTP ha spesso la necessita` di utilizzare icone per rappresentare delle informazioni in modo grafico, per esempio quando si visualizza il contenuto di una directory appartenente alla gerarchia di document root. Sotto questo aspetto, e` conveniente togliere tali icone dalla struttura dei documenti normali, perche' non fanno parte di questi.

Documenti personali

In linea di massima e` concesso agli utenti di creare una propria struttura di documenti ipertestuali. La directory di partenza di questi documenti viene definita come user dir ed e` relativa alla directory personale di questi utenti. E` importante tenere presente che gli utenti hanno tale possibilita`, per configurare opportunamente il servente in modo che questi non possano creare danni.

Avvio e conclusione dell'attivita` del servente

Come gia` descritto, il servizio viene gestito dal demone httpd che puo` essere avviato direttamente dalla procedura di inizializzazione del sistema, oppure puo` essere controllato dal supervisore dei servizi di rete. In questo secondo caso, quando si fanno delle modifiche alla configurazione, non occorre fare in modo che httpd le rilegga, perche' e` costretto a farlo ogni volta che viene risvegliato dal supervisore dei servizi di rete.

Quando httpd e` indipendente dal supervisore dei servizi di rete (standalone), si puo` osservare la presenza di una serie di processi httpd discendenti da uno di origine.

```
# pstree -p[Invio]

init(1)-+-...
        |
        |-httpd(244)-+-httpd(859)
        |            |-httpd(860)
        |            |-httpd(861)
        |            |-httpd(862)
        |            `-httpd(863)
        |-...
        ...
```

Per fare in modo che tutti questi processi rileggano i file di configurazione, basta inviare un segnale SIGHUP a quello principale; in questo caso il numero 244.

```
# kill -HUP 244[Invio]

# pstree -p[Invio]
```

```
init(1)-+-...
        |
        |-httpd(244)-+-httpd(901)
        |            |-httpd(902)
        |            |-httpd(903)
        |            |-httpd(904)
        |            `-httpd(905)
        |-...
        ...
```

Come si puo` osservare, il processo httpd principale rimane attivo, mentre quelli inferiori vengono conclusi e riavviati (lo si vede dal numero PID). Attenzione pero`: se si invia un segnale di questo tipo al processo httpd dopo aver modificato la configurazione in modo errato, questo termina il suo funzionamento.

Configurazione essenziale con httpd.conf

httpd.conf e` il file di configurazione principale di Apache. La sua collocazione dipende dal modo in cui e` stato compilato Apache, oppure dall'opzione -f della riga di comando del demone httpd. Nelle sezioni seguenti vengono descritte solo alcune direttive piu` importanti. Inoltre, nel capitolo 175 viene trattata la configurazione di Apache per la gestione di una cache proxy, cosa che riguarda in modo particolare proprio questo file.

Impostazioni varie

Alcune direttive sono importanti per definire se il demone httpd funziona in modo autonomo o meno; inoltre, nel primo caso, per sapere su quale porta deve restare in ascolto.

Tipo di gestione del demone

```
ServerType { standalone | inetd }
```

La direttiva ServerType permette di informare Apache su come questo viene avviato: in modo autonomo o attraverso il supervisore dei servizi di rete.

ServerType standalone

Nell'esempio si mostra la dichiarazione per il funzionamento autonomo (standalone) che corrisponde alla situazione piu` comune (e anche piu` adatta).
Porta del servizio

```
Port numero_porta
```

Si tratta dell'indicazione della porta (di solito e` 80, corrispondente alla denominazione http), necessaria nel caso in cui il demone sia stato avviato in modo autonomo. Infatti, diversamente, e` il supervisore dei servizi di rete a stare in ascolto della porta corrispondente al servizio http.

Port 80

Listen numero_porta

Se httpd viene utilizzato in modo autonomo, e` possibile richiedere che stia in ascolto anche di un'altra porta, per mezzo della direttiva Listen.

Listen 80
Listen 8080

L'esempio mostra in che modo si possa indicare a httpd di stare in ascolto sia della porta 80 che della 8 080; dove la seconda viene utilizzata normalmente per interrogare un servente proxy.
Indirizzi numerici o nomi di dominio

HostnameLookups { on | off }

Permette di decidere se si intende annotare nei file delle registrazioni l'indirizzo numerico o il nome dei nodi che accedono al servizio. Attivando questa direttiva (on) si registrano i nomi corrispondenti. L'attivazione di questa e` necessaria se si intendono definire dei limiti di accesso basati sul nome di dominio dei clienti, come si vede nell'esempio seguente:

HostnameLookups on

Identificazione

Spesso, un nodo che offre un servizio HTTP puo` essere identificato attraverso degli alias al nome di dominio canonico. Nella configurazione e` opportuno definire un nome corretto, che puo` corrispondere anche a un alias, purche' sia valido. Nello stesso modo, e` importante definire l'indirizzo di posta elettronica presso cui puo` essere raggiunto l'amministratore del servizio (webmaster).

Webmaster

ServerAdmin email

La direttiva ServerAdmin permette di definire l'indirizzo di posta elettronica dell'amministratore del servizio. Generalmente si trattera` dell'utente fittizio webmaster che dovrebbe essere ridiretto automaticamente all'utente root dal sistema di gestione della posta elettronica.

ServerAdmin webmaster@dinkel.brot.dg

L'utilita` di utilizzare un indirizzo di posta elettronica specifico, sta nella facilita` con cui poi si intende il contesto a cui fanno riferimento questi messaggi.
Nome ufficiale del servente

ServerName nome_standard_del_servente

Attraverso questa direttiva si puo` dichiarare espressamente il nome di dominio del servente HTTP. Puo` trattarsi alias di un alias definito nel sistema DNS, ma quello che conta e` che si tratti di un nome valido.

ServerName www.brot.dg

L'esempio dichiara che il nome del nodo che offre il servizio e` www.brot.dg, anche se magari il nome canonico di questo, secondo il DNS, e` diverso. Quello che conta e` che il sistema DNS sia in grado di risolvere anche questo nome qui dichiarato.

Utenti

L'utilizzo di un servizio HTTP e` qualcosa di prettamente anonimo, in quanto la natura di questo, per cui tutto si traduce in semplici richieste seguite da risposte, impedisce una gestione sensata di identificativi utente e delle parole d'ordine relative.

Utente e gruppo per l'accesso al servizio

User { utente | #n }

Group { gruppo | #n }

Queste due direttive permettono di definire l'utente e il gruppo fittizio da abbinare agli accessi fatti al servizio. In pratica, quando si legge un file HTML o si interpella un programma CGI, lo si fa come se si fosse l'utente indicato da queste due direttive. Solitamente, per motivi di sicurezza, si utilizza l'utente e il gruppo nobody, oppure un utente e un gruppo specifici per il servizio HTTP.

Se per qualche motivo si preferisce una notazione numerica, invece di indicare il nome dell'utente e del gruppo si puo` usare il numero UID e GID, preceduto dal simbolo #.

User nobody
Group nobody

Perche' queste direttive possano funzionare, occorre che il demone httpd sia avviato con i privilegi dell'utente root, altrimenti non ha modo di eseguire il cambiamento di utente e gruppo, potendo solo continuare a funzionare con i privilegi ottenuti all'avvio.

Collocazione e denominazione di file e directory

Il file httpd.conf contiene l'indicazione della directory server root, della posizione dei file

delle registrazioni ed eventualmente anche degli altri file di configurazione.

ServerRoot

ServerRoot directory

Rappresenta la directory a partire dalla quale si diramano le informazioni sulla configurazione, sulla registrazione degli eventi e simili. Corrisponde solitamente a qualcosa come /etc/httpd/conf/ o /etc/apache/. Potrebbe essere definita anche attraverso l'opzione -d della riga di comando di httpd.

ServerRoot /etc/apache

L'esempio mostra la posizione piu` conveniente di questa directory per aderire allo standard FHS sulla struttura del file system.
Altri file di configurazione

ResourceConfig config_srm

AccessConfig config_access

Le direttive mostrate servono per definire rispettivamente il nome e la collocazione del file di configurazione delle risorse e del file di configurazione degli accessi. Generalmente, i nomi e la collocazione di questi file non devono essere dichiarate espressamente, perche' e` sufficiente quanto risulta predefinito all'interno del programma stesso.

```
ResourceConfig  conf/srm.conf
AccessConfig    conf/access.conf
```

L'esempio mostra la dichiarazione esplicita dei nomi utilizzati per gli altri file di configurazione. Mancando l'indicazione di un percorso assoluto, si intende che debbano essere discendenti della directory server root.
File delle registrazioni

ErrorLog registro_degli_errori

TransferLog registro_dei_trasferimenti

Queste direttive definiscono i nomi e la collocazione dei file delle registrazioni. Generalmente i percorsi indicati sono relativi, in tal caso si riferiscono alla directory server root come punto iniziale.

```
ErrorLog        /var/log/apache/error_log
TransferLog     /var/log/apache/access_log
```

L'esempio mostra la dichiarazione dei due file delle registrazioni, con un percorso assoluto: /var/log/apache/.

Processo del demone principale

PidFile file_pid

Definisce il nome e la collocazione del file utilizzato per contenere il numero di processo del demone httpd principale, quando questo funziona in modo autonomo.

PidFile /var/run/httpd.pid

L'esempio mostra l'indicazione del file /var/run/httpd.pid, con un percorso assoluto, in modo da non finire al di sotto della directory server root.
Comunicazione interna

ScoreBoardFile file_di_informazioni

Definisce il nome e la collocazione di un file contenente una serie di informazioni sul funzionamento corrente del programma servente, necessarie al servente stesso per la comunicazione tra processi.

ScoreBoardFile /var/run/apache_status

L'esempio mostra l'indicazione del file /var/run/apache_status, con un percorso assoluto, in modo da non finire al di sotto della directory server root.

Configurazione delle risorse con srm.conf

Il file srm.conf e` il file di configurazione delle risorse di Apache. Viene letto subito dopo quello di configurazione del servente. Definisce in particolare dove si trovino i documenti (la directory document root e quella delle pagine degli utenti), gli alias di directory speciali e altre informazioni correlate. La sua collocazione dipende dal modo in cui e` stato compilato Apache, oppure dalla direttiva ResourceConfig del file httpd.conf.

Nelle edizioni piu` recenti di Apache, le direttive del file srm.conf possono risiedere direttamente nel file httpd.conf.

Nelle sezioni seguenti vengono descritte solo alcune direttive piu` importanti.

Documenti HTML

La funzione principale di srm.conf e` quella di definire la collocazione dei documenti ipertestuali, oltre ad altre informazioni di contorno.

DocumentRoot

DocumentRoot directory_iniziale_documenti_html

La direttiva DocumentRoot dichiara la directory da cui si possono diramare i documenti HTML (per qualche motivo oscuro, e` importante che non abbia la barra obliqua finale).

DocumentRoot /home/httpd/html

L'esempio mostra il caso in cui la directory /home/httpd/html/ corrisponda all'inizio dei documenti HTML.
Pagine personali degli utenti

UserDir { directory_iniziale_documenti_utenti | DISABLED [utente] }

La direttiva UserDir dichiara la directory, relativamente alla posizione della directory personale di ogni utente, all'interno della quale ognuno puo` collocare i propri documenti HTML personali. Si accede a questi utilizzando l'URI http://host/~utente.

UserDir public_html

L'esempio mostra la dichiarazione tipica di questa direttiva e significa che ogni utente puo` creare la directory ~/public_html/ all'interno della quale collocare le proprie pagine.

Supponendo di accedere all'URI http://www.brot.dg/~tizio/elenco.html si fa riferimento effettivamente al file ~tizio/public_html/elenco.html. In questo modo, tra le altre cose, si evita di esporre l'intera directory personale dell'utente.

UserDir DISABLED

L'esempio mostra in che modo possa essere impedito ai singoli utenti di creare le proprie pagine HTML nella loro directory personale.

Quando si concede agli utenti di realizzare le loro pagine HTML personali, occorre tenere presente che questo fatto puo` costituire un problema di sicurezza del sistema: un utente potrebbe creare un semplice collegamento simbolico verso un file o una directory che, pur risultando leggibile a tutti gli utenti, non avrebbe dovuto essere accessibile al mondo intero. A questo si puo` porre rimedio, ma per farlo occorre intervenire sul file access.conf, come verra` mostrato in seguito.

UserDir DISABLED root

A partire dalla versione 1.3 di Apache e` possibile specificare a quali utenti vietare la costruzione di pagine HTML personali, come nell'esempio mostrato, in cui questo viene impedito all'utente root.

Indici e file di informazioni

Quando si tenta di accedere a una directory, invece che a un file particolare, si ottiene

l'indice del contenuto, come se si trattasse del protocollo FTP, oppure il contenuto di una pagina predefinita.

File indice

DirectoryIndex file_indice...

Quando si accede a una directory invece che a un file specifico, se questa contiene un file tra quelli elencati nella direttiva DirectoryIndex viene restituito quel file, invece del semplice elenco del contenuto. Solitamente si utilizza il nome index.html. Questo meccanismo permette di mascherare il contenuto effettivo della directory, oltre che di guidare l'utente del servizio in modo che non si perda in una miriade di file.

DirectoryIndex index.html index.htm

L'esempio dichiara due file (index.html e index.htm) come possibili indici da utilizzare quando si fa riferimento a una directory senza indicare un file specifico.
Indice grafico

FancyIndexing { on | off }

La direttiva FancyIndexing permette di definire se, quando viene restituito l'elenco del contenuto di una directory, si vuole una rappresentazione a icone, oppure se si vuole un testo puro e semplice. La parola chiave on attiva la visualizzazione a icone; off la disabilita.

FancyIndexing on

Definizione delle icone

AddIconByEncoding (sigla,file_icona) tipo_codifica...

Questa direttiva abbina un'icona a uno o piu` tipi di codifica. La sigla rappresenta una stringa da utilizzare al posto dell'icona quando non e` possibile la sua rappresentazione (per esempio se si usa il navigatore Lynx).

AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

AddIconByType (sigla,file_icona) tipo_mime/sottotipo

Questa direttiva abbina un'icona a un tipo e sottotipo MIME, eventualmente utilizzando l'asterisco nel sottotipo per includerli tutti. La sigla rappresenta una stringa da utilizzare al posto dell'icona quando non e` possibile la sua rappresentazione.

AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*

AddIcon file_icona estensione...

Questa direttiva abbina un'icona a una o piu` estensioni del nome dei file.

```
AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^
```

DefaultIcon file_icona

Questa direttiva permette di definire un'icona predefinita per i file che non rientrano in una classificazione diversa.

```
DefaultIcon /icons/unknown.gif
```

File da ignorare

IndexIgnore modello_da_ignorare...

Quando si consente di accedere a una directory visualizzandone il contenuto (perche' manca il file index.html o equivalente), si puo` fare in modo che alcuni file non appaiano in elenco. Utilizzando questa direttiva, si possono indicare i modelli di file da non includere. Per questo si possono usare i caratteri jolly consueti (punto interrogativo e asterisco).

```
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS
```

L'esempio mostra l'esclusione dall'elenco di:

- tutti i file che iniziano con un punto e sono lunghi almeno tre caratteri, perche' si vuole continuare a includere il riferimento alla directory precedente;
- tutti i file che terminano con il simbolo tilde, che sono solitamente delle copie di sicurezza di versioni precedenti;
- tutti i file che terminano con il simbolo #, dal momento che anche questi sono generalmente copie di sicurezza di versioni precedenti;
- tutti i file il cui nome inizia per HEADER o README, perche' hanno un ruolo speciale;
- il file RCS.

File ?readme?

HeaderName file_readme_iniziale

ReadmeName file_readme_finale

Attraverso queste due direttive si possono specificare i nomi di file, il cui contenuto si vuole sia incluso nell'elenco della directory. Per la precisione, la direttiva HeaderName specifica il nome di un file da mettere prima dell'elenco; la direttiva ReadmeName specifica il nome di un file da mettere dopo l'elenco. L'esempio permette di chiarire altri dettagli.

HeaderName HEADER
ReadmeName README

In questo caso, viene cercato prima il file HEADER.html. Se viene trovato, viene incluso all'inizio dell'elenco della directory, mantenendo la formattazione HTML. Se manca, ma esiste il file HEADER, questo viene incluso in modo testuale. La stessa cosa vale per il file README.html o soltanto README, con la differenza che questo viene incluso alla fine, dopo l'elenco.

Tipi di file

Il servente ha bisogno di conoscere il tipo di file che si preleva per sapere come comportarsi, ma soprattutto per poterlo comunicare al cliente che lo ha richiesto. Questo si ottiene attraverso la configurazione dei tipi MIME, ma e` pur sempre necessario specificare il tipo predefinito, quando non si riesce a determinarlo altrimenti.

Tipo predefinito

DefaultType tipo_e_sottotipo_mime...

Permette di definire il tipo MIME predefinito di un documento per il quale non si riesca a identificare diversamente. Di solito questo valore predefinito e` text/plain.

DefaultType text/plain

Aggiunta di tipi nuovi

AddType tipo_mime/sottotipo estensione

Con questa direttiva si possono aggiungere dei tipi MIME senza intervenire nel file di definizione di questi, mime.types. Generalmente non e` conveniente intervenire in questo modo; e` sempre meglio utilizzare il file dei tipi MIME.
Codifica

AddEncoding tipo_di_compressione estensione

Questa direttiva permette di abbinare un'estensione a un tipo di codifica. Cio` permette ad alcuni programmi cliente di sapere come gestire tali dati.

AddEncoding x-compress Z
AddEncoding x-gzip gz

L'esempio mostra la configurazione tipica, che serve a informare i programmi cliente quando viene inviato loro un file compresso con compress o con gzip.

Directory alias

Per evitare confusione, oltre che per motivi di sicurezza, e` opportuno dichiarare alcune directory speciali in forma di alias.

Alias normale

Alias directory_fasulla directory_reale

Questo tipo di direttiva, che puo` essere ripetuta, permette di definire delle directory in posizioni diverse da quelle reali. La directory fasulla fa riferimento a una directory indicata nell'indirizzo URI richiesto, mentre quella reale indica la directory effettiva nel file system.

Alias /icons/ /home/httpd/icons/

L'esempio mostra la dichiarazione di una directory cui si accede attraverso l'alias /icons/. In pratica, tutte le volte che viene richiesta una risorsa contenuta nella directory /icons/, questa verra` prelevata dalla directory reale /home/httpd/icons/.

La dichiarazione dell'alias /icons/ e` molto importante nella consuetudine, dal momento che si tratta del riferimento alla directory contenente le icone utilizzare per la visualizzazione degli indici. Si e` visto in un'altra sezione la dichiarazione dell'abbinamento delle icone a seconda dell'estensione dei file, come nell'esempio seguente, dove si fa riferimento a questo alias.

AddIcon /icons/binary.gif .bin .exe

Alias per i programmi CGI

ScriptAlias directory_fasulla directory_reale

Funziona come la direttiva Alias, ma si riferisce ai programmi CGI. Generalmente, i programmi CGI dovrebbero essere collocati esclusivamente all'interno di directory dichiarate attraverso questa direttiva, per non rischiare di creare problemi di sicurezza del sistema.

ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/

Gestori specifici in base all'estensione

E` possibile stabilire un comportamento particolare in base all'estensione dei file. Con questo si intende qualcosa di diverso dalla semplice lettura e invio al cliente che ne fa richiesta. La sintassi generale e` la seguente:

AddHandler nome_dell'azione estensione

Il nome dell'azione definisce un tipo preciso di operazione da abbinare ai file che contengono l'estensione indicata.

Esecuzione di programmi CGI

AddHandler cgi-script estensione

Questa direttiva, usata cosi`, permette di abbinare a un'estensione l'esecuzione automatica come programma CGI. E` decisamente sconsigliabile di permettere l'utilizzo di programmi CGI al di fuori della directory dichiarata con la direttiva ScriptAlias. L'esempio seguente mostra questa direttiva commentata opportunamente per sicurezza.

AddHandler cgi-script .cgi

Configurazione di accesso della directory

E` possibile definire il nome di un file di configurazione che, se presente, serve per definire l'accesso alla directory in cui si trova. Il nome predefinito di questo e` .htaccess. Per questo si utilizza la direttiva AccessFileName, come nell'esempio seguente:

AccessFileName .htaccess

File di messaggi

In occasione di determinate situazioni errore, il programma servente emette delle segnalazioni di errore. Questi messaggi possono essere riscritti in forma di file HTML o di programma CGI. La direttiva per controllare questi messaggi ha tre sintassi possibili.

ErrorDocument n_errore file_alternativo

ErrorDocument n_errore uri_esterno

ErrorDocument n_errore "messaggio

Nel primo caso, l'ultimo argomento e` un file HTML o un programma CGI; nel secondo si tratta di un URI esterno; nel terzo si tratta di una stringa, che viene identificata come tale perche' inizia con gli apici doppi ("). La stringa non deve essere terminata, a meno di volere fare apparire gli apici doppi finali.

ErrorDocument 500 "Errore del servente www.

L'esempio mostra il caso in cui si voglia fare apparire una stringa particolare in occasione del verificarsi dell'errore 500.

ErrorDocument 404 /documento_mancante.html

In questo caso, in occasione dell'errore 404, viene inviato al cliente il file documento_mancante.html che conterra` qualche utile suggerimento per l'utente.

ErrorDocument 404 /cgi-bin/documento_mancante.pl

Questa e` una variante dell'esempio precedente, in cui, invece di inviare un file HTML viene eseguito un programma CGI, /cgi-bin/documento_mancante.pl. Cio` puo` essere utile per comporre una risposta personalizzata, utilizzando le informazioni cui puo` accedere il programma stesso.

ErrorDocument 404 http://roggen.brot.dg/cgi-bin/documento_mancante.pl

Questa e` una variante dell'esempio precedente, in cui si fa riferimento a una risorsa contenuta in un URI esterno al servente dove si manifesta il problema.

Controllare l'accesso con access.conf

access.conf e` il file di configurazione globale che permette di controllare l'accesso alle directory del sistema. La sua sintassi e` diversa da quella degli altri due file di configurazione gia` visti. In particolare, oltre a normali direttive, si utilizzano dei delimitatori simili a marcatori HTML che permettono di definire il contesto a cui si riferiscono le direttive contenute. Piu` precisamente si parla di sezioni.

Nelle edizioni piu` recenti di Apache, le direttive del file access.conf possono risiedere direttamente nel file httpd.conf.

Sezioni di controllo

Le sezioni del file di configurazione degli accessi hanno una forma simile a quella

seguente:

<Nome ...> ... </Nome>

Nel marcatore che ne dichiara l'apertura possono apparire delle opzioni; nella parte compresa tra l'apertura e la chiusura si inseriscono delle direttive riferite a quella sezione particolare. A seconda del contesto, una sezione puo` contenere anche la dichiarazione di altre sezioni in modo annidato.

Sezione ?Directory?

Le sezioni Directory raccolgono le direttive di controllo per una particolare directory e per quelle successive. La direttiva di apertura, ovvero il marcatore <Directory>, deve contenere l'indicazione della directory a cui si riferiscono le direttive della sezione, eventualmente usando anche i caratteri jolly (* e ?) o le espressioni regolari estese.

```
<Directory /home/httpd/html>
      Options Indexes FollowSymLinks
</Directory>
```

Questo esempio e` il piu` comune: si dichiara una sezione riferita alla directory /home/httpd/html/, che qui si vuole intendere corrispondere alla document root.

```
<Directory "^/home/httpd/html/[0-9]{3}">
      Options Indexes FollowSymLinks
</Directory>
```

Questo esempio ulteriore, attraverso un'espressione regolare, dichiara una sezione riferita a tutte le directory discendenti di /home/httpd/html/ che iniziano con tre cifre numeriche.

Quando una sezione si riferisce a una porzione gia` presa in considerazione da un'altra analoga, conviene che queste appaiano in una sequenza tale da porre prima le sezioni generali e dopo quelle piu` particolareggiate, come nell'esempio seguente:

```
<Directory />
      AllowOverride None
</Directory>
...
<Directory /home/httpd/html>
      AllowOverride All
</Directory>
```

Di seguito sono descritte alcune delle direttive che possono essere usate all'interno della sezione Directory.

Options

Options [+|-]opzione...

La direttiva Options permette di definire alcune opzioni in forma di parole chiave. La tabella
160.2 ne riporta l'elenco. In particolare, le opzioni None e All vanno usate da sole.

Tabella 160.2. Alcune opzioni della direttiva Options nella sezione Directory.
Parola chiave Significato
None Disabilita tutto.
All Abilita tutte le opzioni.
FollowSymLinks Abilita l'uso di collegamenti simbolici.
SymLinksIfOwnerMatch Abilita l'uso di collegamenti simbolici solo se il proprietario coincide.
ExecCGI Permette l'esecuzione dei programmi CGI.
Indexes Permette di ottenere il listato del contenuto.
Includes SSI e` permesso.
IncludesNOEXEC SSI e` permesso in parte.

Generalmente, se piu` direttive Options possono applicarsi alla stessa directory, quella riferita alla directory piu` specifica si sostituisce completamente alle altre. Tuttavia, se tutte le opzioni vengono precedute dal segno + o -, queste vengono unite a quelle gia` dichiarate. Le opzioni precedute dal segno + vengono aggiunte; quelle precedute dal segno - vengono eliminate. In ogni caso, per facilitare la lettura sarebbe opportuno dichiarare ogni volta le opzioni che si vuole siano abilitate.

L'esempio seguente mostra la semplice dichiarazione della directory /home/httpd/html/ (corrispondente a document root), in cui e` consentito visualizzare il listato del contenuto e seguire i collegamenti simbolici.

```
<Directory /home/httpd/html>
      Options Indexes FollowSymLinks
</Directory>
```

AllowOverride

AllowOverride opzione...

La direttiva AllowOverride permette di definire quali opzioni possono essere scavalcate dalle dichiarazioni particolari contenute nei file di accesso delle singole directory (.htaccess). La tabella 160.3 ne riporta l'elenco. In particolare, le opzioni None e All vanno usate da sole.

Tabella 160.3. Alcune opzioni della direttiva AllowOverride nella sezione Directory.
Parola chiave Significato
None Impedisce che qualunque direttiva venga scavalcata.
All Permette che siano scavalcate tutte le direttive.
Options Permette l'uso della direttiva Options.
Limit Permette l'uso di direttive di controllo sugli accessi.
AuthConfig Permette l'uso di direttive di autorizzazione.

FileInfo Permette l'uso di direttive di controllo del tipo di documento.
Indexes Permette l'uso di direttive di controllo sul listato delle directory.

L'esempio seguente mostra la semplice dichiarazione della directory /home/httpd/html/ (corrispondente a DocumentRoot), in cui e` consentito visualizzare il listato del contenuto e seguire i collegamenti simbolici. Per questa directory (e per le successive) non e` possibile scavalcare alcuna direttiva utilizzando i file .htaccess.

```
<Directory /home/httpd/html>
    Options Indexes FollowSymLinks
    AllowOverride None
</Directory>
```

Autorizzazioni

Attraverso una serie di direttive e` possibile definire l'autorizzazione all'accesso alla directory, fornendo un nominativo e una parola d'ordine. Questi nominativi e le parole d'ordine cifrate relative devono essere contenuti in un file creato con un programma apposito, htpasswd, ed e` necessario che non coincidano con nominativi e parole d'ordine gia` utilizzati per accedere al sistema. Infatti, il programma cliente memorizza queste informazioni la prima volta che vengono inserite, quindi le fornisce automaticamente a ogni richiesta.

A fianco del file di utenti e parole d'ordine, si puo` creare un file di gruppi che serve solo a facilitare la definizione delle autorizzazioni, quando si vuole fare riferimento a un intero gruppo di utenti, senza doverli elencare.

AuthName nome

La direttiva AuthName permette di definire un nome per identificare il contesto dell'autorizzazione. Questa descrizione viene data all'utente quando gli viene richiesto di inserire il nominativo e la parola d'ordine, in modo da permettergli di distinguere tra autorizzazioni diverse che possono richiedere un'identificazione differente.

AuthType Basic

Questa direttiva e` obbligatorie e specifica il tipo di autorizzazione.

AuthUserFile file_di_utenti_e_parole_d'ordine

Specifica un file da utilizzare come elenco di utenti e parole d'ordine. Questo file viene creato e aggiornato utilizzando il programma htpasswd.

AuthGroupFile file_dei_gruppi

Specifica un file da utilizzare come elenco di gruppi abbinati agli utenti. Non contiene parole d'ordine e viene creato in modo manuale.

require user utente...

require group gruppo...

require valid-user

Una di queste direttive stabilisce la necessita` dell'identificazione attraverso un nominativo-utente e una parola d'ordine. Nel primo caso si indicano precisamente quali utenti possono accedere, nel secondo quali gruppi di utenti e nel terzo si afferma semplicemente che possono accedere tutti.

L'esempio seguente definisce un accesso ristretto e condizionato al riconoscimento degli utenti. In particolare pero`, solo gli utenti tizio, caio e semproni possono accedere.

```
<Directory /home/httpd/html/riservato>

    AllowOverride None
    Options Indexes

    AuthName "Informazioni riservate"
    AuthType Basic
    AuthUserFile /etc/apache/.htpasswd
    AuthGroupFile /etc/apache/.htgroup
    require user tizio caio semproni

</Directory>
```

Limitazione dell'accesso

In origine, queste direttive erano consentite solo nella sezione Limit. Se vi appaiono fuori, indicano che si riferiscono a qualunque metodo di accesso. Quando si utilizzano queste direttive, se si intende fare uso di nomi di dominio e` indispensabile avere attivato la risoluzione dei nomi di dominio attraverso la direttiva HostnameLookups nel file httpd.conf.

order allow,deny

order deny,allow

order mutual-failure

In questo modo si specifica l'ordine in cui devono essere prese in considerazione le direttive deny e allow. Quando si specifica la parola chiave mutual-failure, si intende che possono accedere solo i nodi che appaiono nella lista allow e non appaiono in quella deny.

deny from { all | host... }

Impedisce l'accesso da parte dei nodi elencati. Se si usa la parola chiave all si impedisce a tutti di accedere. I nodi possono essere indicati attraverso il nome di dominio, completo o parziale, e attraverso l'indirizzo numerico, completo o parziale.

allow from { all | host... }

Consente l'accesso da parte dei nodi elencati. Se si usa la parola chiave all si consente a tutti di accedere. I nodi possono essere indicati attraverso il nome di dominio, completo o parziale, e attraverso l'indirizzo numerico, completo o parziale.

L'esempio seguente stabilisce il blocco all'accesso da parte degli utenti del dominio mehl.dg, a partire dalla directory dichiarata nell'apertura della sezione Directory.

```
<Directory /home/httpd/html/polenta>
        AllowOverride None
        Options Indexes

        order allow,deny
        allow from all
        deny from .mehl.dg
</Directory>
```

L'esempio seguente, invece, concede solo al dominio mehl.dg di poter accedere.

```
<Directory /home/httpd/html/polenta>
        AllowOverride None
        Options Indexes

        order deny,allow
        deny from all
        allow from .mehl.dg
</Directory>
```

L'esempio seguente e` una variante del precedente, in cui si utilizza anche l'indicazione di una sottorete in forma di indirizzo numerico.

```
<Directory /home/httpd/html/polenta>
        AllowOverride None
        Options Indexes

        order deny,allow
        deny from all
        allow from .mehl.dg 192.168.2.
</Directory>
```

Sezione ?Limit?

Le sezioni Limit sono usate per racchiudere un gruppo di direttive di controllo di accesso, che riguardano solo i metodi specificati. I metodi di accesso in questione sono, per esempio, GET e POST.

```
<Directory /home/httpd/html/riservato>

    AllowOverride None
    Options Indexes

    AuthName "Informazioni riservate"
    AuthType Basic
    AuthUserFile /etc/apache/.htpasswd
    AuthGroupFile /etc/apache/.htgroup

    <Limit GET POST>
        require valid-user
    </Limit>

</Directory>
```

L'esempio mostra che per la directory specificata e` richiesta l'autenticazione solo in caso di utilizzo dei metodi GET e POST.

Quando si vuole che le direttive di controllo di accesso riguardino tutti i metodi di accesso, non si usa la sezione Limit.

In linea di massima, la sezione Limit puo` contenere ogni direttiva, a esclusione della dichiarazione ulteriore di sezioni Directory e Limit annidate. In pratica, si utilizzano solo direttive per cui abbia senso porre un limite basato sul metodo di accesso. Generalmente ha significato l'utilizzo delle direttive indicate nella tabella 160.4.

Tabella 160.4. Alcune direttive utili nella sezione Limit.

Direttiva	Descrizione
require	Utenti che possono accedere attraverso autenticazione.
order	Ordine di valutazione delle direttive deny e allow.
deny	Specifica i nodi a cui viene negato l'accesso.
allow	Specifica i nodi a cui viene concesso l'accesso.

Sezione ?Location?

Le sezioni Location raccolgono le direttive di controllo per un URI particolare. Si tratta di qualcosa molto simile alla sezione Directory, con la differenza che il riferimento e` fatto all'URI piuttosto che alla directory del file system effettivo.

Questa sezione viene usata prevalentemente per abilitare l'accesso allo stato del servente attraverso l'indicazione di un URI, da parte di un particolare indirizzo autorizzato.

```
<Location /status>
      SetHandler server-status
      order deny,allow
      deny from all
      allow from dinkel.brot.dg
</Location>
```

Nell'esempio viene concesso al nodo dinkel.brot.dg di accedere all'URI /status cui e` abbinata la generazione e la restituzione di informazioni sul sistema. Il risultato potrebbe essere qualcosa di simile a quello che segue.

```
Apache Server Status for dinkel.brot.dg

Current Time: Sun Sep 28 14:00:47 1997
Restart Time: Sun Sep 28 14:00:28 1997
Server uptime: 19 seconds
Total accesses: 0 - Total Traffic: 0 B
CPU Usage: u0 s0 cu0 cs0
0 requests/sec - 0 B/second -

Scoreboard:

K___W__.....
..................................................
..................................................

Key:
"_" Waiting for Connection, "S" Starting up,
"R" Reading Request, "W" Sending Reply,
"K" Keepalive (read), "D" DNS Lookup, "L" Logging

2 requests currently being processed, 5 idle servers

Srv PID   Acc  M CPU  SS Conn Child Slot         Host              Request
0   8449 0/0/0 K 0.00 10 0.0  0.00  0.00
4   8453 0/0/0 W 0.00 0  0.0  0.00  0.00 dinkel.brot.dg GET /status HTTP/1.0
-------------------------------------------------------------------------
 Srv  Server number
 PID  OS process ID
 Acc  Number of accesses this connection / this child / this slot
  M   Mode of operation
 CPU  CPU usage, number of seconds
 SS   Seconds since beginning of most recent request
Conn  Kilobytes transferred this connection
Child Megabytes transferred this child
Slot  Total megabytes transferred this slot
```

Controllare l'accesso con .htaccess

I file .htaccess possono essere usati per definire delle configurazioni specifiche riferite alla directory in cui si trovano. Non e` necessario il loro utilizzo; si tratta solo di una possibilita`, che peraltro deve essere controllata attraverso la direttiva AllowOverride nel file access.conf. In linea di massima, i file .htaccess possono contenere le direttive elencate nella tabella 160.5

Tabella 160.5. Alcune direttive utili nel file .htaccess.
Direttiva Descrizione
Options Opzioni varie.
DefaultType Definisce il tipo e il sottotipo MIME predefinito per la directory.
ErrorDocument Ridefinisce la risposta in caso si verifichino condizioni di errore.
AuthName Nome o descrizione di una zona soggetta ad autenticazione.
AuthType Definizione del tipo di autenticazione.
require Dichiarazione degli utenti autorizzati all'autenticazione.
order Ordine di valutazione delle direttive deny e allow.
deny Specifica i nodi a cui viene negato l'accesso.
allow Specifica i nodi a cui viene concesso l'accesso.

Considerazioni sulla sicurezza

Dalla descrizione dei file di configurazione di Apache si possono intuire i punti su cui agire per cercare di ottenere un servizio HTTP relativamente ?sicuro?. Vale comunque la pena di sottolineare alcune cose.

Il demone httpd viene avviato normalmente con i privilegi dell'utente root, quindi, attraverso delle opportune chiamate di sistema, httpd cambia questi privilegi portandoli a quelli dell'utente e del gruppo specificati con le direttive User e Group del file httpd.conf.

E` molto importante che l'utente e il gruppo corrispondano a nobody, dove questo utente fittizio deve corrispondere idealmente al ?perfetto sconosciuto? che accede al sistema, oppure, ancora meglio, che si tratti di un utente apposito. In questa ottica devono poi essere regolati i permessi delle directory.

I file amministrativi di Apache, cioe` quelli di configurazione e di registrazione degli eventi, non devono essere accessibili in scrittura da parte degli utenti comuni (di qualunque tipo siano, escluso root). Nello stesso modo, non devono essere modificabili le directory che li contengono.

I file che compongono i documenti ipertestuali devono essere accessibili solo in lettura agli utenti comuni, cosi` le directory non devono essere modificabili, eccetto i permessi che puo` avere l'utente root.

E` consigliabile utilizzare la direttiva SymLinksIfOwnerMatch per evitare brutti scherzi da parte degli utenti che hanno la possibilita` di creare documenti HTML a partire dalla loro directory personale.

E` bene evitare di permettere l'utilizzo di programmi CGI al di fuori della directory

definita con la direttiva ScriptAlias nel file srm.conf.
E` opportuno evitare di concedere agli utenti comuni di modificare le impostazioni attraverso i file .htaccess. Si ottiene questo con la direttiva AllowOverride None.

Segue un esempio molto semplice della configurazione del file access.conf.

```
# Prima si impedisce l'accesso alla radice del file system.
<Directory />
        AllowOverride None
        Options None
        order deny,allow
        deny from all
</Directory>

# Si definisce la directory DocumentRoot.
<Directory /home/httpd/html>
        Options Indexes SymLinksIfOwnerMatch
        AllowOverride None
        order deny,allow
        allow from all
</Directory>

# Si concede di accedere alle directory personali degli utenti.
<Directory /home/*/public_html>
        Options Indexes SymLinksIfOwnerMatch
        AllowOverride None
        order deny,allow
        allow from all
</Directory>

# Si concede l'esecuzione ai programmi CGI che si trovano a
# partire dalla directory predisposta per questo.
<Directory /home/httpd/cgi-bin>
        AllowOverride None
        Options ExecCGI
        order deny,allow
        allow from all
</Directory>

# Si concede l'accesso alla directory contenente le icone di sistema.
<Directory /home/httpd/icons>
        AllowOverride None
        Options None
        order deny,allow
        allow from all
</Directory>

# Abilita la lettura dello stato del servente.
```

```
<Location /status>
    SetHandler server-status
    order deny,allow
    deny from all
    allow from .brot.dg
</Location>
```

Come si puo` osservare, non e` stato consentito in alcun caso di utilizzare i file .htaccess e i collegamenti simbolici sono tollerati se il proprietario del collegamento equivale a quello del file o della directory di destinazione. Inoltre sono state prese le misure seguenti:

- e` impedito l'accesso a partire dalla directory radice del file system, obbligando a dichiarare l'accesso alle directory successive;
- viene concesso l'accesso alla directory da cui si diramano i documenti ipertestuali;
- viene concesso espressamente l'accesso alle directory personali degli utenti;
- i programmi CGI possono essere eseguiti solo nella directory preposta a questo, il cui contenuto risulta illeggibile;
- l'accesso alla directory contenente le icone utilizzate da Apache e` stato dichiarato espressamente, altrimenti sarebbero risultate inaccessibili a causa del divieto iniziale sulla directory radice;
- e` stata abilita la lettura dello stato del servente, concedendo l'accesso solo al dominio brot.dg.

Utilizzo del sistema di autenticazione

Il sistema di autenticazione del programma servente permette di consentire l'accesso a directory determinate solo a utenti identificati in base a un nome e a una parola d'ordine. E` molto importante capire come funziona il meccanismo, per non farsi delle illusioni sull'efficienza del sistema.

Figura 160.1. Il programma cliente chiede all'utente di identificarsi quando per la prima volta cio` viene richiesto dal servente HTTP.

figure/a2-web-directory-autorizzazione

La prima volta che l'utente accede, il programma cliente gli presenta la richiesta di inserire il nominativo e la parola d'ordine, poi tutto funziona normalmente. Pero`, essendo il protocollo HTTP privo di stato, non si instaura una connessione stabile; ogni richiesta e` una connessione a parte e ognuna di queste richiede un'autenticazione. In effetti, il programma cliente memorizza i dati inseriti dall'utente e continua a fornirli al servente HTTP. Questo fatto ha due implicazioni: la parola d'ordine viaggia continuamente attraverso la rete; piu` utenti possono accedere simultaneamente da postazioni differenti, utilizzando la stessa identificazione e la stessa parola d'ordine. Sotto questo aspetto, e` importante che le parole d'ordine che si adoperano per queste cose non abbiano alcun nesso con quelle ?serie?.

Per gestire questo tipo di autenticazione, occorre generare un file di utenti e di parole d'ordine, aggiungendo possibilmente anche un file di gruppi. Si utilizza il programma htpasswd che normalmente fa parte del pacchetto di Apache:

htpasswd [-c] file utente

htpasswd crea o aggiorna un file di utenti e di parole d'ordine per l'autenticazione degli accessi a directory protette con il servente Apache. L'opzione -c viene usata per creare il file la prima volta, mentre si inserisce il primo utente. La parola d'ordine viene richiesta subito dopo l'avvio del programma.

Vengono mostrati e descritti alcuni esempi.

htpasswd -c passwd tizio[Invio]

Adding password for tizio.
New password:

Viene inserita la parola d'ordine seguita da [Invio].

Re-type new password:

Viene reinserita la parola d'ordine seguita da [Invio] e si ottiene il file passwd nella directory corrente.

cat passwd[Invio]

tizio:njHIUkjjJLKn

Il file contiene solo i nomi degli utenti e le parole d'ordine cifrate relative.

htpasswd passwd caio[Invio]

Quando si aggiungono utenti, non si utilizza l'opzione -c, altrimenti il file viene cancellato e ricreato.

htpasswd passwd caio[Invio]

Lo stesso programma puo` essere usato per modificare la parola d'ordine di un utente gia` registrato.

Per facilitare la gestione di utenti che utilizzano l'autenticazione per accedere a directory protette, e` possibile realizzare dei raggruppamenti e inserirli in un file senza parole d'ordine. Il formato del file e` molto semplice: ogni record e` costruito secondo la sintassi seguente:

gruppo: utente...

Quindi, i nominativi dei vari utenti sono separati da uno spazio, come nell'esempio seguente:

primo: tizio caio semproni
secondo: cane gatto topo

Nell'esempio sono dichiarati due gruppi: primo e secondo. A primo appartengono gli utenti tizio, caio e semproni; a secondo appartengono gli utenti cane, gatto e topo.

Configurazione

La configurazione delle directory che devono essere accessibili solo attraverso un'autenticazione, avviene nel file access.conf in una sezione Directory. Sono indispensabili le direttive AuthName, AuthType e AuthUserFile con cui si da` un nome all'autenticazione, si definisce il tipo e si indica il nome del file degli utenti e delle parole d'ordine. La direttiva AuthGroupFile serve solo se si intende fare riferimento a gruppi di utenti.

```
<Directory /home/httpd/html/riservato>

   AllowOverride None
   Options Indexes

   AuthName "Informazioni riservate"
   AuthType Basic
   AuthUserFile /etc/apache/passwd
   AuthGroupFile /etc/apache/group

   require valid-user

</Directory>
```

La direttiva require stabilisce a chi, tra gli utenti che sono dichiarati nel file di utenti e di parole d'ordine, sia concesso di accedere. La parola chiave valid-user rappresenta tutti gli utenti che sono stati previsti. In alternativa possono essere elencati gli utenti a cui concedere l'accesso, come nell'esempio seguente;

```
   require user tizio caio semproni
```

oppure si puo` indicare il nome di uno o piu` gruppi.

```
   require group primo terzo quinto
```

Solo nell'ultimo caso e` necessario predisporre e dichiarare la posizione del file dei gruppi.

Siti virtuali

Apache e` in grado di gestire diversi siti virtuali indipendenti sullo stesso elaboratore. In pratica, si distinguono diverse directory per le pagine HTML (diverse directory document root), dove ognuna di queste viene selezionata in base al nome di dominio utilizzato per accedere al servizio.

Evidentemente, per arrivare a questo risultato, occorre che lo stesso elaboratore sia accessibile utilizzando nomi di dominio differenti: si va dall'attribuzione di un semplice alias all'interno del DNS (i record CNAME), fino alla sovrapposizione di indirizzi IP differenti sulle stesse interfacce (con la conseguente attribuzione di nomi di dominio differenti). A proposito della gestione del DNS, si vedano i capitoli 122 e 123.

Quanto visto su Apache fino a questo punto, riguarda la gestione di un sito unico: quello ?reale?. Si osservi in particolare che la direttiva DocumentRoot viene inserita nel file srm.conf. Per definire dei siti virtuali alternativi si interviene nel file httpd.conf, attraverso delle sezioni simili a quelle del file access.conf:

```
<VirtualHost nome_di_dominio >
   direttiva_specifica
   ...
</VirtualHost>
```

In pratica, all'interno del marcatore di apertura dell'ambiente VirtualHost si inserisce il nome del sito virtuale a cui si fa riferimento, mentre all'interno della sezione si inseriscono le direttive specifiche per questo sito.

```
<VirtualHost prova.brot.dg>
   ServerAdmin webmaster@prova.brot.dg
   DocumentRoot /home/httpd/html2
   ServerName prova.brot.dg
   ErrorLog logs/prova.brot.dg-error_log
   TransferLog logs/prova.brot.dg-access_log
</VirtualHost>
```

L'esempio mostra la predisposizione del sito virtuale prova.brot.dg. All'interno della sezione si vedono le dichiarazioni:

- dell'indirizzo di posta elettronica dell'amministratore, con la direttiva ServerAdmin;
- della directory di partenza delle pagine HTML, con la direttiva DocumentRoot;
- del nome di dominio corretto per raggiungere il sito virtuale, con la direttiva ServerName;
- dei percorsi assoluti dei file delle registrazioni, con le direttive ErrorLog e TransferLog.

E` il caso di osservare la stranezza per la quale la direttiva DocumentRoot puo` apparire nella sezione VirtualHost all'interno del file httpd.conf, mentre per il sito reale si usa il file srm.conf.

Nel momento in cui si dichiara l'utilizzo di una nuova directory per i dati (le pagine HTML), ci si deve preoccupare anche di configurare l'accesso a tale directory. Questo si fa nel modo solito all'interno del file access.conf. Seguendo l'esempio mostrato, potrebbe essere necessario aggiungere la sezione seguente:

```
<Directory /home/httpd/html2>
        Options Indexes SymLinksIfOwnerMatch
        AllowOverride None
        order deny,allow
        allow from all
</Directory>
```

cap 16) Servente Samba

I demoni del servente

Un servente Samba si basa su due demoni:

- smbd che fornisce i servizi di condivisione di file stampanti per i clienti SMB (che possono essere macchine MS-Windows o altre macchine GNU/Linux) e si occupa della gestione delle sessioni di comunicazione e delle autenticazioni necessarie all'accesso alle risorse che vengono offerte in condivisione dal servente; il demone avvia una copia di se stesso per ogni richiesta di servizio da soddisfare;
- nmbd che gestisce la distribuzione dell'elenco delle risorse condivise alle altre macchine della rete, puo` mantenere la lista delle risorse condivise (scansione della rete) ed eventualmente risolvere i nomi NetBIOS della rete (servente WINS);

Attivazione del servente Samba

Entrambi i demoni smbd e nmbd possono essere attivati in modo autonomo, o gestiti dal supervisore dei servizi di rete (come Xinetd); qui viene presa in esame solo la prima alternativa che e` di gran lunga la piu` praticata e anche quella predefinita in molte distribuzioni GNU/Linux.

In quasi tutte le distribuzioni si trovano infatti degli script preconfezionati per l'attivazione e la disattivazione di determinati servizi; nel caso della Red Hat si attivano entrambi i demoni con il comando:

```
# /etc/rc.d/init.d/smb start
```

e si disattivano con il comando:

```
# /etc/rc.d/init.d/smb stop
```

ci sono poi anche i comandi:

```
# /etc/rc.d/init.d/smb restart
```

e

```
# /etc/rc.d/init.d/smb status
```

il cui significato dovrebbe essere ovvio.

Configurazione di un servente Samba

La configurazione di un servente Samba si basa su di un file di testo; nella distribuzione

Red Hat (ma anche per altre) e` /etc/samba/smb.conf.

Solitamente viene fornito preconfezionato e commentato, con una configurazione di base gia` pronta all'uso; ovviamente e` possibile intervenire sul file per adattare il comportamento del servente alle proprie esigenze.

Prima di esaminare la struttura del file e i parametri principali di configurazione e` opportuno sottolineare alcuni importanti aspetti generali:

- E` indifferente usare maiuscole o minuscole a meno che tale uso non vada a interferire con le regole del sistema operativo sottostante. Se ad esempio si indica il percorso di una directory condivisa su una macchina GNU/Linux con l'opzione (che verra` descritta piu` avanti) PATH=/USR/LOCAL, Samba non ha alcun problema ad accettare la direttiva ma al momento di collegarsi alla risorsa fallisce in quanto in GNU/Linux quella directory al 99 % non esiste, mentre esiste /usr/local/. E` quindi consigliabile l'uso delle minuscole.
- Le righe di commento iniziano con i simboli ; oppure #.
- Il carattere di continuazione riga e` \.
- Alcune direttive di configurazione di Samba, per ragioni di compatibilita`, sono ridondanti; per questo motivo uno stesso risultato si puo` ottenere in modi diversi.
- Per rendere effettive le variazioni fatte al file di configurazione non e` necessario riavviare i demoni di Samba in quanto il file viene riletto automaticamente ogni 60 s; se si vuole forzare la rilettura basta impartire il comando: kill -SIGHUP n dove n e` il numero del processo corrispondente al demone smbd in funzione (per individuarlo si puo` eseguire ps afx | grep smbd). Occorre comunque notare che non tutti i cambiamenti alla configurazioni vengono necessariamente attuati subito; in particolare le variazioni della configurazione di risorse condivise rimangono congelate finche' c'e` qualche utente connesso a tali risorse.

Il file di configurazione e` suddiviso in sezioni i cui nomi sono racchiusi tra parentesi quadrate.

Ogni sezione corrisponde a una risorsa condivisa a eccezione della sezione global usata per le configurazioni globali. Altre sezioni con un ruolo un po' particolare sono homes e printers.

Sezione [global]

In essa si impostano le informazioni che condizionano tutto il sistema ed eventualmente quei parametri che se non specificati vengono assunti in modo predefinito, ad esempio il nome del gruppo di lavoro (workgroup).

Sezione [homes]

In essa si regolano i parametri di configurazione delle directory personali degli utenti che si collegano al servente Samba.

Sezione [printers]

Consente di impostare le caratteristiche della condivisione di tutte le stampanti installate nella macchina GNU/Linux senza dover definire una condivisione separata per ognuna di esse.

All'interno del file di configurazione e` possibile usare alcune variabili il cui nome viene sostituito dal rispettivo valore quando il file di configurazione viene utilizzato dai demoni smbd e nmbd.

Segue una lista delle variabili piu` importanti con una breve descrizione:

Variabile Descrizione

%S
nome del servizio corrente ([tmp], [homes], ecc.);

%P
directory principale del servizio corrente;

%u
nome dell'utente (GNU/Linux) del servizio corrente;

%g
nome del gruppo primario di %u;

%U
nominativo-utente della sessione;

%D
nome del dominio MS-Windows in cui il servente Samba si integra;

%G
nome del gruppo primario di %U;

%H
directory personale assegnata a %u;

%v
versione in uso di Samba;

%h
nome del nodo che ha avviato il servizio Samba;

%m
nome NetBIOS della macchina cliente;

%L
nome NetBIOS assegnato al servente;

%M
nome Internet della macchina cliente;

%N
nome della directory personale ottenuta dal servizio NIS del servente;

%p
percorso della directory personale ottenuto dal servizio NIS;

%d
numero identificativo del processo corrente;

%a
architettura software della macchina remota (Samba, MS-Windows 95/98/Me/NT), diversamente sara` assegnata la parola UNKNOWN;

%I
indirizzo IP della macchina cliente;

%T
data e ora corrente.

207.3.1 smb.conf: sezione global

E` la sezione che appare in tutte le configurazioni di Samba, anche se non e` obbligatoria. Le opzioni in essa contenute vengono applicate a tutte le altre sezioni.

Viene mostrato un esempio comprendente alcune direttive di uso comune suddivise in blocchi in base alla funzione svolta e intervallate da brevi descrizioni del loro significato:

```
[global]
#
# identificazione del servente
#
   workgroup = INF
   netbios name = pippo
   server string = Samba Server
```

La voce piu` importante e` workgroup che assegna a Samba il dominio o il gruppo di appartenenza. Occorre assegnarla correttamente, pena conflitti nella rete paritetica (peer-to-peer).

La voce netbios name e` di utilizzo meno frequente e serve ad assegnare al servente Samba un nome NetBIOS a piacere. Il nome NetBIOS viene infatti assegnato in modo definito pari a quello ottenuto dal DNS. Ad esempio se il nome DNS del servente fosse muscolis.inf.besta il nome NetBIOS sarebbe muscolis. Uno dei casi in cui e` utile poter impostare un nome NetBIOS diverso da quello predefinito e` quello in cui la rete e` suddivisa in due o piu` domini DNS diversi; in questo caso potrebbe infatti anche esistere una macchina con nome muscolis.mat.besta che verrebbe quindi ad avere lo stesso nome

NetBIOS. Ovviamente il valore di netbios name deve essere assegnato seguendo le regole dei nomi NetBIOS (unica stringa senza punti contenente i simboli alfabetici maiuscoli e minuscoli le cifre e i simboli !, @, #, $, %, ^, &, (,), -, ' e ~).

Con server string si assegna semplicemente la descrizione dell'elaboratore servente.

```
#
# opzioni di rete
#
   hosts allow = 192.168.1. localhost
   hosts deny  = 172.16.244.254
   interfaces  = 192.168.1.1/24 172.16.244.1/16
   bind interfaces only = yes
```

Le direttive Host allow e host deny servono rispettivamente a specificare quali nodi possono e non possono accedere alle risorse condivise dal servente Samba. L'indicazione puo` essere fatta tramite il nome del nodo, il nome di dominio, il numero IP, il numero della sottorete. Nell'esempio viene concesso l'accesso a tutte le macchine della sottorete 192.168.1.* e al localhost (e` opportuno che l'accesso a localhost sia sempre concesso pena possibili malfunzionamenti della scansione delle risorse del servente) e viene negato alla macchina con indirizzo 192.168.1.3. Possono anche essere usate le parole chiave ALL, per designare qualsiasi elaboratore, e EXCEPT per indicare un'eccezione a una regola (ad esempio host allow 192.168.1. EXCEPT 192.168.1.3). Si deve inoltre notare che in caso di assenza delle direttive host allow e host deny, l'accesso e` concesso a tutti in modo predefinito. Infine si tenga presente che tali direttive possono essere inserite anche in specifiche condivisioni ma con grado di priorita` inferiore rispetto a quanto specificato nella sezione global.

La direttiva interfaces e` utile in caso il servente Samba risieda in piu` di una sottorete. Se sull'elaboratore sono presenti piu` interfacce di rete, in modo predefinito, Samba si mette in ascolto di richieste provenienti dagli indirizzi di rete corrispondenti alla rete della prima interfaccia che trova (di solito eth0). Per fare in modo che invece risponda alle richieste provenienti da piu` sottoreti si deve impostare questa opzione. Nell'esempio Samba si pone in ascolto dalle sottoreti 192.168.1.* e 172.16.*.* (si puo` usare anche una notazione con maschera di rete: 192.168.1.1/255.255.255.0 172.16.244.1/255.255.0.0).

Ponendo bind interfaces only = yes (l'alternativa e` ovviamente no oppure si puo` evitare di inserire questa opzione), si forza il servente a rispondere soltanto alle sottoreti corrispondenti alle interfacce indicate in interfaces. In tal caso si deve inserire tra le interfacce anche 127.0.0.1 per permettere al programma smbpasswd di potersi collegare al localhost e funzionare correttamente.

```
#
# opzioni per la stampa
#
   printing = bsd
   printcap name = /etc/printcap
   load printers = yes
```

La direttiva printing permette di specificare il sistema di stampa in uso nel servente; per i sistemi GNU/Linux il valore da indicare e` di solito bsd, l'alternativa piu` diffusa e` sysv.

Le altre due opzioni permettono di caricare automaticamente tutte le stampanti configurate nel sistema senza descriverle singolarmente, indicando a tale scopo il percorso del file printcap contenente la definizione di tali stampanti. La loro configurazione relativamente a Samba viene poi indicata nell'apposita sezione printers descritta piu` avanti.

```
#
# opzioni per il log
#
   log file = /var/log/samba/%m.log
   max log size = 100
   log level = 3
```

La direttiva log file permette di indicare il file delle registrazioni per gli eventi Samba; tale file puo` essere unico oppure, come nell'esempio, diverso per ogni cliente che si collega al servente (il nome del file sara` nome_host.log). Altra possibilita` e` quella di avere un file di registrazioni per ogni utente usando opportunamente la variabile %U o %u.

Con max log size si specifica la grandezza in kibibyte del file delle registrazioni, raggiunta la quale il file stesso viene rinominato con estensione .old e reinizializzato; il file .old eventualmente gia` esistente viene cancellato. Il valore predefinito di questo parametro e` 5 000; il valore zero significa nessun limite di ampiezza (scelta non consigliabile per evitare una crescita abnorme del file o dei file delle registrazioni).

La direttiva log level indica il livello di dettaglio dei messaggi annotati nel registro; il valore predefinito e` zero e corrisponde a nessun messaggio. Aumentando questo valore si hanno messaggi sempre piu` dettagliati; e` comunque sconsigliabile un livello superiore a 3.

```
#
# opzioni per l'accesso alle condivisioni
#
   encrypt password = yes
   null password = yes
   guest account = utentesmb
   security = share
# in alternativa
;  security = user
;  security = server
;  security = domain
# altri parametri nel caso di security = user
```

```
;  smb passwd file = /etc/samba/smbpasswd
;  username map = /etc/samba/smbusers
# altri parametri nel caso di security = server o domain
;  password server = SERVER_NT
```

La direttiva encrypt password = yes e` praticamente obbligatoria se il servente Samba deve ?convivere? con macchine equipaggiate con sistemi operativi MS-Windows 98/NT o piu` recenti che usano le parole d'ordine cifrate.

La direttiva null password = yes permette di avere utenti Samba con parola d'ordine nulla (il valore predefinito e` no.)

La direttiva guest account indica il nome di un utente generico al quale puo` essere consentito l'accesso alle condivisioni (da usare nel caso di security = share, come dettagliato piu` avanti). Tale utente deve essere definito nel sistema GNU/Linux senza directory personale e senza shell, aggiungendo al file /etc/passwd la riga:

utentesmb::499:499:utente generico samba:/dev/null:/dev/null.

Il valore predefinito e` nobody.

Il parametro security e` di importanza fondamentale e richiede una trattazione leggermente piu` ampia.

Livello di sicurezza ?share?

Con l'impostazione security = share si ha il controllo di accesso a livello di condivisione: il cliente che vuole accedere a una risorsa invia ogni volta una parola d'ordine e nessun nominativo-utente. Samba tenta di dedurre il nominativo-utente dalla direttiva valid users eventualmente inserita nella sezione di condivisione di quella risorsa (come illustrato in seguito), oppure dal nome dell'elaboratore cliente, o, in caso di insuccesso, da quanto indicato con il parametro guest account (questo solo se fra i parametri di condivisione e` indicato guest ok = yes e guest only = yes).

Questo livello di sicurezza si usa, soprattutto nel caso di utenti MS-Windows e GNU/Linux non coincidenti, per condividere porzioni di file system quando si ha interesse a far si` che tutti gli utenti abbiano gli stessi diritti sui file condivisi (con l'impostazione guest account = utentesmb il ?proprietario? delle risorse condivise sara` sempre l'utente GNU/Linux utentesmb qualunque sia il nominativo dell'utente MS-Windows che si connette).

Livello di sicurezza ?user?

Con l'impostazione security = user, che e` quella predefinita, si ha il controllo di accesso a livello di utente: il cliente che vuole accedere a una risorsa invia al momento della connessione una coppia utente-parola d'ordine in base alla quale avviene l'autenticazione da parte del servente. Se la connessione viene accettata il cliente puo` accedere a tutte le risorse condivise senza doversi autenticare nuovamente a ogni accesso.

Il controllo delle utenze viene effettuato dal servente Samba in base al contenuto del file dei suoi utenti che solitamente e` /etc/samba/smbpasswd (si puo` cambiare il nome del file o il percorso tramite il parametro smb passwd file =).

Gli utenti Samba vengono aggiunti con il comando smbpasswd illustrato in un paragrafo successivo.

E` di fondamentale importanza notare che comunque il nome utente Samba deve coincidere con il nome di un utente definito nel sistema GNU/Linux, in quanto quest'ultimo deve essere sempre in grado di assegnare un proprietario ?valido? agli eventuali file creati o copiati dall'utente conneso all'interno della risorsa condivisa.

Questa esigenza puo` costituire un problema, ad esempio per la limitazione a otto caratteri dei nominativi-utente di GNU/Linux, che viene in parte superato grazie all'uso di un file di corrispondenza tra utenti GNU/Linux e utenti MS-Windows, il cui nome e` indicato con la direttiva username map =. Un valore abbastanza comune di tale parametro e` /etc/samba/smbusers. Tale file conterra` delle righe cosi` formate:

nome_linux = nome_smb_1 nome_smb_2...

Per ovvi motivi di sicurezza, entrambi i file smbusers e smbpasswd, devono essere accessibili sia in scrittura che in lettura dal solo utente root.

Livello di sicurezza ?server?

Con l'impostazione security = server si ha lo stesso controllo di accesso visto nel caso di user con la differenza che l'utenza viene controllata su un servente esterno (solitamente un PDC MS-Windows) il cui nome (NetBIOS) viene indicato con la direttiva password server =.

Livello di sicurezza ?domain?

Con l'impostazione security = domain si ha ancora lo stesso controllo di accesso visto nel caso di user, ma questa volta il servente Samba va a inserirsi in un dominio MS-Windows NT/2000.

Per ottenere questo risultato occorre fermare i demoni di Samba, quindi aggiungere il servente Samba al dominio NT sul PDC usando il server manager di MS-Windows NT oppure a una active directory sul servente MS-Windows 2000 con la MMC (Microsoft management console) attraverso lo strumento ?Utenti e computer di Active Directory?. A questo punto si deve eseguire il comando:

smbpasswd -j nome_dominio -r nome_server -Unome_utente%parola_d'ordine

Il nome_dominio deve essere lo stesso indicato nella direttiva workgroup di smb.conf; il nome_server deve coincidere con il valore di password server; nome_utente e relativa parola_d'ordine devono rappresentare un'utenza con privilegi sufficienti ad aggiungere un'utenza nuova nella macchina MS-Windows NT/2000.

Ultimate queste operazioni occorre naturalmente riavviare il servizio Samba.

Il vantaggio principale dell'impostazione domain rispetto a quella server consiste nel fatto che il PDC risulta meno carico, in quanto non e` piu` necessaria una connessione di rete permanente tra esso e il servente Samba. Quest'ultimo infatti effettua una chiamata RPC (Remote procedure call) solo al momento dell'autenticazione e non necessita di essere costantemente connesso al PDC come avviene nel caso del livello di sicurezza server.

Per concludere occorre notare che comunque anche con questo tipo di impostazione (come pure per quella server) e` necessario tenere allineati gli elenchi degli utenti dal lato MS-Windows e dal lato GNU/Linux (il motivo e` stato illustrato nel paragrafo del livello user). Ci sono pero` due direttive della sezione global che permettono di automatizzare l'aggiornamento degli utenti GNU/Linux:

```
add user = script_1 %u
delete user = script_2 %u
```

La prima entra in azione quando, a seguito di una connessione di un cliente MS-Windows, Samba si rivolge al servente di dominio per l'autenticazione con esito positivo ma l'utente GNU/Linux corrispondente non esiste; ovviamente lo script script_1 deve essere scritto in modo adeguato affinche' crei l'utente ricevendolo come parametro dalla variabile %u.

In modo speculare si usa l'altra direttiva che entra in azione quando, a seguito di una connessione di un cliente MS-Windows, Samba si rivolge al servente di dominio per l'autenticazione con esito negativo ma il corrispondente utente GNU/Linux esiste; in questo caso script_2 deve provvedere a cancellare l'utente in questione.

smb.conf: sezioni generiche di condivisione

Una sezione indicante una risorsa condivisa puo` avere un nome a piacere purche' sempre racchiuso tra parentesi quadrate.

Vengono adesso illustrati alcuni esempi allo scopo di descrivere almeno le direttive piu` importanti:

```
#
# condivisione 1: una directory accessibile solo a certi utenti
#
[Pagine WWW]
  comment = Dir per le pagine web
  browseable = yes
  public = no
```

```
   path = /var/www
   writable = yes
   valid users = utente1 utente2
```

La direttiva comment serve ad associare una descrizione alla risorsa condivisa.

L'opzione browseable permette di rendere visibile o no la risorsa agli utenti che si connettono al servente.

public e` un sinonimo di guest ok; in questo esempio non si vuole che la risorsa sia accessibile per l'utente generico.

path permette di indicare il percorso della risorsa sul sistema GNU/Linux.

writable serve a concedere o negare l'accesso in scrittura (un sinonimo e` writeable.

valid users indica quali sono gli utenti che possono accedere alla risorsa. E` anche possibile indicare gruppi di utenti GNU/Linux con la sintassi +nome_gruppo e gruppi di utenti NIS (ovviamente deve essere presente in rete un servente NIS) con la sintassi &nome_gruppo e anche entrambi con la sintassi +&nome_gruppo o &+nome_gruppo, o ancora @nome_gruppo.

```
#
# condivisione 2: una directory pubblica = accessibile a tutti
#
[public]
   comment = Dir pubblica
   browseable = yes
   guest ok = yes
   path = /usr/local/public
   writable = yes

#
# condivisione 3: una directory pubblica = accessibile a tutti in cui tutti
# gli utenti possano creare, modificare, cancellare tutti i file
#
[temp]
   comment = Dir pubblica plus
   browseable = yes
   guest ok = yes
   guest only = yes
   path = /tmp
   writable = yes
```

La differenza tra le due condivisioni e` molto sottile ma anche interessante. La presenza di guest ok = yes permette le connessioni anonime; eventuali file creati o copiati nella directory sarebbero di proprieta` dell'utente indicato in guest account in caso di accesso anonimo, oppure dell'utente effettivo in caso esso fosse registrato in GNU/Linux. Se e`

presente anche guest only = yes invece il proprietario e` sempre l'utente fittizio ospite anche nel caso l'utente collegato fosse riconosciuto regolarmente da GNU/Linux; in questo caso quindi tutti gli utenti possono fare tutte le operazioni con qualsiasi file presente nella directory condivisa.

Un modo alternativo per ottenere lo stesso risultato e` quello di usare la direttiva force user = nome_utente; in questo modo Samba assegna lo stesso nominativo-utente a chiunque si connetta alla risorsa.

```
#
# condivisione 4: un cd-rom
#
[cd]
  comment = CD-ROM
  preexec = mount /mnt/cdrom
  postexec = umount /mnt/cdrom
  browseable = yes
  public = yes
  path = /mnt/cdrom
  writable = no
```

Il significato delle impostazioni e` ovvio compreso quello delle due opzioni preexec e postexec. In caso si tema che possano connettersi utenti sprovvisti dei privilegi per montare e smontare il CD, si possono sostituire le due direttive rispettivamente con root preexec e root postexec che svolgono lo stesso compito ma con i privilegi dell'utente root.

```
#
# condivisione 5: una directory privata con permessi preimpostati
#
[privata]
  comment = Dir privata
  browseable = yes
  path = /usr/local/private
  writable = no
  public = no
  write list = pippo pluto
  create mask = 0644
  directory mask = 0644
```

In questo esempio nella directory non sarebbe possibile scrivere, ma la presenza della direttiva write list permette di impostare permessi di scrittura, per gli utenti indicati, indipendentemente da quanto specificato negli altri parametri. Si deve pero` notare che i permessi impostati a livello di sistema su quella risorsa hanno sempre il sopravvento su quanto specificato nel file di configurazione di Samba (in altre parole, se /usr/local/private e` di proprieta` dell'utente root e i permessi sono impostati a 6008, gli altri utenti non possono ne' leggere ne' scrivere alcunche' in quella directory condivisa).

Le ultime due direttive, infine, servono a indicare i permessi con cui verranno creati file e

directory all'interno della risorsa condivisa; il valore predefinito e` 07558

Altre direttive importanti sono:

admin users = tizio
follow symlinks = no

Con la prima si indica che l'utente tizio ha gli stessi privilegi dell'utente root sulla condivisione e quindi non risente di eventuali limitazioni dovute ai permessi sui file; con la seconda si impedisce che vengano seguiti i collegamenti simbolici evitando che chi accede alla condivisione possa accedere anche a file che si trovano all'esterno di questa.

smb.conf: sezione ?homes?

La sezione homes viene utilizzata affinche' ogni utente possa avere accesso a una propria directory personale sul servente Samba che potra` anche coincidere con la directory personale GNU/Linux di quell'utente. Un esempio di definizione puo` essere il seguente:

```
#
# directory personali degli utenti
#
[homes]
   comment = directory home
   browseable = no
   writable = yes
   path = usr/local/samba/%S
```

Qui e` importante l'impostazione che impedisce la scansione della risorsa in modo che essa non appaia con il nome homes a tutti gli utenti.

La presenza di path serve a fare in modo che questa directory non coincida con quella GNU/Linux dell'utente (cosa che sarebbe l'impostazione predefinita).

La logica di funzionamento e` la seguente: quando l'utente si connette, se l'utenza e` accettata, viene creata dal servente Samba una condivisione con le caratteristiche specificate nella sezione homes ma con un nome uguale a quello dell'utente connesso.

smb.conf: sezione ?printers?

Con questa sezione si impostano i parametri di configurazione di tutte le stampanti definite nel sistema GNU/Linux a patto che nella sezione global siano state inserite le direttive seguenti:

load printers = yes
printcap name = /etc/printcap

L'alternativa, che consiste nel definire le varie stampanti come singole risorse condivise, non viene presa in esame in questa sede.

Un esempio di configurazione per le stampanti e` il seguente:

```
#
# Stampanti
#
[printers]
   comment = stampanti
   path = /var/spool/samba
   browseable = no
   printable = yes
   public = yes
   writable = no
```

Qui occorre notare che il parametro path serve a impostare una directory per la coda di stampa, diversa da /tmp/ che e` quella predefinita.

Altra direttiva da segnalare e` printable con la quale si attiva la coda di stampa.

Programmi ausiliari per un servente Samba

Un primo strumento molto utile e` testparm con il quale si verifica la correttezza sintattica delle impostazioni scritte nel file smb.conf.

Il comando da eseguire e`:

```
# testparm /etc/samba/smb.conf
```

si ottiene una risposta suddivisa in due parti: prima il resoconto del controllo sintattico del file di configurazione, poi l'elenco delle risorse condivise descritte in esso.

Altro programma di fondamentale importanza e` smbpasswd, gia` visto in precedenza a proposito della connessione di Samba a un dominio MS-Windows NT, ma che si usa principalmente per definire utenti e parole d'ordine relative. La sintassi in questo caso e`:

```
smbpasswd [-a] [-x] [nominativo]
```

L'opzione -a permette di inserire un nuovo utente e poi di definirne la parole d'ordine; l'opzione -x permette invece di eliminarlo; se non si indica alcuna opzione si esegue solo il cambio della parole d'ordine per l'utente. Il nominativo-utente che si puo` inserire alla fine della riga di comando e` quello sul quale il comando opera (se non viene indicato, si fa riferimento in modo predefinito all'utente GNU/Linux che esegue il comando).

Altro strumento utile e` lo script smbadduser che permette di definire un nuovo utente Samba e contemporaneamente l'associazione con utenti GNU/Linux corrispondenti. La

sintassi e`:

smbadduser utente_linux:utente_smb [utente_linux:utente_smb]...

In pratica questo comando aggiorna i file smbpasswd e smbusers, permettendo di definire la parola d'ordine per i nuovi utenti Samba.

Infine puo` essere molto utile anche il comando smbstatus per avere un rapporto (con l'opzione -d anche dettagliato) delle connessioni Samba attive.

Maggiori dettagli sulla configurazione di un servente Samba si possono trovare anche nella documentazione fornita insieme al pacchetto in /usr/share/doc/samba-x.y.z, oppure consultando la pagina di manuale smb.