UNIVERSITÀ DI PISA

FACOLTÀ DI INGEGNERIA

Amministrazione di un Sistema UNIX in Rete

A cura di
Giuseppe Anastasi
Andrea Passarella

Anno Accademico 2006-07

Indice

CAPITOLO 1

I SISTEMI UNIX

Nel corso della trattazione vengono presi in considerazione i sistemi UNIX. La scelta è stata dettata dal fatto che tali sistemi sono i più utilizzati tra quelli attualmente in circolazione.
La diffusione di questi sistemi è tale per due motivi:

- superiore qualità tecnica rispetto ad altri sistemi operativi;
- distribuzione tramite software libero.

In realtà i primi sistemi non erano *free*: attualmente invece la quasi totalità delle distribuzioni di sistemi UNIX è rilasciato sotto la dicitura di *software libero*.
In questa trattazione, si è scelto di utilizzare la distribuzione FreeBSD per quanto riguarda UNIX, mentre la distribuzione RedHat per quanto riguarda Linux.
Segue una breve storia dei sistemi UNIX. In particolare si è cercato di mettere in evidenza come tali sistemi siano diventati software libero.

UNIX

La prima edizione di UNIX venne sviluppata nel 1969 da Ken Thompson del Research Group presso i Bell Laboratories per utilizzare un elaboratore PDP-7 della DEC, altrimenti inutilizzabile. Ken Thompson è stato presto affiancato da Dennis Ritchie. Insieme ad altri collaboratori lavorarono attivamente sulle prime versioni di UNIX.
Ritchie aveva lavorato in precedenza al progetto MULTICS che ha influenzato notevolmente il nuovo sistema operativo. Infatti basti pensare che persino UNIX è un gioco di parole basato su MULTICS (*UNIX is Not multICS*). Inoltre l'organizzazione di base del file system, l'idea di utilizzare l'interprete dei comandi (shell) come processo utente, l'utilizzo di un processo separato per ogni comando e parecchie altre caratteristiche derivano direttamente da MULTICS.
Ritchie e Thompson hanno lavorato su UNIX per molti anni. Il lavoro svolto sulla prima versione ha permesso loro di passare a un calcolatore PDP-11/20 di maggior potenza per una seconda versione. Una terza versione è nata dalla riscrittura della maggior parte del Sistema Operativo nel linguaggio di programmazione C che ha sostituito il linguaggio Assembly precedentemente utilizzato. A questo punto vennero aggiunti al Sistemo Operativo originario, la multiprogrammazione ed altri miglioramenti volti a sfruttare al meglio l'hardware della macchina su cui girava.
Grazie a questo sviluppo, UNIX si è diffuso prima all'interno dei Bell Laboratories ed ha poi gradualmente conquistato anche alcune università. La prima versione ampiamente disponibile al di fuori dell'ambiente di sviluppo quale erano i Bell Laboratories fu la Versione 6 rilasciata nel 1976. Il numero di versione dei primi sistemi UNIX corrisponde al numero dell'edizione del *Manuale del Programmatore UNIX* in commercio al momento della distribuzione del sistema; in seguito il numero di codice si sviluppò in modo indipendente rispetto ai manuali.

UNIX BSD

I primi utenti di UNIX sono state le università, a cui in particolare questo sistema operativo veniva fornito a costo contenuto, ma senza alcun tipo di supporto tecnico, né 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 università.

Il maggior fermento intorno a UNIX si concentrò presso l'università della California a Berkeley, dove a partire dal 1978 si cominciò a distribuire una variante di questo sistema operativo: BSD (*Berkeley Software Distribution*). Per difendere il software prodotto in questo modo, 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. Ciò fino a quando si decise di ripulire lo UNIX BSD dal codice proprietario.
Il risultato iniziale fu 386BSD, che venne rilasciato nel 1992. Questa edizione libera dello UNIX BSD non ebbe vita facile, dal momento che da quel punto iniziarono delle contese giudiziarie sulla proprietà di alcune porzioni di codice ritenute libere. Dai problemi del 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; più tardi, apparve anche OpenBSD.
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 più importanti riguardano le piattaforme hardware in cui possono essere installate.

GNU

Nel 1985, Richard Stallman fondò la FSF, *Free software foundation*, con lo scopo preciso di creare e diffondere la filosofia del «software libero». Libertà intesa come la possibilità 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 così 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 così a una grande quantità di software utilizzabile sulla maggior parte delle piattaforme UNIX, indirizzando implicitamente il software libero nella direzione dei sistemi di questo tipo.

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, modificato e distribuito, 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 è nato come un progetto personale di studio delle funzionalità di multiprogrammazione dei microprocessori i386 da parte di Linus Torvalds, all'epoca uno studente all'università 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 riuscì ad arrivare a un sistema minimo e soprattutto autonomo da Minix. Il 5 ottobre 1991 inviò il messaggio seguente su *comp.os.minix*:

> *"Rimpiangete i bei tempi andati all'epoca di Minix-1.1, quando gli uomini erano uomini e programmavano da sé i propri device drivers? Vi manca un bel progetto e morite dalla voglia di affondare i vostri denti in un sistema operativo per poterlo modificare secondo le vostre esigenze? Trovate frustrante il fatto che tutto in Minix funzioni perfettamente? Niente più notti in bianco per far funzionare un fetente programma? Allora questa lettera potrebbe fare al caso vostro.*
> *Come dicevo un mese fa, sto lavorando su di una versione gratuita di un sistema simile a Minix per computer AT-386. Sono finalmente arrivato al punto in cui è utilizzabile (ma potrebbe anche non esserlo, dipende da cosa cercate), e voglio distribuire il codice sorgente per una più ampia diffusione. È solo la versione 0.02… ma sono riuscito a farci girare i bash, gcc, gnu-make, gnu-sed, compress, etc."*

L'anno di nascita di un sistema operativo basato sul kernel Linux è quindi il 1991.Linux non è 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 possibilità di modificarlo ecc. In pratica, la fortuna di Linux rispetto a Minix, è stata quella di avere scelto subito la licenza GNU-GPL, quella che ancora oggi rappresenta la difesa ideale per il software che viene scritto perché sia a disposizione di tutti. In questo modo si è superato il limite originale di Minix che lo rendeva interessante solo per professori e studenti.

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 è posto il problema di definire in modo preciso e inequivocabile cosa sia effettivamente il «software libero».
In questa direzione si è distinto particolarmente il gruppo che pubblica la distribuzione Linux Debian, nel definire una serie di punti che devono essere rispettati per l'inserimento del software nella distribuzione stessa.
Al problema dell'ambiguità del concetto, si affiancava l'ambiguità della denominazione: in inglese, free software poteva essere inteso come software gratuito (*free of charge*).
Così, nel 1998, nasce la definizione Open Source, a identificare i principi secondo cui il software può essere ritenuto «libero», riutilizzando gran parte del lavoro del gruppo Debian, ma dandogli un nome inequivocabile e non modificabile.

1.1 Componenti del sistema

I sistemi UNIX sono concettualmente divisibili in due componenti principali: il nucleo (*kernel*) il cui scopo è quello di interagire con l'hardware (incluse le memorie di massa ed i dispositivi di I/O) e gestire processi e file, e le *applicazioni* che si rivolgono al nucleo per ottenere i servizi richiesti dalle loro funzioni.

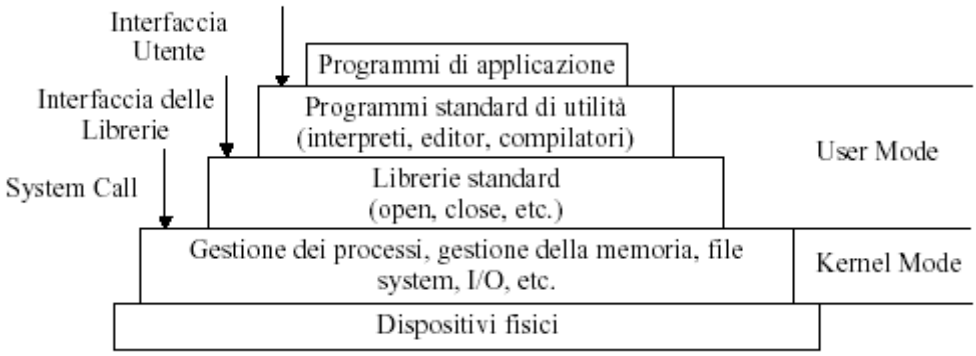


Figura 1 - Struttura di UNIX

Le applicazioni si possono suddividere in tre classi:

1. interpreti dei comandi
2. programmi di sistema
3. programmi utente

Gli interpreti dei comandi (*shell*) sono i programmi che permettono all'utente di interagire con il sistema operativo scrivendo dei comandi su un terminale. I programmi di sistema sono i vari strumenti messi a disposizione dal sistema operativo. I programmi utente sono invece quelli prodotti dall'utente per i propri scopi. Da notare che questa divisione è soltanto concettuale perché nulla vieta che ogni utente possa scrivere un programma, usarlo come quelli forniti dal sistema e metterlo poi a disposizione di altri utenti.

E' bene precisare che esiste un livello intermedio tra i programmi standard di utilità ed il Kernel. Tale livello è occupato dalle librerie standard. Queste sono utilizzate da linguaggi ad alto livello per invocare le chiamate di sistema (*system call*).

1.2 Interfaccia Utente

Generalmente, sia il programmatore che l'utente di un sistema UNIX hanno a che fare con programmi di sistema già scritti e disponibili per l'esecuzione. Questi programmi generano le system call necessarie per supportare le loro funzioni, ma le stesse system call sono contenute all'interno del programma e non devono necessariamente essere note all'utente.

Shell e Comandi

Sia i programmi scritti dall'utente che i programmi di sistema vengono normalmente eseguiti da un interprete di comandi. In sistemi UNIX tale interprete è un processo utente come qualsiasi altro: è denominato *shell* (guscio, conchiglia) in quanto racchiude il kernel del sistema operativo. Gli utenti possono scriversi da soli la propria shell, ed infatti esistono diverse shell generalmente utilizzate. Ad esempio la *Bourne shell* scritta da Steve Bourne è considerata la più diffusa. Anche la shell *Bash* (*Bourne Again SHell*) è molto diffusa ed inoltre è compatibile con la Bourne shell. Invece la *C shell*, scritta per gran parte da Bill Joy (un fondatore della Sun Microsystems) è la più diffusa sui sistemi BSD. La *Korn shell*, infine, è diventata famosa perché combina le caratteristiche della Bourne shell e della C shell.
Le usuali shell hanno in comune gran parte della sintassi del linguaggio dei comandi. Poiché UNIX è un sistema interattivo, la shell indica che il sistema è pronto ad accettare un altro comando visualizzando un prompt in corrispondenza del quale l'utente può digitare un comando su una sola riga. Ad esempio nella riga:

```
$ ls -l
```

il carattere di dollaro rappresenta il prompt usuale della Bourne shell e `ls -l` indica il comando di lista lunga della directory.
Da notare che la shell conserva una lista di parecchie directory, il cosiddetto *percorso di ricerca* o *search path*. Per ogni comando viene effettuata una ricerca ordinata su ciascuna delle directory contenute nel percorso di ricerca, allo scopo di trovare un file con lo stesso nome del comando immesso dall'utente. Se tale file viene trovato, viene caricato ed eseguito. Il percorso di ricerca può anche essere definito dall'utente. Di default le directory `/bin` e `/usr/bin` vengono sempre inserite.

I/O Standard

I vari comandi (e quindi i vari processi) disponibili in un sistema UNIX fanno un intenso uso dell'I/O Standard. In particolare i processi possono aprire file a loro piacimento. In realtà però, la maggior parte dei processi si aspetta che tre descrittori di file identificati dei numeri **0**, **1** e **2** vengano aperti al momento dell'avvio. Questi descrittori di file vengono ereditati attraverso la system call (`fork`) che ha creato il processo. Sono conosciuti come *standard input* (0), *standard output* (1) e *standard error* (2). Tutti e tre vengono normalmente aperti sul terminale dell'utente. Quindi il programma può leggere ciò che l'utente digita, leggendo lo standard input, ed inviare output sullo schermo dell'utente scrivendo sullo standard output. Anche il descrittore di file standard error è aperto per la scrittura e viene utilizzato per la segnalazione di errori. La maggior parte dei programmi può accettare per standard input e standard output anche un file, anziché un terminale. I programmi non si curano della provenienza dei loro input né della destinazione dei loro output: questa è una delle proprietà che caratterizzano i sistemi UNIX.
Le shell comuni hanno una sintassi semplice per cambiare i file aperti per i flussi di I/O standard di un processo. La modifica di un file standard è chiamata *redirezione dell'I/O*.

CAPITOLO 2

COMANDI DI BASE

Una volta avviato un sistema UNIX, prima che sia disponibile l'invito della shell (*prompt*), occorre che l'utente sia riconosciuto dal sistema, attraverso la procedura di accesso (*login*). Quello che viene chiesto è l'inserimento del nome dell'utente (così come è stato registrato) e subito dopo la parola d'ordine (*password*) abbinata a quell'utente. Quindi ogni volta che si tenta di entrare nel sistema si avrà la seguente richiesta:

```
login:
Password:
```

Tale richiesta di informazioni è presente in ogni *terminale*. Generalmente un terminale è il normale mezzo di comunicazione tra l'utente ed il sistema. Senza di esso non ci sarebbe alcuna possibilità di avviare nuovi processi e, di conseguenza, nemmeno di poter compiere alcuna attività. Per questo motivo, l'attivazione di un programma (*emulatore*) per la gestione del terminale è l'ultima fase di una procedura di inizializzazione del sistema e precede immediatamente l'attivazione della procedura di login.
Un terminale generico è normalmente identificato dalla sigla `TTY`: questo perchè alle origini dei sitemi UNIX, il modo comune di interagire con un elaboratore era l'uso della telescrivente (*teletype*). Più propriamente quando si parla di terminale si intende un complesso formato da una tastiera e da un video. Il terminale principale, cioè quello che fa fisicamente parte dell'elaboratore stesso, è detto *console*.
I sistemi UNIX, consentono di gestire diversi terminali sull'unica console esistente effettivamente. Questi vengono definiti ***console virtuali***. Attraverso alcune combinazioni di tasti si riesce a passare da una console virtuale all'altra. Generalmente in un sistema normale sono presenti 6 console accessibili tramite la combinazione [`Alt+F`*n*] dove *n* è un numero da 1 a 6. Ad esempio quando è già stato fatto un accesso, si può iniziare un'altra sessione di lavoro in un'altra console premendo:

[`Alt+F2`]

In questo modo si passa alla seconda console virtuale e su questa si può eseguire un accesso differente. Le attività svolte nelle varie console virtuali sono indipendenti, come se avvenissero attraverso terminali fisicamente distinti.

Prima di utilizzare i vari comandi, è bene aver presenti alcune caratteristiche dei sistemi UNIX: il completamento automatico ed il funzionamento del file system.

Completamento automatico

Una particolarità che caratterizza i sistemi UNIX è il fatto che i nomi dei file possono essere molto lunghi. Per facilitarne la manipolazione, viene utilizzato un sistema attraverso cui la shell aiuta l'utente a completarne il nome. Questo sistema è conosciuto con il nome di **completamento automatico**.
La richiesta di completamento viene fatta attraverso l'uso del tasto [`Tab`].
Vediamo un esempio. Per prima cosa bisogna creare dei file con nomi volutamente lunghi:

```
$ touch microinterruttore[Invio]
$ touch microprocessore[Invio]
$ touch localhost[Invio]
```

Supponendo di voler utilizzare questi nomi all'interno di una riga di comando, si può essere un po' infastiditi dalla loro lunghezza. Utilizzando il completamento automatico si risolve il problema.

```
$ ls -l loc[Tab]
```

Dopo avere scritto solo `loc`, premendo il tasto [`Tab`] si ottiene il completamento del nome, dal momento che non esistono altri file o directory (nella posizione corrente) che inizino nello stesso modo. L'esempio seguente mostra lo stesso comando completato e terminato.

```
$ ls -l loc[Tab]alhost[Invio]
```

Il completamento automatico dei nomi potrebbe essere impossibile. Infatti, potrebbe non esistere alcun nome che coincida con la parte iniziale già inserita, oppure potrebbero esistere più nomi composti con lo stesso prefisso. In questo ultimo caso, il completamento si ferma al punto in cui i nomi iniziano a distinguersi.

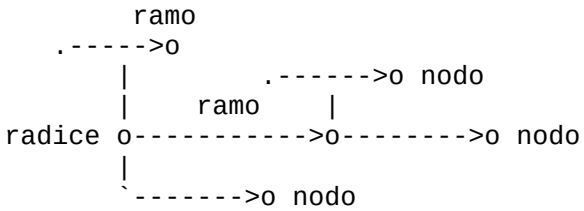
```
$ ls mic[Tab]ro
```

In questo caso, il completamento si spinge fino a `micro` che è la parte comune dei nomi `microinterruttore` e `microprocessore`. Per poter proseguire occorre aggiungere un'indicazione che permetta di distinguere tra i due nomi. Volendo selezionare il primo di questi nomi, basta aggiungere la lettera `i` e premere nuovamente il tasto [`Tab`]. L'esempio seguente rappresenta il procedimento completo.

```
$ ls mic[Tab]roi[Tab]nterruttore[Invio]
```

Nozioni sull’uso del file system UNIX

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 seguente mostra uno schema di un albero.



La radice (o *root*) è 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 è espresso da una sequenza di nomi di nodi che devono essere attraversati, separati da una barra obliqua (`/`). Il percorso `Documenti/Word/tesi` rappresenta un attraversamento dei nodi `Documenti`, Word e tesi.

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

- ***Percorso assoluto***
  Un percorso è 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, è un percorso assoluto. Per esempio, `/tesi/foto` indica un percorso assoluto che parte dalla radice per poi attraversare `tesi` e quindi raggiungere `foto`.
Un albero è un grafo orientato, nel senso che i rami hanno una direzione (archi orientati), ovvero ogni nodo ha un genitore e può 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,`../tesi` rappresenta un percorso relativo in cui si raggiunge il nodo finale, `tesi`, passando prima per il nodo genitore della posizione corrente.
In alcuni casi, per evitare equivoci, può essere utile poter identificare il nodo della posizione corrente. Il simbolo utilizzato è un punto singolo (`.`). Per cui, il percorso `Documenti/Word/tesi` è esattamente uguale a`./Documenti/Word/tesi`.

I sistemi operativi UNIX, come Linux, sono sensibili alla differenza tra le lettere maiuscole e minuscole. La differenza è sostanziale, per cui gli ipotetici file denominati: `Prova`, `pRova`, `PROVA`, ecc. sono tutti diversi. Quando si fa differenza tra maiuscole e minuscole, un sistema è definito *case sensitive* (mentre per converso, quando non si fa differenza, *case insensitive*).
Linux e UNIX sono quindi due sistemi operativi *case sensitive*.

Un albero è tale purché esista uno e un solo percorso dalla radice a un qualunque altro nodo. Nei file system UNIX non è necessariamente così; pertanto sono schematizzabili attraverso grafi orientati, ma non necessariamente degli alberi. Infatti è 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* è un collegamento che una volta creato ha lo stesso livello di importanza dei dati a cui si riferisce e *non è distinguibile* da quelli.

- ***Collegamento simbolico***, ***link simbolico***, ***symlink***
  Il collegamento simbolico, o *link* simbolico, è un file speciale contenente un riferimento a un altro percorso e quindi ad un altro nodo del grafo di directory e file.

In generale si preferisce l'uso di collegamenti simbolici. Utilizzando un collegamento simbolico si dichiara apertamente che si sta indicando una scorciatoia e non si perde di vista il percorso originale.

Una volta completata la procedura di accesso, è possibile utilizzare i comandi messi a disposizione dal sistema. Per quanto riguarda i sistemi Linux ed UNIX si farà riferimento ai comandi utilizzati da entrambi i sistemi operativi. Nel caso in cui fossero presenti dei comandi utilizzati in uno solo dei due sistemi, verrà segnalato.

2.1 cd

cd [*directory***]**
Cambia la directory corrente.

Esercitazione:

Cambiare la directory corrente, spostandosi nella directory genitrice di quella corrente
$ **cd..**

Cambiare la directory corrente, facendola diventare `/tmp/`.
$ **cd /tmp**

Cambiare la directory corrente, spostandosi nella directory personale (*home*) dell'utente.
$ **cd ~** oppure **cd**

Cambiare la directory corrente, spostandosi nella directory personale dell'utente.
$ **cd ~***nomeutente*

2.2 ls

ls [*opzioni***] [***nome...***]**
Visualizza i nomi di file o il contenuto delle directory indicate. In mancanza di questa indicazione viene visualizzato il contenuto della directory corrente e di norma non vengono inclusi i nomi di file e directory il cui nome inizia con un punto poichè questi sono considerati nascosti.
In generale, se non viene indicato diversamente, `ls` genera un elenco ordinato per colonne se lo standard output è diretto allo schermo del terminale, oppure un elenco su un'unica colonna se viene diretto altrove. Questa particolarità è molto importante per poter gestire l'output di questo programma attraverso elaborazioni successive.

Opzioni principali:

-a | `--all`
Vengono elencati anche gli elementi i cui nomi iniziano con punto (i cosiddetti file nascosti).

-A | `--almost-all`
Vengono elencati tutti gli elementi, esclusi i riferimenti alla directory stessa (`.`) e a quella genitrice (`..`).

-F | `--classify`
Se non è già la modalità di funzionamento predefinita, aggiunge un carattere alla fine dei nomi dei file, in modo da riconoscerne il tipo:

- `*` eseguibile;
- `/` directory;

@ collegamento simbolico;
| pipe con nome o FIFO;
= socket.

-i | `--inode`
Emette, alla sinistra delle indicazioni inerenti i file, il numero di inode.

-l | `--format=long` **|** `--format=verbose`
Oltre ai nomi, vengono visualizzati il tipo, i permessi, la quantità di collegamenti fisici, il nome dell'utente proprietario, il nome del gruppo, la dimensione in byte, la data di modifica.

-R | `--recursive`
Vengono elencati i contenuti di tutte le directory in modo ricorsivo.

-r | `--reverse`
Riordina in modo inverso rispetto al normale.

-S | `--sort=size`
Riordina in base alla dimensione in modo decrescente.

-t | `--time=time`
Ordina il contenuto delle directory in funzione della data: dalla più recente alla più antica. Se non viene specificato diversamente, si fa riferimento alla data di modifica.

-u | `--time=atime` **|** `--time=access` **|** `--time=use`
Ordina il contenuto delle directory utilizzando la data di accesso ai file. Se l'opzione `-u` viene usata insieme a `-t` l'elenco viene ordinato in base alla data di accesso.

-w *n_colonne* **|** `--width` *n_colonne*
Definisce la larghezza a disposizione per l'elenco. L'argomento dell'opzione si riferisce al numero di caratteri utilizzabili. Di solito, la larghezza viene determinata in funzione del numero di colonne che ha a disposizione la finestra del terminale.

-X | `--sort=extension`
Riordina in base all'estensione, cioè alla parte di nome che appare dopo l'ultimo punto. I nomi che non contengono alcun punto hanno la precedenza.

-x
Elenca i componenti della directory specificata per linee invece che per colonne

Esercitazione:

Visualizzare un elenco lungo del contenuto della directory corrente.
`$` **`ls -l`**

Spostarsi in `/etc` ed elencare tutti i file che terminano con il suffisso.`conf` che si trovano nella directory corrente.
`$` **`cd /etc`**
`$` **`ls *.conf`** `oppure` **`ls -l *.conf`**
Utilizzando quest'ultimo comando si ottiene un elenco contenente più dettagli sui file essendoci l'opzione `-l` (long listing format).

Spostarsi in `/etc`. Visualizzare il contenuto di tutte le directory in modo ricorsivo
```
$ cd /etc
$ ls –R
```
Per poter vedere ciò che compare sullo schermo è meglio utilizzare **`ls –R | more`**.

Elencare tutti i file che iniziano con "`XF`" a partire dal primo livello dopo la directory genitrice `/etc`.
```
$ cd /etc
$ ls./*/XF*
```

Stampare il numero d'indice di ogni file.
```
$ ls –i
```

Elencare il contenuto della directory corrente compresi i file che iniziano con il punto.
```
$ ls -a
```

2.3 mkdir

`mkdir [`*`opzioni`***`]`** *`directory...`*
Crea una directory.

Esercitazione:

Dalla propria directory personale, creare la directory `prova/`, come discendente di quella corrente.
```
$ cd ~
$ mkdir prova
```

Rimanendo nella propria home directory, creare la directory `prova/`, come discendente di `/miaprova/`.
```
$ mkdir /miaprova
$ cd miaprova/
$ mkdir./prova
```

2.4 rmdir

`rmdir [`*`opzioni`***`]`** *`directory...`*
Elimina le directory specificate solo se sono vuote.

Opzioni principali:

-p | `--parents`
Cancella anche le directory precedenti (padri) se, dopo la cancellazione delle directory finali, queste restano vuote.

Esercitazione:

Cancellare il percorso `~/miaprova/prova/`.
```
$ cd ~/miaprova
$ rmdir /prova
```

```
$ cd..
$ rmdir./miaprova
```

2.5 cp

cp [*opzioni***]** *origine destinazione*
Copia uno o più file in un'unica destinazione.

La copia in un sistema UNIX non funziona come nei sistemi DOS e ciò principalmente a causa di due fattori: i caratteri jolly 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 cioè generata una copia che ha il nome indicato come destinazione. Se il secondo nome indicato è una directory, il file viene copiato nella directory con lo stesso nome di origine. Se vengono indicati più 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.
Gli utenti DOS potrebbero essere abituati ad usare il comando `COPY` per copiare un gruppo di file in un altro gruppo di file con i nomi leggermente modificati, come nel seguente esempio: `COPY *.bak *.doc`. Con i sistemi UNIX, questo tipo di approccio non può funzionare.

I file elencati nell'origine potrebbero essere in realtà dei collegamenti simbolici. Se non viene specificato diversamente attraverso l'uso delle opzioni, questi vengono copiati così come se fossero file normali; cioè la copia sarà ottenuta a partire dai file originali e non si otterrà quindi una copia dei collegamenti.

Opzioni principali:

`-a`
Equivalente a **`-dpR`**, utile per l'archiviazione o comunque per la copia di collegamenti simbolici così 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.

`-p`
Mantiene le proprietà e i permessi originali.

`-r`
Copia le directory ricorsivamente, copiando tutte le non-directory e i link non simbolici (FIFO e file speciali) come se fossero file normali. In altre parole, provando a leggere dati da ciascuna sorgente e scrivendoli sulla destinazione. Perciò, con questa opzione, **cp** può impiantarsi indefinitivamente mentre legge un file FIFO o `/dev/tty`. Questo è un baco. Ciò implica che bisogna evitare `-r` e usare `-R` se non si conosce il contenuto dell'albero che si sta copiando. L'apertura di un device ignoto, uno scanner, per esempio, ha infatti effetti sconosciuti sull'hardware.

Quindi si consiglia di utilizzare l'opzione `-R` se si vuole copiare i file speciali conservando la loro natura di file speciali piuttosto che leggere al loro interno e copiarne il contenuto.

`-R`
Copia le directory ricorsivamente, rispettando le non-directory (vedi **-r** sopra)

Esempi:

Copiare il contenuto della directory `/etc/httpd/conf/` in `~/prova/` copiando anche eventuali sottodirectory contenute.
`$` **`cp -r /etc/httpd/conf/* ~/prova`**
Da notare che se `~/prova` esiste già e non si tratta di una directory, questo file viene sovrascritto, perdendo quindi il suo contenuto originale.

Ripetere l'esercizio precedente mantenendo inalterati i permessi e riproducendo i collegamenti simbolici eventuali.
`$` **`cp -dpR /etc/httpd/conf/* ~/prova`**
In questo caso il comando non mantiene inalterati i permessi perché i file considerati sono di proprietà del superutente root: è quindi una precauzione contro delle possibili intrusioni nel sistema.

2.6 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 più 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 possibilità 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.

Esercitazione:

Cambiare il nome della directory `~/prova` in `prova1`.
`$` **`mv ~/prova prova1`**

2.7 touch

`touch [`*`opzioni`***`]`** *`file...`*
Cambia la data (si intende sia la data che l'ora) di accesso e di aggiornamento dei file. Se non viene specificata una data, viene utilizzata la data e l'ora ottenuta dall'orologio del sistema nel

momento in cui viene eseguito il comando. Se si specificano file che non esistono, questi vengono creati vuoti.

Opzioni principali:

-a | `--time=atime` **|** `--time=access` **|** `--time=use`
Viene cambiata solo la data di accesso.

-c | `--no-create`
Non vengono creati i file che non esistono.

-m | `--time=mtime` **|** `--time=modify`
Cambia solo la data di aggiornamento.

-r *file_di_riferimento* **|** `--file` *file_di_riferimento*
Riproduce gli stessi dati (data ed ora) del file indicato.

-t [[*AA***]***AA***]***MMGGhhmm***[.***ss***]**
Usa l'argomento (mese, giorno, ore, minuti, secolo, anno, secondi) invece di utilizzare la data corrente.

Esercitazione:

Creare il file "`mio`" all'interno della directory personale.
`$` **`touch ~/mio`**

Cambiare la data del file "`mio`" facendola risalire al 6 settembre alle ore 12:15
`$` **`touch –t 09061215 ~/mio`**

Cambiare la data del file "`mio`" facendola risalire al 6 settembre 2001 all'ora 12:15
`$` **`touch –t 200109061215 ~/mio`**

All'interno della propria home directory, creare un nuovo file "`mio1`" e cambiare gli attributi di "`mio`" utilizzando quelli del nuovo file creato
`$` **`touch –r mio1 mio`**

2.8 more / less

Il modo più semplice con cui può essere scritto qualcosa è quello del testo puro. Questo è il caso dei file «leggimi» (*readme*) e simili, oppure di tutta quella documentazione che, per semplicità, è stata convertita anche in questo formato.
La lettura di questi file può essere fatta attraverso due programmi ben conosciuti: `more` oppure `less`. `more` è quello tradizionalmente più diffuso negli ambienti UNIX; `less` è il più pratico ed è decisamente più ricco di piccoli accorgimenti. Le funzionalità essenziali di questi due programmi sono simili, anche se il secondo offrirebbe un gran numero di funzioni aggiuntive.

La sintassi di questi due programmi è la seguente:

`more` **[***`opzioni`***]** **[***`file`***]**`...`
`less` **[***`opzioni`***]** **[***`file`***]**`...`

Solitamente le opzioni non vengono utilizzate.
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 è simile a quello utilizzato da un altro programma per leggere e modificare dei documenti: *vi*. Alcuni comandi richiedono semplicemente la pressione di uno o più 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 tabella mostra l'elenco dei comandi comuni ed essenziali di questi due programmi:

Comando	**Descrizione**
h o **H**	Richiama una breve guida dei comandi disponibili.
Spazio	Scorre il testo in avanti di una schermata.
Invio	Scorre il testo in avanti di una riga alla volta.
b	Quando possibile, scorre il testo all'indietro di una schermata.
/*modello*	Esegue una ricerca in avanti, in base all'espressione regolare indicata.
n	Ripete l'ultimo comando di ricerca.
Ctrl + l	Ripete la visualizzazione della schermata attuale.
q o **Q**	Termina l'esecuzione del programma.

La differenza fondamentale tra questi due programmi sta nella possibilità da parte di `less` di scorrere il testo all'indietro anche quando questo proviene dallo standard input, mentre per `more` non è 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. La tabella seguente mostra l'elenco di alcuni comandi aggiuntivi disponibili con `less`.

Comando	**Descrizione**
y	Scorre il testo all'indietro di una riga alla volta.
?*modello*	Esegue una ricerca all'indietro, in base all'espressione *modello* indicata.
N	Ripete l'ultimo comando di ricerca nella direzione inversa.

`less` è un programma che permette anche un utilizzo molto più complesso di quanto descritto, ma questo va oltre l'uso che se ne fa normalmente. Comunque si consiglia di sfogliare le pagine di manuale tramite il comando `man less`.

Esercitazione:

Posizionandosi in `/etc`, far scorrere sullo schermo l'elenco del contenuto della directory che probabilmente è troppo lungo per essere visualizzato senza l'aiuto di uno di questi due programmi

```
$ cd /etc
$ ls -l | more
```

oppure:

```
$ ls -l | less
```

Visualizzare sullo schermo il contenuto del file `/etc/inittab`

```
$ more /etc/inittab
$ less /etc/inittab
```

2.9 pwd

pwd

Emette il percorso assoluto della directory corrente.

2.10 ln

ln [*opzioni***]** *origine... destinazione*

Crea uno o più collegamenti di file (incluse le directory) in un'unica destinazione.
La creazione di un collegamento è 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'è 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 è 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 più file, l'ultimo nome **deve** corrispondere a una directory. È ammissibile la creazione di collegamenti che fanno riferimento ad altri collegamenti.
Si possono creare due tipi di collegamenti: 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.

Opzioni principali:

-s
Crea un collegamento simbolico.

-f
Sovrascrittura forzata dei file o dei collegamenti già esistenti nella destinazione.

Esercitazione:

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

```
$ ln -s /var/* ~/prova
```

Creare, nella destinazione `~/prova`, un collegamento simbolico corrispondente alla directory `/var`.

```
$ ln -s /var ~/prova
```

Nello stesso modo si realizza un collegamento simbolico corrispondente ad un file di nome `/var`. Notare che se `~/prova` è una directory, viene creato il collegamento `~/prova/var`; se `~/prova` non esiste, viene creato il collegamento `~/prova`.

2.11 rm

rm [*opzioni***]** *nome...*

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

Opzioni principali:

-r | `-R | --recursive`
Rimuove il contenuto delle directory in modo ricorsivo. Al contrario di `rmdir`, qui non c'è bisogno che le directory siano vuote.

-i | `--interactive`
Chiede conferma prima di cancellare.

-f | `--force`
Ignora il parametro `-i` e cancella senza chiedere conferma.

E' consigliabile utilizzare il comando `rm` insieme all'opzione `–i` in quanto non c'e' modo di recuperare i file cancellati (bisogna essere sicuri di ciò che si sta facendo).

Esercitazione:

Eliminare il file `mio` e `mio1`.
```
$ rm –i mio
$ rm –i mio1
```
oppure
```
$ rm mio*
```

Eliminare la directory `~/prova/` che risiede nella directory personale, insieme a tutte le sue sottodirectory eventuali.
```
$ rm -r ~/prova
```

Attenzione:
```
# rm -r.*
```
Elimina tutti i file e le directory a partire dalla directory radice! In pratica elimina tutto! Questo è un errore tipico di chi vuole cancellare tutte le directory nascoste (cioè quelle che iniziano con un punto) contenute nella directory corrente. Il disastro avviene perché nei sistemi UNIX, '**.***' rappresenta anche la directory corrente ('**.**') e la directory precedente o genitrice ('**..**').

2.12 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.

Esercitazione:

Andare in `/etc` e mostrare di seguito il contenuto di `inittab` e `passwd`.
```
$ cd /etc
$ cat inittab passwd |more
```

2.13 Redirezione e pipeline

La shell consente di redirigere l'output di un comando che normalmente sarebbe destinato allo schermo, oppure di inviare dati all'input di un comando, che altrimenti lo attenderebbe dalla tastiera.

Redirezione

La redirezione dirotta i dati in modo da destinarli ad un file o da prelevarli da un file.

```
$ ls -l > elenco
```

Questo comando genera il file `elenco` con il risultato dell'esecuzione di `ls`. Si può controllare il contenuto di questo file con `cat`.

```
$ cat elenco
```

Anche l'input può essere rediretto, quando il comando al quale si vuole inviare è in grado di riceverlo. `cat` è in grado di emettere ciò che riceve dallo standard input.

```
$ cat < elenco
```

Si ottiene in questo modo la visualizzazione del contenuto del file `elenco`, esattamente nello stesso modo di prima, quando questo nome veniva indicato semplicemente come argomento di `cat`. Ma adesso lo si invia attraverso lo standard input per mezzo dell'attività della shell.

La redirezione dell'output, come è stata vista finora, genera un nuovo file ogni volta, eventualmente sovrascrivendo ciò che esiste già con lo stesso nome. Sotto questo aspetto, la redirezione dell'output è fonte di possibili danni.

La redirezione dell'output può essere fatta in aggiunta, creando un file se non esiste, o aggiungendovi i dati se è già esistente attraverso l'uso di **>>**:

```
$ ls -l /tmp >> elenco
```

In tal modo viene aggiunto al file `elenco` l'elenco dettagliato del contenuto della directory `/tmp`.

Esercitazione:

Creare il file `~/prova` come risultato del concatenamento in sequenza di `/etc/hosts` e `/etc/host.conf`.

```
$ cat /etc/hosts /etc/host.conf > ~/prova
```

Aggiungere al file `~/prova` il file `/etc/hosts.allow`.

```
$ cat /etc/hosts.allow >> ~/prova
```

Pipeline

La pipeline è una forma di redirezione in cui la shell invia l'output di un comando come input del successivo.

```
$ cat elenco | sort
```

In questo modo, `cat` legge il contenuto del file `elenco`, ma invece di essere visualizzato sullo schermo, viene inviato dalla shell come standard input di `sort` che lo riordina e poi lo emette sullo schermo.

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

Qui invece l'output del comando `ls -l` è inviato alla shell come input di `more` o `less`.
Una pipeline può utilizzare anche la redirezione, per cui, il comando visto precedentemente può essere trasformato nel modo seguente:

```
$ cat < elenco | sort
```

2.14 Utilizzo del manuale

Sapere utilizzare il manuale presente nei sistemi UNIX è fondamentale per superare tutti quei problemi che si possono incontrare quando non si possiede ancora una padronanza adeguata dei i vari comandi. Infatti il manuale è un prezioso archivio contenete la sintassi dei vari comandi. E' per questo che ne viene fatto un uso intenso anche da parte degli utenti esperti: non a caso il comando `man` è il più utilizzato dalla comunità degli utenti UNIX.

man

`man [`*`opzioni`***`]`** *`nome...`*

Questo è uno dei comandi più utilizzati in assoluto in Linux e in UNIX. Infatti 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.

`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 seguente:

Sezione	Contenuto	Sezione	Contenuto
1	comandi utente	**6**	giochi
2	chiamate di sistema	**7**	varie
3	chiamate di libreria	**8**	comandi di sistema
4	dispositivi	**9**	routine del kernel
5	formati dei file		

Si distinguono diverse sezioni di queste pagine di manuale, a seconda del genere di informazioni in esse contenute. Può infatti accadere che esistano più pagine con lo stesso nome, appartenenti però a sezioni diverse.

Spesso, nella documentazione, quando si vuole fare riferimento ad una pagina di manuale, si indica il nome di un comando seguito da un numero tra parentesi, che ne esprime la sezione. Per cui, *`nomecomando(1)`* indica la pagina di manuale che si riferisce al comando `nomecomando` presente nella prima sezione. Nella pratica *`nomecomando(1)`* corrisponde a digitare sul terminale `man 1 nomecomando`.

Opzioni principali:

`-f`
Si comporta come `whatis`.

`-h`
Visualizza una breve guida di se stesso.

`-k`
Equivalente a `apropos`.

Esercitazione:

Visualizzare la pagina di manuale del programma `ls`.
`$ man ls`

Visualizzare la pagina di manuale nell'ottava sezione, del programma `lilo`.
`$ man 8 lilo`

Visualizzare la pagina di manuale riguardante l'utilizzo del programma `man`.
`$ man man`

whatis

`whatis` *`parola...`*
Cerca all'interno degli elenchi *whatis* una o più parole intere. Questi elenchi sono dei file con lo stesso nome (`whatis`) contenenti una breve descrizione dei comandi di sistema e collocati nelle varie directory *`mandir/`*. Il risultato della ricerca viene emesso attraverso lo standard output. Sono visualizzate solo le corrispondenze con parole intere.

Esercitazione:

Visualizzare le righe degli elenchi *whatis* contenenti la parola "`ls`".
`$ whatis ls`

apropos

`apropos` *`stringa...`*
Cerca all'interno degli elenchi *whatis* una o più stringhe. `apropos` esegue la ricerca negli stessi file utilizzati da `whatis`. Il risultato della ricerca viene emesso attraverso lo standard output. A differenza di `whatis`, sono visualizzate tutte le corrispondenze.

Esercitazione:

Visualizzare le righe degli elenchi *whatis* contenenti la stringa "`keyboard`".
`$ apropos keyboard`

2.15 Arresto e riavvio del sistema

Qualunque sistema operativo in multiprogrammazione, tanto più se anche multiutente, deve prevedere una procedura di arresto del sistema che si occupi di chiudere tutte le attività in corso prima di consentire lo spegnimento fisico dell'elaboratore.
I sistemi UNIX permettono solo all'**`utente root`** di avviare la procedura di arresto del sistema con il comando seguente:

```
# shutdown -h now
```

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

```
# shutdown -r now
```

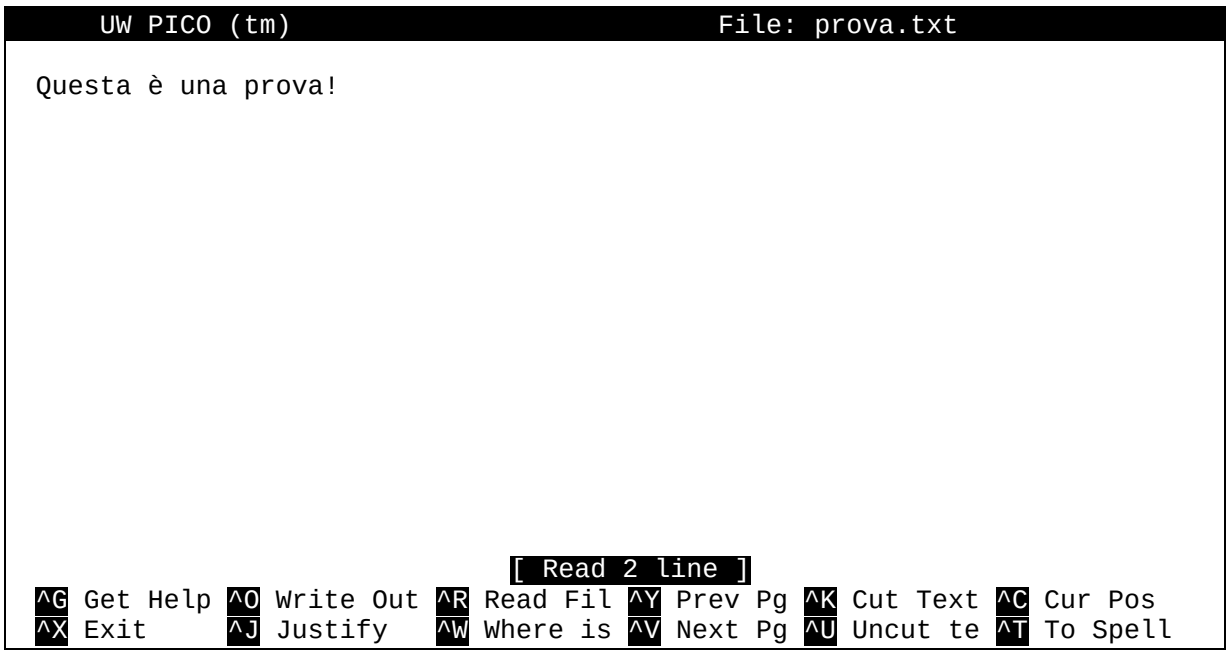
Generalmente, l'unico modo per un **utente comune** di spegnere il sistema, è quello di riavviare attraverso la combinazione di tasti [Ctrl+Alt+Canc]. Non è elegante, ma è il modo migliore per risolvere il problema.

CAPITOLO 3

USO DI UN EDITOR

3.1 Pico

Pico è un editor assai limitato rispetto ad Emacs e *vi* ma molto intuitivo nell'uso. Si avvia semplicemente eseguendo il comando `pico` seguito dal nome del file che si vuole editare.
Per la sua semplicità, viene utilizzato anche dal programma `pine` come editor per la compilazione dei messaggi di posta elettronica.
Ciò che distingue Pico dagli altri programmi di scrittura, è sicuramente il fatto che tutti i comandi che è possibile impostare vengono riportati sinteticamente in una specie di menù visualizzato nelle ultime due righe della finestra di terminale. Ogni comando viene impostato premendo **`CTRL`** insieme ad un altro tasto. A differenza dell'editor Emacs, Pico dispone di un unico `buffer` e quindi può operare su un unico file alla volta.



Utilizzo:

`pico [`*`nomefile_da_modificare`***`]`**

Si aprirà una rudimentale interfaccia grafica nella cui parte inferiore è presente una sorta di barra dei comandi.
E' sufficiente premere il tasto ***`CTRL`*** seguito da una delle lettere indicate per accedere alle numerose funzioni che possiede:

- **`CTRL+X`** una volta digitato permette di uscire dal programma. Se il file è stato modificato, chiede se si vuole salvarlo e con che nome (*exit*);
- **`CTRL+O`** salva il file senza uscire dal programma (o meglio salva su file il testo contenuto nel buffer) (*write out*);
- **`CTRL+V`** ad ogni pressione di questi due tasti scrolla verso il basso di una schermata;
- **`CTRL+Y`** ad ogni pressione di questi due tasti scrolla verso l'alto di una schermata;

- **CTRL+J** “Giustifica”, riallineando i margini destro e sinistro delle righe, il paragrafo corrente (*justify*). Per ritornare al testo non giustificato premere CTRL+U;

- **CTRL+R** inserisce nel punto in cui è posizionato il cursore il contenuto di un altro file, chiedendone il nome (*read file*);

- **CTRL+W** cerca una parola specifica (*Where is*). Il cursore si posiziona sul punto in cui si trova la parola. Premendo nuovamente la stessa combinazione di tasti, il programma ricorda l’ultima parola appena cercata. In questo caso è sufficiente premere Invio per trovare quella successiva;

- **CTRL+G** permette di accedere all’Help in linea che spiega il funzionamento del programma;

- **CTRL+K** taglia (cancellandole) le righe a partire da quella su cui è posizionato il cursore (*cut text*);

- **CTRL+U** incolla il testo nel punto in cui è posizionato il cursore;

- **CTRL+C** fornisce alcune statistiche relative alla riga su cui è posizionato il cursore, come il numero della riga, il numero del carattere e le loro percentuali rispetto al totale;

- **CTRL+SPACE** premendo questi due tasti in sequenza il cursore si sposta all’inizio di ogni parola, in rapida successione;

- **CTRL+T** permette di accedere ad un sottomenu di *spell checking*.

Per cancellare o spostare un intero blocco di testo è possibile utilizzare il comando di “selezione estesa”: ci si posiziona all'inizio del testo da selezionare e si preme **CTRL-^**; muovendosi poi con il cursore, si estende la selezione: il testo marcato viene visualizzato in *reverse* (testo nero, sfondo bianco). A questo punto è possibile tagliare l'intera selezione con il comando **CTRL-k** e poi eventualmente incollarlo (con **CTRL-u**) nella nuova posizione del cursore.
Pico non offre altre funzionalità avanzate: quindi solo lo stretto indispensabile per scrivere un breve file di testo (come un messaggio di posta elettronica, ad esempio) e poco altro. Se si dovessero avere la necessità di eseguire operazioni più complesse sul file, bisognerà utilizzare editor più potenti (e complessi) come Emacs e *vi*.

3.2 Emacs

Emacs è forse un editor più sofisticato, ma altrettanto complesso e potente del famoso editor *vi*. Supporta un linguaggio per la compilazione di *macro* molto simile al *Lisp*, un linguaggio funzionale che opera su liste. Questo linguaggio di macro è potente al punto che sono stati sviluppati dei veri programmi che girano all'interno dell'editor Emacs.
Come struttura il programma si avvicina maggiormente agli editor tradizionali: in questo caso non c'è la doppia modalità inserimento/comandi, ma tutto può essere fatto in un unico ambiente tramite semplici sequenze di tasti.
Esiste anche una versione grafica di Emacs che permette di lavorare in tutta comodità su un terminale grafico, utilizzando per i comandi, invece delle combinazioni di tasti, i consueti menù a tendina selezionabili con il mouse. Spesso Emacs è utilizzato proprio in questa modalità grafica guidata dai menù, che quindi non richiede ulteriori spiegazioni.
Di seguito verranno descritti i principali comandi che devono essere impartiti da tastiera nel caso si utilizzi il programma su un terminale alfanumerico.

```
Buffer Files Tools Edit Search Mule Help
Welcome to GNU Emacs, one component of a Linux-based GNU system.

Get help          C-h  (Hold down CTRL and press h)
Undo Changes      C-x u        Exit Emacs             C-x C-c
Get a tutorial    C-h t        Use Info to read docs  C-h i
Activate menubar  F10 or ESC ‘ or M-‘
(‘C-‘ means use the CTRL key. ‘M-‘ means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the char)

GNU Emacs 20.7.1 (i386-redhat-linux-gnu, X toolkit)
 of Mon Jul 30 2001 on stripples.devel.redhat.com
Copyright (C) 1999 Free software Foundation, Inc.
...

--1-:---F1  *scratch*          (Lisp interaction)--L5--All---------------
For information about the GNU project and its goals, type C-h C-p.
```

Utilizzo:

emacs [*nomefile_da_modificare***]**

Una semplice finestra di lavoro di Emacs contiene generalmente: una barra dei menù, un buffer, una barra di stato ed un Mini-buffer.

Barra dei menù
E’ la linea più in alto presente in una interfaccia Emacs.

Buffer (principale)
E’ l’area principale di Emacs, dove viene caricato e modificato un file. Generalmente è sotto la barra dei menù.

La barra di stato (Status Bar) e il Mini-buffer
Delle ultime due linee in basso nell'interfaccia Emacs, quella superiore è essenzialmente una barra di stato. Contiene informazioni sul buffer in cui si sta lavorando, in quale modalità (*mode*) si trova Emacs e varie altre cose. La linea inferiore è chiamata **mini-buffer**. È separato dal buffer principale dalla barra di stato di cui si è appena detto. Si può identificare il mini-buffer come la "riga di comando" (*command-line*) di Emacs. È dove appaiono i comandi che vengono impartiti ad Emacs ed è dove vengono visualizzati i messaggi di stato in risposta a quello che si sta facendo.
Nella documentazione allegata a Emacs si fa normalmente riferimento alla barra di stato come linea della modalità (*mode line*). È dove Emacs mostra informazioni relative alla modalità corrente che si sta utilizzando o si può utilizzare ed altri dettagli tipo la data e l'ora corrente, il numero di riga e la dimensione del file.

Concetti di base

Buffer e File
Quando si apre un file in Emacs questo non rimane "aperto" tutto il tempo in cui si lavora con esso. Al contrario, Emacs legge il file in un **buffer** in memoria. Mentre si sta editando il buffer e lavorando con i dati, niente è cambiato sul disco. Solo quando di fatto si salva il buffer, allora il file sul disco viene aggiornato.
Quindi il termine "buffer" indica "*una copia del file che si trova attualmente in memoria*". E' importante notare che un buffer non deve sempre essere riferito ad uno specifico file sul disco. Spesso Emacs crea dei buffer come risposta ad alcuni comandi che vengono lanciati. Questi buffer potranno contenere il risultato dei comandi stessi, una lista di selezioni da cui scegliere e così via.

Point e Region (Punto e Regione)
Nel gergo di Emacs, si fa riferimento al **point**. In termini generali il point è il cursore. In realtà c'è una piccola differenza. Il cursore è la rappresentazione visiva del point. Il cursore è sempre "*su*" una particolare posizione del carattere nel buffer corrente. Il point, invece, vive nello spazio *fra i caratteri* che si trovano nel buffer. Quindi si può affermare che se il cursore si trova sulla lettera "h" nella parola "the" allora il point sarà tra la "t" e la "h".
Come molti editor moderni, Emacs permette di effettuare operazioni (taglia, copia, incolla...) su una porzione del buffer corrente. E' possibile evidenziare (o "marcare") un blocco di testo usando la tastiera o il mouse e poi eseguire operazioni solo sul blocco selezionato di testo. In Emacs, quel blocco di testo è chiamato **region** (regione).

Finestre
Una **finestra** in Emacs è una particolare area dello schermo nel quale è visualizzato un buffer. Ad esempio quando Emacs viene avviato per la prima volta, quella che compare sullo schermo è proprio una finestra. Alcune funzioni di Emacs (tipo l'help e la documentazione) spesso aprono (temporaneamente) una finestra aggiuntiva nella schermata corrente di Emacs.

Utilizzare Emacs

Tasti di comando (Meta, Esc, Control e Alt)
Emacs fa un uso intensivo di combinazioni di più tasti. Al contrario di *vi*, Emacs non è un editor modale e quindi non bisogna preoccuparsi di essere in "modalità comandi" o "modalità inserimento" prima di provare a muovere il cursore o eseguire un comando. In Emacs bisogna soltanto premere una combinazione di tasti.
I tasti di cui Emacs fa maggiore uso sono normalmente abbreviati nella documentazione come `C` (per Control o Ctrl) e `M` per (Meta). Mentre le normali tastiere hanno uno o più tasti etichettati come `Ctrl`, poche ne hanno uno etichettato come `Meta`. In pratica si può sostituire al tasto `Meta` il tasto `Alt` o, in mancanza di questo, il tasto `Esc`. Nella maggior parte delle configurazioni standard, entrambi, `Alt` e `Esc`, faranno essenzialmente le stesse cose.
Quindi quando si trova un riferimento, in qualsiasi documentazione relativa a Emacs, a `C-x f` significa "*premere Ctrl-x e poi f*". Inoltre se si trova un riferimento a qualcosa del tipo `M-x shell` significa "*premere Alt-x e digitare la parola shell*".
Un comando veramente utile per chi inizia è `M-x apropos` o `C-h a`. `apropos` cercherà nella documentazione in linea di Emacs tutte le funzioni e cercherà l'espressione regolare che verrà digitata. Ad esempio, per scoprire tutti i comandi relativi a un frame basta semplicemente digitate `C-h a` e poi `frame`.

Muoversi in un Buffer
Quella che segue è una lista delle combinazioni di tasti più comuni per muoversi in un buffer:

Tasti	Azione
C-p	Indietro di una riga
C-n	Avanti di una riga
C-f	Avanti di un carattere
C-b	Indietro di un carattere
C-a	Inizio di una riga
C-e	Fine di una riga
C-v	Avanti di una schermata
M-v	Indietro di una schermata
M-f	Avanti di una parola
M-b	Indietro di una parola
M-<	Inizio del buffer
M->	Fine del buffer
C-g	Cancella l'operazione corrente

Da notare che i tasti cursore (o tasti freccia) funzionano normalmente senza utilizzare comandi particolari. Invece, a seconda della versione Emacs posseduta, il tasto `Backspace` potrebbe anche non funzionare.

Comandi essenziali
Prima di vedere alcuni comandi di base bisogna descrivere il modo in cui questi lavorano.
Tutti i "tasti di comando" in Emacs (quelli che sono `M-x` *`qualcosa`* o `C-`*`qualcosa`*) sono di fatto delle scorciatoie a delle funzioni che fanno parte di Emacs. Si può chiamare una qualsiasi di queste funzioni digitando `M-x` *`nomefunzione`* e premendo Invio. Per quelle funzioni che ne possiedono una, si può anche usare la scorciatoia da tastiera. Ad esempio, la funzione di Emacs che salva un buffer su disco è chiamata `save-buffer`. Per default è anche

vincolata a `C-x C-s`. Quindi, si può usare sia la scorciatoia da tastiera per salvare il buffer corrente, sia digitare `M-x save-buffer` per raggiungere esattamente lo stesso risultato.
Tutte le funzioni più comuni hanno delle scorciatoie da tastiera per default. Alcune di esse sono elencate di seguito:

Tasti	**Funzione**	**Descrizione**
C-x C-f	find-file	Apre un file dal disco
C-x C-s	save-buffer	Salva il buffer corrente su disco
C-x u	undo	Annulla l'ultima operazione
C-x C-c		Termina l'esecuzione del programma
C-s	isearch-forward	Cerca avanti una stringa
C-r	isearch-backward	Cerca indietro una stringa
	replace-string	Cerca e rimpiazza una stringa
	replace-regexp	Cerca e rimpiazza usando regexp
C-h t	help-with-tutorial	Usa la guida interattiva
C-h f	describe-function	Mostra aiuto per una funzione
C-h v	describe-variable	Mostra aiuto per una variabile
C-h x	describe-key	Mostra che cosa fa una sequenza di tasti
C-h a	apropos	Cerca aiuto per una stringa/regexp
C-h F	view-emacs-FAQ	Mostra le FAQ di Emacs
C-h i	info	Legge la documentazione di Emacs
C-x r m	bookmark-set	Imposta un segnalibro. Utile nelle ricerche
C-x r b	bookmark-jump	Salta ad un segnalibro.

La maggior parte di queste funzioni, chiedono di inserire qualcosa al prompt: questa richiesta viene fatta sempre nel mini-buffer.
Emacs ha centinaia di funzioni incorporate disponibili. La lista riportata sopra è un campione minimo che rappresenta quelle che di solito vengono più utilizzate. Per approfondimenti si consiglia di utilizzare l'aiuto in linea.

Navigazione tra finestre
Ogni volta che si usa il comando `C-x C-f` per aprire un nuovo file, questo viene caricato in un buffer: i buffer precedentemente caricati non verranno eliminati. Quindi si avranno più buffer aperti e distinti, ognuno contenente il testo di un file.
Si può passare da un buffer all'altro utilizzando le seguenti combinazioni di tasti che gestiscono le finestre di Emacs:

Tasti	**Descrizione**
C-x C-b	Elenca i buffers attivi;
C-x C-f *nomebuffer*	Se il buffer è già stato caricato, lo visualizza nella finestra attiva, altrimenti prima carica il file con quel nome, lo inserisce in un nuovo buffer e lo visualizza nella finestra attiva;
C-x 2	Divide lo schermo a metà creando una seconda finestra;
C-x o	Se lo schermo visualizza più di una finestra (attivata con C-x 2) passa da una finestra all'altra (*other*);
C-x 1	Visualizza sullo schermo solo la finestra attiva (quella in cui si trova il cursore).

Completamento con Tab

Come la maggior parte delle shell UNIX, Emacs offre il completamento automatico del comando tramite il tasto `Tab`. Da notare che il completamento del comando nella shell Bash venne preso a modello dopo che era stato utilizzato in Emacs.

Come esempio, provare `M-x search` e poi premere `Tab`. Emacs aggiungerà un trattino per indicare che ci sono molti possibili completamenti ma questi hanno tutti un trattino come carattere successivo. Se si preme `Tab` ancora una volta, Emacs mostrerà una lista delle possibili corrispondenze da cui poter scegliere. Notare che farà questo in una *nuova finestra*. Temporaneamente dividerà lo schermo in due finestre: una che contiene il buffer che si sta editando e l'altra che contiene la lista dei possibili completamenti per "`search-`". Si può premere `C-g` per uscire fuori dal processo di selezione e chiudere la nuova finestra.

Tutorial, Aiuto e Info

Emacs è dotato di un potente sistema di *help on-line* e di un *tutorial*, una sorta di breve corso guidato sulle funzioni principali del programma. Per accedere all'help, ed avere istruzioni sull'uso di un certo comando, basta digitare `C-h c` e poi la sequenza di caratteri su cui si vuole un aiuto. Ad esempio:

```
>> C-h c C-p
C-p runs the command previous-line
```

Per avere informazioni più complete su una certa sequenza di tasti si può usare il comando `C-h k` seguito dal comando di cui si vogliono le informazioni. Emacs visualizza il testo di help in una finestra separata, quindi per ripristinare la finestra su cui si stava lavorando, eliminando l'help, bisogna digitare `C-x 1` (passaggio alla prima finestra creata).

Si raccomanda di consultare a fondo il tutorial per imparare ad utilizzare Emacs. Si può entrare nel tutorial tramite `C-h t`. Il tutorial è una auto-guida e aiuta le persone che hanno appena iniziato con Emacs. Inoltre, per vedere l'intero volume della documentazione disponibile per Emacs, bisogna digitare `M-x info` o `C-h i` che lancia Info, il browser per la documentazione di Emacs.

CAPITOLO 4

SCRIPT DI COMANDI

Col termine shell script (o *shell program*) si indicano generalmente file di comandi del Sistema Operativo, scritti nel linguaggio di shell, così da risultare veri e propri programmi. Sono principalmente usati per semplificare operazioni ripetitive, per rimpiazzare con un unico comando due o più comandi da eseguirsi insieme e per scrivere semplici applicazioni interattive. Uno shell program si puo' creare come un semplice file, usando un qualsiasi text editor.

4.1 Introduzione

I vari interpreti dei comandi che si possono trovare su un sistema UNIX (sh, csh, tcsh, ksh, bsh) sono molto simili fra loro; inoltre tutti funzionano anche come linguaggi di programmazione. Infatti la maggior parte deriva dalla `sh` (shell), il primo interprete scritto per il sistema UNIX che si trova in qualsiasi macchina UNIX-like nella directory `/bin`.
Nel seguito della trattazione si farà riferimento alla shell più utilizzata negli ambienti UNIX, la shell *Bash* (*Bourne Again SHell*).

Una sequenza di comandi shell puó essere data direttamente al terminale una riga dopo l'altra:

```
$ ls -l
$ date
$ cp fileA fileB
```

oppure scritta su un unica riga:

```
$ ls -l; date; cp fileA fileB
```

oppure inserita in un file di comandi, che viene detto *`Shell SCRIPT`*.

Uno shell script puó essere eseguito con:

```
$ sh scriptfile
```

oppure puó essere trasformato in un nuovo comando del sistema rendendolo eseguibile:

```
$ chmod a+x scriptfile
$ scriptfile
```

In ogni caso, il contenuto del file *`scriptfile`* viene interpretato usando la shell. Se si vuole che sia interpretato usando la Bourne shell sh bisogna aggiungere, come prima riga del file:

```
#!/bin/sh
```

Invece per la Bash shell bisogna aggiungere:

```
#!/bin/bash
```

La stessa cosa vale se si usa la shell come linguaggio di programmazione. Si puó lanciare un programma shell in modo interattivo:

```
$ cd ~
$ sh  /* entra in ambiente shell */
sh-2.05$ for i in $( ls )
> do
> echo item: $i
> done
item: autosave
item: Desktop
....
sh-2.05$ exit  /* esce dalla shell e torna in csh */
```

Oppure si può scrivere uno shell script e lanciarlo (dopo averlo reso eseguibile):

```
$ pico myscript
      #!/bin/bash
      for i in $( ls )
      do
      echo item: $i
      done
$ chmod a+x myscript  /*rende eseguibile il file di script!*/
$./myscript
item: autosave
item: Desktop
....
```

Sulla seconda riga del file script, `i` è la variabile che prenderà i differenti valori contenuti in `$(ls)`. La terza riga potrebbe essere più lunga se necessario, o ci potrebbero essere più righe prima del `done`. `done` indica che il codice che ha utilizzato il valore di `$i` è terminato e `$i` può ricevere un nuovo valore.
Quindi questo script prende gli *item* (elementi) di uscita del comando `ls` e li fa vedere preceduti da `item:`.
Per eseguire lo script al di fuori della propria home directory bisogna digitare il percorso in cui si trova lo script preceduto da`/`:

```
$ cd /
$./home/nomeutente/myscript
item: bin
item: boot
item: dev
…
```

Quindi si vede che lo script agirà, facendo un `ls`, sul percorso `/` (*root*), cioè sulla directory corrente.

La shell usa la variabile *path* settata nel file`.bash_profile` (a seconda della shell installata) presente in ogni directory utente per stabilire in quale directory cercare i comandi lanciati dall'utente stesso. Ogni utente può definire una propria directory di comandi, e poi modificare la variabile *path* in modo da lanciare i propri comandi senza specificare ogni volta il loro pathname (cioé proprio come se fossero un normale comando UNIX). Di solito la directory predefinita è `~/bin`: senza cambiare il file`.bash_profile`, si può creare la directory `~/bin` e metterci dentro i file di script.
In fondo al file`.bash_profile` è presente la riga:

```
PATH=$PATH:$HOME/bin  /* aggiunge a path il nuovo pathname */
```

Per modificare `PATH` basta aggiungere dopo `HOME/` la directory dove si devono mettere i file script (in questo caso la directory predefinita è `~/bin`).
Eseguire poi il comando:

```
source.bash_profile  /* rilegge il file.bash_profile */
```

Ora i comandi dentro la directory specificata possono essere lanciati direttamente, come un normale comando UNIX.

4.2 Variabili d'ambiente

La shell conosce un solo tipo di variabili: stringhe di caratteri.
Il nome di una variabile shell puó essere una qualsiasi sequenza di caratteri senza spazio e metacaratteri non quotati. Lo stesso ambiente shell definisce ed usa un insieme di variabili per gestire correttamente la connessione dell'utente.

Eseguendo il comando **set** si possono vedere tutte le variabili correntemente definite nell'ambiente di lavoro. Per definire una nuova variabile shell basta assegnarle un valore:

```
$ myvar=pippo /* niente spazi! */
```

Per utilizzare il valore di una variabile definita occorre anteporle il simbolo **$**:

```
$ echo $myvar
pippo
$
```

Per cancellare una variabile basta digitare: **unset** *nomevar*.

Assegnazione variabili da terminale

Creare lo script `myscript` ed eseguirlo:

```
#!/bin/bash
echo "inserire nome e cognome:"
read nome cognome
echo "nome inserito = " $nome
echo "cognome inserito = " $cognome
$ myscript
inserire nome e cognome:
Mario Rossi
nome inserito = Mario
cognome inserito = Rossi
$
```

Le variabili di shell

Il valore di una variabile si usa direttamente:

```
$ myvar1=/etc/passwd
$ ls -l $myvar1
r-xr-xr-x    root   4570    /etc/passwd
```

Si puó concatenare ad un altro:

```
$ myvar2=.old
$ ls -l $myvar1$myvar2
 r-xr-xr-x    root   2620    /etc/passwd.old
```

Si puó concatenare una variabile ad una stringa qualsiasi racchiudendo il nome della variabile tra graffe o terminandolo con il back slash \:

```
$ myvar1=/etc/pass
$ ls -l ${myvar1}wd
r-xr-xr-x    root   4570    /etc/passwd
```

oppure:

```
$ ls -l $myvar1\wd.old
r-xr-xr-x    root   2620    /etc/passwd.old
```

Non si può però concatenare una variabile ad una stringa qualsiasi *direttamente*, perché quello diventa un nuovo nome di variabile non definita:

```
$ myvar1=/etc/pass
$ more $myvar1wd
usage: more [...] [...]  /*ERRORE nell'uso del more */
```

Le variabili non definite hanno il valore Null.

Alcune variabili predefinite

- **`$#`**: numero di argomenti passati ad uno script shell
- **`$*`** o **`$@`**: tutti gli argomenti passati
- **`$$`**: PID della shell corrispondente
- **`$?`**: valore restituito dall'ultimo comando eseguito
- **`$!`**: Identificativo di elaborazione (PID) dell'ultimo comando iniziato con `&`
- **`$-`**: opzioni fornite all'interprete
- **`$0`**: nome dello script shell
- **`$`***n*: valore dell'ennesimo argomento passato allo script (*n* va da 0 a 9)
- **`$HOME`**: home directory dell'utente
- **`$PWD`**: current working directory
- **`$PATH`**: elenco delle directory in cui cercare i comandi
- **`$PS1`**: stringa del simbolo che segnala il prompt (per default il segno è `$`)
- **`$PS2`**: stringa del simbolo che segnala che la riga di comando continua (il default è >)

Script di esempio (file `myscript1`):

```
      #!/bin/bash
      echo “questo script si chiama “ $0 “ ed ha PID = “ $$
      echo “é stato chiamato con “ $# “argomenti”
      echo “il secondo argomento é “ $2
      echo “é stato lanciato nella directory “ $PWD

$ sh myscript1 prova 1 2
      questo script si chiama myscript1 ed ha PID 1178
      é stato chiamato con 3 argomenti
      il secondo argomento é 1
      é stato lanciato nella directory /home/usename/
```

4.3 Strutture

Per la formulazione di comandi complessi si possono usare le strutture di controllo e di iterazione tipiche dei linguaggi di programmazione più comuni. Queste strutture sono particolarmente indicate per la preparazione di script di shell, ma possono essere usate anche nella riga di comando di una shell interattiva.
È importante ricordare che il punto e virgola singolo (`;`) viene utilizzato per indicare una separazione e può essere rimpiazzato da uno o più codici di interruzione di riga.

for

Il comando `for` esegue una scansione di elementi e in corrispondenza di questi esegue una lista di comandi.

```
for variabile [in valore...]
do
    lista_di_comandi
done
```

L'elenco di parole che segue la sigla `in` viene espanso, generando una lista di elementi; la variabile indicata dopo `for` viene posta, di volta in volta, al valore di ciascun elemento di questa lista; infine, la lista di comandi che segue `do` viene eseguita ogni volta (una volta per ogni valore disponibile). Se la sigla `in` (e i suoi argomenti) viene omessa, il comando `for` esegue la lista di comandi (`do`) una volta per ogni parametro posizionale esistente (`$1`, `$1`,...). In pratica è come se fosse stato usato: `in $@`.
Il valore restituito da `for` è quello dell'ultimo comando eseguito all'interno della lista `do`, oppure zero se nessun comando è stato eseguito.

Esempi:

L'esempio seguente mostra uno script che, una volta eseguito, emette in sequenza gli argomenti che gli sono stati forniti.

```
#!/bin/bash
for i in $*
do
   echo $i
done
```

L'esempio seguente mostra uno script un po' più complicato che si occupa di archiviare ogni file e directory indicati come argomenti.

```
#!/bin/bash
ELENCO_DA_ARCHIVIARE=$*
for DA_ARCHIVIARE in $ELENCO_DA_ARCHIVIARE
do
    tar czvf ${DA_ARCHIVIARE}.tgz $DA_ARCHIVIARE
done
```

Creare il file `archivio` e scriverci in colonna i nomi seguenti:

```
ciao
ciao1
miaprova
```

A questo punto si può fare uno script per cercare le occorrenze della parola `ciao` tramite l'uso del comando `grep` (file `script2`):

```
#!/bin/bash
for i
do
    grep $i archivio
done
```

if

Il comando `if` permette di eseguire liste di comandi differenti, in funzione di una o più condizioni, espresse anch'esse in forma di lista di comandi.

```
if lista_condizione
then
    lista_di_comandi
[elif lista_condizione
then
    lista_di_comandi]
...
[else
    lista_di_comandi]
fi
```

Inizialmente viene eseguita la lista che segue `if` che costituisce la condizione. Se il valore restituito da questa lista è zero (cioè *Vero*), allora viene eseguita la lista seguente `then` e il comando termina. Altrimenti viene eseguita ogni `elif` in sequenza, fino a che ne viene trovata una la cui condizione si verifica. Se nessuna condizione si verifica, viene eseguita la lista che segue `else`, sempre che esista.
Il valore restituito è quello dell'ultimo comando eseguito, oppure zero se non ne è stato eseguito alcuno.

Esempi:

L'esempio seguente mostra uno script che fa apparire un messaggio di avvertimento se non è stato utilizzato alcun argomento, altrimenti si limita a visualizzarli.

```
#!/bin/bash
if [ $# = 0 ]
then
    echo "devi fornire almeno un argomento"
else
    echo $*
fi
```

L'esempio seguente mostra uno script attraverso il quale si tenta di creare una directory e se l'operazione fallisce viene emessa una segnalazione di errore.

```
#!/bin/bash
if ! mkdir deposito
then
    echo "Non è stato possibile creare la directory \"deposito\""
else
    echo "È stata creata la directory \"deposito\""
fi
```

while

Il comando `while` permette di eseguire un gruppo di comandi in modo ripetitivo fintanto che una certa condizione è *Vera*.

```
while lista_condizione
do
    lista_di_comandi
done
```

Il comando `while` esegue ripetitivamente la lista che segue `do` finché la lista che rappresenta la condizione continua a restituire il valore zero (*Vero*). Il valore restituito dal comando è lo stesso di quello della lista che segue `do`, oppure zero se la condizione non si è mai verificata.

Esempi:

Lo script dell'esempio seguente contiene un ciclo perpetuo, in cui viene richiesto di inserire qualcosa, ma solo se si inserisce la stringa `fine` si conclude l'iterazione.

```
#!/bin/bash
RISPOSTA="continua"
while [ $RISPOSTA != "fine" ]
do
    echo "usa la parola fine per terminare"
    read RISPOSTA
done
```

All'interno dei comandi composti si utilizzano spesso delle condizioni racchiuse tra parentesi quadre. L'uso delle parentesi quadre è una forma abbreviata del comando interno `test`:

```
test espressione_condizionale
[ espressione_condizionale ]
```

`test` risolve (valuta) l'espressione indicata (la seconda forma utilizza semplicemente un'espressione racchiusa tra parentesi quadre). Il valore restituito può essere *Vero* (corrispondente a zero) o *Falso* (corrispondente a uno) ed è pari al risultato della valutazione dell'espressione. Le espressioni possono essere unarie o binarie. Le espressioni unarie sono usate spesso per esaminare lo stato di un file. Vi sono operatori su stringa e anche operatori di comparazione numerica. Ogni operatore e operando deve essere un argomento separato.

Se si usa la forma tra parentesi quadra, è indispensabile che queste siano spaziate dall'espressione da valutare.

Per ulteriori approfondimenti si rimanda alle pagine di manuale.

until

Il comando `until` permette di eseguire un gruppo di comandi in modo ripetitivo fintanto che una certa condizione è *Falsa*.

```
until lista_condizione
do
    lista_di_comandi
done
```

Il comando `until` è analogo a `while`, cambia solo l'interpretazione della lista che rappresenta la condizione nel senso che il risultato di questa viene invertito (negazione logica).

case

Il comando `case` permette di eseguire una scelta nell'esecuzione di varie liste di comandi. La scelta viene fatta confrontando una parola (di solito una variabile) con una serie di modelli. Se viene trovata una corrispondenza con uno dei modelli, la lista di comandi relativa viene eseguita.

```
case parola in
    [modello [ | modello]... ) lista_di_comandi;; ]
...
    [*) lista_di_comandi;; ]
esac
```

La parola che segue `case` viene espansa e quindi confrontata con ognuno dei modelli, usando le stesse regole dell'espansione di percorso (i nomi dei file). La barra verticale (`|`) viene usata per separare i modelli quando questi rappresentano possibilità diverse di un'unica scelta.
Quando viene trovata una corrispondenza, viene eseguita la lista di comandi corrispondente. Dopo il primo confronto riuscito, non ne vengono controllati altri dei successivi. L'ultimo modello può essere `*)`, corrispondente a qualunque valore, che si può usare come alternativa finale in mancanza di altro.
Il valore restituito è zero se nessun modello combacia. Altrimenti, è lo stesso valore restituito dall'ultimo comando eseguito, contenuto all'interno della lista.

Esempi:

L'esempio seguente mostra uno script che fa apparire un messaggio diverso a seconda dell'argomento fornitogli.

```
#!/bin/bash
case $1 in
    -a | -A | --alpha)    echo "alpha" ;;
    -b)                   echo "bravo" ;;
    -c)                   echo "charlie";;
    *)                    echo "opzione sconosciuta";;
esac
```

Come si può notare, per selezionare `alpha` si possono utilizzare tre opzioni diverse.

4.4 Funzioni

Attraverso le funzioni è possibile dare un nome a un gruppo di liste di comandi, in modo da poterlo richiamare come si fa per un comando interno normale. Sotto questo aspetto, le funzioni vengono impiegate normalmente all'interno di file script.

```
[function] nome () {
    lista_di_comandi
}
```

Le funzioni vengono eseguite nel contesto della shell corrente e quindi non vengono attivati altri processi per la loro interpretazione (ciò al contrario di quanto capita quando viene avviata l'interpretazione di un nuovo script). La lista di comandi viene eseguita ogni volta che il nome della funzione è utilizzato come comando. Il valore restituito dalla funzione è quello dell'ultimo comando a essere eseguito all'interno di questa.
All'interno della funzione possono essere dichiarate delle variabili locali attraverso il comando interno `local`.

È possibile utilizzare il comando interno `return` per concludere anticipatamente l'esecuzione della funzione. Al termine dell'esecuzione della funzione, i parametri posizionali riprendono il loro contenuto precedente e l'esecuzione dello script riprende dal comando seguente alla chiamata della funzione.
Le funzioni possono essere esportate e rese disponibili a una subshell utilizzando il comando interno `export`.

Esempi:

L'esempio seguente mostra uno script che prima dichiara una funzione denominata `messaggio` e subito dopo la esegue semplicemente nominandola come un comando qualsiasi.

```
#!/bin/bash
messaggio () {
    echo "ciao,"
    echo "bella giornata vero?"
}

messaggio
```

Nell'esempio seguente, una funzione si occupa di emettere il riepilogo della sintassi per l'uso di un ipotetico script.

```
function sintassi () {
    echo "myscript {-cd | -ls | -txt }"
    echo ""
    echo "-cd           esegue il cd nella directory corrente;"
    echo "-ls           esegue ls nella directory corrente;"
    echo "-txt          crea un file di testo vuoto;"
}
```

Nell'esempio seguente, si utilizza il comando `return` per fare in modo che l'esecuzione della funzione termini in un punto determinato restituendo un valore stabilito. Lo scopo dello script è quello di verificare che esista il file `inittab` nella directory `/etc/`.

```
#!/bin/bash
function verifica() {
    if [ -e "/etc/$1" ]
    then
        return 0
    else
        return 1
    fi
}

if verifica inittab
then
    echo "Il file inittab esiste"
else
    echo "Il file inittab non esiste"
fi
```

CAPITOLO 5

GESTIONE DEI PROCESSI

Un programma singolo, nel momento in cui viene eseguito, è un ***processo***. La nascita di un processo, cioè l'avvio di un programma, può avvenire solo tramite una richiesta da parte di un altro processo già esistente. Si forma quindi una sorta di gerarchia dei processi organizzata ad albero. Il processo principale (*root*) che genera tutti gli altri, è quello dell'eseguibile `init` che a sua volta è attivato direttamente dal kernel.
Si preferisce utilizzare il nome Init per identificare il processo principale, tenendo conto che questo si concretizza generalmente nell'eseguibile `init`.

5.1 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 genitore* (cioè il processo che ha generato quello a cui si fa riferimento);
- il *nome del dispositivo di comunicazione* se il processo è controllato da un terminale;
- il *numero di identificazione dell'utente*;
- il *numero di identificazione del gruppo*.

/proc/

Il kernel dei sistemi UNIX 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 è sempre il primo a essere messo in funzione. Il listato seguente mostra il contenuto che potrebbe avere il file `/proc/1/status`.

```
Name:   init
State:  S (sleeping)
Pid:    1
PPid:   0
Uid:    0       0       0       0
Gid:    0       0       0       0
Groups:
VmSize:      764 kB
VmLck:         0 kB
VmRSS:        16 kB
VmData:       64 kB
VmStk:         4 kB
VmExe:        24 kB
VmLib:       628 kB
SigPnd: 0000000000000000
...
```

5.2 Nascita e morte di un processo

La nascita di un processo, cioè l'avvio di un programma, può avvenire solo tramite una richiesta da parte di un altro processo già esistente, utilizzando la chiamata di sistema `fork()`. Per esempio, quando si avvia un programma attraverso il terminale, è 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***. È 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 necessità attraverso un segnale `SIGCHLD`. Questo segnale viene inviato proprio dalla funzione di sistema `exit()`, ma se il meccanismo non funziona come previsto, si può inviare manualmente un segnale `SIGCHLD` al processo genitore. In mancanza d'altro, si può 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, cioè 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 è denominato `core`, dal nome del primo tipo di memoria centrale: 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
```

Quello che conta è non confondere *core* con spazzatura: ad esempio ci possono essere dei file chiamati `core` per qualche motivo, che nulla hanno a che fare con lo scarico della memoria.

5.3 Comunicazione tra processi

Nel momento in cui l'attività di un processo dipende da quella di un altro ci deve essere una forma di comunicazione tra i due. Ciò viene definito IPC, o *Inter Process Communication*.
I metodi utilizzati normalmente sono di due tipi: invio di segnali e pipe.

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 è 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.

I segnali più comuni sono i seguenti:

- **`SIGINT`**
  È 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`**
  È 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`**
  È un segnale di conclusione intercettabile, inviato normalmente da un altro processo. Di solito, provoca la conclusione del processo che ne è il destinatario.

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

- **`SIGHUP`**
  È 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. Può essere generato anche a causa della «morte» del processo controllante.

La tabella seguente elenca alcuni dei segnali più importanti mentre l'elenco completo può essere ottenuto consultando *signal*(7):

Segnale	Azione	Descrizione
SIGHUP	A	Il collegamento con il terminale è stato interrotto.
SIGINT	A	Interruzione attraverso un comando dalla tastiera.
SIGQUIT	A	Conclusione attraverso un comando dalla tastiera.
SIGILL	A	Istruzione non valida.
SIGABRT	C	Interruzioni di sistema.
SIGFPE	C	Eccezione in virgola mobile.
SIGKILL	AEF	Conclusione immediata.
SIGSEGV	C	Riferimento non valido a un segmento di memoria.
SIGPIPE	A	Pipe interrotta.
SIGALRM	A	Timer.
SIGTERM	A	Conclusione.
SIGUSR1	A	Primo segnale definibile dall'utente.
SIGUSR2	A	Secondo segnale definibile dall'utente.
SIGCHLD	B	Eliminazione di un processo figlio.
SIGCONT		Riprende l'esecuzione se era stato fermato.
SIGTSTOP	DEF	Ferma immediatamente il processo.
SIGTSTP	D	Stop attraverso un comando della tastiera.
SIGTTIN	D	Processo sullo sfondo che richiede dell'input.
SIGTTOU	D	Processo sullo sfondo che deve emettere dell'output.

Le lettere contenute nella seconda colonna rappresentano il comportamento predefinito dei programmi che ricevono tale segnale:

- `A` termina il processo;
- `B` il segnale viene ignorato;
- `C` la memoria viene scaricata (*core dump*);
- `D` il processo viene fermato;
- `E` il segnale non può essere catturato;
- `F` il segnale non può essere ignorato.

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 è possibile collegare più 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 prova | 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 è il primo a uscire).

È possibile creare esplicitamente dei ***serbatoi FIFO*** di questo genere, in modo da poterli gestire senza dover fare ricorso alle funzionalità della shell. Questi, sono dei file speciali definiti proprio "FIFO" e vengono creati attraverso il programma `mkfifo`. Da notare che i file FIFO, data la loro affinità 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 della pipeline, 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*.

5.4 Scheduling e priorità

La gestione simultanea dei processi è 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 è lo *scheduling*, ovvero la pianificazione di questi processi.

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

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

5.5 Privilegi dei processi

Nei sistemi operativi UNIX c'è la necessità 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 è rappresentato virtualmente dai processi che lui stesso mette in esecuzione; pertanto, un'informazione essenziale riferita ai processi è quella che stabilisce l'appartenenza a un utente e a un gruppo. In altri termini, ogni processo porta con sé l'informazione del numero UID (User-ID) e del numero GID (Group-ID), in base ai quali ottiene i privilegi relativi e gli viene concesso o meno di compiere le operazioni per cui è stato avviato.

5.6 Monitoraggio 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 è standardizzato per tutti i sistemi UNIX, i programmi che ne permettono la consultazione hanno raramente un funzionamento interno simile tra loro. Ciò non impedisce però che i comandi (e di conseguenza anche la sintassi) utilizzati per la gestione dei processi siano standard per tutti i sistemi UNIX.

5.6.1 Process status

Il controllo dello stato dei processi esistenti avviene fondamentalmente attraverso l'uso di `ps`, `pstree` e `top`.

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

`top` invece è 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 però della possibilità di interferire con i processi inviandovi dei segnali o cambiandone il valore nice.

Intestazioni

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

Sigla	Descrizione
UID	Il numero di UID dell'utente proprietario del processo.
PID	Il numero del processo, cioè il PID.
PPID	Il PID del processo genitore (quello da cui ha avuto origine).
USER	Il nome dell'utente proprietario del processo.
PRI	La priorità 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 quantità di memoria condivisa utilizzata dal processo.
WCHAN	L'evento per cui il processo è 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 più lettere alfabetiche il cui significato viene riassunto nella tabella seguente:

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 (è spostato completamente nella memoria virtuale).
N	Ha un valore nice positivo (in pratica è rallentato).

5.6.2 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. Si avrà un output del genere:

```
# ps

  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
```

Opzioni principali:

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 più 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 è 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 così quelli che per qualunque motivo sono in uno stato di pausa.

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

t*x*
Permette di ottenere l'elenco dei processi associati al terminale *x*. Per identificare un terminale, si può 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 è troppo lunga consente la visualizzazione di una riga in più: l'opzione può essere indicata più 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) seguente:

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	Priorità.

x
Visualizza anche quei processi che non provengono da terminali.

Esercitazione:

Elencare i processi appartenenti all'utente che dà il comando.
$ **ps**

Elencare tutti i processi utilizzando un formato più ampio in modo da fornire più dettagli sui processi.
$ **ps a l**

Elencare tutti i processi in funzione escludendo quelli in pausa.
$ **ps a r**

Elencare 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 Our** oppure: $ **ps a l --sort=uid,rss**

Elencare tutti i processi, anche quelli non provenienti da terminali, in formato utente.
$ **ps -aux**

5.6.3 pstree

pstree [*opzioni***] [***PID* **|** *utente***]**
Visualizza uno schema ad albero dei processi in corso di esecuzione. È 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`.
Si avrà un output del tipo:

```
$ pstree -u -p

      init(1)-+-crond(173)
              |-gpm(314)
              |-inetd(210)
              |-kerneld(23)
              |-kflushd(2)
              |-klogd(162)
              |-kswapd(3)
              |-login(374)--bash(382)
              |-login(381)--bash(404,federico)--startx(415)--xinit(416)-...
              |-lpd(232)
              |-mingetty(380)
              |-mingetty(375)
              |-mingetty(376)
              |-mingetty(377)
              |-named(221)
          ...
              |-portmap(184)
              |-rpc.mountd(246)
              |-rpc.nfsd(255)
              |-rxvt(433)---bash(434,federico)---su(444,root)---bash(445)
              |-rxvt(436)---bash(437,federico)---pstree(608)
              |-sendmail(302)
              |-snmpd(198)
              |-syslogd(153)
              `-update(379)
```

Opzioni principali:

`-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.

5.6.4 top

`top [`*`opzioni`***`]`**
Visualizza la situazione sull'utilizzo delle risorse di sistema attraverso una tabella dell'attività principale della CPU, cioè 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. Esempio di schermata del comando `top`:

```
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
```

Opzioni principali:

`-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 è di cinque secondi.

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

`-s`
Disabilita la possibilità di utilizzare alcuni comandi in modo interattivo. Può 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 modalità di funzionamento.

`k`
Permette di inviare un segnale a un processo che verrà 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 quantità di processi da visualizzare. Il numero che esprime questa quantità viene richiesto successivamente. Il valore predefinito di questa quantità è 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 però può essere attribuito solo dall'utente root, migliora le prestazioni. Se il valore nice non viene specificato, per default si ha un aumento di +10 rispetto al valore corrente.

`S`
Attiva o disattiva la modalità 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 può attribuire il valore zero che implica un aggiornamento continuo. Il valore predefinito di questa durata è di cinque secondi.

u
Permette di visualizzare i processi dell'utente specificato. Dopo l'inserimento della lettera `u`, viene richiesto il nome dell'utente di cui si vogliono vedere i processi.

5.6.5 Altri comandi

I comandi visti sono i più importanti strumenti per monitorizzare i processi. Accanto a questi ce ne sono però altri che, anche se utilizzati in maniera minore, possono comunque essere utili a chi amministra il sistema.

fuser

`fuser [`*`opzioni`***`]`** *`file...`*

A volte è importante conoscere se un file è utilizzato da qualche processo. Per questo si utilizza il programma `fuser` che è in grado di dare qualche informazione aggiuntiva del modo in cui tale file viene utilizzato.
Il compito normale di `fuser` è 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 può essere utilizzato anche dagli utenti comuni per buona parte delle sue funzionalità.
Quando si utilizza `fuser` per ottenere l'elenco dei processi che accedono a determinati file, 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.

Opzioni principali:

`-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`).

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

-v
Mostra una tabella dei processi abbinati ai file, in forma più chiara rispetto alla visualizzazione normale.

Esempi:

```
# fuser /dev/* -v
```

Mostra i processi che accedono ai file della directory `/dev` utilizzando una tabella.

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`.

uptime

`uptime [`*`opzioni`***`]`**

Permette di conoscere da quanto tempo è in funzione il sistema senza interruzioni. Emette una sola riga contenente:

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

Esempio:

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

free

`free [`*`opzioni`***`]`**

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

Opzioni principali:

-b o **-k**
Nel primo caso i valori vengono espressi in byte. Nel secondo caso i valori vengono espressi in Kilobyte (modalità predefinita).

-t
Visualizza anche una riga contenente i totali.

-s *secondi_di_dilazione*
Permette di ottenere un aggiornamento continuo a intervalli regolari stabiliti dal numero di secondi indicato come argomento. Questo numero può essere anche decimale.

Esempio:

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

5.7 Cambiamento priorità di esecuzione

La priorità di esecuzione di un processo può essere modificata attraverso il valore **nice** che viene sommato, in modo algebrico, al valore di questa.
Quando si parla di priorità occorre però fare attenzione al contesto: di solito, un valore basso significa precedenza (quindi priorità) rispetto ai valori superiori.
Dal momento che il valore nice viene sommato alla priorità, se nice è pari a zero non altera la priorità di esecuzione di un processo, se ha un valore positivo ne rallenta l'esecuzione, se invece ha un valore negativo ne accelera il funzionamento.

Alcuni programmi ricevono dal sistema un valore di priorità particolarmente basso per motivi fisiologici di funzionamento del sistema stesso. Il pretendere di portare agli stessi livelli di priorità altri programmi potrebbe comportare il blocco del sistema operativo. In pratica, anche se si tenta di dare a un processo un valore nice negativo, spesso il sistema non reagisce con un'eguale diminuzione del valore della priorità. Inoltre, solo l'utente root può attribuire ai processi valori nice inferiori a zero.

Di solito quindi, il valore nice viene usato per ridurre la velocità di esecuzione di un processo in modo da alleggerire l'impiego di risorse da parte dello stesso. Spesso si combina questa tecnica assieme all'utilizzo di elaborazioni sullo sfondo.

A fianco del problema della modifica della priorità di esecuzione di un programma c'è anche quello di mantenere in funzione un programma anche dopo la disconnessione del terminale dal quale questo viene avviato.

5.7.1 nice

`nice [`*`opzioni`***`] [`***`comando`** **`[`***`argomenti`***`]]`**
Esegue un comando con un valore nice diverso dal normale. Minore è questo valore, maggiori saranno le risorse (in termini di rapidità di esecuzione) che il sistema gli concede. L'esecuzione senza alcun argomento visualizza il livello attuale del valore nice. Se viene specificato il nome di un comando, ma non viene indicato il livello di variazione (*adjustment*), il valore nice del comando indicato come argomento sarà incrementato di 10 rispetto al valore attuale. Il livello di variazione può andare da un minimo di -20 a un massimo di +19, ma solo l'utente *root* può attribuire variazioni negative.

Opzioni principali:

> **`-n`** *`variazione`*
> Definisce esplicitamente il livello di variazione del valore nice da attribuire al comando da eseguire.

Esercitazione:

> Rallentare il comando `pico` eseguendolo in background (utilizzare `&` finale).
> `$` **`nice -n`** `19 pico` **`&`**

5.7.2 renice

renice *priorità* **[[**-p**]** *pid*...**] [[**-g**]** *pid_di_gruppo*...**] [[**-u**]** *utente*...**]**
renice modifica il valore nice di uno o più processi. È possibile indicare un processo singolo, un processo che faccia capo a un gruppo (un processo dal quale discendono altri processi), oppure tutti i processi che appartengono a un certo utente.

Opzioni principali:

-p *pid*
Indica esplicitamente che si fa riferimento a un processo singolo, indicato attraverso il numero PID.

-g *pid_di_gruppo*
Indica un processo, attraverso il numero PID, riferendosi anche a tutti i suoi processi discendenti.

-u *utente*
Indica che si fa riferimento a tutti i processi avviati con i privilegi dell'utente indicato per nome.

Esercitazione:

Compilare ed eseguire (tramite compilatore gcc) con il nome di cpu.exe il seguente programma scritto in linguaggio C:

```
//file cpu.c
#include <unistd.h>

void main()
{
      while (1){ }
}
```

Aprire 2 terminali: in uno loggarsi come root e sull'altro come utente normale. Sul terminale dell'utente comune lanciare i comandi:
```
$./cpu.exe &
$./cpu.exe &
```
in modo da avere due processi eseguiti in modalità background. Facendo un ps si avrà una situazione del genere:

```
 PID  TTY       TIME  CMD
1042  tty2  00:00:00  bash
...
1133  tty2  00:05:33  cpu.exe
1134  tty2  00:05:37  cpu.exe
1135  tty2  00:00:00  ps
```

A questo punto andare sul terminale dell'utente root ed eseguire top. Premendo il carattere s e digitando 1, si imporrà al programma di aggiornare automaticamente la situazione ogni secondo. In più premendo u e digitando il nome dell'utente normale che ha aperto il terminale si possono vedere i processi di quell'utente soltanto.

In questo caso si nota che i due processi `cpu.exe` si dividono in maniera pressoché identica la percentuale di utilizzo della CPU (%CPU). Ritornando al terminale dell'utente normale si può eseguire il comando:

```
$ renice 19 -p 1134
```

che aggiunge un valore nice positivo pari a 19 al processo identificato dal PID 1134 (uno dei due `cpu.exe`). Se si torna al terminale utilizzato dall'utente root si verifica che il processo cpu.exe di PID 1134 risulta avere una percentuale di utilizzo della CPU molto inferiore (13,8%) rispetto al processo `cpu.exe` di PID 1133 (85,1%). Questo perché il processo di PID 1134 è stato *rallentato* di molto.

Provare ora ad assegnare un valore nice negativo (anche utilizzando i comandi interattivi di `top`) al processo `cpu.exe` di PID 1134 e vedere cosa succede.

5.7.3 Processi background

Per mezzo della shell è possibile avviare dei comandi sullo sfondo, ovvero in *background*, in modo che si renda nuovamente disponibile l'invito per inserire altri comandi.

```
$ yes > /dev/null &
```

Questo comando avvia `yes` dirottando l'output nel file `/dev/null` che in realtà è un dispositivo speciale paragonabile a una pattumiera senza fondo (tutto ciò che vi viene scritto è eliminato). Il simbolo "e-commerciale" ('**&**'), posto alla fine del comando, dice alla shell di eseguirlo sullo sfondo. Naturalmente, ha senso eseguire un comando sullo sfondo quando questo non richiede input da tastiera e non emette output sul terminale.

5.7.4 nohup

nohup *comando* **[***argomenti***]**

Esegue un comando facendo in modo che questo non sia interessato dai segnali di interruzione di linea (`SIGHUP`). Da questo punto di vista, `nohup` permette di avviare dei processi che non devono interrompersi nel momento in cui l'utente che li avvia termina la sua sessione di lavoro (chiude la connessione con il terminale). Naturalmente, questo ha senso se i programmi vengono avviati sullo sfondo (esecuzione in *background*).
In base a questo principio, cioè quello per cui si usa `nohup` per avviare un programma sullo sfondo in modo che continui a funzionare anche quando l'utente si scollega, la priorità di esecuzione viene modificata, aumentando il valore nice di cinque unità.

Il comando indicato come argomento non viene messo automaticamente sullo sfondo: per ottenere questo occorre aggiungere il simbolo **&** (e-commerciale) alla fine della riga di comando.

Quando il comando indicato come argomento utilizza il terminale per emettere l'output, sia lo standard output che lo standard error vengono ridiretti verso il file./`nohup.out`, oppure, se i permessi non lo consentono, verso il file `~/nohup.out`. Se questo file esiste già i dati vengono aggiunti.

Esercitazione:

Vediamo come effettivamente si comporta `nohup`. Si comincia dall'avvio di una nuova copia della shell Bash nel modo seguente:

```
$ bash
```

Avviare sullo sfondo il programma `yes` e redirigere il suo output verso `/dev/null`:

```
$ nohup yes > /dev/null &
[1] 1304
```

Il processo corrispondente ha il numero `PID 1304`. Si controlla lo stato dei processi attraverso `ps`:

```
$ ps
        PID   TTY   STAT  TIME   COMMAND
            ...
       1304   tty1   RN   2:39     yes
            ...
```

Dalla colonna `STAT` si può osservare che `yes` ha un valore nice positivo (si osserva per questo la lettera `N`). Si controlla lo stato dei processi attraverso `pstree`:

```
$ pstree -p
      init(1)-+-...
                  |
              ...
                  |-login(370)--bash(387)--bash(1303)-+-pstree(1341)
                  |                                   `-yes(1304)
              ...
```

Si può osservare che `yes` è un processo figlio della shell Bash (l'eseguibile `bash`) avviata poco prima.

Si conclude l'attività della shell provocando un segnale di interruzione di linea per i processi che dipendono da questa tramite:

```
$ exit
```

Si controlla nuovamente lo stato dei processi attraverso `pstree`:

```
$ pstree -p
      init(1)-+-...
                  |
              ...
                  |-login(370)---bash(387)---pstree(1359)
              ...
                  |-yes(1304)
              ...
```

Adesso, `yes` risulta essere un processo figlio del processo principale (Init, ovvero l'eseguibile `init`).

Per interrompere l'esecuzione dell'eseguibile `yes`:

```
$ kill PID_yes
```

Probabilmente, facendo qualche esperimento, si può osservare che i processi sullo sfondo non terminano la loro esecuzione quando si conclude la sessione di lavoro della shell che li ha avviati, senza bisogno di utilizzare `nohup`. Tuttavia ci sono situazioni in cui `nohup` è indispensabile. Per esempio, se si sta lavorando con l'ambiente grafico X e si chiude una finestra di terminale, un eventuale programma sullo sfondo viene eliminato sicuramente, a meno di usare `nohup`.

5.8 Invio di segnali ai processi

I segnali sono dei numeri ai quali i programmi attribuiscono significati determinati, relativi a quanto accade nel sistema. I segnali rappresentano sia un'informazione che un ordine: nella maggior parte dei casi i programmi possono intercettare i segnali e compiere delle operazioni correlate prima di adeguarsi al nuovo stato, oppure addirittura rifiutare gli ordini; in altri casi sono sottomessi immediatamente agli ordini.
I numeri dei segnali sono stati abbinati a nomi standard che ne rappresentano in breve il significato (in forma di abbreviazione o di acronimo). I numeri dei segnali non sono standard tra i vari sistemi UNIX e dipendono dal tipo di architettura hardware utilizzata. Anche all'interno di Linux stesso ci possono essere differenze a seconda del tipo di macchina che si utilizza.
Questo particolare è importante sia per giustificare il motivo per cui è opportuno fare riferimento ai segnali in forma verbale, sia per ricordare la necessità di fare attenzione con i programmi che richiedono l'indicazione di segnali esclusivamente in forma numerica (per esempio `top`).

Segnali attraverso la tastiera

Alcuni segnali possono essere inviati al programma con il quale si interagisce attraverso delle combinazioni di tasti. Di solito si invia un segnale `SIGINT` attraverso la combinazione [`Ctrl+c`], un segnale `SIGTSTP` attraverso la combinazione [`Ctrl+z`] e un segnale `SIGQUIT` attraverso la combinazione [`Ctrl+\`].
L'effetto di queste combinazioni di tasti dipende dalla configurazione della linea di terminale. Questa può essere controllata o modificata attraverso il programma `stty`. Come si può vedere dall'esempio seguente, alcune combinazioni di tasti (rappresentate nella forma ^*x*) sono associate a delle funzioni:

```
$ stty -a

speed 38400 baud; rows 28; columns 88; line = 204;
intr = ^C; quit = ^\; erase = ^H; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
...
```

5.8.1 kill

`kill [`*`opzioni`***`] [`***`PID...`***`]`**

`Kill` permette di inviare un segnale a uno o più processi identificati attraverso il loro numero PID. Il nome `kill` deriva in particolare dall'effetto che si ottiene utilizzandolo senza l'indicazione esplicita di un segnale da inviare: quello predefinito è `SIGTERM` attraverso il quale si ottiene normalmente la conclusione del processo destinatario.
Attraverso `kill` si riesce solitamente a ottenere un elenco dei segnali disponibili con il loro numero corrispondente. Ciò è molto importante per conoscere esattamente quale numero utilizzare con i programmi che non permettono l'indicazione dei segnali in forma verbale.

Opzioni principali:

> **`-s`** *`segnale`*
> Specifica il nome o il numero del segnale da inviare. La lettera `s` può anche essere ignorata: basta che ci sia il trattino `-`.

-l
Mostra l'elenco dei segnali disponibili con i numeri corrispondenti.

Esempi:

$ **kill -s SIGHUP 1203**
Invia il segnale SIGHUP al processo corrispondente al numero 1203.

$ **kill –l**

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGIOT       7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR
```

Mostra l'elenco dei segnali disponibili.

$ **kill -s 1 1203**
Stesso comportamento del primo esempio.

$ **kill -9 1203**
$ **kill –KILL 1203**
Invia il segnale SIGKILL al processo di PID 1203.

5.8.2 killall

killall [*opzioni***] [***-segnale***] [***comando...***]**
Invia un segnale a tutti i processi che eseguono i comandi specificati. Si utilizza quindi killall per inviare un segnale a dei processi identificati per nome. Se non viene specificato il segnale da inviare, si utilizza SIGTERM. I segnali possono essere indicati per nome o per numero.

Opzioni principali:

-l
Mostra l'elenco dei segnali disponibili con i numeri corrispondenti.

Esempi:

$ **killall -HUP** *nome_comando*
Invia il segnale SIGHUP a tutti i processi avviati con il comando *nome_comando*. I processi soggetti a questo sono solo quelli che appartengono all'utente che invia il segnale.

5.9 Esercitazione sui processi

I processi vengono eliminati automaticamente una volta che questi terminano regolarmente. A volte ci può essere la necessità di eliminare forzatamente un processo. Per verificare questa situazione si può passare sulla seconda console virtuale e da lì avviare un programma inutile che verrà eliminato attraverso la prima console.

[*Alt+F2*]

Se fosse necessario eseguire l'accesso, è questo il momento di farlo.

```
$ yes
y
y
y
y
...
```

Attraverso `yes` si ottiene un'emissione continua di lettere «y». Si può passare alla prima console e osservare la situazione.

[*Alt+F1*]

```
$ ps -a
PID TTY STAT TIME COMMAND
077  1   SW  0:01 (login)
078  2   SW  0:01 (login)
091  1    S  0:01 -bash
132  2    S  0:01 -bash
311  2    R  0:26 yes
```

Si decide di eliminare il processo generato da `yes`, e questo attraverso l'invio di un segnale di conclusione.

```
$ kill 311
```

Il numero 311 è il numero abbinato al processo, o PID, che si ottiene osservando le informazioni emesse da `ps`. Tornando sulla console in cui era stato eseguito `yes` si potrà osservare che questo ha terminato di funzionare.

[*Alt+F2*]

```
...
y
y
Terminated
```

Per gli utenti più esperti provare a compilare ed eseguire (tramite compilatore `gcc`) il seguente programma scritto in linguaggio C.

```
//file cpu.c
#include <unistd.h>

void main()
{
      while (1){
            // usleep (1000); //indica il numero di microsecondi
                              // per cui il programma si addormenta
               }
}
```

Provare ad eseguire il programma così come si presenta e "ucciderlo" utilizzando la procedura vista in precedenza.

In seguito provare a togliere il commento ad "`usleep (1000)`", compilare ed eseguire il programma con un altro nome (es. `cpu1.exe`) e vedere con il comando top le differenze con il programma `cpu.exe`. Per comodità si può eseguire il processo in background:

```
$./cpu.exe &
```

CAPITOLO 6

IL FILE SYSTEM

Il file system di UNIX supporta due entità principali: file e directory. Le directory sono viste come file con un formato speciale: quindi il concetto fondamentale utile per comprendere il file system dei sistemi UNIX è costituito da come vengono rappresentati i file.

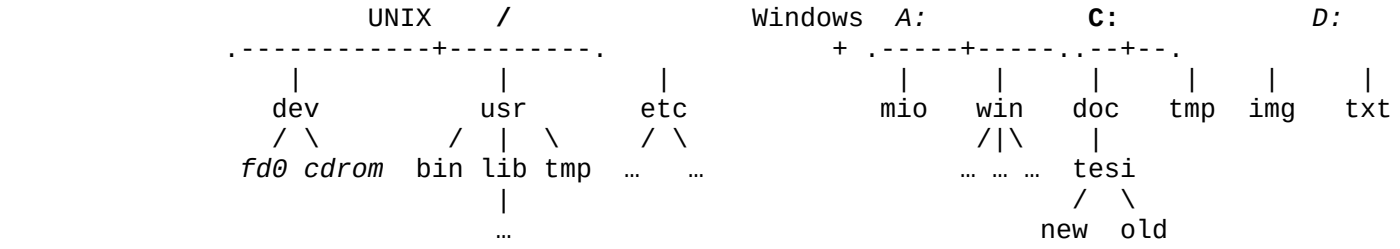
6.1 Manipolazione dei file

In UNIX un file è una sequenza di byte. Programmi diversi possono necessitare di diversi livelli di struttura, ma il kernel non impone una struttura ai file.
I file sono organizzati in directory strutturate ad albero. Le stesse directory sono file che contengono informazioni su come trovare altri file.

UNIX ha path-name *assoluti* e *relativi*. I path-name assoluti iniziano dalla radice del file system e sono contraddistinti da una barra all'inizio del path-name stesso: `/usr/local/font`. I path-name relativi iniziano dalla directory corrente e la sintassi è del tipo: `local/font` oppure.`/local/font`.

Un file può essere noto con più di un nome in più di una directory. I nomi multipli sono noti con il termine *link* e sono tutti trattati nello stesso modo dal sistema operativo. I sistemi UNIX supportano anche *link simbolici* che sono dei file contenenti il path-name di un altro file. Infatti esistono due tipi di link: *hard link* e *soft link*. I soft link diversamente dagli hard link possono puntare su directory e possono superare i confini del file system. In una directory il nome di file "." è un hard link alla directory stessa. Invece il nome di file ".." è un hard link alla directory padre. Più precisamente un collegamento fisico o *hard link*, è un collegamento che una volta creato ha lo stesso livello di importanza del file riferito e non è distinguibile da quello. Invece un collegamento simbolico, o *soft link*, è un file speciale contenente un riferimento ad un altro percorso e quindi ad un altro nodo del grafo di directory e file.
Nel file system i dispositivi hardware hanno dei nomi. Questi *file speciali di dispositivo* sono riconosciuti dal kernel come interfacce dei dispositivi a cui l'utente può accedere con le stesse system call con cui accede agli altri file.

Il file system UNIX è un file system detto a *radice unica* contrapponendosi a quello a *radice multipla* presente nei sistemi Windows:



Notare come le radici dei dispositivi presenti in Windows (`A:` e `D:`) siano allo stesso livello della radice del file system principale (**`C:`**): per tale motivo il sistema Windows è detto sistema a radice multipla. Il contrario avviene nei sistemi UNIX in cui si ha una radice unica contrassegnata da `/` e i dispositivi sono rappresentati dai cosiddetti *file di dispositivo* che si

trovano in un ramo dell'albero di directory. Inoltre in UNIX un dispositivo può essere montato in qualsiasi parte del filesystem, "a valle" della radice `/`.

Da una analisi del file system si trova che:

- la directory radice (`/`) contiene soltanto un piccolo numero di directory;
- `/dev` contiene i file speciali di dispositivo;
- `/bin` contiene i file binari dei programmi di sistema essenziali di UNIX.
- Altri file binari possono trovarsi in `/usr/bin` mentre i file di libreria si trovano in `/lib` o `/usr/lib`.
- I file degli utenti sono memorizzati in una directory distinta per ogni utente, chiamata normalmente `/home/`*`nomeutente`*. In un grande sistema queste directory possono essere raggruppate ulteriormente per facilitarne l'amministrazione.
- I file ed i programmi amministrativi come i file di password si trovano in `/etc`.
- I file temporanei possono essere inseriti in `/tmp`, che normalmente viene cancellata durante il boot del sistema, oppure in `/usr/tmp`.
- Il kernel del sistema operativo necessita soltanto di `/etc/init`, che viene utilizzata per inizializzare i processi di attivazione dei terminali.

6.2 Inode

Un file è rappresentato tramite un record che memorizza la maggior parte delle informazioni relative al file stesso: tale struttura è detta *inode*. Il termine *inode* deriva da "index node" ed originariamente veniva indicato con "i-node" o anche con "I node".
L'inode contiene gli identificatori dell'utente e del gruppo di file, l'ora dell'ultima modifica e dell'ultimo accesso al file, un contatore del numero di hard link (elementi di directory) al file ed il tipo di file (file di base, directory, link simbolico, dispositivo a caratteri, dispositivo a blocchi o socket). Inoltre l'inode contiene 15 puntatori ai blocchi del disco con il contenuto di dati del file. I primi 12 puntatori puntano a blocchi *diretti*: contengono cioè indirizzi di blocchi che contengono dati del file. Quindi ai dati dei piccoli file (massimo 12 blocchi) è possibile fare riferimento immediato poiché, mentre il file è aperto, in memoria centrale viene conservata una copia dell'inode. I tre puntatori successivi dell'inode puntano a blocchi *indiretti*. Se il file è abbastanza grande da giustificare l'utilizzo di blocchi indiretti, ognuno di essi avrà la dimensione del blocco maggiore. Il primo puntatore ad un blocco indiretto è l'indirizzo di un *blocco indiretto singolo* cioè di un blocco indice che non contiene dati ma gli indirizzi che contengono dati. Inoltre vi è un puntatore ad un *blocco indiretto doppio* cioè l'indirizzo di un blocco che contiene gli indirizzi di blocchi che contengono i puntatori agli effettivi blocchi di dati. L'ultimo puntatore contiene infine l'indirizzo di un *blocco indiretto triplo*.

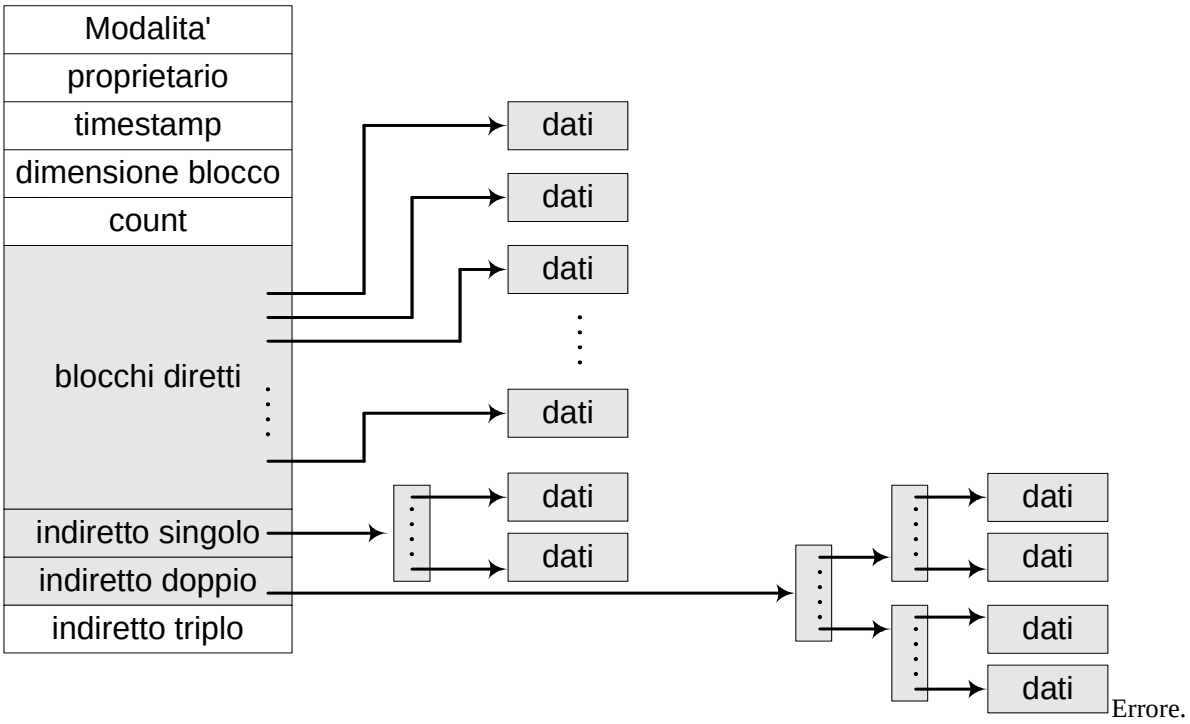


Figura 2 - Struttura di un inode

6.3 Directory

Non esiste una vera e propria distinzione tra file di base e directory: il contenuto delle directory viene conservato in blocchi di dati e le directory sono rappresentate da un inode, così come avviene per i file di base. Soltanto il campo `type` dell'inode permette di distinguere i file dalle directory.
Nei sistemi UNIX i nomi dei file hanno lunghezza variabile fino a 255 byte, per cui anche gli elementi delle directory hanno lunghezza variabile. Ogni elemento contiene la lunghezza dell'elemento, il nome del file e l'inode number. Questo elemento di lunghezza variabile complica la gestione delle directory e le routine di ricerca ma migliora la possibilità degli utenti di scegliere nomi significativi per i loro file e le loro directory senza limiti di lunghezza del nome.
I primi due nomi di ogni directory sono "." e "..". I nuovi elementi della directory vengono aggiunti nel primo spazio disponibile nella directory, generalmente dopo i file esistenti. Per tale scopo viene utilizzata una ricerca lineare.
L'utente fa riferimento ad un file tramite il path-name, mentre il file system utilizza l'inode come propria definizione del file. Quindi il kernel deve mappare il path-name fornito dall'utente su un inode e a questo scopo si utilizzano le directory.
Una volta che l'inode è stato trovato viene allocata una *struttura di file* che punta all'inode. Il descrittore di file dato all'utente fa riferimento a questa struttura. I sistemi UNIX utilizzano una *cache* di nomi di directory per mantenere le traduzioni directory-inode più recenti: ciò permette un notevole aumento delle prestazioni del file system.

Fin qui si è parlato delle caratteristiche del file system di un sistema UNIX. In seguito verranno approfondite le caratteristiche di un file system molto utilizzato, derivante da quello UNIX: il file system ext2 di Linux.

6.4 Il file system di Linux

Il file system di Linux è conosciuto con il nome di file system esteso. In particolare esistono:

- file system esteso (Ext fs): usato nelle vecchie versioni di Linux;
- secondo file system esteso (Ext2fs): è il file system più utilizzato in Linux;
- terzo file system esteso (Ext3fs): è il nuovo *journaled* file system di Linux.

Secondo file system esteso (Ext2fs)

Il secondo file system esteso è probabilmente il più utilizzato file system nella comunità Linux. Fornisce la semantica UNIX standard per i file e caratteristiche avanzate. Inoltre, grazie alle ottimizzazioni incluse nel codice del kernel, è robusto ed offre prestazioni eccellenti.
Poichè il file system è stato progettato tenendo conto di possibili miglioramenti ed aggiunte, ext2fs contiene hooks (agganci) che possono essere utilizzate per aggiungere nuove caratteristiche.

Il secondo file system è nato per risolvere alcuni problemi presenti nel primo file system esteso. L'obiettivo era quello di fornire un file system potente, con la semantica del file di UNIX e caratteristiche avanzate.
Ovviamente Ext2fs doveva essere progettato in modo che avesse prestazioni eccellenti. Inoltre doveva anche essere in grado di fornire un file system robusto in modo da ridurre il rischio di una perdita dei dati. In ultimo, Ext2fs doveva prevedere la possibilità di estensioni in modo da permettere agli utenti di usufruire di nuove capacità senza riformattare i dischi.

Caratteristiche

Ext2fs supporta tutti i tipi di file standard UNIX: file regolari, directory, device speciali e link simbolici.
Inoltre è in grado di gestire file system creati in partizioni molto grandi. Mentre il codice originale nel kernel limitava il file system a 2 GB, un recente progetto ha alzato questo limite a 4 TB. Quindi è ora possibile usare dischi molto grandi senza la necessità di creare diverse partizioni.
Ext2fs gestisce i nomi lunghi in quanto usa voci nella directory di lunghezza variabile. Il limite massimo per i nomi è 255 caratteri ma può essere esteso a 1012 se necessario.
Ext2fs riserva alcuni blocchi per il super utente (*root*). Normalmente il 5% dei blocchi è riservato. Questo permette all'amministratore di correggere facilmente situazioni dove i processi di un utente riempiono il file system.

Caratteristiche avanzate

In aggiunta alle caratteristiche standard di UNIX, Ext2fs supporta alcune estensioni che non sono normalmente presenti nel file system UNIX.

Ext2fs permette all'amministratore di scegliere le dimensioni del blocco logico quando viene creato un file system. Le dimensioni del blocco possono tipicamente essere di 1024, 2048 e 4096 byte. Usare un blocco di grandi dimensioni può velocizzare le operazioni di I/O per le ridotte richieste di I/O e quindi per i minori spostamenti della testina del disco che devono essere fatti per accedere ad un file. D'altro canto blocchi grossi sprecano più spazio su disco: in media l'ultimo blocco di ogni file è pieno per metà, quindi, al crescere delle dimensioni del

blocco, più spazio viene sprecato nell'ultimo blocco. Inoltre, molti dei vantaggi nell'usare blocchi grandi sono ottenuti dalle tecniche di pre allocazione del file system Ext2fs.

Ext2fs supporta i link simbolici veloci. Un link simbolico veloce non usa alcun blocco dati del file system. Il nome del file target non è salvato nel blocco dati ma nell'inode stesso. Questa politica permette di risparmiare spazio su disco (non servono blocchi dati) e velocizza le operazioni sui link (non è necessario accedere a blocchi dati). Naturalmente lo spazio all'interno degli inode è limitato per cui non tutti i link possono essere realizzati come link veloci. Le dimensioni massime del nome del file target sono 60 caratteri. Si prevede di poter estendere questo limite in futuro.

Ext2fs tiene traccia dello stato del file system. Un campo speciale nel super blocco è usato dal kernel per indicare lo stato del file system. Quando un file system è montato in lettura e scrittura, il suo stato è posto a 'Not Clean'. Quando è smontato o rimontato in sola lettura, il suo stato è posto a 'Clean'. Durante il boot, il controllore del file system usa queste informazioni per decidere se un file system deve essere controllato. Il codice del kernel tiene traccia anche degli errori in questo campo. Quando una inconsistenza è rilevata dal codice del kernel, il file system è marcato come 'Erroneus'. Il controllore del file system verifica questo in modo da forzare il controllo del file system indipendentemente dal suo stato.

Saltare sempre i controlli del file system può essere a volte pericoloso, per cui Ext2fs fornisce due modi per forzare i controlli ad intervalli regolari. Uno di questi modi è l'utilizzo di un contatore di mount gestito nel super blocco. Ogni volta che il file system viene montato in lettura e scrittura, il contatore viene incrementato. Quando raggiunge il valore massimo (anch'esso memorizzato nel super blocco), il controllore del file system forza una verifica anche se il file system è 'Clean'. Anche la data dell'ultimo controllo ed il massimo intervallo consentito tra due controlli sono salvati nel super blocco. Questi due campi permettono all'amministratore del sistema di forzare controlli periodici. Quando l'intervallo massimo viene superato, il file system viene controllato indipendentemente dal suo stato.

Un attributo permette agli utenti di richiedere la cancellazione dei file in modo sicuro. Quando un file viene cancellato in maniera sicura, dati casuali vengono scritti nei blocchi su disco in precedenza usati dal file. Questo non permette a malintenzionati di ottenere l'accesso al precedente contenuto del file usando un disk editor.

Struttura fisica

La struttura fisica di un file system Ext2fs è stata fortemente influenzata dal layout del file system BSD. Un file system è costituito da gruppi di blocchi (ereditati dai gruppi di cilindri utilizzati nel file system BSD). Tuttavia i gruppi di blocchi non sono legati alla disposizione fisica dei blocchi sul disco poichè i dischi moderni tendono ad essere ottimizzati per un accesso sequenziale e nascondono la loro geometria fisica al sistema operativo.
La struttura fisica di un file system è rappresentata nella tabella seguente:

Boot Sector	Block Group 1	Block Group 2	... ...	Block Group N

Ogni gruppo di blocchi contiene una copia ridondante delle informazioni di controllo cruciali del file system (il super blocco e i descrittori del file system) e contiene anche una parte del file system (un blocco bitmap, una bitmap degli inode, una parte della tabella degli inode e i blocchi dato). La struttura del gruppo di blocchi è rappresentata in questa tabella:

Super Block	FS descriptors	Block Bitmap	Inode Bitmap	Inode Table	Data Blocks

Usare i gruppi di blocchi è un grosso vantaggio per l'affidabilità: poiché le strutture di controllo sono replicate in ogni gruppo, è facile ripristinare un file system con il super blocco danneggiato. Questa struttura aiuta anche ad ottenere delle buone prestazioni: riducendo la distanza tra la tabella degli inode ed i blocchi dato, è possibile ridurre gli spostamenti della testina del disco durante l'I/O sui file.
In Ext2fs, le directory sono gestite con liste collegate con voci di dimensioni variabile. Ogni voce contiene il numero di inode, le dimensioni della voce, il nome del file e la sua lunghezza. Usando voci di dimensioni variabili, è possibile gestire i nomi lunghi senza sprecare spazio su disco.
La struttura di una voce di directory è rappresentata nella tabella seguente:

inode number	entry length	name length	filename

Come esempio, la tabella seguente rappresenta la struttura di una directory contenente tre file: `file1`, `long_file_name`, and `f2`:

i1	16	05	`file1`
i2	40	14	`long_file_name`
i3	12	02	`f2`

Ottimizzazioni
In Linux, il codice Ext2fs nel kernel contiene molte ottimizzazioni per le prestazioni, le quali tendono a migliorare la velocità di I/O durante la lettura e la scrittura dei file.
Ext2fs si avvantaggia della gestione del cache buffer per eseguire delle letture in anticipo: quando un blocco deve essere letto, il codice del kernel richiede l'I/O di diversi blocchi contigui. In questo modo, si cerca di assicurare che il blocco successivo da leggere sia già caricato nel buffer. Le letture in anticipo sono normalmente effettuate durante l'accesso sequenziale dei file; ext2fs le estende anche alla lettura delle directory.
Ext2fs contiene anche molte ottimizzazioni di allocazione. I gruppi di blocchi sono utilizzati per raggruppare insieme inode e dati correlati: il codice del kernel cerca sempre di allocare blocchi dato (data blocks) per un file, nello stesso gruppo di blocchi del suo inode. Questo per ridurre gli spostamenti della testina del disco quando si leggono un inode ed i suoi blocchi dato.
Nello scrivere dati in un file, Ext2fs prealloca fino a 8 blocchi adiacenti quando alloca un nuovo blocco. La preallocazione ha una percentuale di successo di circa il 75% anche su un file system molto pieno. Con questa preallocazione si raggiungono buone prestazioni in scrittura a pieno carico. Ciò permette anche che blocchi contigui siano allocati per lo stesso file, velocizzando così le future letture sequenziali.
Queste due ottimizzazioni nelle allocazioni producono una localizzazione molto buona di:

- file collegati fra di loro, attraverso i gruppi di blocchi;
- blocchi collegati fra di loro, attraverso gli 8 bit di raggruppamento della allocazione dei gruppi.

Strategie di allocazione
Nella figura seguente sono illustrate le strategie di allocazione:

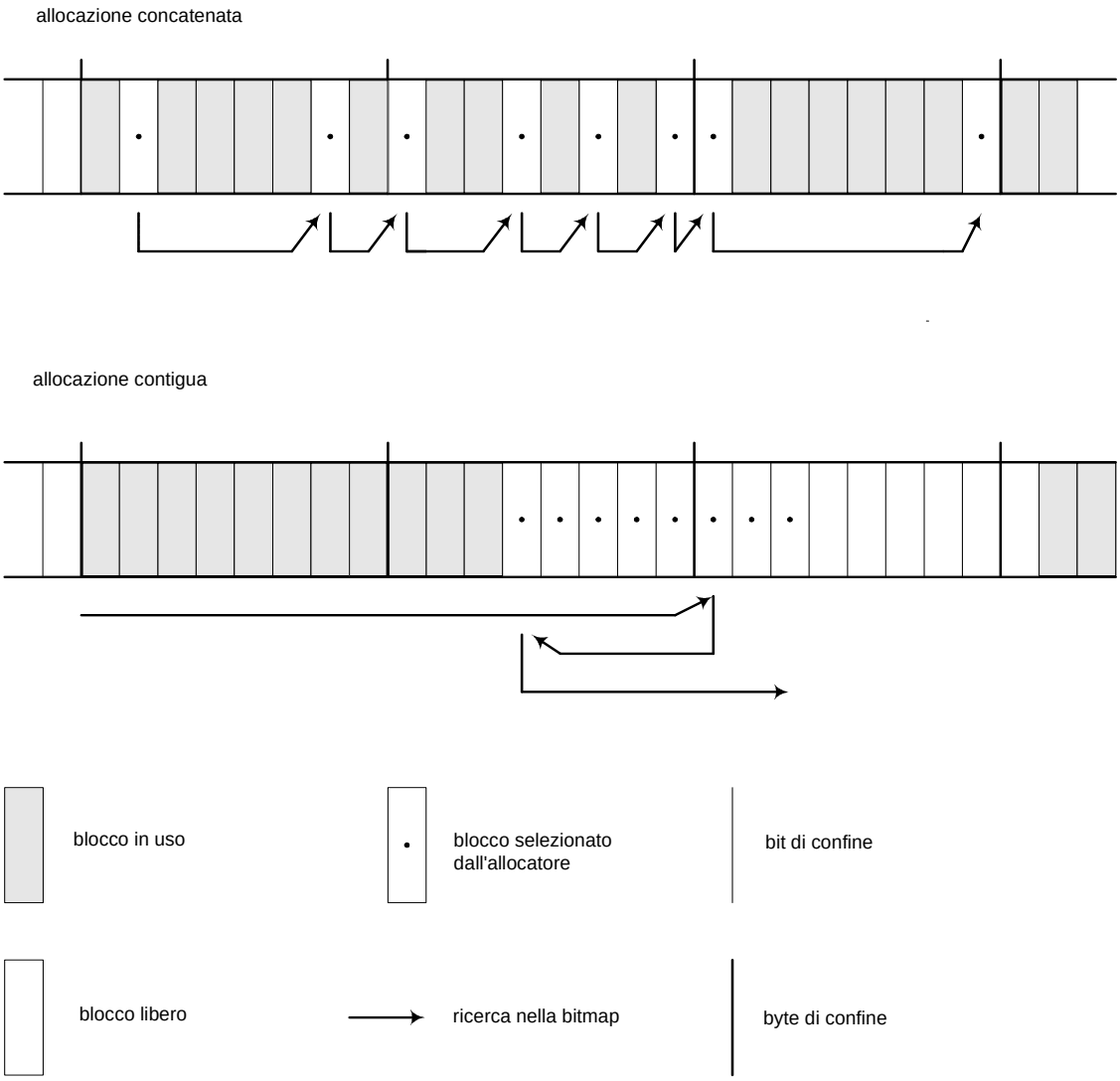


Figura 3 - Strategie di allocazione in un file system ext2

Ogni riga rappresenta una sequenza di bit posti ad uno o a zero nella mappa di bit dell'allocazione. Questi bit contrassegnano i blocchi liberi e i blocchi in uso del disco. La prima possibilità è che si trovino dei blocchi liberi sufficientemente vicini al punto d'inizio della ricerca, in tal caso essi sono allocati indipendentemente dal fatto che il loro insieme possa essere frammentato. Infatti dato che i blocchi sono vicini tra loro, è probabile che possano essere letti senza dover eseguire operazioni di ricerca sul disco, cosa che compensa parzialmente la frammentazione. Inoltre, allocare tutti questi blocchi ad un solo file è una scelta che a lungo termine si rivela migliore dell'allocazione di blocchi isolati a file distinti, perché a lungo andare regioni ampie di spazio libero su disco divengono rare. La seconda possibilità è che non si sia trovato un blocco libero nelle vicinanze del punto d'inizio della ricerca, quindi si procede nella ricerca di un intero byte libero nella mappa di bit. Se si allocasse questo byte senza prendere alcuna precauzione si creerebbe un frammento di spazio libero prima di esso: ciò è sconveniente perché è necessario tornare indietro per colmare lo spazio rimasto tra il primo blocco allocato precedentemente ed il blocco in questione. Infine, per rispettare la dimensione di default di un'allocazione, che è di otto blocchi, si estende l'allocazione in avanti fin quando è necessario.

Terzo file system esteso (Ext3 FS)

Ext3 possiede le stesse caratteristiche di ext2 ma include anche il journaling. Da notare che un file system di tipo "journaling" usa un'area separata chiamata log o journal. Prima di effettuare ogni cambiamento ai metadati (links, inode), questo è registrato in un'area separata. Il cambiamento viene quindi successivamente effettuato. Se il sistema va in crash durante l'operazione, ci sono informazioni sufficienti nel log per riprendere e completare l'operazione.
Questo approccio non richiede una verifica completa del file system, rendendo così molto veloce la verifica di file system molto grossi, in genere pochi secondi per un file system di molti gigabyte. Inoltre, poichè tutte le informazioni sulle operazioni in sospeso sono salvate, non sono necessarie rimozioni e troncamenti. Lo svantaggio dei file system di tipo "journaling" è però la maggiore lentezza rispetto ad altri file system.

6.5 Dischi

La gestione dei dischi nei sistemi UNIX è piuttosto laboriosa per chi si avvicina la prima volta alla sua filosofia. Inoltre bisogna prestare attenzione anche al nome con cui si denota un preciso supporto di memorizzazione: infatti i nomi delle periferiche (device) possono variare a seconda che ci si riferisca ad UNIX o a Linux.

Nomi delle periferiche in Linux

Linux utilizza dei nomi bene ordinati per i file di dispositivo, ma questi possono confondere chi proviene dall'esperienza DOS. La tabella seguente mostra l'elenco di alcuni nomi di dispositivo riferiti alle unità di memorizzazione principalmente utilizzate in Linux:

Descrizione	**Nome**	**Dos**
prima unità floppy	/dev/fd0	A:
seconda unità floppy	/dev/fd1	B:
primo disco fisso ATA (IDE)	/dev/hda	
secondo disco fisso (o CD-ROM) ATA (IDE)	/dev/hdb	
terzo disco fisso (o CD-ROM) ATA (IDE)	/dev/hdc	
quarto disco fisso (o CD-ROM) ATA (IDE)	/dev/hdd	
primo disco SCSI	/dev/sda	
secondo disco SCSI	/dev/sdb	
terzo disco SCSI	/dev/sdc	
	...	

I dischi che non rientrano nella categoria dei floppy, sono suddivisi in partizioni, dove per fare riferimento a queste si aggiunge un numero alla fine del nome. Per esempio, `/dev/hda1` è la prima partizione del primo disco ATA, `/dev/sda2` è la seconda partizione del primo disco SCSI.

La distinzione tra i nomi usati per le partizioni primarie e le partizioni logiche contenute in quelle estese, può creare confusione ulteriore. Volendo prendere come esempio il primo disco fisso ATA, le prime quattro partizioni normali (primarie ed estese) hanno nomi che vanno da `/dev/hda1` a `/dev/hda4`, mentre le partizioni logiche utilizzano nomi da `/dev/hda5` in poi.

Nomi delle periferiche in FreeBSD

La seguente tabella mostra una lista di periferiche supportate in FreeBSD ed il nome associato dal sistema operativo alle periferiche stesse:

Descrizione	Nome
IDE hard drives	ad
IDE CDROM drives	acd
SCSI hard drives and USB Mass storage devices	da
SCSI CDROM drives	cd
Floppy drives	fd
SCSI tape drives	sa
IDE tape drives	ast

Differenze fra i nomi device di Linux e quelli FreeBSD

I nomi dei device, ad esempio l'hard disk IDE `/dev/ad0`, non sono gli stessi per chi arriva dal mondo Linux. Segue un breve elenco dei dispositivi più utilizzati in particolare i nomi dei dispositivi relativi ad hard disk, slice, partizioni, porte seriali

Dischi ide:

Descrizione	Linux	FreeBSD
Disco master canale ide primario	hda	ad0
Disco slave canale ide primario	hdb	ad1
Disco master canale ide secondario	hdc	ad2
Disco slave canale ide secondario	hdd	ad3

Dischi SCSI:

Descrizione	Linux	FreeBSD
Primo disco SCSI	sda	da0
Secondo disco SCSI	sdb	da1
Terzo disco SCSI	sdc	da2

Partizioni primarie / slice:

La particolarità del sistema di partizionamento di FreeBSD consiste nella creazione di "**SLICE**". Tale struttura, per quanto riguarda l'hard disk è una partizione primaria al cui interno verrano create le "LABELS", ossia una sorta di sottopartizioni che formeranno il file system vero e proprio del FreeBSD. Praticamente si può immaginare lo SLICE come un guscio ermetico al cui interno esistono partizioni di filesystem e swap.

Descrizione	Linux	FreeBSD
Disco master primario ide, prima partizione/slice	hda1	ad0s1
Disco master primario ide, seconda partizione/slice	hda2	ad0s2
...		

Partizioni all'interno di una slice:

Descrizione	FreeBSD
Disco master primario ide, prima slice, prima partizione	ad0s1a
Disco master primario ide, prima slice, seconda partizione	ad0s1b
...	

Porte seriali e parallele:

Descrizione	Linux	FreeBSD
COM1	cua0 e/o ttyS0	cuaa0
COM2	cua1 e/o ttyS1	cuaa1
COM3	cua2 e/o ttyS2	cuaa2
LPT1	lpt0	lpt0

6.6 Attivazione dei file system

Il file system è 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 possibilità di distinguere tra un'unità 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'unità di memorizzazione contenente questo file system, occorre interrompere il collegamento. Ciò significa che non si può inserire un dischetto, accedervi immediatamente e toglierlo quando si vuole: occorre dire al sistema di collegare il file system del dischetto, quindi lo si può usare come parte dell'unico file system globale. Al termine si deve interrompere questo collegamento e solo allora si può rimuovere il dischetto.

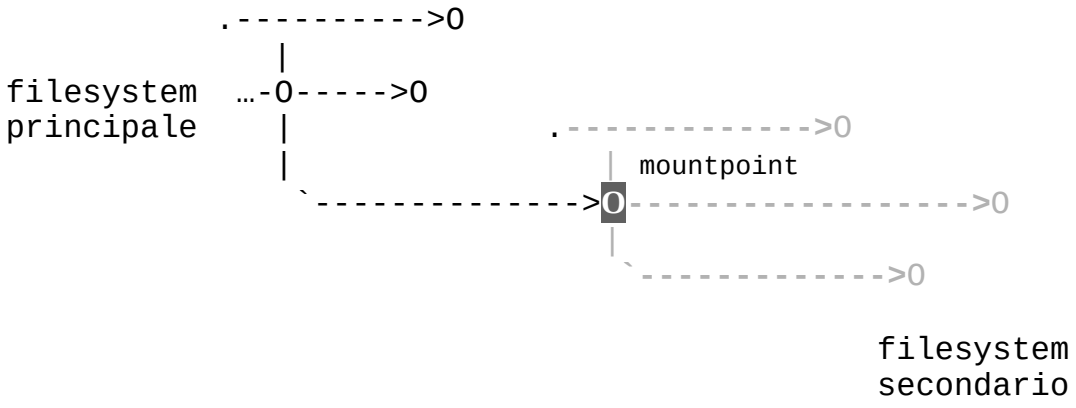


Figura 4 - Collegamento di un file system secondario in corrispondenza di un punto di innesto (mount point)

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 è il **mount point** o *punto di innesto*.

Quindi per poter accedere ai dati di un'unità di memorizzazione organizzata con un file system, è necessario prima montare il suo file system in quello globale. Prima di estrarre una di queste unità, o comunque, prima di poter spegnere un elaboratore, occorre eseguire l'operazione opposta di distacco (unmount dell'unità).

6.6.1 Tipi di file system

Quando si monta un file system è necessario che il modo con cui questo è organizzato (cioè il tipo) sia riconoscibile e gestito dal kernel. Nella tabella seguente, sono elencati i nomi che identificano i tipi di file system riconoscibili da un sistema UNIX:

Tipo di file system	Descrizione
minix	Minix
ext2	Second-extended
umsdos	Linux su Dos-FAT
msdos	Dos-FAT (nomi 8.3)
vfat	Dos-VFAT (nomi lunghi)
nfs	NFS o file system di rete
iso9660	CD-ROM
smbfs	SMB (rete NetBIOS-TCP/IP)
proc	file system virtuale /proc
swap	partizione di scambio

6.6.2 mount

`mount [`*`opzioni`***`] [`***`dispositivo`***`] [`***`directory`***`]`**

`mount` permettere di montare un file system all'interno del sistema. Il programma opposto è `umount` e serve per smontare un file system montato precedentemente. La forma normale e più semplice di utilizzo di `mount` è la seguente:

```
mount -t tipo_di_file_system dispositivo punto_di_innesto
```

In questo modo si richiede al kernel di montare il file system del dispositivo specificato nella directory indicata (punto di innesto).
Per conoscere la situazione dei dispositivi collegati attraverso questo sistema, si può usare la sintassi seguente:

```
mount [ -t tipo_di_file_system]
```

Se viene specificato il tipo di file system, si ottiene un elenco limitato a quei dispositivi.
La maggior parte delle unità di memorizzazione sono indicate nel modo consueto utilizzando nomi di file di dispositivo (`/dev/...`), ma ci possono essere altre possibilità, come quando si vuole montare un file system di rete o NFS, dove si usa la forma *`host`*`:`*`/directory`*.
Il file `/etc/fstab` viene utilizzato per automatizzare il collegamento dei file system più importanti al momento dell'avvio del sistema. Questo viene letto utilizzando la forma seguente:

```
mount -a [ -t tipo_di_file_system]
```

Di solito si trova una chiamata di questo tipo all'interno di uno degli script che compongono la procedura di inizializzazione del sistema (`/etc/init.d/*` oppure `/etc/rc.d/rc*`). La presenza del file di configurazione `/etc/fstab` è utile anche per semplificare il montaggio (e poi anche l'operazione inversa) di un file system che sia stato previsto al suo interno.

In linea di principio, solo l'utente *root* può montare un file system. Per permettere agli utenti comuni di montare e smontare un'unità di memorizzazione (come nel caso di un CD-ROM o di un dischetto), la si può indicare nel file `/etc/fstab` con l'opzione *user*. Nell'esempio

seguente, si vede un record di `/etc/fstab` attraverso il quale si definisce il montaggio facoltativo di un CD-ROM in sola lettura con la possibilità anche per gli utenti di eseguire l'operazione.

```
/dev/cdrom  /cdrom  iso9660  ro,user,noauto,unhide
```

In tal modo, qualunque utente potrà eseguire uno dei due possibili comandi seguenti.

```
$ mount /dev/cdrom
$ mount /cdrom
```

La coppia di programmi `mount` e `umount` mantiene una lista dei file system montati correntemente. Quando `mount` viene avviato senza argomenti si ottiene l'emissione del contenuto di questa lista.
Per maggiori dettagli vedere *mount*(8).

Opzioni principali:

`-a`
Utilizza `/etc/fstab` per eseguire automaticamente l'operazione: Vengono montati tutti i file system a esclusione di quelli segnati come `noauto`.

`-t [``no`**`]`***`tipo_di_file_system`***`[,...]`**
Specifica il tipo di file system. Sono riconosciuti i nomi indicati nella tabella del paragrafo precedente. Se il nome del tipo di file system viene preceduto dalla sigla `no`, si intende che quel tipo deve essere escluso. Se si vogliono indicare più tipi di file system questi vengono separati da virgole.
Quando si usa questa opzione con l'indicazione di più tipi, o con il prefisso `no`, lo si fa quasi sempre con l'uso dell'opzione `-a`, come nell'esempio seguente:

```
# mount -a -t nomsdos,nonfs
```

In questo caso si intende eseguire il montaggio di tutti i file system indicati all'interno di `/etc/fstab`, a esclusione dei tipi `msdos` e `nfs`.

`-o` ***`opzione_di_file_system`*****`[,...]`**
Questa opzione permette di specificare uno o più nomi di opzioni, separati da virgole, legati alla gestione del file system.

Esempi:

```
# mount -t ext2 /dev/hda2 /mnt
```

Monta il file system di tipo ext2 contenuto nella seconda partizione del primo disco fisso ATA, a partire dalla directory `/mnt`.

```
# mount -t vfat /dev/fd0 /floppy
```

Monta il file system di tipo Dos-VFAT (cioè Dos-FAT con le estensioni per i nomi lunghi) contenuto in un dischetto inserito nella prima unità, a partire dalla directory `/floppy/`.

```
# mount -t nfs roggen.brot.dg:/pubblica /roggen
```

Monta il file system di rete offerto dall'elaboratore `roggen.brot.dg`, corrispondente alla sua directory `/pubblica/` (e discendenti), nella directory locale `/roggen/`.

6.6.3 umount

umount [*opzioni***] [***dispositivo***] [***directory***]**
umount esegue l'operazione inversa di mount: smonta i file system. L'operazione può avvenire solo quando non ci sono più attività in corso su quei file system, altrimenti l'operazione fallisce.

Opzioni principali:

-a
Vengono smontati tutti i file system indicati in /etc/fstab.

-t [no**]***tipo_di_file_system***[,...]**
Indica che l'azione deve essere eseguita solo sui file system specificati. Se si usa il prefisso no, l'azione si deve compiere su tutti i file system a esclusione di quelli indicati.

Esempi:

umount /dev/hda2
Smonta il file system montato precedentemente, riferito al dispositivo /dev/hda2.

umount /mnt
Smonta il file system montato precedentemente nella directory /mnt.

6.6.4 /etc/fstab

Il file /etc/fstab viene utilizzato per definire le caratteristiche e le directory di collegamento (punti di innesto) dei vari file system, usati di frequente nel sistema. Si tratta di un file che viene solo letto dai programmi e il suo aggiornamento viene fatto in modo manuale dall'amministratore del sistema.
Il file è organizzato in record (corrispondenti alle righe) divisi in campi separati da uno o più spazi (inclusi i caratteri di tabulazione). Le righe che iniziano con il simbolo #, le righe vuote e quelle bianche sono ignorate e trattate eventualmente come commenti. Un esempio di record è il seguente:

```
/dev/hda3  /  ext2  defaults   0    1
```

Analisi dei campi:

1. Il primo campo definisce il tipo di dispositivo o il file system remoto da montare.

2. Il secondo campo definisce la directory che funge da punto di innesto per il file system.

3. Il terzo campo definisce il tipo di file system e ne viene indicato il nome in base alla tabella precedente.
   Se in questo campo viene indicato il termine ignore, si intende fare riferimento a una partizione presente, ma inutilizzata, per la quale non si vuole effettuare alcun collegamento. Di fatto, i record che contengono questa indicazione vengono ignorati.

4. Il quarto campo descrive le opzioni speciali per il tipo di montaggio che si intende eseguire. Si tratta delle stesse opzioni speciali descritte in *mount*(8) in occasione della spiegazione dell'uso dell'opzione `-o` (a esclusione dell'opzione `remount`).

5. Il quinto campo viene utilizzato per determinare quali file system possono essere utilizzati per lo scarico dei dati (*dump*).

6. Il sesto campo viene utilizzato dal programma `fsck` per determinare l'ordine in cui il controllo dell'integrità dei file system deve essere effettuato nel momento dell'avvio del sistema.
   Il file system principale dovrebbe avere il numero uno in questo campo, mentre gli altri, il numero due (o anche valori superiori). Se questo campo contiene il valore zero, significa che il file system in questione non deve essere controllato.

Campo `Opzioni` **valido per filesystem UNIX:**

`default`
Utilizza le impostazioni predefinite: `rw`, `suid`, `dev`, `exec`, `auto`, `atime`, `nouser`, `async`.

`sync` | **`async`**
Esegue gli I/O sui file system in modo sincrono o asincrono. La modalità sincrona è più sicura, ma il suo utilizzo rallenta e appesantisce l'attività del disco.

`atime` | **`noatime`**
Aggiorna o meno la data di accesso ai file. Può essere utile eliminare questo tipo di aggiornamento per ridurre l'attività del disco.

`auto` | **`noauto`**
Permette o impedisce il montaggio automatico quando si utilizza il file `/etc/fstab`.

`dev` | **`nodev`**
Considera validi, o esclude la validità dei file di dispositivo che dovessero essere contenuti nel file system.

`exec` | **`noexec`**
Permette o impedisce l'esecuzione di file binari.

`suid` | **`nosuid`**
Consente o impedisce che i bit SUID (*Set user ID*) e SGID (*Set group ID*) abbiano effetto. Disattivando questa possibilità (cioè utilizzando l'opzione `nosuid`), si vuole evitare che gli eseguibili contenuti nel file system che si intende montare, possano ottenere privilegi particolari.

`user` | **`nouser`**
Permette o impedisce all'utente comune di montare e smontare il file system. L'opzione `user` implica l'attivazione automatica di `noexec`, `nosuid` e `nodev`, a meno che queste siano annullate da successive indicazioni contrarie come nella lista seguente: `user,exec,suid,dev`.

ro
Sola lettura.

rw
Lettura e scrittura.

Campo `Opzioni` **valido per i file system FAT (msdos, vfat):**

uid=*identificativo_utente*
Permette di stabilire il proprietario dei file e delle directory contenute nel file system. Se non viene specificato, si intende appartengano all'utente che esegue il montaggio.

gid=*identificativo_gruppo*
Permette di stabilire il gruppo proprietario dei file e delle directory contenute nel file system. Se non viene specificato, si intende appartengano al gruppo dell'utente che esegue il montaggio.

umask=*maschera*
Permette di stabilire quali permessi inibire nel file system. Si tratta di un numero ottale, composto da tre cifre numeriche, dove la prima cifra rappresenta i permessi per il proprietario, la seconda per il gruppo, la terza per il resto degli utenti:

- 0 rappresenta un qualsiasi tipo di accesso;
- 1_8 rappresenta un permesso in esecuzione;
- 2_8 rappresenta un permesso in scrittura;
- 4_8 rappresenta un permesso in lettura.

Di conseguenza:

- 3_8 rappresenta un permesso in scrittura ed esecuzione;
- 5_8 rappresenta un permesso in lettura ed esecuzione;
- 6_8 rappresenta un permesso in lettura e scrittura;
- 7_8 rappresenta un permesso in lettura, scrittura ed esecuzione.

Bisogna fare attenzione però che il valore che si inserisce rappresenti un impedimento all'uso di quel permesso: di conseguenza, la maschera 022_8 indica che è consentito al proprietario qualunque tipo di accesso (lettura, scrittura ed esecuzione), mentre agli altri utenti non è consentito l'accesso in scrittura.
Se non viene definito si utilizza il valore predefinito per la creazione dei file nei file system normali: `default`.

quiet
I file system FAT non sono in grado di memorizzare informazioni sulle proprietà e i permessi dei file. Di conseguenza, i programmi che tentano di modificare i valori predefiniti, ottengono una segnalazione di errore dalle funzioni di sistema. L'opzione `quiet` inibisce queste segnalazioni di errore.

Per approfondimenti leggere le pagine di manuale: *mount*(8) e *nfs*(5).

Esercitazione:
Analizzare il seguente esempio tratto da un file `/etc/fstab`:

```
# path/IP      Punto di mount   Tipo     Opzioni           Dump Check

/dev/hda3            /            ext2     defaults           0     1
/dev/hdb1            /home        ext2     defaults           0     2
proc                 /proc        proc     defaults           0     0
/dev/hda2            none         swap     sw
/dev/hda1            /mnt/dosc    vfat     quiet,umask=000    0     0
/dev/sda             /mnt/dosd    vfat     user,noauto,quiet  0     0
/dev/sda1            /mnt/scsimo  ext2     user,noauto        0     0
/dev/cdrom           /mnt/cdrom   iso9660  ro,user,noauto     0     0
roggen.brot.dg:/     /mnt/roggen  nfs      ro,user,noauto     0     0
/dev/fd0             /mnt/dosa    vfat     user,noauto,quiet  0     0
```

Tutte le unità che non sono unite stabilmente al corpo fisico dell'elaboratore, hanno l'opzione `noauto` che impedisce il montaggio automatico all'avvio del sistema. Queste possono essere attivate solo manualmente, attraverso `mount`, con il vantaggio di potere indicare semplicemente la directory di collegamento (punto di innesto) o il nome del dispositivo.

Il dispositivo `/dev/hda3` viene utilizzato come file system principale e per questo è il primo a essere attivato. L'ultimo campo (*check*) riporta il valore uno, perché si vuole fare in modo che questo file system venga controllato per primo al momento dell'avvio del sistema.

Il dispositivo `/dev/hdb1` viene utilizzato come file system per contenere le directory personali degli utenti. L'ultimo campo riporta il valore due, perché si vuole fare in modo che questo file system venga controllato per secondo al momento dell'avvio del sistema, dopo il controllo del file system principale.

Il file system virtuale `proc` viene inserito correttamente nella directory `/proc/`. Il nome utilizzato nel campo del nome, `proc`, non significa nulla, ma è preferibile al consueto `none` che si usa spesso in questo caso.

Il dispositivo `/dev/hda1` corrisponde a una partizione Dos-FAT con la gestione dei nomi lunghi. In particolare, viene permesso a ogni utente di accedere ai suoi file in tutti i modi possibili.

Il dispositivo `/dev/sda` rappresenta un cosiddetto *superfloppy*, cioè un disco rimovibile (in questo caso SCSI) che non è in grado di gestire partizioni, esattamente come fanno i dischetti. Trattandosi di un disco rimovibile viene concesso a tutti gli utenti di eseguire il montaggio e questo non viene effettuato automaticamente al momento dell'avvio del sistema.

Il dispositivo `/dev/sda1` rappresenta la stessa unità a dischi rimovibili, ma in questo caso viene vista come la prima partizione di uno di questi dischi. Anche qui viene concesso agli utenti comuni di montare e smontare il disco.

Il dispositivo `/dev/cdrom` rappresenta il lettore di CD-ROM. In particolare, viene specificato che l'accesso può avvenire in sola lettura.

L'elaboratore `roggen.brot.dg` condivide tutto il proprio file system (a partire dalla directory radice) attraverso il protocollo NFS. Viene consentito l'accesso in sola lettura.

Il dispositivo `/dev/fd0`, il dischetto, può essere utilizzato da tutti gli utenti e si prevede di accedere sempre solo al formato Dos-FAT con l'estensione per i nomi lunghi.

6.7 Montare e smontare i dischetti

Nei sistemi UNIX e derivati, per poter accedere a un'unità di memorizzazione occorre che il file system di questa sia **montato** in quello globale. Non si può indicare semplicemente una directory o un file di un certo dispositivo. Il montaggio è l'operazione con cui si innesta il file system di un'unità di memorizzazione in corrispondenza di una directory del file system già attivo. La directory che si utilizza generalmente per montare provvisoriamente le unità esterne è `/mnt/`.

Si procede inserendo il dischetto formattato con il formato Dos-FAT e montandolo in `/mnt/`.

```
# mount -t msdos /dev/fd0 /mnt
```

Da questo momento, la directory `/mnt/` è l'inizio del dischetto.

```
# touch /mnt/miaprova.txt
```

Con il comando appena visto, si vuole creare un file vuoto con un nome che rispetta lo standard Dos-FAT utilizzato dai nomi dei file: kkkkkkkk.xxx. Volendo si possono anche copiare dei file nel dischetto.
Solitamente, l'invito della shell torna a disposizione prima che le operazioni di scrittura siano state completate, e questo a causa della presenza di una memoria cache. Anche per questo motivo, il dischetto non può essere estratto semplicemente alla fine delle operazioni che con questo si vogliono svolgere.

Finché il dischetto risulta montato, si può trattare come parte del file system globale.

```
# ls -l /mnt
      total 403
      -rwxr-xr-x 1 root root 412516 set 3 11:00 ammini~1.zip
      -rwxr-xr-x 1 root root      0 set 9 14:33 miaprova.txt
```

Trattandosi di un dischetto con un file system Dos-FAT, tutti i file hanno gli stessi permessi. L'utente proprietario di tutti i file è root essendo stato lui a montare il dischetto. I nomi dei file vengono troncati.
Volendo utilizzare un dischetto Dos-FAT per memorizzare nomi lunghi, nello stesso modo in cui fa Windows 95/98, si poteva montare facendo riferimento al tipo di file system `vfat`.
Una volta terminato il lavoro con il dischetto, lo si smonta:

```
# umount -t msdos /dev/fd0 /mnt
```

oppure:

```
# umount /mnt
```

Ricordarsi che non è possibile smontare un disco se prima non è terminata l'attività con questo. Per cui, se la directory corrente è posizionata su una directory appartenente al file system del disco che si vuole smontare, non si riesce a eseguire l'operazione di distacco.

Esercitazione:

Provare a montare un dischetto formattato in Windows (file system `vfat`). Crearci un file attribuendogli un nome molto lungo. Esaminare il contenuto del dischetto. Smontare il dischetto.

6.8 NFS

Il servizio di NFS (*Network File System*) permette la condivisione dei file system in rete. Il suo funzionamento è molto semplice, infatti prevede un server sul quale si vanno a configurare le directory da condividere e, sui vari client, occorre un semplice `mount` per rendere disponibile quella directory.

Le funzionalità più utili offerte da NFS sono:

- possibilità di concentrare i dati a cui spesso accedono tutti gli utenti su un host centrale e di renderli disponibili ai client al momento del boot;
- possibilità di conservare su un unico host tutti i dati ed i programmi che occupano molto spazio su disco.

Per poter condividere file attraverso NFS, sia come client che come server, occorre includere il supporto al file system NFS nel kernel.
Per controllare che questo supporto esista, è sufficiente leggere il contenuto del file raggiungibile tramite: `/proc/filesystems`.
L'esempio seguente rappresenta una situazione in cui è possibile accedere a file system NFS (è la riga `nodev nfs` a rivelarlo).

```
        ext2
        …
        msdos
nodev   proc
        …
nodev   nfs
        …
nodev   smbfs
        iso9660
```

6.8.1 Lato Server

Di seguito sono elencati i punti da seguire per configurare il servizio dal lato server:

1. (Ri)Controllare se il servizio NFS è supportato dal kernel quindi leggere il contenuto del file `/proc/filesystems` e controllare la presenza della riga `nodev nfs`.

2. Lanciare il demone `portmapper` (di solito viene lanciato automaticamente all'avvio del sistema).

3. Lanciare il demone `rpc.mountd` che si occupa di gestire il montaggio dei file system di rete dal lato del server. Mantiene il file `/etc/rmtab` che elenca i montaggi attivi. Tuttavia, non è garantito che il contenuto di questo file sia esatto, per cui non lo si può utilizzare per determinare con certezza quali siano le connessioni in corso.

4. Lanciare il demone `rpc.nfsd` che si occupa di gestire le richieste, da parte dei clienti, per i servizi NFS. Deve essere in funzione nel server. Viene avviato generalmente dalla procedura di inizializzazione del sistema, subito dopo `mountd`. Il funzionamento di questo programma dipende dal file `/etc/exports`.

5. Configurare il file `/etc/exports`.

6.8.2 /etc/exports

Il file `/etc/exports` contiene l'indicazione delle porzioni di file system locale da concedere in condivisione alla rete NFS.
Se il file manca o è vuoto, non viene concesso l'utilizzo di alcuna parte del file system locale all'esterno.
Ogni record del file è composto da:

- l'indicazione di una directory a partire dalla quale si concede la condivisione;
- una serie di nodi o reti cui viene concesso l'utilizzo di questa directory con l'eventuale specificazione di opzioni di accesso.

In pratica si utilizza la sintassi seguente:

```
directory_di_partenza [host][(opzioni)]...
```

Quando si fanno modifiche a questo file, è necessario riavviare il sistema di gestione del servizio NFS. Di solito è sufficiente inviare un segnale di aggancio ai demoni `mountd` e `nfsd`.
Se non si riesce in questo modo, si può provare eliminando del tutto i due processi, riavviandoli manualmente subito dopo.
Per maggiori dettagli si veda la pagina di manuale *exports*(5).

Identificazione degli elaboratori:
Gli elaboratori che possono accedere alla directory condivisa possono essere specificati in vari modi, alcuni dei quali sono elencati di seguito:

- **indicazione di un nodo singolo**
  quando si utilizza un nome o un indirizzo IP che fa riferimento ad un elaboratore specifico;
- **uso di caratteri jolly**
  possono essere utilizzati i caratteri jolly `*` e `?` per indicare un gruppo di **nomi** di elaboratore con una sola notazione, tenendo presente che questi simboli non possono sostituirsi ai punti di un nome di dominio;
- **rete IP**
  attraverso la notazione *`indirizzo_ip/maschera_di_rete`* è possibile indicare simultaneamente tutti gli elaboratori collocati all'interno della rete o della sottorete a cui si fa riferimento.

Opzioni principali:
Alcune delle opzioni da applicare sono elencate di seguito:

`ro`
Consente l'accesso in sola lettura. Questa è la modalità di funzionamento predefinita.

`rw`
Consente l'accesso in lettura e scrittura.

noaccess
Non concede la directory in condivisione. Può essere utile quando si vuole evitare l'accesso a una sottodirectory di una directory già concessa in condivisione.

link_relative
Trasforma i collegamenti simbolici assoluti, contenuti nel file system da condividere, in collegamenti relativi.
Si ricorda che un percorso è assoluto quando parte dalla directory radice (`/`), mentre è relativo quando parte dalla posizione corrente. Nello stesso modo, un collegamento simbolico può essere fatto utilizzando l'indicazione di un percorso assoluto o relativo. Quando si utilizza un file system di rete, difficilmente si ricostruisce la situazione del file system contenuto nell'elaboratore che opera da server, di conseguenza, gli eventuali collegamenti simbolici assoluti, non sarebbero più validi. Questo tipo di trasformazione risolve il problema ed è anche la modalità di funzionamento predefinita.

link_absolute
Mantiene il collegamenti simbolici così come sono.

root_squash
Si tratta di un'opzione di sicurezza, attraverso la quale si impedisce l'accesso come utente root. In pratica, quando un utente root presso un client utilizza il file system condiviso, viene trattato come utente *nobody*.
L'utente nobody corrisponde spesso al numero UID 65534 o -2. Tuttavia, questo utente non ha un numero UID standard, tanto che in alcuni sistemi si preferisce utilizzare un numero più basso di quelli assegnati agli utenti comuni.

no_root_squash
Non effettua la trasformazione dell'UID root e ciò è necessario quando si utilizzano client NFS senza disco fisso.

Per verificare l'utilizzo effettivo del servizio da parte dei client, è disponibile il programma `showmount`:

showmount [*opzioni***] [***host***]**

Questo si presta anche all'utilizzo dal lato client per conoscere le directory esportate da un server NFS. In particolare è possibile utilizzare alcune opzioni:

-a | `--all`
Elenca i clienti che utilizzano il proprio servizio e anche le directory che questi hanno montato.

-e | `--exports`
Elenca le directory esportate dal server locale o dal server remoto (se indicato come ultimo argomento del comando).

Esempi:

`/usr *.brot.dg(ro)`
Concede ai nodi del dominio `brot.dg` l'accesso in lettura alla directory `/usr/` e seguenti.

`/roggen.brot.dg(ro, root_squash)`
Concede a `roggen.brot.dg` di accedere in sola lettura a partire dalla directory radice, escludendo i privilegi dell'utente `root`.

`/home roggen.brot.dg(rw) weizen.mehl.dg(rw)`
Concede a `roggen.brot.dg` e a `weizen.mehl.dg` di accedere in lettura e scrittura alla directory `/home/`.

```
/home       10.10.10.11(rw)
/condivisa  10.10.10.11(rw, root_squash)
```
La prima permette di esportare la cartella `/home` al client 10.10.10.11 con diritti di lettura/scrittura. La seconda riga di questo file permette di esportare al client 10.10.10.11 la cartella `/condivisa` in lettura/scrittura e, l'opzione `root_squash`, indica di trattare l'utente *root* dei vari client come l'utente *nobody* nell'ambito del file system di rete.

6.8.3 Lato Client

Dal lato client non occorrono programmi demone: basta il normalissimo `mount`. Il montaggio di un file system di rete avviene in modo analogo a quello di una normale unità di memorizzazione, con una sintassi leggermente diversa:

`mount -t nfs` *`host_remoto`*`:`*`directory_remota`* `directory_locale [`*`options`*`]`

La notazione sopra riportata rappresenta la porzione di file system remoto cui si vuole accedere, attraverso l'indicazione simultanea dell'elaboratore e della directory di partenza.
Supponendo che l'elaboratore `mio.server.it` conceda l'utilizzo della directory `/usr/` e successive, l'elaboratore `mio.client.it` potrebbe connettersi attraverso il programma `mount` nel modo seguente:

```
mount -t nfs mio.server.it:/usr /usr
```

Inoltre, nell'elaboratore `client` si potrebbe aggiungere una riga nel file `/etc/fstab` in modo da automatizzarne la connessione.

```
mio.server.it:/usr  /usr         nfs      defaults
```

Sia attraverso il programma `mount` (preceduto dall'opzione `-o`), che nel file `/etc/fstab` (nel campo delle opzioni) possono essere specificate delle opzioni particolari riferite a questo tipo di file system:

`rsize``=`*`n`* e **`wsize`**`=`*`n`*
Permette di specificare la dimensione dei datagram (dal momento che si tratta del protocollo UDP, che è di tipo non connesso) utilizzati rispettivamente in lettura e scrittura da parte del client NFS. Il valore predefinito è di 1024 byte.

`timeo``=`*`n`*
Permette di definire il valore di *timeout*, espresso in decimi di secondo, per il completamento delle richieste. In pratica, se entro quel tempo non si ottiene una conferma, si verifica un *minor timeout* e l'operazione viene ritentata con una durata di

timeout doppia. Quando si raggiunge un *timeout* massimo di 60 secondi si verifica un *major timeout*. Il valore predefinito è sette, corrispondente a 0,7 secondi.

`hard`
Stabilisce che la connessione deve essere ritentata all'infinito, anche dopo un *major timeout*. È la modalità di funzionamento predefinita.

`soft`
Stabilisce che venga generato un errore di I/O non appena si verifica un *major timeout*.

`intr`
Permette l'interruzione di una chiamata NFS attraverso l'uso di segnali. Può essere utile per interrompere una connessione quando il server non risponde.

CAPITOLO 7

PERMESSI

I sistemi UNIX sono 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 è quell'utente che possiede speciali privilegi sulla macchina:

- può leggere tutti i file del sistema prescindendo dallo schema dei permessi;
- può gestire il meccanismo degli account e dei permessi come vuole;
- …

L'utente comune è quello che utilizza il sistema con delle restrizioni: ad esempio è impossibilitato nell'accedere a dati che non lo riguardano (dati del superutente o di altri utenti).

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

Per poter utilizzare un sistema di questo tipo, occorre essere stati registrati, ovvero, è necessario avere ottenuto un **account**. Dal punto di vista dell'utente, l'*account* è 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 ad un gruppo di utenti.

7.1 Gestione permessi

I file di un file system UNIX appartengono simultaneamente ad un utente e ad un gruppo di utenti. Per questo si parla di utente e gruppo proprietari, oppure semplicemente di proprietario e di gruppo. L'utente proprietario può modificare i permessi di accesso ai suoi file, limitando questi anche per se stesso. In particolare 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, ammesso che si tratti di un eseguibile binario o di uno script di qualunque tipo.

Per le *directory*:

- l'accesso in lettura permette di leggerne il contenuto, e cioè di poter conoscere l'elenco dei file in esse contenuti (di qualunque tipo essi siano). Quindi la mancanza del permesso di accesso in lettura impedisce di poter effettuare un **`ls`** sulla directory.
- 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. Quindi la mancanza del permesso di accesso in esecuzione impedisce di eseguire il comando **`cd`** sulla directory.

I permessi di un file consentono di attribuire privilegi differenti per gli utenti, a seconda che si tratti del proprietario del file, di utenti appartenenti al gruppo proprietario (dove per gruppo proprietario si intende quello dell'utente proprietario), oppure si tratti di utenti diversi. Così,

per ogni file, un utente può 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 ottale.

7.1.1 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 (**-**):

```
rw- r-- ---
```

I permessi possono essere di tre tipi:

- **r** lettura;
- **w** scrittura;
- **x** esecuzione.

I primi tre caratteri della stringa rappresentano i privilegi concessi al proprietario stesso; i tre caratteri successivi rappresentano i privilegi degli utenti appartenenti al gruppo; il gruppo finale di tre caratteri rappresenta i privilegi concessi agli altri utenti:

```
rwx  rw-  r--
 |    |    |
 |    |   altri
 |    |
 |  utenti gruppo
 |
proprietario
```

Esempi:

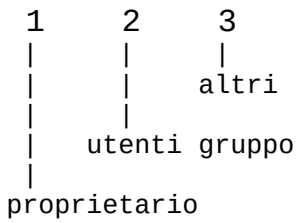
rw-r--r--
L'utente proprietario può accedervi in lettura e scrittura, mentre sia gli appartenenti al gruppo che gli altri utenti possono solo accedervi in lettura.

rwxr-x---
L'utente proprietario può 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 può accedervi in lettura e scrittura, mentre tutti gli altri non possono accedervi affatto.

7.1.2 Permessi espressi in forma numerica

I permessi possono essere rappresentati attraverso un numero ottale composto da una serie di tre cifre numeriche: la prima rappresenta i privilegi dell'utente proprietario, la seconda quelli del gruppo e la terza quelli degli altri utenti:



Per quanto riguarda la corrispondenza permessi-numero:

Permesso	**Numero**
Lettura	4
Scrittura	2
Esecuzione	1

Il numero che rappresenta il permesso attribuito ad un tipo di utente, si ottiene sommando i numeri corrispondenti ai privilegi che si vogliono concedere.

Notare che se il numero non utilizza tutte le cifre, si intende che manchino quelle anteriori e che queste siano semplicemente azzerate.

Esempi:

`644`
L'utente proprietario può 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 può 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 può accedervi in lettura e scrittura (4+2), mentre tutti gli altri non possono accedervi affatto.

7.1.3 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 modalità:

- **SUID** (*Set User ID*) attiva il numero dell'utente (UID) durante l'esecuzione, ovvero, attribuisce all'eseguibile in funzione i privilegi dell'utente a cui appartiene;
- **SGID** (*Set Group ID*) attiva il numero del gruppo (GID) durante l'esecuzione, ovvero, attribuisce all'eseguibile in funzione i privilegi del gruppo a cui appartiene;
- **Sticky** (*Save Text Image*) se si tratta di un eseguibile, durante l'esecuzione salva l'immagine testo nella memoria virtuale.

Si tratta di attributi speciali che riguardano prevalentemente i file eseguibili. E' per questo che spesso i permessi espressi in forma numerica (ottale) sono di quattro cifre (in totale 12 bit), con la prima che riguarda gli S-bit e normalmente è azzerata.
L'indicazione della presenza di questi bit attivati può 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 può osservare, questa indicazione prende il posto del permesso in esecuzione. Nel caso in cui il permesso in esecuzione corrispondente non sia attivato, la lettera ('**s**' o '**t**') appare maiuscola.

Importanza del bit SUID
Problema: Come può un utente senza i diritti di super-user, modificare la propria password senza interagire direttamente con l'amministratore del sistema? Infatti basta fare:

```
$ ls -l /etc/passwd

      -rw-r--r--   root   root   /etc/passwd
```

per rendersi conto che il file `/etc/passwd` è inaccessibile in scrittura da un utente normale.

Soluzione: Se si fa un **`ls -l`** di `/usr/bin/passwd` si ottiene:

```
$ ls -l /etc/bin/passwd

      -r-sr-xr-x   root   root   /usr/bin/passwd
```

Da notare il permesso **`s`** al posto del solito permesso **`x`**. Questo perché il programma `/usr/bin/passwd` è un programma set-user-ID dell'utente root e possiede il permesso di esecuzione **`x`** per qualsiasi utente. Ciò significa che il file è eseguibile e che qualsiasi utente può farlo girare come se lo stesse eseguendo con i privilegi del superuser. A questo punto poiché il file `/etc/passwd` possiede il permesso di accesso in scrittura per l'utente root, gli utenti normali possono utilizzarlo per cambiare la loro password (attraverso, appunto, il programma `/usr/bin/passwd` cioè **`passwd`**).
Anche se il meccanismo SUID è "*un'idea semplice ed allo stesso tempo elegante*" (Dennis Ritchie), bisogna dire che è anche un meccanismo molto pericoloso per la sicurezza del sistema. E' per questo che se ne sconsiglia l'uso soprattutto ai principianti.

S-bit e le directory
I 3 bit iniziali della modalità dei permessi meritano un po' di attenzione quando si tratta di directory, e non solo di file eseguibili.
La directory che abbia il bit *Sticky* attivo (**`d--x--x--t`**) non consente la cancellazione e la ridenominazione di un file da parte di un utente diverso da quello proprietario, anche se questo tentativo viene fatto da chi ha il permesso in scrittura sulla directory. Il bit Sticky viene attribuito generalmente alla directory `/tmp/` (oltre che a `/var/tmp/`) quando questa risulta accessibile da ogni utente in tutti i modi: **`drwxrwxrwt`**. Ciò permette di evitare che i file possano essere cancellati o rinominati da utenti diversi dai proprietari.
La directory con il bit *SGID* attivo (**`d--x--s--x`**) fa in modo che i file (e le directory) che verranno creati al suo interno appartengano al gruppo della directory stessa.

7.1.4 Maschera dei permessi: umask

Quando viene creato un file, questo appartiene automaticamente all'utente che lo crea ed al gruppo principale dell'utente stesso. I permessi gli vengono attribuiti in base alla maschera dei permessi (*umask*). Questa maschera rappresenta i permessi che non vengono attribuiti.
Di solito, il suo valore è tale da non attribuire il permesso di scrittura né al gruppo proprietario, né agli altri utenti. Il valore di questa maschera può essere modificato attraverso un comando interno di shell: **`umask`**.

7.2 chown

`chown [`*`opzioni`***`] [`***`utente`***`][:|.][`***`gruppo`***`]`** `file...`
Cambia la proprietà dei file. Se viene fornito solo il nome dell'utente o il suo numero UID, questo diviene il nuovo proprietario dei file. Se il nome dell'utente, o il suo numero, è seguito da due punti (`:`) oppure dal punto (`.`) e dal nome o dal numero di un gruppo (GID), vengono cambiate la proprietà dell'utente e la proprietà del gruppo. Se dopo '`:`' o '`.`' non segue il nome del gruppo, viene attribuito il gruppo a cui appartiene l'utente. Se prima di '`:`' o '`.`' non viene indicato il nome dell'utente, viene cambiata solo la proprietà del gruppo.

Opzioni principali:

> **`-R`**
> Esegue l'operazione anche nelle sottodirectory.

Esempi:

> `#` **`chown federico prova`**
> L'utente root cambia l'utente proprietario del file `prova`, facendo in modo che diventi `federico`.
>
> `#` **`chown fede.users prova`**
> L'utente root cambia l'utente ed il gruppo proprietario del file `prova`, facendo in modo che diventino rispettivamente `fede` e `users`.
>
> `$` **`chown.users prova`**
> L'utente proprietario del file `prova` cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

7.3 chgrp

chgrp [*opzioni*] *gruppo file...*
Cambia il gruppo proprietario di file e directory. Il gruppo, nell'argomento del comando, può essere espresso con il nome o con il numero GID. È equivalente a `chown` quando non si specifica l'utente.

Opzioni principali:

> **-R**
> Esegue l'operazione anche nelle sottodirectory.

Esempi:

> $ **chgrp users mio_file**
> L'utente proprietario del file `mio_file` cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

7.4 chmod

chmod [*opzioni*] *modalità_dei_permessi file...*
Cambia la modalità dei permessi sui file indicati come argomento. Le modifiche della modalità dei permessi avvengono in base alle specifiche indicate nell'argomento precedente all'elenco dei file, e si possono esprimere con la sintassi seguente:

> **[**u|g|o|a**]**...**[{**+|-|=**}{**r|w|x|X|s|t|u|g|o**}**...**][**,...**]**

Una combinazione delle lettere `u`, `g`, `o`, `a` controlla il tipo di utenti a cui si vuole riferire il cambiamento di permesso. I segni **+**, **-**, **=** indicano il tipo di cambiamento sui permessi; il gruppo finale di lettere `r`, `w`, `x`, `X`, `s`, `t`, `u`, `g`, `o` indica i permessi su cui agire.

Utenti (ugoa):

Simbolo	Descrizione
u	Utente proprietario del file.
g	Gruppo proprietario del file.
o	Gli altri utenti.
a	Tutti.

Se l'indicazione degli utenti su cui intervenire non viene fornita, la variazione agisce in funzione della maschera dei permessi che può essere modificata attraverso il comando di shell `umask`. In pratica, la variazione riguarda tutti i tipi di utente, a esclusione dei bit attivati nella maschera dei permessi.

Variazione dei permessi (+-=):

Simbolo	Descrizione
+	Le modalità dei permessi indicate vengono aggiunte.
-	Le modalità dei permessi indicate vengono tolte.
=	Le modalità dei permessi vengono modificate in modo da diventare esattamente come indicato.

Permessi (rwxXstugo):

Simbolo	Descrizione
r	Permesso di accesso in lettura.
w	Permesso di accesso in scrittura (modifica).
x	Permesso di esecuzione o di attraversamento se si tratta di directory.
X	Come **x**, ma interviene sulle directory e solo sui file che hanno già un permesso in esecuzione per un utente qualunque. In pratica, si cerca di intervenire solo sui file per i quali il permesso di esecuzione (o di attraversamento) può avere senso.
s	Riguarda solo i file eseguibili (ed eventualmente le directory). Attiva il bit SUID, o il bit SGID a seconda che il cambiamento intervenga sull'utente, sul gruppo o su entrambi.
t	Riguarda solo i file eseguibili (ed eventualmente le directory). Attiva il bit Sticky.
u	Attribuisce le stesse modalità dei permessi che ha già l'utente proprietario di quel file.
g	Attribuisce le stesse modalità dei permessi che ha già il gruppo proprietario di quel file.
o	Attribuisce le stesse modalità dei permessi che hanno già gli altri utenti per quel file.

Non è possibile cambiare i permessi dei collegamenti simbolici: se si interviene su un collegamento
simbolico si agisce in realtà sul file di destinazione.

Opzioni principali:

-R
Esegue l'operazione anche nelle sottodirectory.

Esempi:

$ **chmod -R go-rwx ~/***
Toglie sia al gruppo che agli altri utenti la possibilità di accedere in qualunque modo ai file della propria directory personale e anche nelle sottodirectory successive.

CAPITOLO 8

GESTIONE UTENZA

Le informazioni sugli utenti registrati nel sistema sono raccolte principalmente all'interno di `/etc/passwd`. Anche se il nome suggerisce che debba contenere le password, in realtà il suo scopo è più ampio e la sua accessibilità in lettura è essenziale per tutti i programmi che hanno qualcosa a che fare con gli utenti. Per questo motivo, in molti sistemi (ad esempio Linux) si preferisce trasferire le password in un altro file con meno possibilità di accesso: `/etc/shadow`.
Il file `/etc/group` permette di raccogliere le notizie sui gruppi e in particolare di stabilire la possibile appartenenza da parte di un utente a più gruppi.

8.1 Password cifrate

Le password utilizzate per accedere vengono annotate in forma cifrata, nel file `/etc/passwd`, oppure nel file `/etc/shadow`. La cifratura genera una stringa che può essere usata per verificare la correttezza della password, mentre da sola, questa stringa non permette di determinare quale sia la password di origine. In pratica, data la password si può determinare la stringa cifrata, ma dalla stringa cifrata non si ottiene la password.
La verifica dell'identità avviene quindi attraverso la generazione della stringa cifrata corrispondente: se corrisponde a quanto annotato nel file `/etc/passwd`, oppure nel file `/etc/shadow`, la password è valida, altrimenti no.

L'algoritmo usato per generare la password cifrata non è uguale in tutti i sistemi. Per quanto riguarda i sistemi UNIX si distinguono due possibilità: l'algoritmo tradizionale DES, che accetta password con un massimo di otto caratteri, e l'algoritmo MD5 che al contrario non pone limiti.
La gestione dell'algoritmo di cifratura delle password è a carico della funzione **`crypt()`** (descritta in *crypt*(3)).

8.2 Utenti e gruppi

I nuovi utenti possono essere aggiunti solo da parte dell'utente root, ma poi possono essere loro stessi a cambiare alcuni elementi della loro registrazione. Il più importante è naturalmente la password.

8.2.1 adduser, useradd

`adduser [`(*opzioni*)**`]`**`...`
`useradd [`(*opzioni*)**`]`**`...`
Il programma in questione può avere due nomi alternativi: **`adduser`** o **`useradd`**. Questo permette all'utente root di aggiungere un nuovo utente all'interno del file `/etc/passwd`, assegnandogli un UID, un GID, una password, una shell e creando la sua directory personale.
Per mantenere la compatibilità con alcuni vecchi programmi, il nome dell'utente non deve superare gli otto caratteri. Inoltre, è opportuno limitarsi all'uso di lettere non accentate e di numeri; qualunque altro simbolo, compresi i segni di punteggiatura, potrebbero creare problemi di vario tipo. Si consiglia di utilizzare **`adduser –silent`** in modo che le informazioni richieste per l'inserimento del nuovo utente siano solo quelle necessarie.

8.2.2 passwd

`passwd` [*utente*]
Permette di cambiare la password registrata all'interno di `/etc/passwd` (oppure all'interno di `/etc/shadow`). Solo l'utente root può cambiare la password di un altro utente. Gli utenti comuni devono utilizzare una password non troppo breve composta sia da maiuscole che minuscole o simboli diversi. Alcune password simili al nome utilizzato per identificare l'utente, non sono valide.
Se non si dispone di un mezzo per l'inserimento di un nuovo utente, come quello fornito da `adduser`, è possibile aggiungere manualmente un record all'interno del file `/etc/passwd` senza l'indicazione della password che poi potrà essere specificata attraverso l'eseguibile `passwd`.

8.2.3 groups

`groups` [*utente...*]
Visualizza i gruppi ai quali l'utente o gli utenti appartengono. Il risultato è equivalente al comando seguente:

```
id -Gn [nome_utente]
```

8.2.4 id

`id` [*opzioni*] [*utente*]
Visualizza il numero UID (*User ID*) e il numero GID (*Group ID*) dell'utente selezionato o di quello corrente.

Opzioni:

`-u |` `--user`
Emette solo il numero dell'utente (UID).

`-g |` `--group`
Emette solo il numero del gruppo (GID).

`-G |` `--groups`
Emette solo i numeri dei gruppi supplementari.

`-n |` `--name`
Emette il nome dell'utente, del gruppo o dei gruppi, a seconda che sia usato insieme a `-u`, `-g` o `-G`.

`-r |` `--real`
Emette i numeri UID o GID reali invece di quelli efficaci (ammesso che ci sia differenza). Si usa insieme a `-u`, `-g` o `-G`

8.3 etc/passwd

È un elenco di utenti, parole d'ordine, directory *home* (directory personali), shell e altre informazioni personali utilizzate. La struttura dei record (righe) di questo file è molto semplice:

```
utente:parola_d'ordine_cifrata:uid:gid:dati_personali:directory_home:shell
```

Segue la descrizione dei campi:

1. *utente*
   È il nome utilizzato per identificare l'utente logico che accede al sistema.

2. *parola_d'ordine_cifrata*
   È la parola d'ordine cifrata. Se questa indicazione manca, l'utente può accedere senza indicare alcuna parola d'ordine. Se questo campo contiene un asterisco (`*`) l'utente non può accedere al sistema. Con questa tecnica è possibile impedire temporaneamente l'accesso, con la possibilità di ripristinarlo successivamente con la stessa parola d'ordine, togliendo semplicemente l'asterisco.

3. *uid*
   È il numero identificativo dell'utente (*User ID*).

4. *gid*
   È il numero identificativo del gruppo a cui appartiene l'utente (*Group ID*).

5. *dati_personali*
   Di solito, questo campo contiene solo l'indicazione del nominativo completo dell'utente (nome e cognome), ma può contenere anche altre informazioni.

6. *directory_home*
   La directory assegnata all'utente.

7. *shell*
   La shell assegnata all'utente.

Quindi riassumendo si avrà:

```
rossi: *: 100: 0: Paolo Rossi: /usr/users/rossi: /bin/bash
|      |     |   |           |             |                |
|  Password  |   |           |             |                |
|  cifrata   |   |           |             |                |
|            |  Group ID     |             |                |
Utente       |               |             |                |
             |               |             |    Interprete dei comandi
          User ID            |             |
                             |         Home directory
                             |
                      Informazioni utente
```

Da notare che in quei sistemi in cui è attivata la gestione delle password shadow (ad esempio Linux), rientra nella manipolazione della password anche il file `/etc/shadow`. Questo serve a contenere le parole d'ordine cifrate, togliendole dal file `/etc/passwd`. Facendo in questo

modo, è possibile inibire la maggior parte dei permessi di accesso a questo file, proteggendo le parole d'ordine. Non è però possibile impedire l'accesso in lettura del file `/etc/passwd` in quanto fornisce una quantità di informazioni sugli utenti, indispensabili a molti programmi.

Esempi:

> `tizio:724AD9dGbG25k:502:502:Tizio Tizi:/home/tizio:/bin/bash`
> L'utente `tizio` corrisponde al numero UID `502` e al numero GID `502`; si chiama Tizio Tizi; la sua directory personale è `/home/tizio/`; la sua shell è `/bin/bash`. Di questo utente, personalmente, non si conosce niente altro che il nome e il cognome. Il fatto che UID e GID corrispondano dipende da una scelta organizzativa dell'amministratore del sistema.
>
> `tizio:*:502:502:Tizio:/home/tizio:/bin/bash`
> Questo esempio mostra una situazione simile a quella precedente, ma la password è salvata in un altro file (di solito in Linux in `/etc/shadow`).

8.4 etc/group

È l'elenco dei gruppi di utenti. La struttura delle righe di questo file è la seguente:

```
gruppo:parola_d'ordine_cifrata:gid:lista_di_utenti
```

Segue la descrizione dei campi:

1. *gruppo*
   È il nome utilizzato per identificare il gruppo.

2. *parola_d'ordine_cifrata*
   È la parola d'ordine cifrata. Di solito non viene utilizzata e di conseguenza non viene inserita. Se è presente una parola d'ordine, questa dovrebbe essere richiesta quando un utente tenta di cambiare gruppo attraverso `newgrp`.

3. *gid*
   È il numero identificativo del gruppo.

4. *lista_di_utenti*
   È la lista degli utenti che appartengono al gruppo. Si tratta di un elenco di nomi di utente separati da virgole.

Quindi riassumendo si avrà:

```
utenti: *: 40: bianchi,rossi
|       |    |          |
|  Password  |          |
|  cifrata   |       Nomi utenti
|            |       del gruppo
Nome       Group ID
Gruppo
```

Esempi:

`group::502:federico`
Il gruppo `group` non ha alcuna parola d'ordine e ad esso appartiene solo l'utente `federico`.

`users::100:antonio,luca,marco`
In questo caso, gli utenti `antonio`,`luca`,`marco` appartengono al gruppo `users`.

8.5 Password shadow

Il meccanismo delle password shadow si basa su un principio molto semplice: nascondere le password cifrate ai processi che non hanno i privilegi dell'utente root. Infatti, nei sistemi in cui le password shadow non sono attivate, è il file `/etc/passwd`, leggibile a tutti i tipi di utenti, che contiene tali password cifrate. Il problema nasce dal fatto che è possibile scoprire la password degli utenti attraverso programmi specializzati che scandiscono un vocabolario alla ricerca di una parola che possa corrispondere alla password cifrata.

Con le password shadow attivate si aggiunge il file `/etc/shadow` a fianco del consueto `/etc/passwd`. In quest'ultimo vengono tolte le password cifrate e al loro posto viene inserita una **x** (minuscola) come nell'esempio seguente:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
...
fede:x:1000:1000:Federico:/home/federico:/bin/bash
...
```

Nel file `/etc/shadow`, oltre alle password cifrate, vengono inserite altre informazioni sulle utenze che permettono di aumentare la sicurezza del sistema.

8.5.1 /etc/shadow

La presenza del file `/etc/shadow` indica l'attivazione delle password shadow. I record di questo file sono organizzati in campi, separati attraverso il simbolo due punti, secondo la sintassi seguente:

```
utente:password:modifica:valid_min:valid_max:preavviso:tempo_riserva:termine:riservato
```

I campi che rappresentano una data possono contenere un numero intero che indica il numero di giorni trascorsi dal 1/1/1970, mentre quelli che rappresentano una durata, possono contenere un numero intero che esprime una quantità di giorni.
Gli altri campi sono:

1. *utente*
   Il nominativo dell'utente.

2. *password*
   La password cifrata, quella tolta dal file `/etc/passwd`.

3. *modifica*
   Data in cui è stata modificata la password per l'ultima volta.

4. *validità_minima*
   Numero di giorni di validità minima della password; entro questo tempo, l'utente non può cambiare la password.

5. *validità_massima*
   Numero di giorni di validità massima della password; prima che trascorra questo tempo, l'utente deve cambiare la password.

6. *preavviso*
   Numero di giorni, prima della scadenza della password, durante i quali l'utente viene avvisato della necessità di modificarla.

7. *tempo_di_riserva*
   Durata massima di validità dell'utenza dopo che la password è scaduta.

8. *termine*
   Data di scadenza dell'utenza.

9. *riservato*
   Riservato per usi futuri.

A titolo di esempio, viene mostrato un caso sufficientemente completo nella figura seguente:

```
rossi:wsLHjp.FutW0s:10459:0:30:7:10:10824:
|      |              |   |  | |  |   |
|    password         |   |  | |  |   termine dell'utenza il
|    cifrata          |   |  | |  |   giorno 21/08/1999
|                     |   |  | |  |
utente                |   |  | |  dieci giorni di riserva
                      |   |  | |
                      |   |  | sette giorni di preavviso
                      |   |  |
      la password è stata | trenta giorni di validità massima
     modificata il giorno |
     21/08/1998           l'utente può cambiare password
                          in ogni momento
```

Perché il sistema delle password shadow possa dare la sicurezza che promette, è necessario che il file `/etc/shadow` appartenga all'utente root e abbia esclusivamente il permesso di lettura per il proprietario.

8.6 Procedimento manuale di gestione account

Creazione di un utente

Si esaminerà ora il procedimento manuale utilizzato per aggiungere un utente senza utilizzare i comandi per la gestione dell'utenza forniti nei sistemi UNIX. Ovviamente per fare ciò bisogna loggarsi come root.
I passi da seguire in un sistema che utilizza le password shadow sono:

1. Editare il file `/etc/passwd` aggiungendo una riga per il nuovo utente:
   `federico:x:501:501:Federico:/home/federico:/bin/bash`
   E' importante dare un User ID ed un Group ID superiore a quelli attribuiti all'ultimo utente inserito in modo da evitare conflitti con altri ID già occupati.

2. Editare `/etc/group` aggiungendo un nuovo gruppo per il nuovo utente:
   `federico:x:501:`
   Notare che non è necessario aggiungere un nuovo gruppo: si può anche aggiungere il nome del nuovo utente ad un gruppo di un altro utente. Cioè:
   `luca:x:500:federico`

3. Editare `/etc/shadow` aggiungendo una riga per il nuovo utente:
   `federico::::::::`

4. Creare la home directory del nuovo utente (`/home/`*`nomeutente`*):
   `# mkdir /home/federico`

5. Ricreare l'ambiente base nella nuova home directory:
   `# cp /etc/skel/.* /home/federico/`
   La directory `/etc/skel/` viene utilizzata normalmente come directory personale tipica per i nuovi utenti. Quando si aggiunge un nuovo utente e gli si prepara la sua directory personale, viene copiato all'interno di questa il contenuto di `/etc/skel/`. Il nome *skel* sta per *skeleton*, cioè scheletro. In effetti rappresenta lo scheletro di una nuova directory personale.
   Il contenuto di questa directory è molto importante: infatti è realizzato in modo che ogni nuovo utente trovi subito una serie di file di configurazione necessari per utilizzare le shell previste nel sistema.

6. Modificare il proprietario della home directory:
   `# chown -R `*`nomeutente:nomegruppo`*` /home/`*`nomeutente`*
   cioè:
   `# chown -R federico:federico /home/federico`

7. Modificare i permessi della home:
   `# chmod 700 /home/`*`nomeutente`*
   cioè, in questo caso:
   `# chmod 700 /home/federico`

8. Modificare la password dell'utente:
   `# passwd `*`nomeutente`*
   in questo caso:
   `# passwd federico`

Per verificare la correttezza di quanto fatto, provare su un nuovo terminale a loggarsi con il nome e la password dell'utente creato.

Eliminazione di un utente

L'eliminazione di un utente dal sistema non è gestibile attraverso un programma di servizio standard di uso generale: la particolare distribuzione Linux può fornire degli strumenti adatti, oppure si deve agire manualmente.
In alcune distribuzioni Linux (come la RedHat) è presente il comando:

```
userdel nomeutente
```

con il quale è possibile cancellare automaticamente un utente. Per fare ciò, ovviamente è necessario avere i permessi di amministratore del sistema (root). Utilizzando `userdel nomeutente` non viene però cancellato il contenuto della home dell'utente (cioè `/home/nomeutente`): bisognerà quindi procedere manualmente.

Se si cancella un utente ancora connesso, tale utente rimarrà connesso con i suoi stessi privilegi. Le modifiche avverranno solo al prossimo tentativo di login da parte di quell'utente.

Poiché raramente si hanno a disposizione dei comandi per l'eliminazione di un utente, si descrive anche come si può intervenire manualmente per cancellare un utente. Fondamentalmente si tratta di agire su due punti:

- l'eliminazione dell'utente dai file `/etc/passwd` e `/etc/group` (ed eventualmente anche da `/etc/shadow`);
- l'eliminazione dei file appartenenti a quell'utente.

I file di un utente possono trovarsi ovunque gli sia stato consentito di scriverli. In particolare:

- la directory personale;
- la directory delle caselle postali (`/var/mail/` o in certi casi `/var/spool/mail/`, a meno che questa non sia già inserita direttamente nelle directory personale);
- la directory `/var/spool/cron/` per eventuali applicazioni a esecuzione pianificata.

Per elencare tutti i file appartenenti a un certo utente, è possibile usare il programma `find` in uno dei seguenti modi:

```
find / -uid numero_utente -print
find / -user utente –print
```

anche se poi bisogna prestare attenzione a cosa cancellare.
In generale i passi da seguire per cancellare un utente sono i seguenti (si farà riferimento all'utente creato con la procedura manuale vista sopra):

1. Rimuovere la riga riguardante l'utente da cancellare in `/etc/passwd`:
   `federico:x:501:501:Federico:/home/federico:/bin/bash`

2. Rimuovere la riga riguardante l'utente da cancellare in `/etc/group`:
   `federico:x:501:`
   oppure togliere l'utente dalla lista di un gruppo.

3. Rimuovere la riga riguardante l'utente da cancellare in `/etc/shadow`:
   `federico:$1$9rbw1deB$...:11944:…:::`

4. Rimuovere la directory home dell'utente:
   `# rm -rf /home/federico/`

CAPITOLO 9

PIANIFICAZIONE DEI PROCESSI

La pianificazione dei processi riguarda l'esecuzione di processi in date e orari stabiliti. Il mezzo attraverso il quale si controlla l'avvio di un processo in un momento stabilito è fornito dal sistema Cron, ovvero dal demone omonimo (`cron`).

Cron è il sistema che si occupa di eseguire, attraverso il demone `cron`, dei comandi in momenti determinati in base a quanto stabilito all'interno della sua configurazione, rappresentata dai file ***crontab***. Questi file possono essere diversi, solitamente uno per ogni utente che ha la necessità di pianificare l'esecuzione di alcuni comandi e uno generale per tutto il sistema.
I file crontab vengono creati attraverso il programma `crontab` oppure editati manualmente. L'esistenza del programma `crontab` permette di evitare che i file crontab siano accessibili a utenti che non ne siano i proprietari. Inoltre, non è necessario preoccuparsi di avvisare il demone `cron` dell'avvenuto cambiamento nella situazione dei piani di esecuzione.
L'output dei comandi che il sistema Cron mette in esecuzione, se non è stato rediretto in qualche modo, per esempio a `/dev/null` o a un file, viene inviato con un messaggio di posta all'utente cui appartiene il file crontab.
Il demone `cron` viene avviato di norma durante la procedura di inizializzazione del sistema. Di questo demone ne esistono almeno due tipi diversi per i sistemi GNU: Vixie Cron e Dillon's Cron, dai nomi dei loro autori. Nelle sezioni seguenti si fa riferimento in particolare al sistema Cron di Paul Vixie.
Vedere *cron*(1), *crontab*(1) (o *crontab*(8)) e *crontab*(5).

9.1 cron

`cron`

`cron` è un demone funzionante sullo sfondo (*background*) che si occupa di interpretare i file crontab collocati in `/var/spool/cron/crontabs/` oltre a uno speciale, `/etc/crontab`, il cui formato è leggermente diverso.
Dal momento che la sostanza del funzionamento di questo programma sta nell'interpretazione dei file crontab, le altre notizie sul suo utilizzo sono riportate in occasione della presentazione di questi file.

9.2 crontab

`crontab [`*`opzioni`***`]`**

`crontab` permette di creare o modificare il file crontab di un utente determinato. In particolare, solo l'utente root può agire sul file crontab di un altro utente. Di solito, `crontab` viene utilizzato con l'opzione `-e` per modificare o creare il file crontab.
I file crontab vengono poi utilizzati dal demone `cron` che si occupa di eseguire i comandi lì indicati.

Opzioni:

`[-u` ***`utente`*** **`]`** ***`file`***	Sostituisce il file crontab con il contenuto del file indicato come argomento.
`-l`	Visualizza il file crontab dell'utente.

-e	Crea o modifica il file crontab dell'utente.
-r	Cancella il file crontab dell'utente.

Esempi:

`$ crontab -e`
Inizia la modifica del file crontab dell'utente.

`$ crontab -l`
Visualizza il contenuto del file crontab dell'utente. Il suo contenuto potrebbe apparire come nel listato seguente:

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.1466 installed on Thu Aug 21 17:39:46 1997)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie
Exp $)
10 6 * * * echo "prova miaprova"
…
```

`$ crontab -r`
Elimina il file crontab dell'utente.

9.2.1 /var/spool/cron/*

I file contenuti nella directory `/var/spool/cron/` o `/var/spool/cron/crontabs/` sono i file crontab degli utenti comuni, creati generalmente attraverso il programma `crontab`. Ogni utente ha il proprio con il nome corrispondente all'utente stesso; i comandi contenuti al suo interno vengono eseguiti con i privilegi dell'utente proprietario del file crontab.
Le righe vuote sono ignorate e così anche quelle dove il primo carattere diverso da uno spazio lineare (sia spazi veri e propri che caratteri di tabulazione) è il simbolo `#`, che serve così a introdurre dei commenti. Un record significativo può essere un assegnamento di una variabile di ambiente o un comando Cron.
L'assegnamento di una variabile può avvenire nel modo consueto,

nome = valore

dove gli spazi attorno al segno di uguaglianza sono facoltativi e il valore assegnato può essere indicato eventualmente con virgolette (singole o doppie).
Il demone `cron` utilizza una serie di variabili di ambiente per determinare il proprio comportamento. Alcune di queste ricevono un valore predefinito dal demone `cron` stesso, ma tutte, tranne `LOGNAME`, possono essere modificate attraverso un assegnamento all'interno del file crontab.

Segue un elenco delle variabili più importanti:

SHELL
Di solito il valore iniziale è `/bin/sh` stabilendo così che i comandi di Cron devono essere eseguiti facendo uso della shell Bourne.

LOGNAME
Il valore iniziale è il nome dell'utente e non può essere modificato.

HOME
Il valore iniziale è la directory personale dell'utente.

`MAILTO`
Non viene preassegnata dal demone `cron` e se risulta definita, ma non vuota, viene utilizzata per determinare il destinatario dei messaggi di posta elettronica che vengono generati. Se il contenuto di questa variabile è la stringa nulla (`""`), non viene inviato alcun messaggio. Se la variabile non esiste, il destinatario dei messaggi di posta elettronica è lo stesso utente a cui appartiene il file crontab.

Un file crontab tipico può contenere solo comandi di Cron. Il formato di questo può essere riassunto brevemente nel modo seguente:

`data_orario comando`

Il comando viene eseguito attraverso la shell indicata all'interno della variabile `SHELL`, mentre l'indicazione data-orario si scompone in altri cinque campi:

```
minuti   ore   giorni_del_mese   mesi   giorni_della_settimana
```

I campi possono contenere un asterisco (`*`) e in tal caso rappresentano ogni valore possibile di quel campo. Per esempio, `* * * * *` rappresenta ogni minuto di ogni ora di ogni giorno del mese di ogni mese di ogni giorno della settimana.
A parte il caso degli asterischi, all'interno di questi campi si possono indicare dei valori numerici. Notare che i giorni della settimana sono identificati da zero a sette, dove sia zero che sette corrispondono alla domenica.
Per ognuno di questi campi, i valori possono essere indicati in vari modi con diversi significati:

- **Valori singoli**
  Un numero isolato all'interno di un campo indica che il comando deve essere eseguito quando l'orologio del sistema raggiunge quel valore. Per esempio, `10 6 * * *` rappresenta esattamente le ore 06:10 di ogni giorno.

- **Intervalli**
  Un intervallo, rappresentato da una coppia di numeri separati da un trattino, indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei valori possibili. Per esempio, `10 6 1-5 * *` rappresenta esattamente le ore 06:10 dei primi cinque giorni di ogni mese.

- **Elenchi**
  Un elenco, rappresentato da una serie di numeri separati da una virgola (senza spazi), indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei valori. Per esempio, `10 6 1-5 1,3,5 *` rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, marzo e maggio.
  Gli elenchi possono essere anche combinati con gli intervalli. Per esempio, `10 6 1-5 1-3,5-7 *` rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, febbraio, marzo, maggio, giugno e luglio.

- **Passo**
  Invece di indicare momenti precisi, è possibile indicare una ripetizione o un passo. Questo può essere rappresentato con una barra obliqua seguita da un valore e indica che il comando deve essere eseguito ogni volta che è trascorsa quella unità di tempo. Per esempio, `*/10 6 * * *` rappresenta le ore 06:10, 06:20, 06:30, 06:40, 06:50 e 06:00. In pratica, corrisponde a `0,10,20,30,40,50 6 * * *`.

Il passo può essere combinato opportunamente con gli intervalli. Per esempio, `0-30/10 6 * * *` rappresenta le 6:00, le 6:10, le 6:20 e le 6:30. In pratica, corrisponde a `0,10,20,30 6 * * *`.

Quello che appare dopo i cinque campi dell'orario viene interpretato come un comando da eseguire. Più precisamente, viene considerato tale tutto quello che appare prima della conclusione della riga o di un segno di percentuale (%). Quello che eventualmente segue dopo il primo segno di percentuale viene interpretato come testo da inviare allo standard input del comando stesso. Se all'interno del testo da inviare appaiono altri segni di percentuale, questi vengono trasformati in codici di interruzione di riga.
Segue un esempio commentato di file crontab:

```
# Utilizza «/bin/sh» per eseguire i comandi, indipendentemente da
# quanto specificato all'interno di «/etc/passwd».
SHELL=/bin/sh

# Invia i messaggi di posta elettronica all'utente «fede»,
# indipendentemente dal proprietario di questo file crontab.
MAILTO=fede

# Esegue 5 minuti dopo la mezzanotte di ogni giorno.
5 0 * * *       $HOME/bin/salvataggiodati

# Esegue alle ore 14:15 del primo giorno di ogni mese.
# L'output viene inviato tramite posta elettronica all'utente
«tizio».
15 14 1 * *     $HOME/bin/mensile

# Esegue alle 22 di ogni giorno lavorativo (da lunedì al venerdì).
# In particolare viene inviato un messaggio di posta elettronica a
«fede».
0 22 * * 1-5    mail -s "Sono le 22" fede%Fede,%%è ora di smettere!%

# Esegue 23 minuti dopo mezzanotte, dopo le due, dopo le quattro,...,
# ogni giorno.
23 0-23/2 * * * echo "Ciao ciao"

# Esegue alle ore 04:05 di ogni domenica.
5 4 * * 0       echo "Buona domenica"
```

9.2.2 /etc/crontab

Il file `/etc/crontab` ha un formato leggermente diverso da quello dei file crontab normali. In pratica, dopo l'indicazione dei cinque campi data-orario, si inserisce il nome dell'utente in nome del quale deve essere eseguito il comando indicato successivamente.
Nell'esempio seguente, tutti i comandi vengono eseguiti per conto dell'utente root, ovvero, vengono eseguiti con i privilegi di questo utente:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# Run any at jobs every minute
* * * * * root [ -x /usr/sbin/atrun ] && /usr/sbin/atrun

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 1 * * * root run-parts /etc/cron.daily
```

```
02 2 * * 0 root run-parts /etc/cron.weekly
02 3 1 * * root run-parts /etc/cron.monthly

# Remove /tmp, /var/tmp files not accessed in 10 days (240 hours)
41 02 * * * root /usr/sbin/tmpwatch 240 /tmp /var/tmp

# Remove formatted man pages not accessed in 10 days
39 02 * * * root /usr/sbin/tmpwatch 240 /var/catman/cat?
```

Una parte dell'esempio mostrato è abbastanza comune nelle varie distribuzioni GNU e merita una spiegazione aggiuntiva. A metà dell'esempio appare l'avvio del comando `run-parts` a cadenza oraria, giornaliera, settimanale e mensile. Per esempio, la direttiva:

```
01 * * * * root run-parts /etc/cron.hourly
```

avvia il comando `run-parts /etc/cron.hourly` ogni ora. `run-parts` è un programma (a volte realizzato in forma di script) che avvia tutti gli eseguibili contenuti nella directory indicata come argomento; per cui, `run-parts /etc/cron.hourly` serve ad avviare tutto quello che c'è nella directory `/etc/cron.hourly/`.
Quindi per inserire un'elaborazione nei momenti più comuni, basta mettere il programma o lo script relativo nella directory che rappresenta la cadenza desiderata.

9.3 Anacron

Cron è un sistema di pianificazione adatto principalmente per gli elaboratori che restano in funzione ininterrottamente per molto tempo; infatti, se non si accende mai l'elaboratore nell'intervallo di tempo in cui sarebbe previsto l'avvio di elaborazioni a cadenza giornaliera, settimanale o mensile, queste verrebbero automaticamente escluse. Per risolvere il problema e per garantire l'avvio di quelle elaborazioni, si può utilizzare Anacron.
Anacron è un sistema di pianificazione molto semplice, che permette soltanto di programmare l'esecuzione di elaborazioni determinate a cadenza giornaliera, o a multipli di giorni. La sua logica è molto semplice: utilizza un file di configurazione, `/etc/anacrontab`, concettualmente analogo al crontab di Cron, in cui si indica semplicemente l'intervallo in giorni per l'esecuzione di processi determinati. Per mantenere memoria di ciò che è stato fatto, utilizza dei file nella directory `/var/spool/anacron/`, annotando in che giorno ha eseguito un certo job per l'ultima volta.
Questo sistema è gestito in pratica dall'eseguibile `anacron`, che si comporta normalmente come un demone, che resta in funzione solo per il tempo necessario a completare il suo lavoro: il giorno successivo, verrà riavviato automaticamente dal sistema.

9.3.1 /etc/anacrontab

Il file `/etc/anacrontab` si utilizza per configurare il comportamento di Anacron. Il file può contenere la definizione di alcune variabili di ambiente ed è composto da una serie di record (righe), che descrivono i vari job da gestire. Come nel caso dei file crontab normali, le righe bianche e quelle vuote vengono ignorate; nello stesso modo sono ignorate le righe che iniziano con il simbolo `#`.

Il formato del record è il seguente:

```
n_giorni   n_minuti_ritardo   nome_attribuito_al_job   comando
```

I record che definiscono i job di Anacron sono composti da campi separati da spazi bianchi, Il significato dei vari campi è il seguente:

1. il primo campo è un numero che esprime la cadenza in giorni con cui deve essere eseguito il comando;

2. il secondo campo è un altro numero che esprime un ritardo in secondi, che deve essere atteso prima di cominciare;

3. il terzo campo attribuisce un nome al job;

4. l'ultimo campo è il comando corrispondente al job e, in questo caso particolare, può contenere spazi.

Il campo ritardo deve essere stabilito in modo tale da evitare che in un certo momento possano essere messi in funzione simultaneamente troppi processi, tenendo conto che è normale inserire l'avvio di Anacron all'interno della stessa procedura di inizializzazione del sistema.
È necessario attribuire un nome ad ogni record (il job, secondo questa logica), per permettere a Anacron di annotarsi quando il comando relativo viene eseguito, in modo da determinare ogni volta se il tempo previsto è scaduto o meno.
La definizione di variabili di ambiente può essere necessaria, specialmente quando si prevede l'avvio di Anacron in modo automatico, attraverso la procedura di inizializzazione del sistema; in tal caso diventa fondamentale attribuire un valore alle variabili SHELL e PATH. Si osservi l'esempio seguente:

```
# /etc/anacrontab

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

1       5       cron.daily      run-parts /etc/cron.daily
7       10      cron.weekly     run-parts /etc/cron.weekly
30      15      cron.monthly    run-parts /etc/cron.monthly
```

Oltre alla definizione delle variabili, si può vedere la dichiarazione di tre job che riguardano l'esecuzione di altrettanti comandi a cadenza giornaliera, settimanale e mensile. I tre job vengono avviati a distanza di cinque minuti uno dall'altro e anche il primo di questi attende cinque minuti per sicurezza. Questa pianificazione si affianca a quella del crontab di sistema, in modo da garantire l'esecuzione degli script contenuti nelle directory `/etc/cron.daily`, `/etc/cron.weekly` e `/etc/cron.monthly`.

9.3.2 anacron

`anacron [`*`opzioni`***`]`**

L'eseguibile `anacron` è quello che svolge in pratica il lavoro di gestione del sistema Anacron. Può essere avviato anche da utenti comuni, ma in generale questo non si fa, lasciando che sia la stessa procedura di inizializzazione del sistema a preoccuparsene.
Quando viene avviato, si mette normalmente a funzionare sullo sfondo, disimpegnandosi dal processo che lo ha avviato (diventando figlio del processo principale); quindi legge il file di configurazione e controlla nella directory `/var/spool/anacron/` quando è stata l'ultima volta che ha eseguito ogni job previsto (se si aprono i vari file si vede che contengono una data); di

conseguenza avvia i comandi relativi ai job scaduti, aggiornando i file relativi nella stessa directory. L'avvio di questi comandi avviene di norma rispettando il ritardo indicato nel file di configurazione. Al termine, l'eseguibile `anacron` termina di funzionare.
Di solito, Anacron viene gestito direttamente dalla procedura di inizializzazione del sistema; in tal caso, è normale che l'eseguibile sia avviato con l'opzione `-s`, che fa in modo di mettere in serie l'esecuzione dei job. In pratica, si evita che venga avviato un altro job prima che sia terminata l'esecuzione di quello precedente.
Per evitare conflitti tra Anacron e Cron, quando potrebbe essere consentito a Cron di eseguire le stesse elaborazioni, conviene fare in modo che Cron «avvisi» Anacron nel momento in cui queste vengono svolte. In pratica, si utilizza l'opzione `-u` di `anacron`, con la quale si ottiene proprio questo: aggiornare le date annotate nei file contenuti nella directory `/var/spool/anacron/`.
Per comprendere meglio la cosa, si pensi alla situazione tipica, ovvero quella in cui si predispongono le solite directory `/etc/cron.daily/`, `/etc/cron.weekly/` e `/etc/cron.monthly/`, che devono contenere gli script da eseguire con cadenza giornaliera, settimanale e mensile, da parte di Cron o Anacron, indifferentemente. Per evitare che Anacron rifaccia quello che potrebbe avere già fatto Cron, si mette lo script `0anacron` in ognuna di queste directory (lo zero iniziale garantisce che questo sia il primo script a essere avviato). Nel caso si `/etc/cron.daily/0anacron`, il contenuto potrebbe essere:

```
#!/bin/bash

anacron -u cron.daily
```

Questo dice a Anacron di aggiornare la data abbinata al job `cron.daily`, in modo da evitare di ripeterne l'esecuzione prima del necessario; inoltre, questo script non crea problemi a Anacron stesso, nel momento in cui dovesse essere avviato come parte del comando relativo a un job.

CAPITOLO 10

BACKUP

L'amministrazione di un sistema UNIX è da sempre un grosso problema sotto tutti i punti di vista. Il primo tra tutti è quello della salvaguardia dei dati, ma al rischio della loro perdita si pone rimedio solo attraverso una corretta gestione delle copie di sicurezza.

10.1 Scelta del sistema di copia

Gli strumenti a disposizione per eseguire copie di sicurezza sono molti e si possono distinguere due estremi possibili:

- copiare i file e le directory;
- copiare una partizione o un disco intero.

La copia di una partizione o di un disco può avere il vantaggio di permettere l'archiviazione della situazione esatta in cui si trova, problemi inclusi. Inoltre la data di lettura dei file non viene modificata. Lo svantaggio fondamentale di tale tipo di copia è che questa è riferita a un disco particolare (o a una partizione) di una macchina particolare: è molto poco probabile che si possano recuperare dati archiviati in questo modo in un disco fisso diverso. Questa tecnica, più che per eseguire delle copie di sicurezza, viene utilizzata per archiviare dischetti nel loro stato originale.
La copia di file e directory non tiene conto del supporto fisico in cui si trovano e nemmeno del tipo di file system utilizzato. Questo comporta una serie di conseguenze:

- i file e le directory vengono scanditi in lettura, alterando quindi le date di lettura;
- i collegamenti simbolici vanno copiati come tali e non devono essere copiati gli oggetti a cui questi puntano;
- i collegamenti fisici potrebbero non essere distinti.

In generale, dal momento che una copia di file e directory è portabile, mentre una copia di un dispositivo intero non lo è (quando non si tratta di un dispositivo standard come i dischetti), dovrebbe essere preferibile la prima di queste due soluzioni.

Di solito si preferisce la tecnica dell'archiviazione dei dati in un file unico (archivio), assieme a tutte le informazioni necessarie per riprodurre i file e le directory originali. In questo modo si possono utilizzare unità di memorizzazione di qualunque tipo, eventualmente suddividendo l'archivio in pezzi più piccoli contenibili al loro interno.

10.1.1 Utenti e gruppi proprietari

Tra gli attributi dei file, è molto importante l'indicazione degli utenti e dei gruppi proprietari. I programmi di archiviazione potrebbero non essere in grado di memorizzare il numero UID e GID, limitandosi ad annotare solo i nomi degli utenti e dei gruppi. In tal modo, nel momento del recupero, i numeri UID e GID verrebbero riprodotti in base alle caratteristiche del sistema, cioè in base alla particolare configurazione dei file `/etc/passwd` e `/etc/group`.
Il fatto che il programma di archiviazione memorizzi i numeri UID e GID, oppure che memorizzi i nomi di utenti e gruppi, ha delle implicazioni che si traducono, a seconda delle circostanze, in vantaggi o svantaggi.
Se i dati archiviati devono essere riprodotti in un sistema diverso da quello di origine, in cui ci sono gli stessi nomi di utenti e di gruppi, che però potrebbero corrispondere a numeri UID e GID differenti, diventa conveniente un metodo di archiviazione che ignori i numeri degli

utenti e dei gruppi. Tuttavia, se alcuni nomi di utenti o gruppi non sono presenti nel sistema di destinazione, la proprietà di questi file verrebbe assegnata automaticamente all'utente root.
Quando si esegue una copia di sicurezza di un intero sistema e poi lo si vuole riprodurre altrove, si agisce per mezzo di un sistema operativo minimo, avviato probabilmente attraverso dischetti. In queste condizioni, lo scopo è quello di riprodurre esattamente il sistema originale, per cui, i numeri UID e GID andrebbero rispettati fedelmente, nell'attesa che sia ripristinato tutto, compresi i file `/etc/passwd` e `/etc/group` originali.

Quando il programma di archiviazione memorizza entrambe le informazioni, sia UID/GID che i nomi, nel momento del recupero si pone il problema di come comportarsi quando questi non corrispondono. Si presentano queste alternative:

- se i numeri corrispondono ai nomi, non si pongono problemi nell'estrazione;
- se i numeri e i nomi non sono utilizzati nel sistema di destinazione, si possono estrarre i dati utilizzando i numeri originali, anche se non sono abbinati ad alcun nome;
- se i numeri non sono utilizzati nel sistema di destinazione, mentre i nomi sì, si possono cambiare i numeri in modo che corrispondano i nomi;
- se i nomi non sono utilizzati nel sistema di destinazione, mentre i numeri corrispondono a nomi differenti, non si può fare altro che recuperare i numeri, assegnando in pratica la proprietà a utenti e gruppi con nomi differenti;
- se sono presenti sia i nomi che i numeri, ma questi non hanno lo stesso abbinamento (cioè i numeri corrispondono a nomi diversi), dovrebbe essere preferibile cambiare i numeri in modo che corrispondano ai nomi.

10.1.2 Percorsi assoluti o relativi

Quando si intende archiviare una porzione di file system e quindi solo ciò che si trova a partire da una certa directory in poi, è importante sapere come si comporta il programma di archiviazione al riguardo della registrazione dei percorsi. Se si vuole archiviare la directory `/home/tizio/esempi/`, il programma di archiviazione potrebbe registrare il suo contenuto in uno dei tre modi seguenti.

1. `/home/tizio/esempi/*`
2. `home/tizio/esempi/*`
3. `./*`

Naturalmente, ciò dipende anche dal modo in cui vengono date le istruzioni al programma stesso.
Nel primo caso, quando dovesse rendersi necessario il recupero dei dati, questi verrebbero collocati esattamente nella directory indicata, in modo assoluto. Nel secondo, verrebbero collocati in modo relativo a partire dalla directory corrente, ottenendo così la directory.`/home/tizio/esempi/*`. Nel terzo caso si avrebbe il recupero del contenuto di quella directory senza informazioni sul percorso precedente.

10.2 Strategia nelle copie

Le copie di sicurezza permettono di conservare la situazione dei dati in un istante determinato, ma i dati sono soggetti a continui aggiornamenti. Per questo occorre una procedura attraverso la quale si possa avere una gestione ordinata e ragionevolmente sicura delle copie.

Generazione delle copie

Per distinguere una copia effettuata in un momento rispetto a quella fatta in un altro, si parla di ***generazione***. In pratica, l'ultima copia di sicurezza effettuata è l'ultima generazione, mentre le altre sono tutte generazioni precedenti. Questo termine si riferisce naturalmente a copie fatte sullo stesso insieme di dati. Di solito si utilizzano almeno tre generazioni di copie: l'ultima, quella precedente e quella ancora precedente.

Livelli delle copie

Quando si esegue una copia dei soli file che risultano diversi rispetto all'ultima copia completa, si parla di ***copia di primo livello***; quando se ne esegue un'altra in un momento successivo per le variazioni avvenute dopo quella di primo livello, si parla di ***secondo livello*** e così di seguito.

A questo punto si possono porre dei problemi nel momento in cui dovesse essere necessario un ripristino dalle copie. Si dovrebbe ripristinare l'ultima copia completa, seguita da tutte quelle aggiuntive dei soli file modificati, nello stesso ordine in cui sono state fatte: dalla più vecchia alla più recente. Altrimenti quando non si vuole ripetere una copia completa troppo frequentemente, si cerca almeno di eseguire copie successive sempre di primo livello (si hanno quindi più generazioni di copie di primo livello). In tal modo, un eventuale recupero richiederebbe solo il ripristino dell'ultima generazione di copia completa e dell'ultima generazione di copia di primo livello.

Distribuzione delle responsabilità

In un sistema monoutente, l'unico utilizzatore è anche l'amministratore del proprio sistema e di conseguenza anche l'unico responsabile: è il solo a sapere esattamente cosa ha fatto e cosa è necessario copiare per sicurezza.

In un sistema multiutente o comunque quando si condividono dati in gruppo, anche se a prima vista potrebbe sembrare conveniente la gestione delle copie in modo centralizzato, è comunque utile affidarla in parte anche alla responsabilità dei singoli:

- periodicamente, un amministratore potrebbe occuparsi di eseguire la copia complessiva di tutto il sistema;
- ogni giorno, gli utenti dovrebbero preoccuparsi di eseguire le copie dei dati di loro competenza.

Nel momento in cui dovesse essere necessario, si dovrebbero recuperare i dati dalle copie generali fatte dall'amministratore e successivamente da quelle particolari dei singoli utenti.

Se determinate attività vengono svolte in gruppo, si potrebbe eleggere ugualmente un responsabile all'interno di questo che si occupi delle copie di quell'attività.

Il vantaggio di questo metodo sta nell'alleggerimento delle responsabilità dell'amministratore e nella soluzione più facile di piccoli problemi locali.

Scelta del supporto

La scelta del supporto di conservazione della copia è importante e comporta alcune conseguenze:

- il costo;
- la disponibilità di unità in grado di utilizzarli;
- la facilità o difficoltà nel recupero di porzioni dei dati archiviati.

I supporti maggiormente utilizzati sono:

1. *Dischetti*: vengono utilizzati quando i dati da archiviare sono pochi. Ovviamente sono molto economici ed in più le unità a dischetti sono disponibili ovunque.

2. *CD-R*: sono economici e di grande capacità. Unico svantaggio: è possibile una sola registrazione. La soluzione a questo problema è l'utilizzo di *CD-RW* (*ReWritable*). Il CD-ROM è un supporto molto sicuro in quanto ha una vita media molto lunga, garantendo la durata delle copie di sicurezza.

3. *Dischi rimovibili*: sono di grandi capacità a prezzi ragionevolmente bassi. Questi hanno il vantaggio di poter essere utilizzati come dischi normali, pertanto, il recupero di dati parziali diventa molto più facile, anche quando la copia di sicurezza avviene per mezzo di un'archiviazione tradizionale.

10.3 Compressione

Il problema della dimensione dei dati da archiviare viene ridotto con l'aiuto della compressione. La tecnica della compressione può essere applicata all'archiviazione in due modi possibili:

- prima della costruzione dell'archivio, ottenendo così un *archivio di file compressi*;
- dopo la costruzione dell'archivio, ottenendo così un *archivio compresso*.

La differenza tra i due modi è enorme. Notare che la compressione introduce un elemento di rischio maggiore nella perdita di dati: se una copia di sicurezza viene danneggiata parzialmente, l'effetto di questo danno si riflette in una quantità di dati maggiore (spesso è compromesso tutto l'archivio).

Compressione prima dell'archiviazione

I programmi di archiviazione compressa maggiormente diffusi negli ambienti DOS utilizzano la tecnica della compressione prima dell'archiviazione. È questo il caso degli archivi.`zip`,.`arj`,.`lzh`. Tale sistema ha il vantaggio di permettere una facile scansione dell'archivio alla ricerca di file da estrarre (e decomprimere) o un ampliamento dell'archivio in un momento successivo alla sua creazione. Un altro vantaggio è la minore sensibilità alla perdita dei dati: se una parte dell'archivio è danneggiato, dovrebbe essere possibile ripristinare almeno il resto. Lo svantaggio principale è che la compressione fatta in questo modo, a piccoli pezzi, non è molto efficiente.

Compressione dopo l'archiviazione

La compressione fatta dopo l'archiviazione elimina ogni possibilità di accedere ai dati contenuti nell'archivio e di poterlo ampliare, se non dopo averlo decompresso. Questo significa anche che un danneggiamento parziale dell'archivio implica la perdita di tutti i dati da quel punto in poi.

Un altro tipo di problema deriva dalla difficoltà di distribuire un archivio compresso suddividendolo su più unità di memorizzazione. In questo caso però, l'efficienza della compressione è massima. Negli ambienti UNIX, di fatto, è questa la scelta preferita.

10.4 Utilizzo del comando find

Il programma `find` esegue una ricerca, all'interno di uno o più percorsi, per i file che soddisfano delle condizioni determinate, legate alla loro apparenza esterna e non al loro contenuto. Per ogni file o directory trovati, può essere eseguito un comando (programma, script o altro) che a sua volta può svolgere delle operazioni su di essi.
Per approfondimenti consultare *find.info* oppure *find*(1).

Avvio di find

La sintassi di `find` è piuttosto insolita, oltre che complessa. In particolare è indispensabile tenere a mente che molti dei simboli utilizzati negli argomenti di `find` potrebbero essere interpretati e trasformati dalla shell, di conseguenza occore utilizzare le tecniche che la shell stessa offre per evitarlo.

`find [`*`percorso...`***`] [`***`espressione`***`]`**

`find` esegue una ricerca all'interno dei percorsi indicati per i file che soddisfano l'espressione di ricerca. Il primo argomento che inizia con `-`, `(`, `)`,, o `!` (trattino, parentesi tonda, virgola, punto esclamativo) viene considerato come l'inizio dell'espressione, mentre gli argomenti precedenti sono interpretati come parte dell'insieme dei percorsi di ricerca.
Se non vengono specificati percorsi di ricerca, si intende la directory corrente; se non viene specificata alcuna espressione, viene emesso l'elenco dei nomi trovati.

Espressioni di find

Il concetto di espressione nella documentazione di `find` è piuttosto ampio e bisogna fare un po' di attenzione. Si può scomporre idealmente in

`[`*`opzione...`***`] [`***`condizioni`***`]**

e a loro volta le condizioni possono essere di due tipi: test e azioni. Ma, mentre le opzioni devono apparire prima, test e azioni possono essere mescolati tra loro.
In generale quindi:

`find [`*`percorso...`***`] [[`***`opzione...`***`] [`***`condizioni`***`]]`**

Le opzioni rappresentano un modo di configurare il funzionamento del programma, così come di solito accade nei programmi di servizio. Le condizioni sono espressioni che generano un risultato logico e come tali vanno trattate: per concatenare insieme più condizioni occorre utilizzare gli operatori booleani.

Opzioni principali

Come già accennato, dopo l'indicazione dei percorsi e prima delle condizioni (test e azioni) vanno indicate le opzioni.

`-depth`

Elabora prima il contenuto delle directory: in particolare processa il contenuto di una directory prima dell'inodo corrispondente alla directory stessa.. In pratica si ottiene una scansione che parte dal livello più profondo fino al più esterno.

`-xdev | -mount`

Questa opzione dice a `find` di *non* cambiare filesystem. Ciò è molto utile quando occorre cercare qualcosa che si sa essere nel file system di root: infatti per default `find /` esaminerebbe tutti i file! Quindi non esegue la ricerca nelle directory contenute all'interno di file system differenti da quello di partenza. E’ preferibile usare `-xdev`.

-noleaf
Non ottimizza la ricerca. In particolare non fa usare l'ottimizzazione che dice “una directory contiene esattamente due sottodirectory in meno del suo numero di hard link”. Di solito tutte le directory hanno un riferimento col proprio nome nella directory padre, uno come. all'interno di se stesse (ecco il valore di due indicato sopra) e uno come.. in tutte le directory figlie. Ma ci sono due eccezioni: un filesystem non UNIX (ad esempio montato via NFS o anche partizioni DOS), e i link simbolici. Quindi in generale questa opzione è necessaria quando si effettuano ricerche all'interno di file system che non seguono le convenzioni UNIX.

Esempi:

```
$ find. -xdev -print
```
Elenca tutti i file e le directory a partire dalla posizione corrente restando nell'ambito del file system di partenza.

Test o condizioni

I test sono condizioni che vengono valutate per ogni file e directory incontrati. Il risultato delle condizioni può essere *Vero* o *Falso*. Quando vengono indicate più condizioni, queste devono essere unite in qualche modo attraverso degli operatori booleani per ottenere una sola grande condizione. Se non viene specificato diversamente, viene utilizzato automaticamente l'operatore AND: `-and`.

Valori numerici:
All'interno dei test, gli argomenti numerici possono essere preceduti o meno da un segno.

Simbolo	Descrizione
+*n*	Indica un numero maggiore di *n*.
-*n*	Indica un numero minore di *n*.
n	Indica un numero esattamente uguale a *n*.

Proprietà:

Simbolo	Descrizione
-uid *n*	Si avvera quando il numero UID (utente) del file o della directory è uguale a *n*.
-user *nome_dell'utente*	Si avvera quando il file o la directory appartiene all'utente indicato.
-nouser	Si avvera per i file e le directory di proprietà di utenti non esistenti.
-gid *n*	Si avvera quando il numero GID (gruppo) del file o della directory è uguale a *n*.
-group *nome_del_gruppo*	Si avvera quando il file o la directory appartiene al gruppo indicato.
-nogroup	Si avvera per i file e le directory di proprietà di gruppi non esistenti.

Permessi:

Simbolo	Descrizione
-perm *permessi*	Si avvera quando i permessi del file o della directory corrispondono esattamente a quelli indicati con questo test. I permessi si possono indicare in modo numerico (ottale) o simbolico.
-perm *-permessi*	Si avvera quando i permessi del file o della directory comprendono almeno quelli indicati con questo test.
-perm *+permessi*	Si avvera quando alcuni dei permessi indicati nel modello di questo test corrispondono a quelli del file o della directory.

Caratteristiche dei nomi:

-name *modello*
Si avvera quando viene incontrato un nome di file o directory corrispondente al modello indicato, all'interno del quale si possono utilizzare i caratteri jolly. La comparazione avviene utilizzando solo il nome del file (o della directory) escludendo il percorso precedente. I caratteri jolly (`*`, `?`, `[`, `]`) non possono corrispondere al punto iniziale (`.`) che appare nei cosiddetti file nascosti.
-iname *modello*
Si comporta come `-name`, ma non tiene conto della differenza tra maiuscole e minuscole.

-lname *modello*
-ilname *modello*
Si avvera quando si tratta di un collegamento simbolico e il suo contenuto corrisponde al modello che può essere espresso utilizzando anche i caratteri jolly. Un collegamento simbolico può contenere anche l'indicazione del percorso necessario a raggiungere un file o una directory reale. Il modello espresso attraverso i caratteri jolly non tiene conto in modo particolare dei simboli punto (`.`) e barra obliqua (`/`) che possono essere contenuti all'interno del collegamento.

-path *modello*
-ipath *modello*
Si avvera quando il modello, esprimibile utilizzando caratteri jolly, corrisponde a un percorso. Per esempio, un modello del tipo`./i*no` può corrispondere al file`./idrogeno/ossigeno`.

-regexp *modello*
-iregexp *modello*
Si avvera quando l'espressione regolare indicata corrisponde al file o alla directory incontrati. Per la verifica della corrispondenza, attraverso l'espressione regolare, viene utilizzato anche il percorso e non solo il nome del file o della directory. Quindi, per ottenere la corrispondenza con il file`./carbonio` si può utilizzare l'espressione regolare`.*bonio` oppure`.*bo..o`, ma non `c.*io`.

Data di modifica:

Simbolo	Descrizione
-mmin *n*	Si avvera quando la data di *modifica* del file o della directory corrisponde a *n* minuti fa.
-mtime *n*	Si avvera quando la data di *modifica* del file o della directory

	corrisponde a *n* giorni fa. Più precisamente, il valore *n* fa riferimento a multipli di 24 ore.
-newer *file*	Si avvera quando la data di *modifica* del file o della directory è più recente di quella del file indicato.

Data di accesso:
Si utilizza la stessa sintassi della data di modifica, solo che al posto della *m* iniziale si utilizza una *a*:

```
-mmin n       diventa -amin n
-mtime n      diventa -atime n
-newer file   diventa -anewer file
```

Data di creazione o cambiamento di stato:
Si utilizza la stessa sintassi della data di modifica, solo che al posto della *m* iniziale si utilizza una *c*:

```
-mmin n       diventa -cmin n
-mtime n      diventa -ctime n
-newer file   diventa -cnewer file
```

Dimensione:

Simbolo	Descrizione
-empty	Si avvera quando il file o la directory sono vuoti.
-size *n*[b|c|k|w]	Si avvera quando la dimensione del file o della directory ha una dimensione pari a *n*. L'unità di misura è rappresentata dalla lettera che segue il numero: • b blocchi da 512 byte (valore di default); • c byte (caratteri); • k blocchi da 1024 byte (1 Kb); • w parole di 2 byte.

Varie:

Simbolo	Descrizione
-true	Sempre vero.
-false	Sempre falso.
-type *categoria*	Si avvera se l'elemento analizzato appartiene alla categoria indicata: • b dispositivo a blocchi; • c dispositivo a caratteri; • d directory; • p file FIFO (pipe con nome); • f file normale (*regular file*); • l collegamento simbolico; • s socket.

Operatori booleani
Le condizioni possono essere costruite anche utilizzando alcuni operatori booleani e le parentesi:

Simbolo	Descrizione
(*expr* **)** **\(** *expr* **\)**	Le parentesi tonde stabiliscono la precedenza nell'esecuzione dei test.
! *expr* **-not** *expr*	NOT logico. Cambia il valore di verità dell'espressione, vale a dire che se *expr* era vera, diventa falsa.
expr1 expr2 *expr1* **-a** *expr* *expr1* **-and** *expr2*	AND logico. In mancanza dell'indicazione di un operatore logico tra due condizioni si intende AND.
expr1 **-o** *expr2* *expr1* **-or** *expr2*	OR logico.

Azioni
Le azioni sono delle operazioni da compiere per ogni file o directory che si ottiene dalla scansione. Queste azioni generano però un risultato che viene interpretato in maniera logica. Dipende da come vengono concatenate le varie condizioni (test e azioni) se, e in corrispondenza di quanti file, verranno eseguite queste azioni.
Le azioni si possono dividere in due grandi categorie:

1. Azioni che *stampano* qualcosa.
   La più semplice di queste (quella di default) è **`-print`**, che scrive semplicemente il nome completo dei file che avverano l'insieme delle condizioni. Da notare che ritorna sempre `true`. Una semplice variante di `-print` è **`-fprint`** *file*, che scrive su *file* invece dello standard output.

2. Azioni che *eseguono* qualcosa.
   Vediamone la sintassi:

   - **`-exec`** *`comando;`*
     Esegue il comando indicato, nella directory di partenza. Restituisce `true` se il comando ritorna 0, cioè è stato eseguito senza problemi. La ragione per avere il; è che `find` non può sapere dove termina il comando visto che ce ne potrebbe essere più di uno. Allora si usa il punto e virgola. Se questo viene interpretato dalla shell, occorre quotarlo (`/;`). All'interno del comando, la stringa `{ }` viene interpretata come sinonimo del file che è attualmente in corso di elaborazione.

   - **`-ok`** *`comando;`*
     Si comporta come `-exec`, con la differenza che, prima di eseguire il comando, chiede conferma all'utente. Se il comando non viene eseguito, restituisce il valore `false`.

Esempi:

```
$ find. -name prova\* -print
```
cerca tutti i nomi che iniziano con `prova`. Se la stringa da cercare fosse contenuta nel nome, è meglio scrivere `"*pippo*"` invece di `\*pippo\*`.

`$ find / -name "lib*" -print`
Esegue una ricerca su tutto il file system globale, a partire dalla directory radice, per i file e le directory il cui nome inizia per `lib`. Dal momento che si vuole evitare che la shell trasformi `lib*` in qualcosa di diverso, si utilizzano le virgolette.

`# find / -xdev -nouser -print`
Esegue una ricerca nel file system principale a partire dalla directory radice, escludendo gli altri file system, per i file e le directory appartenenti a utenti non registrati (che non risultano da `/etc/passwd`).

`$ find /usr -xdev -atime +90 -print`
Esegue una ricerca a partire dalla directory `/usr/`, escludendo altri file system diversi da quello di partenza, per i file la cui data di accesso è più vecchia di 2160 ore (24 * 90 = 2160).

`$ find / -xdev -type f -name core -print`
Esegue una ricerca a partire dalla directory radice, all'interno del solo file system principale, per i file `core` (solo i file normali).

`$ find / -xdev -size +5000k -print`
Esegue una ricerca a partire dalla directory radice, all'interno del solo file system principale, per i file la cui dimensione supera i 5000 Kb.

`$ find ~/dati -atime +90 -exec mv \{\} ~/archivio \;`
Esegue una ricerca a partire dalla directory `~/dati/` per i file la cui data di accesso è più vecchia di 90 giorni, spostando quei file all'interno della directory `~/archivio/`. Il tipo di shell a disposizione ha costretto a usare spesso il carattere di escape (\) per poter usare le parentesi graffe e il punto e virgola secondo il significato che gli attribuisce `find`.

`$ find /usr –newer /home/a.txt`
Esegue una ricerca nel file system principale a partire dalla directory `/usr` per i file e le directory più nuovi rispetto al file `/home/a.txt`.

`$ find /home –name "pro*" !-type d`
Esegue una ricerca nel file system principale a partire dalla directory `/home` per i file il cui nome inizia con `pro`. Da notare che nella ricerca vengono escluse le directory ed inoltre poiché si vuole evitare che la shell trasformi `pro*` in qualcosa di diverso, si utilizzano per tale scopo le virgolette.

`$ find /usr !prova.txt > lista`
Esegue una ricerca nel file system principale a partire dalla directory `/usr` per i file e le directory dal nome diverso da `prova.txt`. Il risultato è rediretto in un file chiamato `lista`. Tale file può essere utile nel momento in cui si decide di creare e comprimere un archivio con i programmi di compressione come tar.

10.5 Archiviazione e recupero attraverso tar e gzip

La coppia `tar` e `gzip` rappresenta lo standard nell'archiviazione dei dati: `tar` genera un archivio non compresso che può comprendere anche collegamenti simbolici e file speciali; `gzip` lo comprime generando un archivio più piccolo.
In particolare `tar` è in grado di utilizzare `gzip` direttamente senza dover far uso di pipeline, purché il risultato dell'archiviazione non debba essere suddiviso su più supporti.

10.5.1 tar

`tar` *`opzione_di_funzionamento`* **[***`opzioni`***]** *`file...`*

`tar` (*Tape ARchive*) è un programma di archiviazione nato originariamente per essere usato con i nastri. Il primo argomento deve essere una delle opzioni che ne definisce il funzionamento. Alla fine della riga di comando vengono indicati i nomi dei file o delle directory da archiviare. Se non viene specificato diversamente con le opzioni, l'archivio viene emesso attraverso lo standard output.
Il `tar` tradizionale ammette l'uso di opzioni senza il trattino anteriore (`-`) consueto.
Per la descrizione completa di questo programma, conviene consultare *tar*(1).

Opzioni di funzionamento:
Un gruppo di opzioni rappresenta l'operazione da compiere. Di solito, data la loro importanza, queste opzioni appaiono all'inizio degli argomenti di `tar`.

Simbolo	Descrizione
`A` **|** `-A` **|** `--catenate`	Aggiunge dei file `tar` a un archivio già esistente.
`C` **|** `-c` **|** `--create`	Crea un nuovo archivio.
`d` **|** `-d` **|** `--diff` **|** `--compare`	Trova le differenze tra l'archivio e i file esistenti effettivamente.
`--delete`	Cancella dall'archivio i file indicati.
`-f` *`file`* **|** `--file=`*`file`*	Salva l'archivio nel file o nel dispositivo. Se si tratta di un file normale, questo viene creato.
`r` **|** `-r` **|** `--append`	Aggiunge dati a un archivio già esistente.
`t` **|** `-t` **|** `--list`	Elenca il contenuto di un archivio.
`u` **|** `-u` **|** `--update`	Aggiunge solo i file più recenti rispetto a quanto già contenuto nell'archivio.
`-v` **|** `--verbose`	Elenca i file che vengono elaborati.
`x` **|** `-x` **|** `--extract`	Estrae i file da un archivio.
`-z` **|** `--gzip` **|** `--ungzip`	Filtra l'archivio attraverso il programma `gzip`.

Alcune opzioni standard di `tar` possono fare a meno del trattino iniziale, come nel caso di `c` e `x`. Altre opzioni hanno quel trattino, ma possono essere aggregate in un'unica serie di lettere, come nel caso di `czvf`, dove si ha l'unione di: `-c`, `-z`, `-v` e `-f`.

Altre opzioni:

Simbolo	Descrizione
-h | --dereference	Non copia i collegamenti simbolici, ma i file a cui questi fanno riferimento.
-k | --keep-old-files	In fase di estrazione da un archivio, non sovrascrive i file eventualmente già esistenti.
-L *Kbyte*	Definisce la dimensione massima dei vari segmenti di copia multivolume.
-m | --modification-time	In fase di estrazione da un archivio, non viene ripristinata la data di modifica dei file.
-M | --multi-volume	Permette di creare, elencare o estrarre, un archivio multivolume.
-N *data* | --newer *data*	Archivia solo i file la cui data è più recente di quella indicata come argomento.
-p | --same-permissions	Estrae tutti i permessi associati ai file. Se non viene usata questa opzione, i file ottengono i permessi predefiniti, anche in funzione della maschera dei permessi dell'utente che esegue l'operazione.
-P | --absolute-path	Estrae i file utilizzando i percorsi assoluti, cioè senza eliminare la prima barra (/) che appare nei nomi di percorso (pathname).
--remove-files	In fase di creazione di un nuovo archivio, elimina i file archiviati.
--same-owner	Durante l'estrazione da un archivio, assegna ai file estratti gli utenti e i gruppi proprietari originali.
-W | --verify	Cerca di verificare la validità dell'archivio dopo averlo creato.
-j | --bzip2	Filtra l'archivio attraverso il programma di compressione bzip2 (sia per la compressione che per l'estrazione).

In generale le opzioni più utilizzate sono le seguenti:

- **-cvf** per Compattare;
- **-xvf** per Scompattare.

E' consigliabile comunque prendere dimestichezza anche con le altre opzioni in quanto il comando tar è molto utilizzato anche con le opzioni che indicano file di archivio, invocazione di un programma di compressione…

Esempi:

```
# tar -c -f /dev/fd0 -L 1440 -M -v /usr
# tar cf /dev/fd0 -L 1440 -M -v /usr
# tar cvf /dev/fd0 -L 1440 -M /usr
```

Archivia la directory /usr/ con tutto il suo contenuto, comprese le sottodirectory, utilizzando i dischetti (da 1440 Kb). Con la copia multivolume, come in questo caso, non è possibile utilizzare la compressione automatica attraverso l'opzione -z.

```
# tar -cfv /dev/fd0 -L 1440 -M /usr
# tar cfv /dev/fd0 -L 1440 -M /usr
```

Questi due esempi sono identici ed **errati** ugualmente. Non è possibile accodare lettere di altre opzioni dopo la «`f`», dal momento che questa richiede un argomento.

```
$ tar -t -f /dev/fd0 -L 1440 -M -v
```
Visualizza l'elenco del contenuto dell'archivio fatto su dischetti.

```
$ tar -x -f /dev/fd0 -L 1440 -M -v -p --same-owner
$ tar xpvf /dev/fd0 -L 1440 -M --same-owner
```
Estrae il contenuto dell'archivio su dischetti a partire dalla posizione corrente. Di solito l'opzione `--same-owner` è già predefinita all'interno di `tar`, ma in generale vale la pena di specificarla. Tuttavia, in questi esempi, dal momento che si tratta di un utente comune (lo si vede dal dollaro che viene indicato come invito), non ha significato, dal momento che l'utente comune non ha la possibilità di assegnare a un altro la proprietà dei file che crea. Provare, una volta diventati superutenti, ad archiviare file di altri utenti utilizzando all'atto dell'estrazione l'opzione `--same-owner`.

```
$ tar -c -f /tmp/archivio.tgz -z -v /usr
$ tar czvf /tmp/archivio.tgz /usr
```
Archivia il contenuto della directory `/usr/` nel file `/tmp/archivio.tgz`; il secondo esempio fa la stessa cosa con in più la compressione di `gzip`.

Se si vuole utilizzare come elenco dei file da archiviare un file creato con la redirezione del comando `find`, è possibile farlo utilizzando la seguente sintassi:

$ **tar -cvf** *nome_archivio path* **–T** *nomefile*

dove *nomefile* è il nome del file di lista creato con `find`. Quindi `tar` includerà nell'archivio i file elencati nel file di lista di nome *nomefile*. Per utilizzare tale procedura, è consigliabile sapere usare bene il comando `find`.

10.5.2 gzip, gunzip, zcat

gzip [*opzioni***] [***file...***]**
gunzip [*opzioni***] [***file...***]**
`gzip` è un programma di compressione attraverso il quale viene creato un file compresso per ogni file indicato negli argomenti. `gzip` è in grado di comprimere solo file normali, in modo singolo: per ogni file ne viene generato un altro con l'estensione.`gz` o un'altra se specificato diversamente con le opzioni. Se non viene indicato alcun file o se si utilizza espressamente un trattino isolato (`-`), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.
`gunzip` è un collegamento a `gzip`: avviare `gunzip` equivale ad avviare `gzip` con l'opzione `-d`. Si veda in particolare *gzip.info* o *gzip*(1).

Opzioni principali:

Simbolo	Descrizione
-c | `--stdout`	Emette il risultato attraverso lo standard output
-d | `--decompress`	Decomprime un file compresso. `gzip` si comporta con questa opzione predefinita quando viene eseguito con il nome `gunzip`.

-r | --recursive	Se tra i nomi indicati nella riga di comando appaiono delle directory, vengono compressi o decompressi tutti i file in esse contenuti.
-t | --test	Controlla l'integrità dei file compressi.
-1 | -2 |…| -9	Permette di definire il livello di compressione: -1 rappresenta la compressione minima; -9 rappresenta la compressione massima. Per default si utilizza -6.

Esempi:

$ **gzip *.sgml**
Comprime i file *.sgml, utilizzando un livello intermedio di compressione, sostituendo i file originali con quelli compressi: *.sgml.gz.

$ **gzip -d *.sgml.gz**
Espande i file corrispondenti al modello *.sgml.gz, togliendo loro l'estensione.gz.

$ **cat pippo | gzip -9 > pippo.gz**
Genera il file pippo.gz come risultato della compressione di pippo. In particolare, viene utilizzato il livello di compressione massima e il file originale non viene cancellato.

$ **cat pippo.gz | gzip -d > pippo**
Fa l'opposto dell'esempio precedente: espande il file pippo.gz generando pippo, senza cancellare il file originale.

10.5.3 bzip2, bunzip2

bzip2 [*opzioni***] [***file...***]**
bunzip2 [*opzioni***] [***file...***]**
bzip2 è un programma di compressione funzionalmente analogo a gzip, nel senso che viene creato un file compresso per ogni file indicato negli argomenti. bzip2 è in grado di comprimere solo file normali, in modo singolo, dove per ogni file ne viene generato un altro con l'estensione.bz2. Se non viene indicato alcun file o se si utilizza espressamente un solo trattino isolato (-), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.

bzip2 utilizza un algoritmo di compressione differente, rispetto a gzip, con un carico di elaborazione maggiore, che diventa efficace solo in presenza di file di grandi dimensioni. In generale, per garantire la massima portabilità di un archivio compresso, conviene utilizzare gzip, salvo quando le dimensioni dell'archivio sono tali da rendere realmente conveniente l'utilizzo di bzip2.
bunzip2 è un collegamento a bzip2, il quale, se avviato con questo nome, utilizza implicitamente l'opzione -d per decomprimere i file indicati alla fine della riga di comando.

La sintassi di bzip2 è molto simile a quella di gzip. Anche le opzioni sono per la maggior parte identiche. Per altre informazioni conviene leggere la documentazione *bzip2*(1).

10.6 Esempi di archiviazione e compressione

Negli esempi seguenti si immagina di dover archiviare il contenuto della directory `~/lettere/`, equivalente a `/home/fede/lettere/`, comprese eventuali sottodirectory discendenti.

Archiviazione diretta su floppy

I dischetti possono essere utilizzati in modo sequenziale, ma questo lo si fa solo quando l'archivio generato non è contenibile in un solo dischetto: si ha quindi una copia multivolume e in tal caso non è ammissibile l'uso della compressione.

```
# tar cf /dev/fd0 -M ~/lettere
```

In questo caso, l'opzione `-M` sta proprio per *Multivolume* indicando quindi la possibilità che il supporto di destinazione non sia in grado di contenere l'intero archivio. In tal modo, `tar` si prende cura di sospendere l'archiviazione ogni volta che viene raggiunta la capienza massima. Notare che `tar` non è in grado di determinare da solo questa capacità, per cui l'esempio non è corretto. Si utilizza quindi l'opzione `-L` seguita dalla dimensione del dischetto:

```
# tar cf /dev/fd0 -M -L 1440 ~/lettere
```

Provare a sviluppare un esempio del genere utilizzando anche il comando `mount` in modo da visualizzare il contenuto del dispositivo.

Archiviazione normale su file

Quando l'archiviazione può essere fatta su dischi di dimensione sufficiente a contenere l'archivio intero, invece di utilizzare l'opzione `-f` per specificare un file di dispositivo, si può indicare direttamente un normalissimo file al loro interno, come nell'esempio seguente:

```
$ tar cf /mnt/prova/lettere.tar ~/lettere
```

In pratica, nel caso appena visto, si utilizza un disco montato nella directory `/mnt/prova/` e si crea il file `lettere.tar` al suo interno.

L'archiviazione compressa, con l'utilizzo di `gzip`, può essere ottenuta semplicemente con l'opzione `-z`, come nell'esempio seguente:

```
$ tar czf /mnt/prova/lettere.tar.gz ~/lettere
```

In tal caso l'estensione standard utilizzata (ma non obbligatoria) è.`tar.gz` che rende esplicito il fatto che la compressione è stata fatta dopo l'archiviazione. In alternativa si può usare anche.`tgz`, diffusa nei sistemi DOS.

Archiviazione e percorsi

Gli esempi seguenti, pur archiviando gli stessi dati, mostrano un modo diverso di registrare i percorsi all'interno dell'archivio. La directory di lavoro nel momento in cui si avvia il comando, è `/home/fede/`, corrispondente alla directory personale dell'utente.

```
$ tar czf /mnt/prova/lettere.tar.gz ~/lettere
$ tar czf /mnt/prova/lettere.tar.gz /home/fede/lettere
$ tar czf /mnt/prova/lettere.tar.gz lettere
$ tar czf /mnt/prova/lettere.tar.gz./lettere
```

Nei primi due esempi, viene archiviata l'indicazione del percorso precedente, ma pur essendo stato dato in modo assoluto (`/home/fede/lettere`), questo viene reso relativo da `tar`, eliminando la prima barra obliqua che si riferisce alla directory radice.

Negli ultimi due esempi, viene archiviata l'indicazione della sola directory `lettere/`, sempre in modo relativo.

Archiviazione di periodi

Poiché i file sono forniti di informazioni orarie è possibile eseguire delle copie di sicurezza riferite a dei periodi. Le copie di sicurezza a più livelli possono essere ottenute utilizzando dell'opzione `-N` seguita da una data di partenza: si ottiene l'archiviazione di quanto variato a partire da una certa data (di solito si utilizza quella dell'ultima archiviazione completa).

```
$ tar czf /mnt/prova/lettere.tar.gz -N 20020906 ~/lettere
```

In questo caso, la data che segue l'opzione `-N` rappresenta la mezzanotte del sei settembre 2002.

```
$ tar czf /mnt/mo1/lettere.tar.gz -N "20020906 15:30" ~/lettere
```

Questo ultimo esempio aggiunge alla data l'indicazione di un'ora particolare, 15:30. Per evitare che sia interpretato in maniera errata, il gruppo data-orario viene racchiuso tra virgolette.

Estrazione dei percorsi

Quando si accede all'archivio per estrarne il contenuto o per compararlo con i dati originali, entra in gioco il problema dei percorsi. I dati vengono estratti normalmente nella directory corrente, oppure vengono comparati utilizzando come punto di partenza la directory corrente. Quindi, se l'archivio contiene la directory degli esempi precedenti, registrata a partire dalla radice (ma come visto, senza l'indicazione della radice stessa), questi verranno estratti in`./home/fede/lettere/`, oppure comparati con i dati contenuti a partire da questo percorso.
Se in fase di estrazione o comparazione si vuole fare riferimento a percorsi assoluti, si può utilizzare l'opzione `-P`. In questo modo si afferma esplicitamente che i percorsi indicati nell'archivio vanno considerati come discendenti dalla directory radice.

Questo particolare della gestione dei percorsi è molto importante quando si fanno le copie di sicurezza: spesso si hanno dischi montati su punti di innesto provvisori, pertanto non è molto conveniente memorizzare anche il percorso su cui sono montati.

Recupero

Per poter effettuare un recupero di dati da un archivio è necessario conoscere in particolare il modo in cui questo era stato creato: normale, compresso, multivolume.
In generale, per recuperare dati da un archivio si utilizza l'opzione `x` (*Extract*) al posto di `c` e ad essa si devono eventualmente aggiungere `-z` nel caso di estrazione da un archivio compresso con `gzip` o `-M` nel caso di un archivio multivolume.
Durante il recupero di una copia di sicurezza può essere importante fare in modo che i dati riprodotti mantengano gli stessi attributi originali (permessi e proprietà). Per questo si aggiungono le opzioni `-p` (riproduce i permessi) e `--same-owner` (riproduce le proprietà: UID e GID).
L'esempio seguente mostra un recupero da un archivio multivolume su dischetti.

```
$ tar x -M –L 1440 -p --same-owner -f /dev/fd0
```

L'esempio seguente mostra un recupero con percorso assoluto: i percorsi indicati all'interno dell'archivio vengono aggiunti alla directory radice.

```
$ tar xz -P -p --same-owner -f /mnt/prova/lettere.tar.gz
```

Recupero parziale

Il recupero parziale del contenuto di un archivio `tar` può essere fatto per file singoli o per directory, oppure attraverso l'uso di caratteri jolly. In questo ultimo caso però, occorre fare attenzione a evitare che la shell esegua l'espansione: è compito di `tar` determinare a cosa corrispondano all'interno dei suoi archivi.

L'esempio seguente mostra in che modo potrebbero essere recuperati i file contenuti nella directory `home/fede/lettere/nuove/`:

```
$ tar xz -P -p -f /mnt/prova/lettere.tar.gz home/fede/lettere/nuove
```

L'esempio seguente mostra l'estrazione di tutti i file e delle directory corrispondenti a `home/fede/lettere/ve*`. Gli apici sono necessari per evitare che intervenga la shell a espandere l'asterisco.

```
$ tar xz -P -p -f /mnt/mo1/lettere.tar.gz 'home/fede/lettere/ve*'
```

Nel caso di un recupero parziale può essere utile avere un elenco del contenuto di un archivio per compararne i dati con quelli originali. Per questo scopo valgono le stesse regole del recupero dei dati in generale. In particolare, al posto dell'opzione `x` si deve utilizzare `t` per gli elenchi e `d` per la comparazione.

CAPITOLO 11

PROCEDURA DI INIZIALIZZAZIONE DEL SISTEMA

Quando un sistema UNIX 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'attività 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 modalità monoutente.

11.1 Init

Init è 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` può 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`). Se ne consiglia l'uso solo ad utenti esperti.

Opzioni principali:

`-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 possibilità 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 è stato modificato).

`S |` `s`
Richiede di passare alla modalità monoutente, ma non è 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`.

11.2 /etc/inittab

Il file `inittab` descrive quali processi vengono eseguiti 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 può essere stabilito un gruppo diverso di processi da avviare.
La struttura dei record che compongono le direttive di questo file può essere schematizzata nel modo seguente:

```
id:livelli_di_esecuzione:azione:processo
```

1. *id* -- è 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 (o script) da eseguire.

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 e Init attende che termini prima di eseguirne degli altri.

- **`once`**

Il processo viene eseguito una volta.

- **`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 nulla.

- **`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.

- **`ctrlaltdel`**
  Il processo viene eseguito quando Init riceve il segnale `SIGINT`. Ciò significa che è 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, può contenere diversi caratteri che stanno a indicare diversi livelli di esecuzione possibili. Per esempio, la stringa `123` indica che il processo specificato verrà eseguito indifferentemente per tutti i livelli di esecuzione da uno a tre. Questo campo può 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: è la prima cosa a essere eseguita dopo l'avvio del sistema stesso. In pratica viene avviato lo script `/etc/rc.d/rc.sysinit`.

`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.

`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.

`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 perché è indifferente.

`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.

`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.

```
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 è `mingetty` a essere responsabile dell'attivazione delle console virtuali (Red Hat).

`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.

11.3 Inizializzazione del sistema nella distribuzione Red Hat

La distribuzione Red Hat utilizza un sistema di configurazione composto da script, che non dovrebbero essere modificati, e da file di configurazione utilizzati dagli script, modificabili attraverso programmi utilizzabili soltanto dall'amministratore del sistema.

La procedura di inizializzazione del sistema è attivata dall'eseguibile `init` attraverso le indicazioni di `/etc/inittab`. I livelli di esecuzione (***runlevels***) sono:

- **`0`** arresto del sistema;
- **`1`** singolo utente;
- **`2`** multiutente senza l'utilizzo di eventuali NFS;
- **`3`** multiutente;
- **`4`** non definito (disponibile);
- **`5`** multiutente con procedura di accesso grafica;
- **`6`** riavvio.

Il file `/etc/inittab` è quello che dirige il funzionamento di Init:

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

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

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

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

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

# When our UPS tells us power has failed, assume we have a few
minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
```

```
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting
Down"

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


# Run gettys in standard runlevels
1:12345: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

# Run xdm in runlevel 5
x:5:respawn:/usr/bin/X11/xdm –nodaemon
```

11.3.1 Componenti della procedura di inizializzazione sistema

Attraverso il file `/etc/inittab` vengono indicati due script fondamentali, attraverso cui si articola la procedura di inizializzazione del sistema. Si tratta di `/etc/rc.d/rc.sysinit` e `/etc/rc.d/rc`. Il primo viene utilizzato a ogni avvio del sistema e da questo dipendono le operazioni che vanno svolte una volta sola in quella occasione; il secondo serve ogni volta che si cambia il livello di esecuzione.
Più in dettaglio:

`/etc/rc.d/`
È la directory che raccoglie gli script utilizzati nella fase di avvio del sistema e in quella di arresto.

`/etc/rc.d/rc.sysinit`
È lo script di inizializzazione del sistema. In particolare:

- attiva la gestione della memoria virtuale per l'uso delle aree di scambio previste nelle partizioni, in base al contenuto del file `/etc/fstab`;
- verifica il file system principale e al termine ne esegue il montaggio;
- verifica la dipendenza dei moduli;
- avvia `kerneld` per automatizzare il caricamento dei moduli del kernel;
- verifica gli altri file system indicati per questo nel file `/etc/fstab` e al termine ne esegue il montaggio;
- elimina o ripulisce i file utilizzati nella sessione di lavoro precedente, che servivano per segnalare lo stato di funzionamento;
- configura l'orologio del sistema;
- attiva la gestione della memoria virtuale per l'uso delle aree di scambio previste sui file;
- eventualmente esegue `/etc/rc.d/rc.serial` per l'attivazione della porta seriale;
- eventualmente esegue `/etc/rc.d/rc.modules` per l'attivazione di moduli del kernel.

`/etc/rc.d/rc`
È lo script principale per il controllo dei livelli di esecuzione. Viene utilizzato da Init, attraverso le indicazioni di `/etc/inittab`, con un argomento corrispondente al numero di livello di esecuzione da attivare.

/etc/rc.d/rc *livello_di_esecuzione*
Si limita a determinare se è stato richiesto un cambiamento nel livello di esecuzione, quindi avvia tutti gli script che trova nella directory /etc/rc.d/rc*n*.d/ (dove *n* rappresenta il livello di esecuzione richiesto), cominciando da quelli che iniziano con la lettera K e terminando con quelli che iniziano con la lettera S.
In realtà, questi script sono contenuti nella directory /etc/rc.d/init.d/, con nomi più espressivi, mentre nelle directory /etc/rc.d/rc*n*.d/ sono contenuti solo dei collegamenti simbolici con nomi scelti appositamente per definire l'ordine in cui le operazioni devono essere svolte.
In questo modo, è possibile definire il livello di esecuzione numero quattro, lasciato a disposizione, semplicemente copiandovi all'interno i collegamenti simbolici necessari e senza toccare alcuno script.

/etc/rc.d/rc*n*.d/
Si tratta delle directory riferite a ogni livello di esecuzione (*n* rappresenta il numero del livello stesso). Al loro interno si trovano solo collegamenti simbolici riferiti agli script che si vuole siano eseguiti.
Quando viene selezionato il livello di esecuzione relativo, vengono eseguiti in ordine alfabetico, prima gli script (o meglio i collegamenti) che iniziano con la lettera K (*Kill*) nella forma:

```
/etc/rc.d/rcn.d/script stop
```

allo scopo di disattivare il servizio particolare cui si riferiscono, quindi quelli che iniziano con la lettera S (*Start*), nella forma seguente:

```
/etc/rc.d/rcn.d/script start
```

/etc/rc.d/init.d/
È il vero contenitore degli script utilizzati dalla procedura di inizializzazione del sistema. Questi vengono utilizzati indirettamente attraverso collegamenti simbolici contenuti nelle directory /etc/rc.d/rc*n*.d/.
Molti di questi script possono essere utilizzati dall'amministratore per disattivare o attivare un servizio particolare, senza dover utilizzare un livello di esecuzione diverso e senza dover ricordare tutte le implicazioni di un particolare servizio.

/etc/rc.d/init.d/functions
Si tratta di uno script utilizzato da quasi tutti gli altri, per definire alcune funzioni standard. In particolare, queste funzioni, servono per evitare l'avvio di demoni già in funzione e comunque per uniformare il sistema di avvio e conclusione del loro funzionamento.

/var/lock/subsys/
Questa directory viene utilizzata dagli script contenuti in /etc/rc.d/init.d/ per annotare la presenza in funzione di un servizio determinato: se esiste un file (vuoto) con quel nome, il servizio è considerato attivo.

11.3.2 etc/rc.d/rc?.d/

Le directory `/etc/rc.d/rcn.d/` servono a contenere una serie di collegamenti simbolici che puntano a script della directory `/etc/rc.d/init.d`. Il listato seguente dovrebbe chiarire il meccanismo:

```
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
lrwxrwxrwx 1 root root 11 12:44 S99local ->../rc.local
```

I collegamenti che iniziano con la lettera «`S`» vengono avviati ordinatamente all'attivazione del livello di esecuzione corrispondente, con l'argomento `start`, mentre quelli che iniziano con la lettera «`K`» vengono avviati prima di passare a un nuovo livello di esecuzione, con l'argomento `stop`.
Il numero che segue la lettera «`S`» e «`K`», serve a definire un ordine (alfabetico) corrispondente a quello in cui i servizi vanno avviati o interrotti. Se si volesse predisporre uno script, nella directory `/etc/rc.d/init.d/` per la gestione di un servizio addizionale, si dovrebbero predisporre due collegamenti nella directory del livello di esecuzione prescelto, simili a quelli già visti, con un numero adatto a collocarli nella posizione giusta nell'ordine delle azioni da compiere.
Osservare la presenza del collegamento `S99local` che punta allo script `/etc/rc.d/rc.local`. Si tratta di un'anomalia nella logica generale, dal momento che si fa riferimento a qualcosa di esterno alla directory `/etc/rc.d/init.d/`. Il numero, `99`, è obbligatorio, dal momento che l'esecuzione di questo deve avvenire alla fine dell'avvio di tutti gli altri servizi.

11.4 Registrazione e controllo

In ogni sistema operativo multiutente c'è la necessità di controllare gli accessi, per mezzo della registrazione degli utenti e della registrazione degli eventi. Nei sistemi UNIX un utente che può accedere ha un *account*: si tratta di un «accredito», o meglio ancora una specie di contratto di utenza, che gli permette di esistere nel sistema in qualità di «utente logico».

11.4.1 Registro del sistema

Il registro del sistema (*system log*, o anche *syslog*) è la procedura di registrazione degli eventi importanti all'interno di un cosiddetto file di *log*, ovvero un file delle registrazioni. Questa procedura è gestita principalmente dal demone `syslogd`, che viene configurato attraverso `/etc/syslog.conf`. Altri programmi o demoni possono aggiungere annotazioni al registro inviando messaggi a `syslogd`.

11.4.2 I file di log

Spostarsi nella directory `/var/log` ed eseguire il comando `ls`; quelli che si vedono sono i file di log del sistema: qui sono conservate tutte le informazioni sul normale funzionamento della macchina e soprattutto le registrazioni di errori e problemi; leggendo questi file si possono ottenere preziosi dettagli su come risolvere i problemi di funzionamento del proprio computer.
Ogni file contiene i log di specifici aspetti del sistema; il file principale che determina la configurazione del processo di logging è `/etc/syslog.conf`.
La sintassi dei file di log è uniforme: ogni messaggio viene scritto su una singola riga: inizia sempre con l'indicazione del momento in cui viene effettuata la registrazione, il nome del computer su cui gira il programma che ha generato il log, di solito (anche se non è obbligatorio) il nome del programma stesso ed infine, preceduto da un due punti, il messaggio vero e proprio.
Ad esempio, nel file `/var/log/cron` si può trovare:

```
Sep 8 10:49:04 localhost anacron[963]: Anacron 2.3 started on 2002-09-08
```

che significa questo: l' 8 settembre alle ore 10, 49 minuti e 4 secondi il daemon di log ha ricevuto una richiesta dal processo `anacron`, che in quel momento stava girando con PID 963 sulla macchina `localhost` (che è il nome locale del computer); il messaggio era `" Anacron 2.3 started on 2002-09-08"` e serve solo per segnalare che qualcuno lo ha eseguito.
Ovviamente questo era un tipico messaggio puramente informativo; altri possono essere di maggior aiuto nella risoluzione dei problemi. Si può analizzare ad esempio una riga di `/var/log/messages.3`:

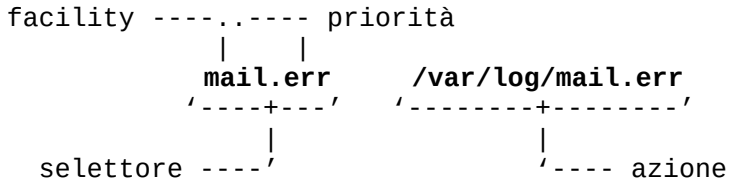
```
Aug 16... login[988]: FAILED LOGIN 1 FROM (null) FOR root Authentication failure
```

indica che il programma `login` ha incontrato un errore piuttosto grave non essendo riuscito ad autenticare un utente (magari non c'è riuscito perché l'utente non possedeva un account!). Poiché non è riuscito a continuare ha smesso di operare ed ha segnalato la cosa nei log: grazie a questo avvertimento si può riuscire immediatamente a capire cosa era andato storto ed a risolvere la situazione.
Se si pensa che anche i programmi del sottosistema di rete si appoggiano su syslog per registrare il proprio normale funzionamento e soprattutto per segnalare eventi sospetti, appare immediatamente chiaro quanto sia importante imparare ad utilizzare al meglio i file di log.
Si può ora analizzare il funzionamento del sistema: i programmi che hanno necessità di lasciare una traccia nei log si appoggiano sul sistema di logging che nelle moderne installazioni è formato da due daemon: **syslogd** e **klogd** che si occupano rispettivamente di gestire i messaggi provenienti dai normali programmi e dal kernel; normalmente `klogd` si appoggia su `syslogd`, il cui file di configurazione `/etc/syslog.conf` è quindi la parte più interessante da analizzare. Inoltre è interessante sottolineare che nel sistema UNIX FreeBSD esiste solo il demone `syslogd`: manca quindi il demone che gestisce i messaggi provenienti dal kernel, il `klogd`.

11.4.3 /etc/syslog.conf

Questo file contiene una serie di righe formate da coppie di campi **selettore** e **azione**, dove selettore definisce *cosa* loggare, e azione *dove* scrivere i log; i due campi sono separati da spazi o tabulazioni.
Ogni selettore è definito da una coppia **facility** e **priorità** separati da un punto: facility stabilisce il *genere* di dati da considerare per i log (ad esempio le informazioni provenienti dal

processo che si occupa di gestire le stampanti, `lpd`), mentre priorità pone un filtro alla *gravità* dei messaggi da loggare.
Da notare che vengono loggati nella directory indicata i messaggi di livello *maggiore o uguale* rispetto alla priorità specificata.
Si può analizzare un esempio banale:



significa che di tutti i messaggi provenienti dai programmi che gestiscono il sistema di posta (ad esempio `sendmail`, `postfix`, `qmail`, `smail` o simili) bisogna loggare nel file `/var/log/mail.err` solamente quelli la cui priorità *corrisponde o supera* il livello `err`.

Cosa è possibile loggare
Le facility predefinite si possono trovare nel file `/usr/include/sys/syslog.h`:

- **`auth`**: messaggi relativi alla sicurezza del sistema, come ad esempio registrazioni di login e logout (è usato da programmi come `su(1)`e `login(1)`).
- **`authpriv`**: come `auth`, ma il messaggio potrebbe contenere dati sensibili, quindi è bene dirigerlo su un file separato che ci si preoccuperà di non rendere leggibile a tutti.
- **`cron`**: il sottosistema di temporizzazione degli eventi: comprende ad esempio daemon come `cron(8)`e `anacron(8)`.
- **`daemon`**: generico per i daemon di sistema.
- **`kern`**: messaggi del kernel.
- **`lpr`**: il sottosistema che gestisce le stampanti.
- **`mail`**: tutto ciò che ha a che fare con la consegna o ricezione di email.
- **`news`**: gestione delle news Usenet.
- **`syslog`**: messaggi generati dallo stesso syslog.
- **`user`**: messaggi generici provenienti da programmi in user space (in contrapposizione ai log generati dal kernel).
- **`uucp`**: il sottosistema UUCP.
- **`ftp`**: riservato ai server ftp.
- da **`local0`** a **`local7`**: riservati per utilizzi locali.

Priorità:
Insieme al tipo di dato da loggare, l'applicazione specifica anche la gravità e l'urgenza del messaggio. Sempre in `/usr/include/sys/syslog.h` si trovano le definizioni dei vari livelli di priorità. Vediamoli in ordine dal meno importante al più urgente:

- **`debug`**: non mostrati normalmente ma solo quando si vuole ottenere informazioni dettagliate da un programma.
- **`info`**: informazioni poco significative circa il funzionamento del programma.
- **`notice`**: segnala una situazione significativa, ma sempre all'interno del normale funzionamento di un programma.
- **`warning`**: messaggio generato in seguito a comportamenti più o meno anomali, che potrebbero essere legati a problemi di funzionamento.
- **`err`**: si è verificata un errore.
- **`crit`**: indica una situazione critica, che compromette seriamente il normale funzionamento del sottosistema coinvolto.

- **`alert`**: condizione che impone un immediato intervento da parte dell'operatore.
- **`emerg`**: il sistema risulta inutilizzabile.

Oltre a questi esistono anche `warn`, `error` e `panic` che però sono obsolete e corrispondono rispettivamente a `warning`, `err` ed `emerg`.

Sintassi avanzata di /etc/syslog.conf
È possibile creare righe di configurazione anche piuttosto complesse combinando in vari modi più facility e più livelli di priorità. Prima di tutto è possibile utilizzare il carattere **`*`** come jolly per tutte le facility o priorità, e la parola chiave **none** per indicare nessuna priorità.

Vediamo alcuni esempi:

```
lpr.*     /var/log/lpr.log
```

raccoglie i messaggi del sottosistema di stampa di ogni priorità e li scrive nel file `/var/log/lpr.log` (nella pratica è del tutto equivalente a `lpr.debug`).

È poi possibile specificare più facility con la medesima priorità separandole con virgole:

```
user,daemon.crit     /var/log/misc.crit
```

salva nel file indicato i messaggi critici appartenenti alle facility `user` e `daemon`.

```
*.emerg     /var/log/all.emerg
```

memorizza tutti i messaggi relativi ad emergenze.

È anche possibile scegliere diversi tipi di dati da inviare allo stesso file; basta separare le diverse entry con punti e virgola:

```
kern.*;news.info     /var/log/uucp_and_news.log
```

salva tutti i messaggi provenienti dal kernel e i messaggi che siano di informazione (o più gravi) dal news server.

```
*.*;auth,authpriv.none     /var/log/quasi_tutto.log
```

salva ogni tipo di messaggio da ogni sistema, tranne tutti i messaggi (grazie alla keyword `none`) relativi alla sicurezza.

Come detto quando si specifica una priorità significa che tutti i messaggi del dato livello e tutti i messaggi più gravi vengono salvati nel file indicato. È anche possibile specificare che si desidera esclusivamente una data priorità anteponendole un uguale:

```
mail.=info        /var/log/mail.info
mail.=warning     /var/log/mail.warn
```

In questo modo vengono salvati in `/var/log/mail.warn` solo i messaggi di warning del sottosistema che si occupa della gestione delle mail, mentre in `/var/log/mail.info` si troveranno solo quelli informativi: se qui non si fosse aggiunto l'uguale avrebbe raccolto anche i messaggi di tipo `notice`, `warning` e degli altri livelli superiori.

Sempre per quanto riguarda le priorità è possibile invertire la selezione facendola precedere da un punto esclamativo:

```
mail.*;mail.!notice     /var/log/mail.log
```

che salva solo i messaggi di tipo `debug` e `info`, o anche:

```
mail.*;mail.!=debug     /var/log/mail.nodebug
```

che memorizza tutti i messaggi del dato sottosistema ad esclusione di quelli relativi al debug (questo esempio equivale quindi a `mail.info`).

In ogni punto lungo le righe di configurazione è possibile andare a capo terminando la riga con un backslash:

```
*.=info;*.=notice;*.=warning;\
        auth,authpriv.none;\
```

```
            cron,daemon.none;\
            mail,news.none    /var/log/messages
```

salva tutti i messaggi di tipo `info`, `notice` o `warning` tralasciando quelli dai sistemi di sicurezza, cron, daemon vari, mail e news.

Esempi di /etc/syslog.conf

Vediamo ora due esempi completi tratti dalle installazioni di default di *Debian* e *RedHat*; le righe che iniziano con un cancelletto (#) sono considerate commenti ed ignorate.

```
# Tratto dal pacchetto RedHat

# Trascrive tutti i messaggi del kernel sulla console.
# Come si vede è commentata, per evitare che di default
# si ricevano troppi messaggi non graditi.
#kern.*                                                 /dev/console

# Logga nel file /var/log/messages tutti i messaggi (a parte quelli
# riferiti alle email, alla sicurezza e a cron) di livello info o
# superiore.
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# Il file /var/log/secure ha i permessi settati in modo da restringerne
# l'accesso al solo amministratore di sistema.
authpriv.*                                              /var/log/secure

# Logga tutto ciò che ha a che fare con le email.
mail.*                                                  /var/log/maillog

# Logga tutti i messaggi provenienti da cron
cron.*                                                  /var/log/cron

# Chiunque sia loggato riceve i messaggi che segnalano emergenze.
*.emerg                                                 *

# Salva i messaggi superiori o uguali al livello crit dei sottosistemi
# UUCP e news in un file a parte.
uucp,news.crit                                          /var/log/spooler

# Salva i messaggi mostrati durante il boot.
local7.*                                                /var/log/boot.log


# Tratto dal pacchetto Debian

# Log divisi per facility.
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          /var/log/mail.log
user.*                          -/var/log/user.log
uucp.*                          -/var/log/uucp.log

# Log separati per i livelli più significativi
# per quanto riguarda il mail server.
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err
```

```
# Idem per il news server.
news.crit                       /var/log/news/news.crit
news.err                        /var/log/news/news.err
news.notice                     -/var/log/news/news.notice

# I messaggi di debug.
*.=debug;\
        auth,authpriv.none;\
        news.none;mail.none     -/var/log/debug

# I messaggi informativi.
*.=info;*.=notice;*.=warn;\
        auth,authpriv.none;\
        cron,daemon.none;\
        mail,news.none          -/var/log/messages

# Segnala a tutti le emergenze.
*.emerg                         *

# Alcuni messaggi che potrebbe tornare utile mostrare su una
# console separata.
#daemon,mail.*;\
#       news.=crit;news.=err;news.=notice;\
#       *.=debug;*.=info;\
#       *.=notice;*.=warn       /dev/tty8

# Idem sulla xconsole.  Lanciare il programma xconsole(1x)
# per vedere i log.
daemon.*;mail.*;\
        news.crit;news.err;news.notice;\
        *.=debug;*.=info;\
        *.=notice;*.=warn       |/dev/xconsole
```

Come si vede le politiche seguite sono piuttosto diverse: sicuramente la *Debian* punta a fornire un sistema che presenti i log in maniera più organica dividendo logicamente i messaggi e creando anche alcuni file contenitori di informazioni di varia provenienza, mentre *RedHat* crea solo log di dati realmente essenziali nel tentativo di mantenere basso lo spazio occupato su disco.

Crescita dello spazio utilizzato dai log

Con tutti questi file nella directory `/var/log` che non fanno altro che accrescersi continuamente è possibile ritrovarsi in breve tempo con una directory che occupa tantissimo spazio su disco, peraltro con dati che difficilmente interessano, specie quelli più vecchi di un certo periodo di tempo. Per ovviare a questo problema esistono una serie di tool normalmente inseriti in ogni distribuzione che si occupano di prendere periodicamente i file di log, copiarne il contenuto in un file che verrà poi compresso per risparmiare ulteriore spazio ed infine azzerarne la lunghezza; dopo un certo periodo di tempo gli stessi file compressi contenenti i vecchi log vengono cancellati definitivamente.
In *RedHat* si può fare riferimento a `logrotate(8)`; inoltre si può fare in modo che venga eseguito periodicamente tramite `cron`.

Analisi approfondita di syslog

Quello che si è visto finora rappresenta quanto necessario conoscere per utilizzare in modo corretto il sistema di log; si cercherà ora di analizzare più da vicino il funzionamento di syslog, in modo da fare un quadro teorico più preciso.

Il daemon principale di tutto il sistema è `syslogd(8)`, il cui comportamento è stabilito dal file di configurazione `/etc/syslog.conf`; si occupa di raccogliere i messaggi da loggare e di scriverli nei rispettivi file (o di trasferirli ad un host remoto o visualizzarli sulla console, a seconda di come è stato configurato). Di solito viene avviato tramite i normali file di inizializzazione (ad esempio `rc.*`).
Una volta lanciato di default apre in lettura la UNIX domain socket `/dev/log` dove i programmi interessati potranno inviare, tramite opportune funzioni di libreria, i propri messaggi da far loggare; se avviato con l'opzione `-r` (e con permessi di superutente) si metterà anche in ascolto sulla porta UDP 514 in modo che altre macchine possano inviargli i propri log.
Se syslogd è in grado di ricevere i messaggi provenienti dai normali programmi, va ricordato che esiste un'altra fonte importantissima di log: il kernel. Per questo esiste il daemon `klogd(8)` che può essere lanciato nel medesimo modo di syslogd (per la precisione è bene lanciare klogd *dopo* syslogd, e disattivarli in ordine inverso).
Il kernel utilizza la funzione `printk()` per inviare i propri messaggi: se nessuno li intercetta di default essi vengono stampati sulla console. Il daemon klogd si pone in "ascolto" del file `/proc/kmsg` (il filesystem `/proc` è un filesystem virtuale: i suoi file non occupano spazio su disco). Il formato di default dei messaggi provenienti dal kernel è:

```
<[0-7]>Testo del messaggio.
```

dove il numero tra 0 e 7 racchiuso dalla coppia <…> indica la priorità del messaggio; queste priorità sono del tutto analoghe a quelle già viste per syslogd, con <7> ad indicare i messaggi di `debug` e <0> per riportare che il sistema è in uno stato inutilizzabile (`emerg`).
A questo punto quando klogd intercetta un messaggio proveniente dal kernel di default lo passa a syslogd con `kern` come facility e il corrispondente livello di priorità; in alternativa klogd può essere lanciato con l'opzione `-f file` grazie alla quale i messaggi verranno loggati nel file indicato. Oltre a ciò i messaggi il cui livello sia inferiore a quello prestabilito (il default è il livello 6, quindi tutti i messaggi dei livelli da 0 a 5) vengono anche mostrati in console; questo comportamento può essere modificato con l'opzione `-c numero` che consente di cambiare questa impostazione; ad esempio può essere una buona idea usare `-c 4`, specie su macchine con parecchi utenti in modo da limitare i messaggi visualizzati in console a quelli di emergenza, di allerta, critici e di errore.
Il kernel può trovarsi nella condizione di dover comunicare informazioni parecchio importanti, quando si trova in situazioni critiche. In caso di serio errore interno viene generato un *Oops* che mostra lo stato dei registri, il contenuto della stack del kernel e il codice che si stava eseguendo quando è intervenuto il problema. Sfortunatamente i dati mostrati sono di norma prettamente numerici e dipendenti da come è stato compilato il kernel: quindi sono di difficile interpretazione. Per ovviare a questo problema in fase di compilazione del kernel viene creato un file di nome `System.map` che mappa funzioni e variabili del kernel facendo corrispondere ai loro indirizzi numerici un nome significativo.
Il daemon klogd cerca questo file nell'ordine in `/boot/System.map`, `/System.map` e `/usr/src/linux/System.map` (oppure si può indicare quale file usare da riga di comando con l'opzione `-k file`); se non lo trova bisognerà manualmente utilizzare l'utility `ksymoops` che si trova nella directory `scripts/ksymoops` tra i sorgenti del kernel per tradurre i valori numerici; di norma si troverà l'oops tra i file di log, ma non è però così raro trovarsi nella situazione in cui il *general protection fault* sia talmente grave da uccidere klogd o impedire la scrittura su disco fisso: in questo caso è necessario trascrivere su carta quanto mostrato sullo schermo.
Tanto syslogd quanto klogd rispondono ad una serie di segnali (che l'utente può inviare tramite il comando `kill(1)`). Il segnale maggiormente utilizzato è `SIGHUP` tramite il quale è possibile far rileggere il file di configurazione a syslogd.

Entrambi i daemon utilizzano dei file nella directory `/var/run` per indicare i propri PID: i file sono `syslogd.pid` e `klogd.pid`.

11.4.4 Sicurezza

Un meccanismo così utile e tutto sommato poco interagente con il mondo esterno quale è il sistema di registrazione e controllo può anche rappresentare un problema per la sicurezza.
Infatti nei log di sistema finiscono informazioni molto importanti proprio sulla sicurezza stessa del sistema; è quindi bene porre massima attenzione nello scegliere permessi d'accesso e proprietario dei file in cui i messaggi verranno scritti.
Anche un corretto funzionamento può provocare dei seri rischi: nei log inviati tramite facility `auth` e `authpriv` finiscono le informazioni su login, logout e altre procedure di autenticazione degli utenti: molto facilmente possono comparire informazioni sensibili, ed è quindi decisamente consigliabile assicurarsi che i permessi di questi file di log non ne consentano la lettura a chiunque; analoghi problemi vengono posti da `pppd(8)` e `chat(8)` (usati per il collegamento dial-up al provider) che potrebbero loggare la procedura di connessione che mostrerebbe anche username e password dell'utente.
Può succedere, per errore o per deliberato attacco da parte di un utente locale o via rete, che la stessa capacità di syslog di registrare tutti i messaggi possa diventare un problema: se uno stesso messaggio venisse inviato migliaia di volte si rischia certo di rallentare leggermente le prestazioni del sistema ma anche di esaurire lo spazio su disco. Non ci sono molti modi per proteggersi efficacemente da simili attacchi: per quanto riguarda i messaggi di eventi registrati dalla rete molti programmi salvano solo il primo di una raffica ravvicinata di numerosi eventi identici, e in seguito di indicare semplicemente quante volte il fatto si è ripetuto (ad esempio con un messaggio come "*last message repeated 10 times*"). Un’altra soluzione potrebbe essere quella di fare in modo che la directory `/var` o `/var/log` sia su una partizione separata dal resto del sistema.

CAPITOLO 12

L’EDITOR VI

vi [*opzioni***] [***file...***]**

L’eseguibile `vi` può 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, verrà creato.

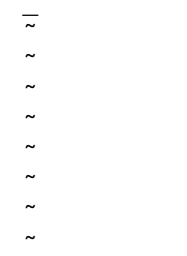
È anche possibile l’indicazione di alcune opzioni, tra cui, le più importanti sono elencate di seguito:

-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.

Quando si avvia *vi* senza indicare alcun file, appare una schermata simile a quella della figura seguente in cui le righe dello schermo contrassegnate dal simbolo tilde (~) rappresentano lo spazio non utilizzato dal file.



Normalmente, l’eseguibile `vi` può essere avviato utilizzando nomi diversi:

view [*opzioni***] [***file...***]**

`view` è 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`.

ex [*opzioni***] [***file...***]**

`ex` è il programma da cui è derivato *vi*. Generalmente, nelle distribuzioni Linux si trova un collegamento con questo nome che punta a *vi*.

12.1 Modalità di funzionamento

vi distingue diverse modalità di funzionamento, altrimenti definibili come stati o contesti. Quando si avvia *vi*, questo si trova di solito nella modalità di comando che permette di usare tasti determinati, attribuendogli significati speciali (di comando).
Esistono due tipi di comandi:

- visuali (*visual*): sono i più semplici e si compongono di sequenze di uno o più tasti il cui inserimento non appare in alcuna parte dello schermo e si concludono senza la pressione del tasto Invio;
- due punti (*colon*): iniziano tutti con il simbolo:, terminano con *Invio*, e dal momento che possono essere un po' più complicati, durante la digitazione appaiono sulla riga inferiore dello schermo. In particolare, questi ultimi sono quelli derivati da `ex`.

Quando si vuole agire per inserire o modificare del testo, occorre utilizzare un comando con il quale *vi* passa alla modalità di inserimento e modifica. La modalità di inserimento si riferisce al momento in cui è possibile modificare il testo. Per passare dalla modalità 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 modalità di comando, da quella di inserimento, è sufficiente premere il tasto **[*Esc*]**. Quando ci si trova già nella modalità di comando, la pressione del tasto [*Esc*] non produce alcunché o al massimo interrompe l'introduzione di un comando.Lo svantaggio principale di questo tipo di approccio è quello di dover passare alla modalità di comando per qualunque operazione diversa dal puro inserimento di testo.
Nelle vecchie versioni anche lo spostamento del cursore avveniva attraverso dei comandi, obbligando l'utente a premere il tasto [*Esc*] prima di poter utilizzare i tasti per il suo spostamento. Attualmente è possibile utilizzare i tasti freccia nel modo consueto dei programmi di scrittura più recenti.
Prima di poter continuare bisogna definire due concetti molto utilizzati in *vi*: la posizione attiva ed i moltiplicatori.

Posizione attiva

Per la descrizione del funzionamento di *vi* è importante definire il concetto di ***posizione attiva*** che si riferisce al punto in cui si trova il cursore. Estendendo il significato, si può parlare di riga attiva, colonna attiva e parola attiva, intendendo quelle su cui si trova il cursore.

Moltiplicatori

E' importante sottolineare che l'effetto di molti comandi può essere ***moltiplicato*** utilizzando un numero. Il concetto è molto semplice: 2a = a+a.

12.2 Inserimento

Si può inserire o modificare del testo solo quando si passa alla modalità 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 può 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 modalità di inserimento e passare a quella di comando.

Con le realizzazioni di *vi* più sofisticate, è 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 modalità di comando.

Il primo documento scritto con *vi* potrebbe essere il seguente:

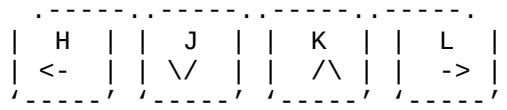
```
Non è facile impararlo, ma ne vale ugualmente la pena_
~
~
~
~
~
~
-- INSERT --
```

I comandi a disposizione per passare alla modalità di inserimento sono molti e non si limitano quindi ai due modi appena descritti. La tabella seguente ne elenca alcuni:

Simbolo	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).

12.3 Navigazione

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* più recenti permettono l'uso dei tasti freccia durante la modalità di inserimento (oltre che durante la modalità di comando), ma questo è solo un aiuto minimo: in generale è necessario tornare alla modalità 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.



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 è possibile utilizzare il comando *n***G** o:*n* (in quest'ultimo caso, seguito da [*Invio*]). Per esempio, per raggiungere la decima riga di un documento ipotetico, si può 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 si riassumono in questa tabella:

	Descrizione		Descrizione
h	Sposta il cursore a sinistra di un carattere.	**H**	Sposta il cursore sulla prima riga che appare sullo schermo.

j	Sposta il cursore in basso nella riga successiva.	**M**	Sposta il cursore sulla riga centrale dello schermo.
k	Sposta il cursore in alto nella riga precedente.	**L**	Sposta il cursore sull'ultima riga che appare sullo schermo.
l	Sposta il cursore a destra di un carattere.	**G**	Sposta il cursore sull'ultima riga del file.
-	Sposta il cursore all'inizio della riga precedente.	*n***G**	Sposta il cursore sulla riga identificata dal numero *n*.
+	Sposta il cursore all'inizio della riga successiva.	**|**	Sposta il cursore sulla prima colonna (all'inizio della riga).
w	Sposta il cursore all'inizio della parola successiva.	*n***|**	Sposta il cursore sulla colonna identificata dal numero *n*.
e	Sposta il cursore alla fine della parola successiva.	**:***n*	Sposta il cursore sulla riga identificata dal numero *n*.
b	Sposta il cursore all'inizio della parola precedente.	**Ctrl+B**	Fa scorrere il testo all'indietro di una schermata.
^	Sposta il cursore all'inizio della prima parola della riga.	**Ctrl+F**	Fa scorrere il testo in avanti di una schermata.
0	Sposta il cursore all'inizio della riga.	**Ctrl+U**	Fa scorrere il testo all'indietro di mezza schermata.
$	Sposta il cursore alla fine della riga.	**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.

12.3.1 Modificatori

I comandi di spostamento o di navigazione, esclusi quelli che iniziano con i due punti: e quelli che si ottengono per combinazione ([*Ctrl+*...]), possono essere utilizzati come ***modificatori*** di altri comandi.
Per definire l'estensione di un comando lo si può far precedere da un moltiplicatore (un numero) e in più, o in alternativa, lo si può 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, e su questa zona interverrà il comando precedente. Tuttavia, per poter applicare questo concetto, è necessario che i comandi da utilizzare in associazione con i modificatori (di spostamento), siano stati previsti per questo. Deve trattarsi cioè di comandi che richiedono questa ulteriore aggiunta.
Ad esempio 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, e per tale ragione, a questo genere di comandi non si può far seguire alcun modificatore. Questo tipo di comandi può 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.

12.4 Cancellazione

Durante la fase di inserimento è 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 modalità di comando.
I comandi di cancellazione più importanti sono **x**, **d** seguito da un modificatore, e **dd**. Il primo cancella il carattere che si trova in corrispondenza della posizione attiva, cioè 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 è 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**.
Ricapitolando:

Simbolo	**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.
d*mod*	Cancella dalla posizione attiva fino all'estensione indicata dal modificatore.
D	agisce come `d$`.

Esempi:

5x
Ripete cinque volte la cancellazione di un carattere: cancella cinque caratteri.
2dd
Ripete due volte la cancellazione di una riga: 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: 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.

12.5 Sostituzione

La modifica del testo inserito può avvenire attraverso i comandi di cancellazione già 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 più 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.
In particolare i vari comandi sono riassunti nella tabella seguente:

Simbolo	Descrizione
C	Sostituisce dalla posizione attiva alla fine della riga.
cc	Sostituisce la riga attiva a partire dall'inizio.
c*mod*	Sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore.
r*x*	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 dieci caratteri.

12.6 Copia e spostamento di porzioni di testo

La gestione della copia e dello spostamento di testo attraverso *vi* è abbastanza complicata. Per questa attività 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, e questo giustifica 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 più semplice di gestire questo meccanismo è quello di usare l'area temporanea senza nome. Per copiare una porzione di testo si può 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 può utilizzare il comando **p** oppure **P**, a seconda che si intenda incollare prima o dopo la posizione del cursore.
Da notare che i comandi **p** e **P** non cancellano il contenuto dell'area temporanea, di conseguenza, se serve si può ripetere l'operazione di inserimento riutilizzando questi comandi. Se invece di copiare si vuole spostare il testo, al posto di usare i comandi di estrazione si possono usare quelli di cancellazione i quali, prima di cancellare il testo, fanno una copia nell'area temporanea.

Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso dell'area temporanea di memoria senza nome:

Simbolo	Descrizione
yy	Copia la riga attiva nell'area temporanea.
y*mod*	Copia nell'area temporanea il testo fino all'estensione indicata dal modificatore.
dd	Trasferisce la riga attiva nell'area temporanea.
d*mod*	Trasferisce nell'area temporanea il testo fino all'indicazione 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.

12.6.1 Copia e spostamento con nome

Quando si vogliono utilizzare delle aree temporanee di memoria specifiche, cioè identificate attraverso le lettere dell'alfabeto, si procede nei modi già visti nel caso dell'uso dell'area temporanea senza nome, con la differenza che i comandi sono preceduti da "*x*, dove *x* è la lettera che si vuole usare.
In questo caso è possibile aggiungere del testo a un'area temporanea: basta indicarla attraverso una lettera maiuscola.
L'elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso delle aree temporanee con nome è il seguente:

Simbolo	Descrizione
"*x***yy**	Copia la riga attiva nell'area temporanea *x*
"*x***y***mod*	Copia nell'area temporanea *x* il testo fino all'indicazione dal modificatore.
"*x***dd**	Trasferisce la riga attiva nell'area temporanea *x*.
"*x***d***mod*	Trasferisce nell'area temporanea *x* il testo fino all'indicazione dal modificatore.
"*x***p**	Incolla il contenuto dell'area temporanea *x* prima del cursore.
"*x***P**	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.

12.7 Ricerche

Per effettuare delle ricerche all'interno del documento aperto con *vi* si possono utilizzare delle espressioni regolari attraverso due comandi: **/** e **?**. La sintassi per la ricerca in avanti è:

/modello

mentre per la ricerca a ritroso è 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*]. I comandi di ricerca attraverso espressioni regolari sono i seguenti:

/*modello_regexp*	Cerca in avanti una corrispondenza con il modello indicato.
?*modello_regexp*	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 può essere utilizzato con *vi* è solo quello elementare e valgono in
particolare le regole indicate nella tabella seguente:

.	Corrisponde a un carattere qualsiasi.
\	Fa perdere il significato speciale che può 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 (l'uso della barra obliqua inversa serve per togliere il significato speciale che ha il punto in un'espressione regolare).

12.7.1 Ricerche e sostituzioni

La ricerca e sostituzione sistematica avviene attraverso un comando particolare che inizia con i due punti. La sua sintassi è la seguente:

:*inizio*,*fine***s/***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 **$** può 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 è fisso, ma può 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/prova/g
Sostituisce ogni occorrenza della parola "pippo" con la parola "prova". La ricerca viene effettuata a partire dalla prima riga fino all'ultima.
:%s/pippo/prova/g
Questo è 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 dieci righe a partire da quella attiva.
:%s/^..//gc
Esegue la stessa operazione dell'esempio precedente, applicando la sostituzione su tutto il file, richiedendo però conferma per ogni sostituzione.
:.,10s/^/xxxx/g
Inserisce la stringa "xxxx" all'inizio di dieci righe a partire da quella attiva.

12.8 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.
In particolare:

u: annulla l'ultimo comando.

U: annulla le modifiche sulla riga attiva.

12.9 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 può 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'attività di *vi* si ottiene con il comando:**`q`**. I comandi possono essere combinati tra loro, per esempio quando si vuole salvare e concludere l'attività simultaneamente con il comando:**`wq`**. Il punto esclamativo (**`!`**) può essere usato alla fine di questi comandi per forzare le situazioni, come quando si vuole concludere l'attività 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.
Riassumendo:

Simbolo	**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	Uscita dal programma.
:q!	Uscita dal programma (senza salvare).

Esempio in cui si vede *vi* durante l'esecuzione del comando:**`w`**, prima della pressione conclusiva del tasto [*Invio*]:

```
Il mio primo documento scritto con vi.
Non è facile, ma è bene impararlo!~
~
~
~
~
~
                                :w esempio_
```

CAPITOLO 13

SERVIZI DI RETE

13.1 Il paradigma client-server

La comunicazione su Internet si basa sul modello CLIENT – SERVER, in cui il client inoltra una richiesta verso il server che accetta la richiesta e risponde al client con le informazioni desiderate:

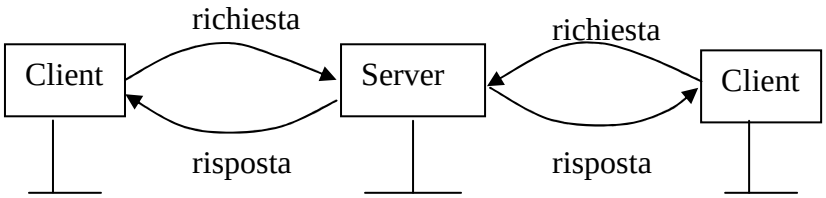


Figura 7: modello Client-Server.

Client e server sono due processi che si scambiano informazioni e come tali possono girare sia su macchine distinte come sullo stesso calcolatore; sarebbe quindi inesatto, anche se ormai facente parte dell'uso comune, usare questi nomi per riferirsi non al programma, ma alla macchina.
La figura sopra riportata lascia intuire come il server possa ricevere anche molte richieste contemporaneamente; nasce quindi la necessità di definire secondo quale ordine tali richieste vengono soddisfatte. Se scegliessimo una soluzione del tipo proposto dall'algoritmo FCFS, incontreremo sicuramente grossi disagi: infatti anche una piccola richiesta potrebbe aspettare molto tempo prima di essere eseguita perché preceduta da una richiesta più lunga. In questo modo il client riceverà con molto ritardo le informazioni che aveva richiesto al server e l'utente non vedrà apparire nulla sullo schermo per un tempo troppo lungo.
Per risolvere questo problema si possono adottare diverse strategie,una di queste consiste nella generazione di *thread:* il server genera "un processo figlio" di tipo thread, cioè un processo che condivide, principalmente, lo spazio di indirizzamento con il processo padre. La sequenza di azioni che esegue il server una volta mandato in esecuzione è quindi la seguente:

- il server rimane in attesa finché non riceve una richiesta, a questo punto genera un thread secondario che si occupa di soddisfare la richiesta, mentre il thread principale ritorna in ascolto di nuove richieste. Esiste quindi una copia del server che si occupa di ogni nuova richiesta,in questo modo i client che richiedono trasferimenti di file di piccole dimensioni non devono attendere il completamento di operazioni potenzialmente molto lunghe.
- un server del tipo descritto si dice *concorrente;* tale concorrenza si riferisce alla possibilità del server di erogare contemporaneamente lo stesso servizio a più client senza che questi debbano mettersi in coda; questa è una caratteristica di fondamentale importanza per il modello client –server.

13.2 Servizi di rete d uso comune

13.2.1 WWW (world wide web)

Tra i vari servizi che internet mette a disposizione degli utenti, il servizio World Wide Web (www) è, oggi, quello più famoso e usato per l'accesso e la visualizzazione di informazioni.
Per il servizio Web il programma che gira sul lato client prende il nome di BROWSER mentre quello su lato server si chiama SERVER WEB. Il servizio www si basa sul protocollo http, che è un protocollo di livello applicativo che gira sopra a TCP.
Il comportamento del server web nel gestire e rispondere alle richieste dei client è stabilito appunto dal protocollo http e dai parametri di configurazione scelti per quel server. Intervenendo su tali parametri abbiamo la possibilità di "guidare" il comportamento del server web (come vedremo più avanti per Apache).
Per capire il funzionamento di questo servizio, supponiamo di richiedere il caricamento della seguente pagina web:

`http: //www.ingegneria.unipi.it /index.html`

in generale una stringa di questo tipo prende il nome di URL e si compone delle seguenti parti:

protocollo: // indirizzo simbolico: porta / path del documento richiesto

la parte che precede i due punti specifica il protocollo usato nella comunicazione, la seconda parte (`www.ingegneria.unipi.it`) è l'indirizzo simbolico della macchina su cui si trova l'informazione (supponiamo che su tale macchina sia in esecuzione un server) mentre l'ultima parte, `index.html`, permette al server di identificare il documento che ho richiesto.
La parola prima dei due A questo punto:

1. il browser trasforma l'indirizzo simbolico che ha ricevuto in un indirizzo fisico;
2. il browser apre una connessione con il server sulla macchina remota sulla porta 80;
3. una volta portata a termine con successo la connessione il browser/client inoltra le sue richieste al server secondo il protocollo http;
4. il server risponde alla richieste del client inviando i dati richiesti o un opportuno messaggio di notifica se si sono verificati degli errori;
5. il client riceve i dati, se necessita di altre informazioni necessarie alla visualizzazione della pagina riparte con una nuova richiesta al server;
6. infine,sempre il browser, visualizza per l'utente il documento richiesto.

Analizziamo in dettaglio le prime 4 fasi.

TRASFORMAZIONE dell'INDIRIZZO da SIMBOLICO a FISICO

Come verrà chiarito più approfonditamente nel capitolo 2, gli indirizzi Internet che siamo abituati a conoscere, sono in realtà degli indirizzi simbolici usati per semplicità dagli utenti ma incomprensibili per il nostro calcolatore.
Il protocollo di comunicazione usato in Internet prevede invece l'utilizzo di indirizzi IP ovvero stringhe di 32 bit contenenti un numero binario. Al fine di colmare il divario tra gli indirizzi simbolici usati dagli utenti e quelli IP utilizzati dal software di rete, esiste il cosiddetto servizio di risoluzione dei nomi: **DNS**.
Nel nostro esempio il nome simbolico del destinatario è:

```
www.ingegneria.unipi.it
```

se il nostro calcolatore non è in grado da solo di trovare l'indirizzo IP corrispondente, inoltra una richiesta di risoluzione al DNS server che esegue per lui la ricerca e gli risponde con le informazioni desiderate.
Adesso che abbiamo a disposizione l'indirizzo IP del destinatario, possiamo provare a stabilire la connessione.

CONNESSIONE tra CLIENT e SERVER

Quando precedentemente parlavamo del server, abbiamo dato per scontato che i due processi, client e server, comunicassero tra di loro; vediamo adesso come effettivamente viene implementato il paradigma di comunicazione client-server che è stato introdotto.
Il problema da risolvere per garantire la comunicazione in Internet è la 'distanza': i due processi girano su macchine distinte e fisicamente distanti nello spazio.
Le modifiche da apportare al sistema in questo caso rispetto a quello in cui i due processi risiedono sullo stesso host non riguardano la struttura dei programmi client-server che rimangono gli stessi ma l'interfaccia di rete che deve permettere ai due processi di stabilire una connessione logica per lo scambio di messaggi. Tale connessione logica dovrà essere supportata da una connessione fisica.
La connessione fisica è formata dai collegamenti che uniscono i vari nodi della rete: cavi di rame, cavi di fibre ottiche ed etere (per le wireless lan).
La connessione logica utilizza i *socket*,oggetti creati da system call del sistema operativo che permettono ad un processo di inviare e ricevere dati da un altro processo che può risiedere su un diverso calcolatore.
Il sistema operativo crea il socket con una system call omonima, *socket ( )*, dopodiché utilizza un'altra system call, *bind ( )*,per associare il socket ad un determinato processo che sta girando su quella macchina,nel nostro caso il client.
Adesso è necessario individuare univocamente il processo remoto(il server) a cui legare il socket. L'aggettivo 'univocamente' si riferisce al fatto che il solo indirizzo IP della macchina non è sufficiente ad identificarne uno specifico processo dal momento che,su tale macchina, sono attivi contemporaneamente più processi. Si risolve questo problema associando ad ogni processo *una porta*.
Per il client il numero di porta associata è indifferente mentre per il server la scelta è più complessa: il server viene contattato da diversi client che,per inviargli una richiesta,devono sapere in anticipo il suo numero di porta. La soluzione è scegliere una porta pubblicamente riconosciuta sulla quale girano tutti i server web: la porta 80.
A questo punto posso associare il socket al processo remoto specificando indirizzo IP della macchina e numero di porta del processo grazie alla system call *connect( )*.
Dal lato server avviene qualcosa di analogo: il sistema operativo della macchina remota crea un socket che viene legato al processo server tramite una bind ( ).

Successivamente viene invocata la system call *listen ( ),*con lo scopo di stabilire il numero massimo di client che possono rimanere in attesa di risposta su quella connessione,cioè su quel socket.
E' ora compito del server mettersi in ascolto di eventuali richieste (sulla porta 80) ed essere pronto ad accettarle; viene invocata per questo la system call bloccante *accept( ),*che obbliga il server in una situazione di stasi fino all'arrivo di una richiesta.
Perché il tentativo di connessione del client abbia successo è necessario fargli eseguire la connect ( ) solo dopo che sul lato server è stata terminata la sequenza di operazioni appena descritte.
Stabilita la connessione client e server si scambiano informazioni grazie alle system call *send ( )* e *receive ( );*è importante ricordarsi di chiudere il socket quando smettiamo di usarlo:system call *close ( ).*

Una comunicazione di questo tipo viene definita *connection-oriented*: prima di poter comunicare client e server devono stabilire tra loro una connessione logica che verrà distrutta alla fine del dialogo. Una comunicazione connection-oriented è affidabile: siamo sicuri che il messaggio arrivi inalterato a destinazione.
Il protocollo che la realizza è TCP (*Trasmission Control Protocol*).

Schema riassuntivo di questo tipo di connessione:

client ⟶ socket ( )	server ⟶ socket ( )
client ⟶ bind ( )	server ⟶ bind ( )
	server ⟶ listen ( )
	server ⟶ accept ( )
client ⟶ connect ( )	
//…… scambio di informazioni…..///	
client ⟶ close ( )	server ⟶ close ( )

Esiste anche un altro tipo di comunicazione *connection-less* in cui server e client si scambiano messaggi senza stabilire prima un canale. In questo caso la comunicazione è inaffidabile: il messaggio può andare perso, alterato o può arrivare con un ordine ribaltato. Il protocollo che la realizza è UDP (*User Datagram Protocol*).

RICHIESTA HTTP del CLIENT:

La richiesta che il client inoltra si compone in realtà di diverse parti che non comprendono solo il nome del documento richiesto ma anche altri tipi di informazioni, vediamo un esempio:

```
GET    /ricerca/rmi.html      http/1.0
accept      text/html
```

```
accept      jpg
if_modified_since   (date)
```

La prima parola indica il *metodo* della richiesta ovvero quale tipo di operazione il client ha richiesto relativamente al documento seguente. I possibili metodi sono:

- **GET e POST**: per ricevere un documento(una pagina html);
- **HEAD**: per ricevere solo l' header del documento (vedi più avanti);
- **PUT**: per inviare e/o memorizzare dati sul server;
- **DELETE**: per cancellare un documento.

La seconda parte (*/ricerca/rmi.html*) prende il nome di *URI* e differisce dall'URL perché rappresenta solo il path del documento richiesto privato dell'indirizzo simbolico del server. Questa forma 'incompleta' di indirizzamento è possibile in una comunicazione connection-oriented (basata sul
intende utilizzare il client nella comunicazione.
L'ultima parte è costituita da un insieme di altre informazioni che impongono dei vincoli sulla risposta del server. Nell'esempio riportato questi vincoli riguardano:

- il tipo di file:
  `accept text/html:` il client accetta solo files di testo e html;
  `accept jpg:` il client accetta anche files jpg (immagini);
- la data di modifica del file:
  `if_modified_since (date)`: accetta il documento solo se è stato modificato dopo la data specificata;

RISPOSTA HTTP del SERVER:

Anche la risposta del server è articolata in diverse parti,vediamone un esempio:

```
http/1.0          200  Documents Follows
SERVER:           APACHE/4.0
DATE:             Monday  31 May 2001
CONTENT_TYPE:     text/html
            …..
// documento byte per byte //
```

La prima riga prende il nome di *Status Line* perché comprende,oltre la versione del protocollo http usato, lo *status code* del server relativo a quella richiesta. Lo status code è composto da un numero (il codice) e da alcune parole che lo specificano; vediamone alcuni esempi:

200 Documents Follows:il documento è stato inviato come richiesto;
304 Not Modified: il documento non è stato più modificato dopo la data richiesta;
401 Not Authorized:il client non è autorizzato a ricevere quel tipo di nformazione;
404 Page Not Found: il server non è riuscito a trovare il documento;
500 Server Error: si è verificato un qualche tipo di errore sul server.

Di seguito alla Status Line sono riportate alcune o tutte le metainformazioni relative al documento richiesto.
Alcune delle possibili metainformazioni sono:

- SERVER : riporta il nome del programma server;
- DATE : indica data e ora corrente;
- CONTENT_TYPE : specifica il tipo del documento che sta per inviare;
- CONTENT_LENGTH : specifica la lunghezza del documento;
- CONTENT_LANGUAGE : specifica la lingua del documento;
- CONTENT_ENCODED : avverte se il contenuto del documento è codificato;
- LAST_MODIFIED : riporta la data dell'ultima modifica.

Status Line e Metainformazioni compongono l'*header,* cioè l' intestazione e sono sempre presenti nella risposta http del server mentre il file vero e proprio (nell'esempio index.html) è inviato a seguito dell'intestazione solo se l'operazione richiesta dal client è una GET o POST e se la richiesta ha avuto successo; quest'ultima informazione può essere facilmente dedotta dallo status code che il server scrive nell'intestazione.

13.2.2 Posta elettronica

La posta elettronica è un altro esempio di servizio basato sul modello di comunicazione client /server.
Per utilizzare questo servizio, l'utente si rivolge ad un programma chiamato *user agent;* grazie ad esso può comporre il testo del messaggio e indirizzarlo. Una volta che il messaggio è stato completato non resta che inviarlo; lo user agent non compie direttamente questa operazione ma passa il documento al *mail transfer* che si occuperà della trasmissione.
La prima azione che compie il mail transfer è quella di controllare il destinatario, dal suo indirizzo simbolico ricava l'indirizzo IP con cui cerca di mettersi in contatto con il processo server sulla macchina remota.
Tale processo prende il nome di *mail server* e si distingue dal server web per il diverso numero di porta cui è associato.
Mail transfer e mail server stabiliscono una comunicazione basata sul protocollo SMTP (Simple Mail Transfer Protocol).
A questo punto il mail transfer invia il messaggio che viene catturato dal mail server e messo nella *mailbox* (casella postale) dell'utente indicato come destinatario, che potrà visualizzarlo grazie al suo user agent.
Nella precedente descrizione abbiamo dato per scontato un requisito non banale per il funzionamento di questo servizio: il mail server deve essere sempre attivo per poter ricevere i messaggi in qualunque momento e di conseguenza il calcolatore su cui risiede deve essere sempre in funzione.
Questa situazione era accettabile per il servizio www dove avevamo solo alcune macchine adibite a server che potevano restare sempre accese; nel caso della posta elettronica invece ogni utente che intende ricevere messaggi dovrebbe lasciare sempre acceso il suo computer per permettere al mail server di intercettarli.
Il modello proposto non è quindi accettabile e deve essere modificato in base ad esigenze reali.
Si può pensare di spostare le caselle postali di più utenti su un'unica macchina server sempre attiva e pronta a ricevere la posta. Tutti i messaggi destinati ad una persona saranno quindi registrati nella relativa mailbox su questo server.

L'utente è così libero di scegliere quando controllare la propria posta:è infatti sufficiente che direttamente dal suo PC si colleghi al server e chieda di scaricare il contenuto della sua casella.
Per implementare questa modifica si usano altri due programmi: uno lato client che si attiva quando l'utente vuole accedere alla sua mailbox e uno lato server che risponde all'interrogazione del client controllando la casella richiesta e inviandogli l'eventuale posta presente all'interno; il client la riceve e infine la passa allo user agent che si occuperà di visualizzarla per l'utente. La comunicazione tra client e server è basata sul protocollo POP3 o IMAP.
E' riportato di seguito una schema riassuntivo dei programmi coinvolti per lo scambio di posta elettronica:

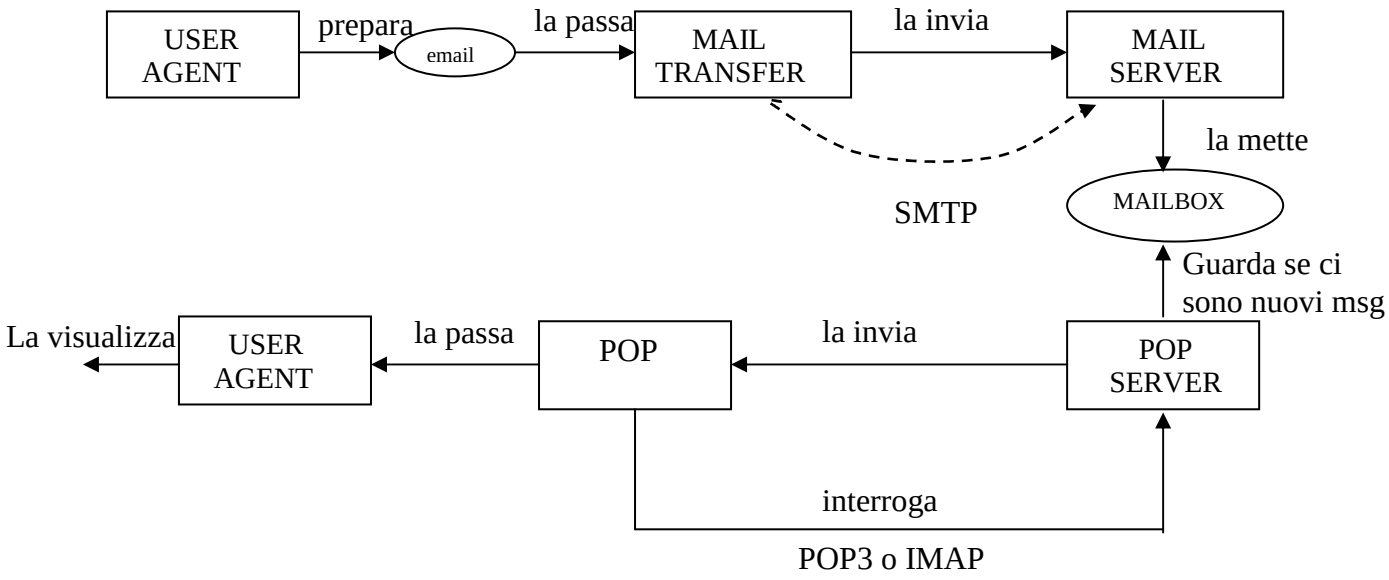


Figura 8: schema dei programmi coinvolti nello scambio di posta elettronica

13.2.3 FTP

Il protocollo FTP per il trasferimento dei files in rete era quello che veniva usato più comunemente prima della nascita del WEB. FTP risultava essere molto più efficiente del protocollo http 1.0 per quanto riguardava lo scambio in serie di più files. Tali prestazioni erano e sono dovute al fatto che FTP, finito di trasferire un file,continua a lasciare aperta la connessione che aveva stabilito e che il protocollo http 1.0 avrebbe invece distrutto e ricostruito per ogni trasferimento.
Attualmente questa mancanza da parte del protocollo http è stata colmata dotando la versione http 1.1 della stessa possibilità di mantenere attiva la connessione.
Come le altre applicazioni di rete, FTP segue il modello client-server:l'utente esegue un'applicazione locale,il client FTP, che si preoccupa di contattare il server FTP,stabilendo una connessione di controllo di tipo TCP;lungo tale connessione il client invia le sue richieste /comandi al server e riceve le relative risposte. Come per gli altri server, anche FTP server è un programma dormiente cioè resta bloccato in attesa di ricevere richieste.
Una differenza con quanto visto finora, è invece rappresentata dalla connessione FTP usata per l'effettivo scambio dei dati. Tale protocollo infatti prevede l'uso di due connessioni

distinte: una connessione di controllo,che abbiamo già citato, usata da client e server per lo scambio di comandi e richieste ed una connessione dati usata per trasmettere il file richiesto. Quest'ultima connessione viene creata un attimo prima di inviare i dati e distrutta subito dopo che è avvenuto il trasferimento; la connessione dati quindi permane solo fintanto che trasferisco il contenuto di un singolo file mentre la connessione di controllo rimane attiva finché è attivo il collegamento tra client e server.
Al fine di evitare possibili conflitti, FTP usa porte diverse per le due connessioni.

Una tipica sequenza di scambio tra client FTP e server FTP potrebbe quindi essere la seguente:

- l'utente A richiede di scaricare un file;
- in un primo momento il client FTP stabilisce con il server FTP solo una connessione di controllo lungo la quale inoltra la sua richiesta;
- il server FTP riceve la richiesta e controlla sia i diritti di A sul documento sia le informazioni che il client gli ha inviato insieme alla richiesta; se tutto corrisponde,il server apre la connessione dati e su tale connessione invia il file;
- il server FTP chiude la connessione dati che aveva creato appena finisce di trasferire l'ultimo byte del file;
- il client FTP riceve il documento richiesto e può subito inoltrare un'altra richiesta sulla connessione di controllo che è rimasta attiva oppure può decidere di terminare la sessione chiudendola.

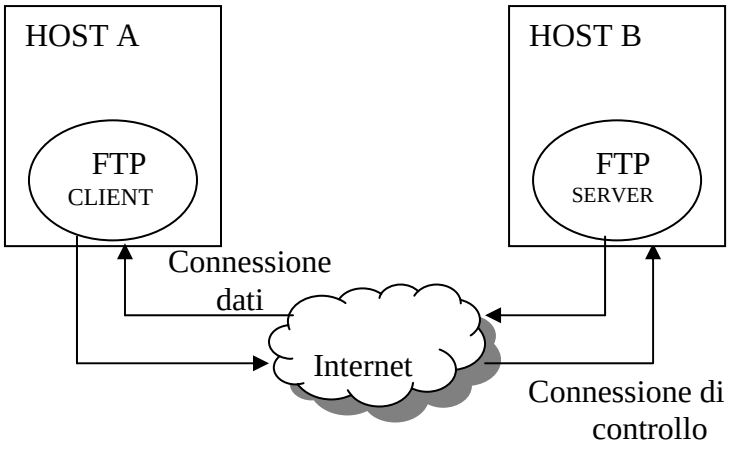


Figura 9: Connessioni FTP

Le frecce indicano chi ha richiesto le connessioni: la connessione di controllo richiesta dal client,quella dati dal server. I ruoli che eravamo abituati a vedere sono quindi invertiti,con il server originario che ricopre il ruolo del client e viceversa.
Il vantaggio di avere due connessioni distinte si riscontra nel caso in cui vogliamo inviare un comando che riguarda lo stesso file che in quel momento sta transitando sulla connessione dati.
Supponiamo ad esempio che il client stia ricevendo un file molto lungo e, per qualche motivo,decida di interrompere la trasmissione; se,come nel caso del web,avessi una solo linea

di comunicazione dovrei aspettare di accodare il comando di interruzione a seguito dei pacchetti dati,nel caso di FTP invece posso inviare la richiesta lungo la connessione di controllo,il mio comando quindi arriva subito al server mentre i dati stanno ancora viaggiando sull'altra linea.
L'applicazione FTP è ancora oggi fra quelle maggiormente usate in Internet e fino a non molto tempo fa il traffico dovuto al trasferimento di files ammontava a un terzo del traffico totale sulla rete mentre la quantità di informazioni generate dalla posta elettronica e dal sistema dei nomi di domino era molto inferiore.
La maggior parte degli utenti di questo servizio oggi necessita solo di una piccola manciata dei comandi di base di FTP.
Dopo aver lanciato il programma FTP dobbiamo impartire il comando *open* per stabilire la connessione di controllo TCP; questo comando richiede il nome del dominio del calcolatore a cui collegarsi. Dopo aver stabilito la connessione, FTP richiede l'identità dell'utente cioè il nome di login e una parola chiave per accedere alla macchina. Il primo serve per determinare a quali file l'utente ha accesso;infatti il nome di login deve corrispondere ad un account regolare sul calcolatore collegato. Ottenuta in questo modo l'autorizzazione, gli utenti possono trasferire file. Infine per chiudere la connessione di controllo l'utente digita *close.*

13.2.4 Mailing list e News

Le *mailing list* sono liste di indirizzi email a cui l'utente può associare un nome;ogni volta che avrà l'esigenza di inviare uno stesso messaggio a tutti gli appartenenti di una lista potrà usare come destinatario il nome associato così,in modo trasparente, il messaggio sarà spedito a tutti i componenti della mailing list selezionata.
Le *news* sono dei gruppi di discussione sugli argomenti più disparati; tali discussioni avvengono in rete ed è quindi necessaria un'applicazione che consenta di inviare e ricevere messaggi da molti altri utenti.
Un piccolo gruppo di discussione può essere trattato come una mailing list, con dei programmi di supporto che aggiungono e tolgono alla lista determinati indirizzi su richiesta degli utenti. In questo modo ogni componente del gruppo vede i messaggi inviati da tutti gli altri utenti e viceversa.
Nella realtà questa soluzione è possibile,come premesso,solo per piccoli gruppi di discussione; infatti, sebbene il singolo componente creda di aver mandato un solo messaggio a tutti gli altri membri, il protocollo di trasferimento ne crea in realtà una copia per ogni altro destinatario all'interno della lista.
Se il numero dei componenti aumenta considerevolmente, nascono molti problemi dovuti alla necessità e al tempo impiegato per riprodurre un così grande numero di messaggi; tali messaggi inoltre devono essere inviati a host situati in luoghi diversi del mondo.
Una soluzione può essere la seguente: si organizzano le mailing list attraverso una rete di *server di news*.
Gli utenti che intendono usufruire del servizio di news si registrano attraverso uno dei vari nodi rappresentati dai server di news, indicando a quale gruppo intendono partecipare, cioè scelgono un argomento. Successivamente,ogni volta che l'utente si riconnette, il server provvede ad inviargli tutti i nuovi messaggi registrati e si occupa di inoltrare al resto del gruppo gli eventuali messaggi che tale utente vuole spedire.
Esiste all'interno di ogni server news una copia di tutti i messaggi che vengono raccolti all'interno della macchina stessa in directory suddivise per argomento.
Il protocollo di comunicazione usata dai server di news per comunicare e scambiarsi i messaggi è NNTP (Network News Transfer Protocol).

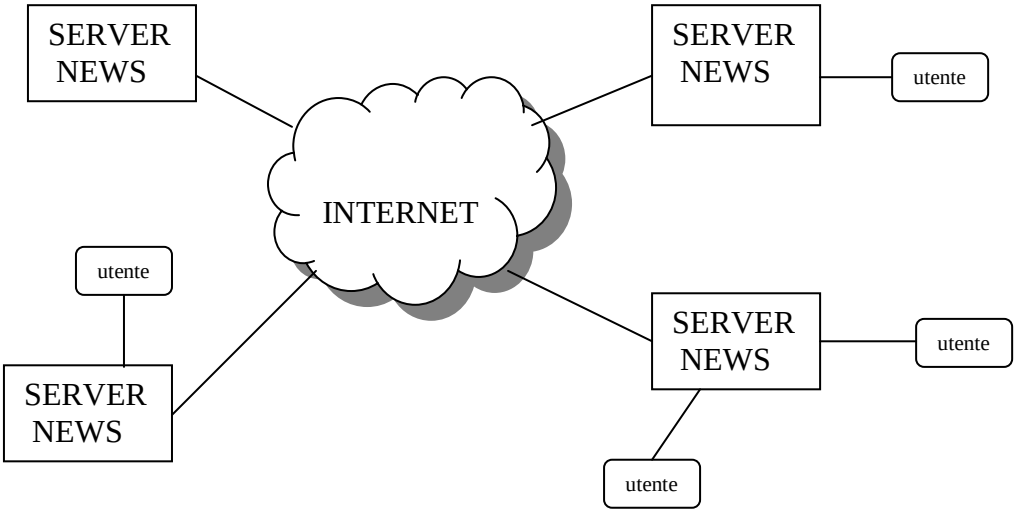


Figura 10: server di news

CAPITOLO 14

CONFIGURAZIONE DELLA RETE

14.1 Indirizzamento IP

Con la nascita di Internet, ovvero con l'interconnessione di più reti fisiche dalle diverse caratteristiche, nasce anche l'esigenza di un sistema di comunicazione universale e omogeneo: un'inter-rete.
ll software dell' inter-rete deve nascondere le caratteristiche fisiche delle sottoreti, offrendo la possibilità a un host di comunicare con un qualunque destinatario, indipendentemente dalla locazione fisica e dalle specifiche della rete fisica a cui quest'ultimo appartiene.
In sostanza quindi l'inter-rete è un'astrazione basata sul software dei protocolli.
Un fattore cruciale nella progettazione dell'inter-rete è l' indirizzamento. Al fine di simulare un sistema di comunicazione omogeneo, tutte le postazioni devono utilizzare lo stesso formato degli indirizzi e allo stesso tempo ogni indirizzo deve identificare univocamente un singolo host sulla rete universale.
Il protocollo dell'inter-rete deve definire uno schema di indirizzamento astratto; gli indirizzi fisici non sono infatti adatti allo scopo perché il loro formato è fortemente dipendente dalla specifica sottorete.
Il formato degli indirizzi per Internet è definito dal **Internet Protocol**, più semplicemente detto schema di indirizzamento IP.
Ogni postazione collegata alla inter-rete è identificata da un **indirizzo IP** formato da una stringa di 32 bit che permette di identificare univocamente tale postazione e di comunicare con essa.

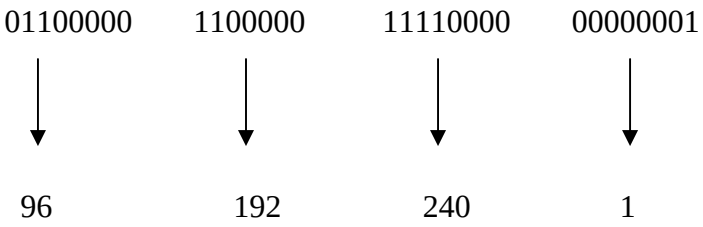
Ogni indirizzo IP è, concettualmente,diviso in due parti: il prefisso e il suffisso.
Il prefisso serve per identificare la rete fisica di appartenenza dell' host, mentre il suffisso individua un determinato calcolatore all'interno di questo sottorete. Questa divisione ha lo scopo di rendere più agevole il meccanismo di instradamento perché virtualmente suddivide la totalità delle postazioni facenti parte della rete universale in sottoreti che a loro volta sono composte dai singoli calcolatori.
Ogni sottorete dunque può essere identificata tramite il suo specifico suffisso, detto numero delle rete, **network number**.
Può accadere che due host su sottoreti diverse abbiano lo stesso suffisso, ma in questo caso hanno sicuramente prefissi diversi, allo stesso modo host sulla stessa sottorete hanno stesso prefisso ma suffissi diversi. In questo modo può essere garantita l'univocità degli indirizzi ma è necessario che sia i numeri di rete sia i suffissi siano assegnati in maniera coordinata.
Esistono delle società dette Fornitori del Servizio Internet (**ISP**) che si occupano della distribuzione dei prefissi garantendone l'univocità. Tali società sono coordinate da un'organizzazione centrale che è l'Ente per l'assegnazione degli indirizzi Internet (**IANA**).
I suffissi possono essere assegnati localmente in modo indipendente; sarà compito dell'amministratore di rete garantire la coordinazione.
Dalla lunghezza del prefisso e del suffisso dipendono le dimensioni della sottorete, visto che la somma totale dei due è fissa e pari a 32 bit.
Se il prefisso è corto e il suffisso lungo, si avranno poche sottoreti con moltissime postazioni ognuna, viceversa si avranno molte sottoreti con poche postazioni ciascuna.
Non poteva essere fatta una scelta forzata della lunghezza di suffisso e prefisso in quanto esistono sia sottoreti del primo tipo come del secondo.Usando ad esempio un indirizzo da

mille postazioni possibili in una rete con solo qualche centinaio di macchine effettive si sarebbero sprecati molti indirizzi.
I progettisti optarono quindi per una suddivisione in classi degli indirizzi IP; tali classi si distinguono oltre che per le dimensioni di suffisso e prefisso anche perché i primi bit dell'indirizzo sono fissi.
Nello schema sottostante sono riportate le 5 classi a cui può appartenere un indirizzo IP:

Le classi A,B,C sono dette primarie perché sono quelle da cui vengono effettivamente assegnati gli indirizzi agli host. La classe D è dedicata alle trasmissioni multicast, mentre la classe E è riservata per usi sperimentali.

14.1.1 Notazione decimale puntata degli indirizzi IP

Da quanto visto finora gli indirizzi IP sono numeri binari di 32 bit.Usati in questa forma però sarebbero particolarmente scomodi e facilmente confondibili.
Per questo motivo il software che interagisce con gli utenti impiega una notazione meno complessa detta notazione **decimale puntata.**
Ogni stringa di 32 bit viene suddivisa in 4 porzioni da 8 bit ciascuna, ogni porzione così composta viene tradotta nell'equivalente numero decimale (ogni byte viene interpretato come un intero senza segno) e divisa dalle altre tramite punti.
Ad esempio l'indirizzo IP ' 01100000110000001111000000000001 ' viene suddiviso in:



e diventa ' 96.192.240.1'.

Gli indirizzi così ottenuti andranno da 0.0.0.0 a 255.255.255.255. Pensate che questo range, che può apparire molto alto, risulta oggi quasi insufficiente a soddisfare tutti i collegamenti richiesti; è stato così messo appunto un nuovo protocollo di rete (IPv6), compatibile con l'attuale, che usa indirizzi IP a 128 bit.
Nella tabella sottostante è riportato per ogni classe l'intervallo di valori decimali che assume il primo byte dell'indirizzo:

CLASSI	INDIRIZZI
A	da 0 a 27
B	da 128 a 191
C	da 192 a 223
D	da 224 a 239
E	da 225 a 255

14.1.2 Indirizzi IP riservati

Anche tra le prime tre classi di indirizzi (A,B,C) non tutti gli indirizzi IP sono disponibili, alcuni vengono riservati per funzioni particolari, quali ad esempio indirizzare tutta una rete o un determinato insieme di calcolatori.
Per questo motivo IP definisce alcune categorie di indirizzi riservati. Tra questi vi sono:

- **Indirizzi delle reti:**
  talvolta può essere conveniente avere la possibilità di denotare con un indirizzo IP il prefisso di una rete fisica; in questo caso useremo il prefisso della rete stessa con suffisso zero. Ad esempio l'indirizzo 220.15.0.0 indica la sottorete di classe C il cui prefisso è 220.15.0.
  Un indirizzo di questo tipo non si riferisce ad una postazione ma alla rete, per questo motivo non dovrebbe mai essere usato come indirizzo del destinatario di un pacchetto;
- **Indirizzo di loopback:** un calcolatore, che sia connesso o meno alla rete fisica, deve avere una connessione virtuale a una rete immaginaria interna al calcolatore stesso. L'accesso a questa rete virtuale avviene tramite un'interfaccia che non esiste nella realtà, denominata 'lo'. L'indirizzo usato per riferire questa interfaccia è sempre lo stesso,127.0.0.1, e prende il nome di indirizzo di loopback. Ovviamente nessun pacchetto recante l'indirizzo di loopback potrà mai attraversare la rete vera e propria in quanto si richiuderà immediatamente sull'host che lo ha inviato;
- **Indirizzo di "questo calcolatore":** ogni calcolatore deve conoscere il proprio indirizzo IP perché i pacchetti che spedisce o riceve dalla rete devono contenere sia l'indirizzo IP del mittente che quello del destinatario. Il protocollo TCP/IP contiene altri protocolli che si occupano di appurare l'indirizzo IP della macchina all'avvio; per comunicare con essi è però necessario che il software specifichi l' IP del calcolatore, che abbiamo detto non è ancora disponibile. Possiamo usare in questi casi l'indirizzo speciale ' 0.0.0.0 ',conosciuto come default route, cioè strada predefinita per l'instradamento dei pacchetti;
- **Indirizzo di broadcast orientato:** un'altra opportunità che ci offrono gli indirizzi riservati è quella di indirizzare non solo il singolo calcolatore ma tutte le postazioni facenti parte di una stessa sottorete. L'indirizzo che viene usato in questo ambito prende il nome di indirizzo di

trasmissione orientata broadcast (direct broadcast address) ed è composto dal prefisso della rete che intendiamo raggiungere ed un suffisso di soli bit a 1. Ad esempio l'indirizzo broadcast della rete 131.5. sarà 131.5.255.255;

- **Indirizzo di broadcast ristretto:** esiste anche un'altra modalità di broadcast, detta ristretta (limited broadcast) e si riferisce alla possibilità di fare una trasmissione broadcast sulla stessa rete a cui appartiene il calcolatore mittente.L'indirizzo IP riservato per questa modalità è composto interamente da 1.

14.1.3 Indirizzi IP statici e dinamici

Un' ulteriore classificazione degli indirizzi IP riguarda la loro modalità di assegnamento.
Un indirizzo IP statico è un indirizzo IP che è assegnato alla macchina una volta per tutte e rimane lo stesso per ognuna delle connessioni effettuate da quel calcolatore.
Un indirizzo IP **dinamico**, invece, non viene associato ad una macchina per sempre ma viene associato a macchine diverse in connessioni diverse; può quindi capitare che l'indirizzo IP che corrispondeva al mio calcolatore nell'ultima connessione sia diverso da quello che le corrisponde nella connessione attuale.
Uno dei protocolli che permette la configurazione dinamica della connessione alla rete di un gruppo di nodi è DHCP (*Dinamic Host Configuration Protocol*).
Cerchiamo di comprendere meglio il problema: immaginiamo una piccola azienda con una rete locale chiusa in cui si vogliono poter collocare dei calcolatori senza dover stabilire a priori gli indirizzi IP e i nomi corrispondenti.
Per poter utilizzare il protocollo DHCP,occorre un server (che chiameremo server DHCP) che risponda alle richieste degli client di assegnazione dell'indirizzo; allo stesso modo è necessario che i client siano in grado di inoltrare tali richieste.
Quando uno dei client contatta il server DHCP per la prima volta, il server gli assegna la configurazione e subito dopo i due stabiliscono l'intervallo di tempo (lease)per cui tale configurazione sarà valida.
Da quel momento in poi è compito del server tenere in memoria i dati delle configurazioni di tutti i nodi che si trovano nella rete di sua competenza; tali dati devono restare uguali per tutto il tempo di lease.
Il client da parte sua richiederà questi dati al server ogni volta che scade il tempo di lease.
Se vogliamo dare la possibilità alle postazioni della nostra rete locale di collegarsi a Internet il server DHCP deve essere dotato di funzionalità simili a quelle di un router (cioè la capacità di instradare i pacchetti che riceve dagli host della rete locale a seconda dell'indirizzo destinatario) e avere a disposizione un opportuno insieme di indirizzi IP validi da assegnare ai client.
Facciamo un'ultima nota su questo protocollo riguardo alle informazioni che il server DHCP può fornire ai client per definire la loro collocazione nella rete circostante.
Le informazioni minime sono naturalmente l'indirizzo IP e la maschera di rete.Le informazioni aggiungibili dipendono dalle caratteristiche del server e possono essere:

- l'indirizzo broadcast;
- il nome del nodo e il dominio relativo;
- l'indirizzo del router predefinito;
- l'indirizzo del DNS server;
- l'indirizzo del server di stampa.

14.1.4 Indirizzi IP pubblici e privati: NAT e IP Masquerading

Esistono alcuni indirizzi IP che non possono essere utilizzati su Internet poichè sono dedicati esclusivamente all'uso su reti private: tali indirizzi sono detti privati.
Grazie agli indirizzi privati possiamo, comprando pochi indirizzi pubblici, costruire una rete anche di grandi dimensioni (la rete privata) i cui computer possono accedere se necessario a Internet.
Gli indirizzi privati sono 10.x.x.x, 172.16.x.x e 192.168.x.x; questi indirizzi non possono essere usati al di fuori della rete privata perché non sono univoci a livello globale, cioè due calcolatori su sottoreti diverse possono avere lo stesso IP privato.
Sono chiamati pubblici tutti gli altri indirizzi IP che hanno libero accesso alla rete.
Abbiamo detto che, in condizioni normali, gli indirizzi privati non avrebbero la possibilità di accedere all'esterno; esistono due tecniche che permettono ugualmente ai calcolatori associati ad un IP privato di collegarsi ad Internet: una tecnica detta NAT (Network Addresses Translation) e un'altra conosciuta come **IP Masquerading** (mascheramento IP).
Iniziamo descrivendo la tecnica NAT: questa prevede che un nodo particolare della rete assuma funzionalità simili a quelle di un router,intervenendo però sui pacchetti allo scopo di sostituire gli indirizzi IP reali, cioè quelli privati, con IP pubblici.
Immaginiamo che la nostra rete privata sia formata da alcune postazioni, alle quali sono assegnati gli indirizzi IP che vanno da 192.168.1.0 a 192.168.1.46. Tutti questi calcolatori sono collegati ad un gateway di default che si occupa, usando il demone **natd**[1], di realizzare il mascheramento.

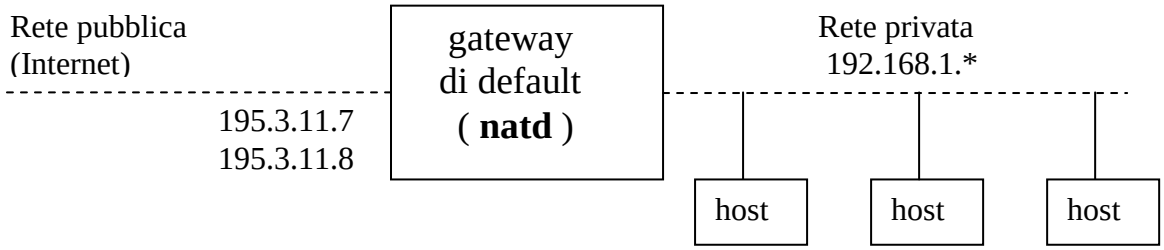


Figura 11: rete privata e rete pubblica

Ogni volta che un host all'interno della rete privata richiede il collegamento all'esterno, il demone natd si occupa di modificare l'indirizzo mittente dei pacchetti inviati dall'host sostituendolo con un opportuno indirizzo IP pubblico a disposizione del gateway di default. In questo modo l'indirizzo privato viene mascherato da uno pubblico consentendo al calcolatore che ne aveva fatto richiesta di accedere a Internet.
Gli indirizzi pubblici a disposizione del gateway di default saranno alquanto limitati, e solo pochi nodi della rete privata potranno accedere contemporaneamente all'esterno. Nell'esempio in figura solo due host per volta possono collegarsi a Internet usando gli indirizzi pubblici 195.3.11.7 e 195.3.11.8.
L'assegnazione degli indirizzi pubblici ai nodi della rete interna è solitamente dinamica, ma talvolta si preferisce abbinare in modo statico alcuni indirizzi univoci ad altrettanti indirizzi della rete privata.Questo permette ai nodi privilegiati di essere raggiungibili anche da un accesso esterno senza che debbano essere loro i primi ad instaurare una connessione.
Passiamo adesso ad analizzare l'altra tecnica di cui abbiamo parlato all'inizio: l'IP Masquerading.

[1] Quanto affermato è valido per il sistema operativo FreeBSD

A differenza di NAT, IP Masquerading permette a più host della rete locale di collegarsi a Internet tramite un unico indirizzo IP pubblico: quello del gateway di default.
Questo è possibile grazie al fatto che IP Masquerading mappa la coppia

< IP-privato, porta >

in una coppia

< IP-pubblico, porta >

cioè ogni volta che un nodo della rete locale cerca di connettersi a Internet, la richiesta arriva al gateway di default su una determinata porta, il gateway esegue la mappatura che abbiamo detto prima e assegna al nodo il proprio indirizzo IP associato ad una delle sue porte disponibili (cioè sostituisce l'indirizzo IP e la porta del mittente contenuto pacchetti del nodo sorgente con il proprio indirizzo e la porta da lui stabilita, quindi li inoltra su Internet).
In questo modo quando l'host destinatario invierà dei pacchetti di risposta tali pacchetti raggiungeranno il gateway di default; a questo punto il gateway capirà dal numero della porta (e grazie alla mappatura che aveva eseguito precedentemente) a quale nodo della rete locale inviare quei pacchetti.
Teoricamente l'IP Masquerading può supportare un numero molto grande di connessioni, fino ad esaurire tutte le porte disponibili per l'IP pubblico che usa.

14.2 Il servizio di risoluzione dei nomi DNS

Nei primi paragrafi di questo capitolo, abbiamo imparato come gli indirizzi IP possano essere riferiti più semplicemente usando la notazione decimale puntata in luogo della stringa da 32 bit definita dal protocollo di rete.
Chiunque conosca Internet però, sa che gli utenti non fanno direttamente uso degli indirizzi IP. Ai calcolatori infatti sono assegnati degli **indirizzi simbolici,** che risultano agli utenti più veloci da usare e più facilmente memorizzabili.
I nomi simbolici risultano molto comodi per gli utenti ma non rispettano il formato previsto dal protocollo IP. Quindi i nomi simbolici adoperati nelle applicazioni di rete devono essere tradotti in indirizzi IP primi di essere utilizzabili.
C'è bisogno perciò di un software che si occupi della traduzione da nome simbolico a indirizzo IP con l'ausilio di un archivio in cui sono registrate tutte le corrispondenze.
Tale archivio dei nomi simbolici non risiede su un solo calcolatore, ma deve essere distribuito su tutta la rete affinché ogni nodo vi possa accedere.
Il software per la traduzione dei nomi simbolici è stato implementato seguendo il modello client – server che abbiamo presentato nel capitolo 1.
L'archivio con le corrispondenze nome simbolico – IP è contenuto all'interno di un sistema di server collocati in diversi punti della rete e in grado di comunicare tra loro. Ogni volta che un'applicazione di rete ha bisogno di una traduzione simbolico – IP, inoltra la sua richiesta ad uno di questi server, diventandone quindi il client; in seguito il server risponderà con l'indirizzo appropriato.
Gli archivi contenuti all'interno dei server non contengono tutte le possibili corrispondenze; per questo motivo qualora un server non sia in grado di eseguire la traduzione richiesta, inoltra la richiesta ad un altro server della rete, diventandone a sua volta client e così via finché non si riesce ad ottenere l'indirizzo IP corrispondente.

14.2.1 Struttura dei nomi simbolici

Da un punto di vista sintattico, i nomi simbolici sono costituiti da un insieme di caratteri alfanumerici separati da punti.

Il metodo di assegnazione di questi nomi segue il cosiddetto Sistema dei nomi di dominio (Domain Name System) ovvero **DNS**; sono i server DNS a fornire il servizio di traduzione che abbiamo spiegato.
I nomi di dominio (così chiameremo in seguito i nomi simbolici) sono gerarchici e hanno la parte più significativa a destra. Se ad esempio consideriamo il nome:

```
mashi.maro.com
```

la prima parte rappresenta un preciso calcolatore (mashi) le parti seguenti sono il gruppo che lo possiede (maro.com).
Il DNS stabilisce solo come scegliere i segmenti più significativi del nome, mentre il numero, il nome e il significato dei restanti segmenti è libera scelta di chi 'acquista' un certo dominio.
Il segmento più significativo,detto anche top-level e stabilito dal DNS, deve essere uno tra i nomi riportati nella seguente tabella:

Nome del dominio	**Proprietario**
arpa	Dominio temporaneo Arpa
com	Società commerciale
edu	Istituzione accademica
gov	Ente dello Stato
int	Organizzazione internazionale
mil	Ente militare
org	Ente diverso dalle categorie precedenti
<sigla nazionale>	Una nazione (esempio.it è Italia)

Se, ad esempio, l'università di Lester (ridente cittadina alla periferia di Londra) desidera ricevere un nome di dominio deve rivolgersi all'ente Internet preposto dal quale riceverà un suffisso di dominio,cioè un nome da anteporre al top level corrispondente. In questo caso il nome del dominio assegnato all'università di Lester potrebbe essere:

```
lester.uk
```

all'interno di questo dominio, il proprietario potrà autonomamente scegliere la struttura dei nomi e la gerarchia. Ad esempio, il calcolatore pekkle del laboratorio di meccanica della facoltà di ingegneria dell'università di Lester, sarà:

```
pekkle.lab.meccanica.ing.lester.uk
```

è possibile intuire da questo esempio come il numero dei segmenti di un nome di dominio corrisponde alla gerarchia dei nomi stessi.
L'ente che assegna i nomi di dominio ne garantisce anche l'univocità, impedendo che due enti abbiano la parte più significativa del nome di dominio uguale.
La gestione dei nomi all'interno di un dominio può essere autonoma grazie al modello client-server sul quale è implementato il servizio DNS. L'intero archivio delle registrazioni è distribuito e suddiviso all'interno dei vari DNS server collegati alla rete. E' sufficiente che gli enti collegati ad Internet mettano a disposizione un DNS server in cui risiede l'archivio con le traduzioni relative ai propri nodi.
I server DNS formano un sistema unitario e sono organizzati secondo una gerarchia che riflette la struttura dei nomi.
Il livello più alto della gerarchia è sempre occupato da un **root server** che rappresenta l'ente autorizzato per i nomi di dominio di livello più alto, come.com,.it,.uk.
Il root server non possiede nessuna informazione riguardo alle traduzioni ma sa come contattare tutti i DNS server del livello inferiore. I DNS server dei livelli inferiori possono a

loro volta o contenere delle traduzioni o limitarsi a instradare la richiesta al livello ancora inferiore.
Ogni server è in grado di contattare un root server e tutti server dei livelli inferiori della gerarchia.
Supponiamo che un calcolatore dell'università di Lester voglia contattare un calcolatore dell'università di Bashey; la richiesta risale la gerarchia di DNS server fino ad arrivare al root server di.uk da lì scende verso i livelli inferiori fino ad arrivare al DNS che contiene l'informazione richiesta.
Il traffico generato sulla rete da un meccanismo di questo tipo sarebbe intollerabile; perciò esistono tecniche di ottimizzazione quali replica e caching.

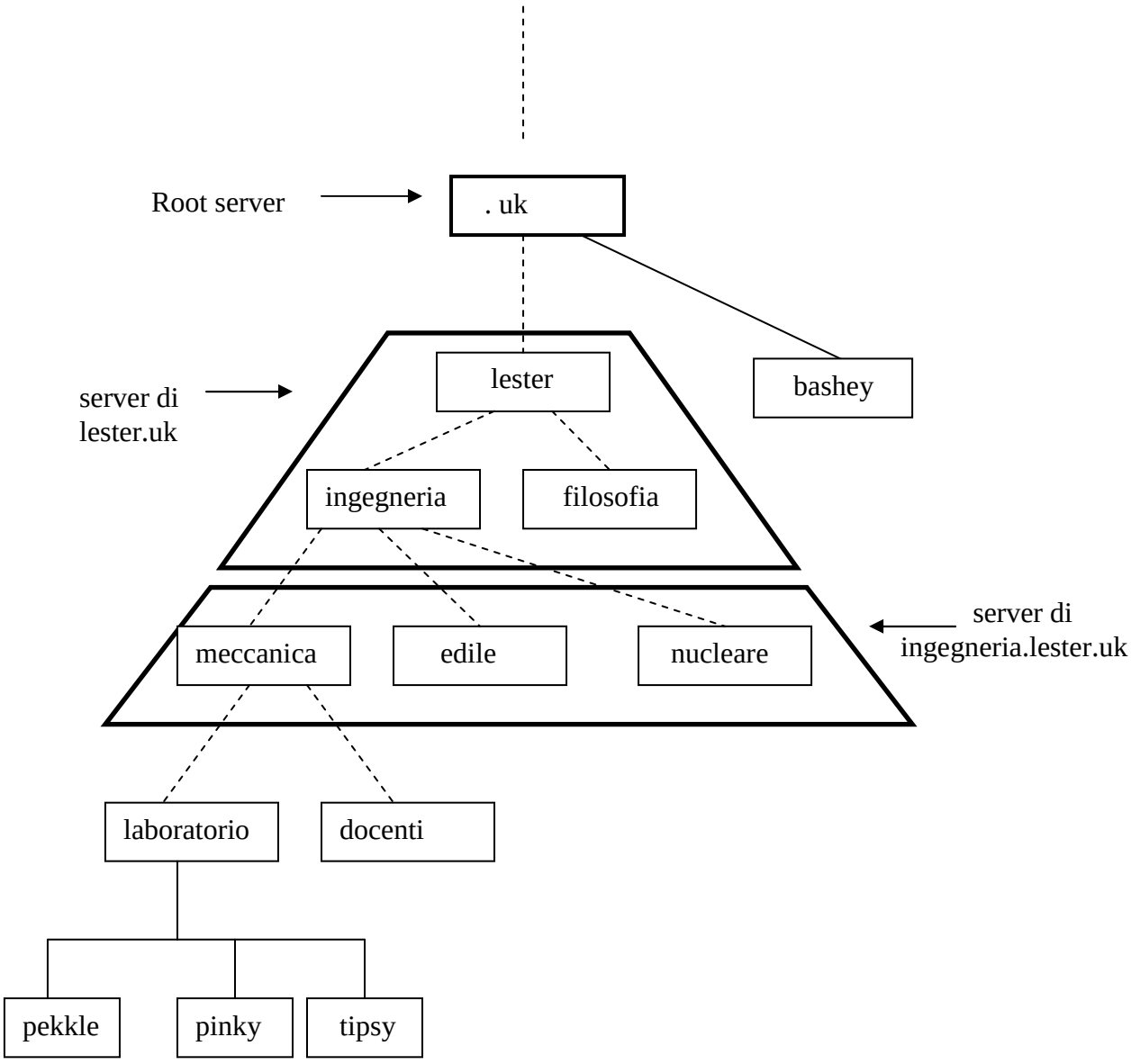


Figura 12: risoluzione DNS

14.2.2 Risoluzione di una richiesta DNS

Ecco i passi che compie il client DNS quando ha bisogno di risolvere un indirizzo simbolico:

1) Legge il file **/etc/host.conf.**
   Si tratta del file di configurazione dei principali servizi di rete. In particolare definisce in che modo si intendono risolvere i nomi di dominio. Ad esempio:
   order hosts, bind
   multi on
   l'opzione order indica l'ordine in cui vanno svolte le operazioni; in questo caso prima utilizzo il file /etc/hosts e poi interpello il servizio di risoluzione dei nomi, bind.

2) Legge il file **/etc/hosts**
   Questo file permette di definire i nomi dei calcolatori abbinati ai loro indirizzi IP, senza ricorre al server DNS. Ad esempio:
   127.0.0.1 localhost.localdomain localhost
   # necessario per il loopback
   193.11.3.7 pinky.pooh.uk pinky
   Se il nome che cerco è già presente risolto in questo file non ho bisogno di inoltrare la richiesta al DNS server.

3) Inoltro una richiesta al DNS server cercando l'indirizzo nel file /etc/resolv.conf.
   Il contenuto di questo file potrebbe essere:
   nameserver 193.32.1.2

14.3 Configurazione dell'interfaccia di rete

Adesso possediamo tutti gli strumenti necessari per capire il significato dei parametri di configurazione della nostra interfaccia di rete.
Il programma attraverso cui avviene la configurazione è 'ifconfig'.

Vediamo quali argomenti possiamo specificare lanciando il comando ifconfig:

- < nessun argomento >**:**
  mostra lo stato delle interfacce di rete;
- **interfaccia**:
  in questo modo specifico il nome dell'interfaccia che voglio configurare.
- **famiglie di indirizzamento:**
  subito dopo il nome posso specificare la sigla di un sistema di protocolli di comunicazione particolare.Questa scelta influisce sul modo di definire gli indirizzi che si attribuiscono alle interfacce. Se non specifico nulla viene scelta di default la famiglia IPv4.
- **indirizzo:**
  l'indirizzo è il modo con cui viene riconosciuta l'interfaccia all'interno del particolare tipo di protocollo che si utilizza.Generalmente si definisce con questo parametro l'indirizzo IP della nostra macchina.
- **opzioni:**
  - *up* | *down*: L'opzione up attiva l'interfaccia. Questa operazione avviene implicitamente ogni volta che assegno all'interfaccia un nuovo indirizzo;
  - *arp* | *-arp* : Abilita o disabilita l'uso del protocollo ARP per questa interfaccia (vedi capitolo 11 paragrafo 4);

- *allmulti | -all multi:* Abilita o disabilita la modalità promiscua dell'interfaccia. Questo permette di monitorare il traffico che transita sulla rete anche se i pacchetti non sono esplicitamente rivolti alla mia interfaccia;
- *mtu* : Permette di specificare l'unità massima di trasferimento dell'interfaccia;
- *broadcast:* Abilita o disabilita la modalità broadcast per l'interfaccia;
- *multicast:* Permette di attivare esplicitamente la modalità multicast;
- *hw:* Solitamente il nome dell'interfaccia rivela il tipo di hardware a cui appartiene l'interfaccia fisica. Se il driver dell'interfaccia è predisposto per diversi tipi di hardware, questa opzione specifica quale scegliere;
- *netmask* < indirizzo di netmask >: Stabilisce l'indirizzo IP della maschera di rete per questa interfaccia. Il valore predefinito è determinato in base alla classe a cui appartiene l'indirizzo IP dell'interfaccia. Se siamo in una sottorete è sempre necessario specificare il valore della netmask.

  Tale parametro ha un ruolo fondamentale ogni volta che si deve individuare il nostro calcolatore all'interno della sottorete. Definisce infatti una maschera che applicata all'intero indirizzo IP ne restituisce solamente il suffisso cioè la parte necessaria a distinguere il singolo calcolatore sulla sottorete.

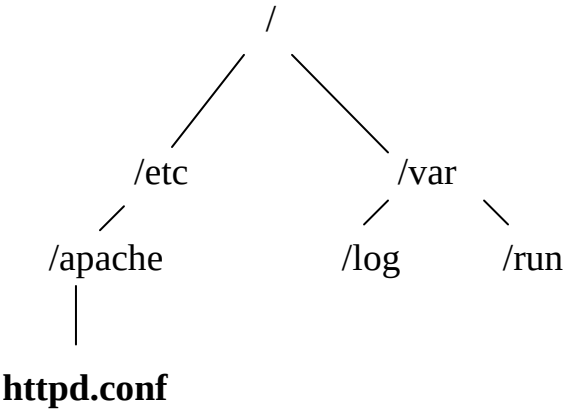
Queste sono solo alcune delle opzioni possibili. Per le altre potete consultare la guida in linea.

CAPITOLO 15

CONFIGURAZIONE DEL WEB SERVER APACHE

15.1 I files di configurazione di Apache

Prendiamo in esame il server Apache che è uno dei più diffusi nel mondo e costituisce lo standard di fatto per Linux e molte altre piattaforme.
Il server Apache è implementato su piattaforma UNIX –like da un processo che prende il nome di demone 'HTTPD'. E' lui che andremo ad analizzare più in dettaglio.
I files di configurazione di questo server sono tre: srm.conf,access.conf,httpd.conf.
Il file srm.conf contiene le direttive necessarie per dire al server dove si trovano i documenti, gli alias di directory speciali e altre informazioni simili sulla locazione dei dati.
Il file access.conf permette invece di controllare l'accesso nelle directory del sistema, definendo la politica di accesso ai dati.
Infine il file httpd.conf serve per configurare il comportamento del server dal punto di vista del sistema operativo.
Oggi c'è la tendenza ad includere le direttive contenute nei primi due files (srm.conf e access.conf) all'interno di httpd.conf così da avere un unico file di configurazione.Visto però,che il demone httpd andrebbe a leggere tutti e tre i files citati,è necessario specificare all'interno di httpd.conf che gli altri due files non devono essere letti (direttiva ResourceConfig e AccessConfig).
Nella figura sottostante è possibile vedere dove si trova normalmente httpd.conf all'interno della struttura del file system per un sistema FreeBSD.
Nel disegno è stata evidenziata anche la posizione di altre due directory: /var /log e /var /run, che interessano da vicino il funzionamento del server.
Quest'ultimo infatti scrive al loro interno particolari files.Il contenuto di questi files riguarda messaggi di logging nel caso della directory /var /log mentre per /var /run riguarda informazioni utili al server per gestire il programma.
Il server si trova quindi a compiere su /var /log e /var /run due diversi tipi di operazioni: aggiunta e lettura di files.



Per compiere tali operazioni è necessario che il processo in questione abbia i permessi di scrittura e lettura sulle directory; visto che un processo ha il livello di privilegio (cioè diritti e

permessi) dell'utente che lo ha mandato in esecuzione è necessario che il server sia avviato da un utente con diritti di lettura e scrittura su /var /log e /var /run.
Inoltre,essendo quest'ultime sottodirectory di /var,è necessario che il processo,e quindi l'utente, abbia anche il diritto in esecuzione su /var; l'unico utente che possiede questi privilegi è root; ne consegue che solo root può lanciare il demone httpd.
Vedremo più avanti che,in realtà, è possibile lanciarlo anche come utente non privilegiato dopo aver effettuato opportune modifiche.

15.2 Le direttive di configurazione

Ritorniamo al file httpd.conf che contiene proprio quei parametri di configurazione (direttive) di cui abbiamo parlato e sui quali possiamo intervenire per guidare il comportamento del server.
Immaginiamo di aprire questo file e vediamo singolarmente le direttive più interessanti cercando di capirne la funzionalità (le descrizioni che affronteremo ora sono valide per qualunque server Apache):

ServerType: I servizi di rete possono venir avviati in due modi: comeprocessi in attesa di richiesta (e in questo caso vengono avviati al boot) o tramite inetd.
Inetd è a sua volta uno dei servizi avviati al boot dallo script /etc /init.d/ inet e serve per avviare servizi di rete solo quando un utente remoto li richiede.
Quando la richiesta è stata servita,il servizio viene terminato.
Questo mi permette di risparmiare risorse che altrimenti sarebbero inutilizzate;
comporta però un carico maggiore dovuto ai possibili avvii ripetuti per un
servizio costantemente richiesto.
La direttiva ServerType permette di informare il server Apache sul modo in cui questo viene avviato; le opzioni possibili sono due
ServerType `standalone`*:* il server viene avviato al boot della macchina. Funziona in modo autonomo, ed è lui che gestisce direttamente le richieste che arrivano creando un processo figlio per ognuna di esse;
ServerType `inetd`: il server viene eseguito su richiesta da inetd che avvia un nuovo processo httpd per ogni richiesta;
In generale è preferibile usare la prima opzione perché i server Web sono caratterizzati per la maggior parte del tempo in cui sono in esecuzione da numerose richieste; meglio quindi avviarli subito al boot e lasciarli attivi.

Sintassi:	`ServerType` *`standalone`*`|`*`inetd`*

ServerRoot: specifica la directory in cui risiede il server, a partire dalla quale si trovano i file per la configurazione di Apache. Ogni volta che all'interno del file' httpd.conf ' trovo un path relativo come opzione di una direttiva, posso ricostruire il path assoluto anteponendovi la stringa che trovo nel ServerRoot. Esempio:

```
ServerRoot  /home /httpd
```

E' necessario che l'amministratore del sistema o comunque colui che ha il compito di manipolare i files di configurazione di Apache, abbia i permessi adeguati sulla directory che abbiamo specificato:in relazione all'esempio precedente, dovrebbe quindi avere il permesso di esecuzione sulla directory /home /httpd.

Sintassi:	`ServerRoot` *`directory-path`*
Di default:	`ServerRoot` *`/usr/local/apache`*

PidFile: definisce il path del file che contiene il PID del processo‘ http’. Tale file si trova solitamente all’interno della directory /var /run per cui un esempio potrebbe essere:

```
PidFile          /var /run /httpd.pid
```

Se il nome del file riportato nella direttiva indica un path relativo si ricorrerà alla direttiva ServerRoot per ricostruire il path assoluto.Questa direttiva si rivela utile ogni volta che nasce la necessità di inviare un segnale al server.
Pensiamo, ad esempio,di aver modificato le impostazioni definite nei files di configurazione del nostro Apache e di voler rendere attive tali impostazioni: sarà necessario inviare un segnale di SIGHUP (kill –1) al process ID del server memorizzato nel PidFile.

Sintassi:	`PidFile` *`filename`*
Di default:	`PidFile` *`logs/httpd.pid`*

ScoreBoardFile: questa direttiva definisce il nome e la posizione di un particolare file che contiene delle informazioni sul funzionamento del processo server al momento, ad esempio:

```
ScoreBoardFile       /var /run / statoServer
```

Il server Apache usa una tabella per le comunicazioni tra il processo padre e i processi figli ed alcune architetture richiedono la presenza di un file che faciliti questa comunicazione.
Se tale file non è specificato, Apache prima tenterà di creare la tabella totalmente in memoria e, se questo non funziona, proverà a creare il file sul disco. Invece specificando questa direttiva, Apache crea sempre il file sul disco.
Lo ScoreBoardFile può essere visualizzato con il comando httpd_monitor incluso nella distribuzione di Apache.

Sintassi:	`ScoreBoardFile` *`file-path`*
Di default:	`ScoreBoardFile logs/apache_status`

ResourceConfig,AccessConfig: queste sono le direttive di cui abbiamo parlato prima che servono in generale per definire nome e luogo dei due file di configurazione srm.conf e access.conf; ma che di solito si usa per escludere tali files (già contenuti in httpd.conf) dalla configurazione con l’opzione:

```
ResourceConfig    / dev / null
```

```
AccessConfig        / dev /null
```

TimeOut: La direttiva che stiamo considerando serve ad impostare un timeout composto dai tempi di attesa di tre eventi distinti. Infatti TimeOut definisce il numero di secondi che Apache aspetta in questi tre casi:

1. il tempo complessivo per ricevere una richiesta GET;
2. il tempo complessivo tra la ricezione di due pacchetti TCP consecutivi di una richiesta PUT o POST
3. il tempo complessivo tra due pacchetti ACK in risposta ad una trasmissione di pacchetti TCP

Ad esempio potrei impostare un Timeout di dieci minuti:

```
TimeOut         600
```

Sintassi:	`TimeOut`	*`seconds`*
Di default:	`TimeOut`	*`300`*

KeepAlive: Nel protocollo di comunicazione http 1.0, client e server chiudevano la loro connessione TCP al termine di ogni singola richiesta e la ristabilivano per la richiesta successiva. Visto che creare la connessione richiede un tempo sufficientemente lungo, la direttiva KeepAlive offre la possibilità di sfruttare la stessa connessione TCP per più richieste successive. In alcuni casi questa possibilità ha portato ad una riduzione quasi del 50% del tempo necessario a scaricare un documento HTML con molte immagini.

La direttiva KeepAlive ha queste opzioni:

KeepAlive *On* *:* la connessione viene mantenuta attiva;

KeepAlive *Off* *:* la connessione viene chiusa ogni volta;

Quando la direttiva KeepAlive è attiva (*On*) è comunque possibile stabilire un numero massimo di richieste permesse per connessione grazie ad un'altra direttiva: MaxKeepAliveRequest.

Sintassi:	`KeepAlive`	*`on |off`*
Di default:	`KeepAlive`	*`on`*

KeepAliveTimeout: La direttiva KeepAliveTimeout specifica quanto tempo aspettare la successiva richiesta sullo stesso canale di connessione prima di chiuderlo. Ad esempio per cinque minuti:

KeepAliveTimeout *100*

Impostare un KeepAliveTimeout troppo alto potrebbe causare problemi alle prestazioni di quei server pesantemente caricati di lavoro; infatti più alto sarà il TimeOut, più saranno i processi server occupati ad aspettare richieste su connessioni di client già sconnessi.

Sintassi:	`KeepAliveTimeOut`	*`seconds`*
Di default:	`KeepAliveTimeOut`	*`15`*

MinSpareServers,MaxSpareServers,StartServer: Ogni volta che lanciamo il server attiviamo, in realtà,un processo httpd PADRE il cui compito è quello di rimanere sempre in ascolto di eventuali richieste per essere pronto a riceverle; tale processo però non si occupa direttamente di gestire le richieste che riceve. Per questo scopo il processo padre genera un insieme di processi FIGLI (uguali a lui nel codice) ai quali affida l'effettiva gestione della richiesta; tra i membri di questo insieme viene effettivamente diviso il carico di lavoro.

I processi figli vengono generati dal server Web in diversi momenti:

- all'avvio del demone httpd: nel momento della sua attivazione il processo padre crea un certo numero di figli stabilito dalla direttiva StartServer;
- all'arrivo di ogni richiesta: ogni volta che arriva una nuova richiesta il padre la affida ad uno dei figli liberi (che non sta ancora gestendo richieste) dopodiché controlla il numero di figli rimasti liberi:
  - se questo numero risulta minore di una soglia, stabilita dalla direttiva MinSpareServers, crea un nuovo figlio in modo che il numero di processi secondari liberi non scenda al di sotto del valore minimo;
  - se il numero risulta compreso tra la soglia inferiore, MinSpareServers, ed una soglia superiore, stabilita da MaxSpareServers, non interviene per modificare il numero dei figli.

Può anche accadere che il processo padre decida di terminare alcuni dei suoi figli. Questo succede ad esempio quando ci sono troppi figli inattivi.Un processo secondario ritorna libero dopo aver eseguito la richiesta che gli era stata affidata e può quindi capitare che, dopo un picco di richieste, i figli rimasti inattivi diventino molti; se tale numero supera il valore stabilito dalla direttiva MaxSpareServers il padre termina i figli in eccesso.
Inoltre vedremo più avanti che il numero di figli che il padre può generare è limitato superiormente dal valore della direttiva MaxClients.
Riassumendo quanto detto finora, nel file httpd.conf esistono alcune direttive che servono per gestire la quantità di processi figli generati dal server padre; le tre che stiamo prendendo in considerazione sono:

MinSpareServers: che specifica il numero minimo di figli che devono essere mantenuti sempre liberi; per cui ogni volta che il padre accetta una nuova richiesta controlla se deve generare o meno un nuovo figlio per mantenere costante questo numero. Ad esempio:

MinSpareServers *10*

Sintassi:	`MinSpareServers`	*number*
Di default:	`MinSpareServers`	*5*

MaxSpareServers: che specifica il numero massimo di figli che possono rimanere inattivi,se tale numero viene superato il padre termina i processi figli in eccesso. Ad esempio:

MaxSpareServers *10*

Sintassi:	`MaxSpareServers`	*number*
Di default:	`MaxSpareServers`	*10*

StartServers: che specifica il numero di processi che il padre crea appena viene lanciato. Ad esempio:

StartServers *5*

Sintassi:	`StartServers`	*number*
Di default:	`StartServers`	*5*

MaxClients,MaxRequestsPerChild: anche queste direttive servono per il controllo dei processi figli e in particolare:

MaxClients: specifica ilnumeromassimo di figli che il server tiene attivi contemporaneamente, e cioè il numero massimo di clienti che possono essere serviti insieme.Le richieste che superano questo numero vengo messe in coda finché il server non è di nuovo disponibile. Ad esempio:

MaxClients *100*

Sintassi:	`MaxClients`	*number*

MaxRequestPerChild: serve per impostare il numero massimo di richieste un processo figlio può gestire prima di venire forzatamente terminato dal padre. Questo è molto utile nei casi in cui ci siano dei figli programmati in modo imperfetto,perché potrebbero ad ogni richiesta bruciare sempre un po' di risorse in più andando ad esaurire quelle disponibili.Ad esempio:

MaxRequestsPerChild *50*

Nel caso in cui scegliessimo l'opzione zero i figli possono gestire un numero infinito di richieste.

Sintassi:	`MaxRequestsPerChild` *`number`*
Di default:	`MaxRequestsPerChild` *`10000`*

Listen: Ad ogni macchina è associato un indirizzo IP che la identifica univocamente. Su una stessa macchina però possono essere attive, al momento, diverse applicazioni; come fa quindi un client a identificare l'applicazione verso cui vuole spedire i dati ? Ai processi in esecuzione viene associata una porta, ciascuna porta viene poi riferita tramite un numero. Le porte che vanno da 0 a 1023 sono porte riservate e possono essere usate solo da processi con livello di privilegio root; le altre sono accessibili anche per programmi di utenti non privilegiati. Nella maggioranza dei casi il server web gira sulla porta 80; questo rappresenta uno degli altri motivi per cui tale processo può essere lanciato solo da root visto che la porta 80 è tra quelle riservate. Dopo questa premessa è possibile capire meglio la funzione delle due direttive Listen e Port. Con Listen è possibile fare in modo che il server stia in ascolto, cioè accetti, le richieste provenienti dalla porta specificata oppure da una combinazione di indirizzo IP–porta ben precisa visto che di default (o se è specificata solo la porta) esso accetterebbe richieste da qualunque indirizzo IP disponibile sulla macchina. Posso inoltre usare più direttive Listen per dire al server di mettersi in ascolto su più indirizzi e porte. Per esempio facciamo in modo che il server accetti connessioni su entrambe le porte 80 e 8080:

Listen *80*
Listen *8080*

La direttiva Listen è oggi sempre richiesta in quanto la sua assenza dal file di configurazione provoca il fallimento dell'avvio del server; questo rappresenta una novità rispetto alle precedenti versioni di Apache.

Sintassi:	`Listen [`*`IP-address`*`:]`*`portnumber`*
Di default:	`è sempre necessario impostare tale direttiva`

Port: questa direttiva serviva a specificare il numero di porta su cui il server web doveva restare in ascolto; oggi però tale direttiva non viene più utilizzata e la si mantiene solo per compatibilità con le vecchie versioni. Risulta comunque interessante fare una nota sull'impostazione del valore di Port.Tale valore,come già accennato, è solitamente il seguente:

Port *80*

ma è possibile scegliere anche altre porte,tale scelta comunque può essere influenzata dalla modalità con cui viene attivato il server.
Quando ServerType è impostato su inetd è preferibile impostare un numero di Port diversa da 80.La ragione sta nel fatto che le porte al di sotto di 1024 sono privilegiate e quindi httpd deve essere lanciato da inetd con il livello di privilegio root. Questo potrebbe dare origine a

un problema di sicurezza perché un intruso malintenzionato potrebbe sfruttare il sito Web per accedere al sistema come root.
L'utilizzo della porta 80 è sicuramente più adatto al caso in cui il ServerType sia standalone, perché il processo primario httpd non fornisce un servizio diretto ai client ma avvia altri processi httpd, con privilegi ridotti che utilizza per fornire il servizio Web.

User,Group: Quando il demone httpd viene avviato, in situazione di default,da root, prima si associa alla porta 80 (direttiva Listen) e poi esegue il forking di un gruppo di processi secondari che forniscono gli effettivi servizi Web, questi processi sono detti figli. Visto che sono i processi figli a gestire effettivamente le richieste,e quindi sono loro ad accedere alle risorse richieste dai clienti, non è opportuno che essi abbiano i permessi di root. I figli lanciati dal server possono sia andare ad eseguire del codice (CGI) sia accedere al file system della macchina con i permessi di root; questo darebbe ad un cliente malintenzionato la possibilità,ad esempio,di andare a leggere il file delle password di sistema. Normalmente quindi si limitano i diritti di accesso del client limitando il livello di privilegio dei figli. A tale scopo le direttive User e Group permettono di definire l'utente e il gruppo fittizi da assegnare a quell' insieme di processi httpd secondari che si occupano di servire direttamente le richieste dei clients. Per sicurezza, tra le possibili opzioni di queste direttive,si preferisce scegliere:

User *nobody*
Group *nobody*

in questo modo i privilegi di chi accede al sistema via rete sono i minori possibili.
Un'alternativa all'uso di nobody potrebbe essere creare un ID utente e un ID gruppo esclusivamente per httpd. Creare UID e GID speciali per il server Web figlio (esempio www_user,www_group) ha il vantaggio di poter avere autorizzazioni per il gruppo, come ulteriore protezione per non dover dipendere completamente dall'autorizzazione generale concessa a nobody.
Esiste inoltre una notazione numerica con cui specificare le opzioni:si usano i numeri UID e GID preceduti dal simbolo ' # '; ad esempio:

User *#100*
Group *#100*

L' opzione noboby corrisponde alla notazione numerica # -1 (si può trovare anche # -2 usato per indicare la stessa cosa). Per quello che abbiamo detto prima riguardo ai processi secondari, il processo httpd padre deve essere lanciato con privilegio root, altrimenti non avrebbe il permesso di modificare utente e gruppo associati ai processi figli che genera, che continuerebbero quindi a girare con il suo livello di privilegio. Queste opzioni vengono ignorate dal server qualora il processo padre non venga lanciato da root.

Sintassi:	`User` *`unix-userid`*
Di default:	`User` *`#-1 (#-2)`*

Sintassi:	`Group` *`unix-groupid`*
Di default:	`Group` *`#-1 (#-2)`*

ServerAdmin: definisce l'indirizzo di posta elettronica del server Web che il server stesso include nei messaggi di errore, dovuti a problemi di funzionamento, che invia ai clients. Ad esempio:

ServerAdmin `you@your.address`

Sintassi:	`Serveradmin` *`email_address`*

ServerName: definisce il nome simbolico e la porta che il server utilizza per identificare se stesso e che usa nel creare l'URL di risposta alle richieste dei clients. Ad esempio, se il nome della macchina che ospita il nostro server Web è new.hostname.com, ma la stessa macchina possiede anche un DNS alias www.hostname.com desideriamo che il server venga identificato con quest'ultimo nome,dobbiamo impostare la direttiva ServerName come segue

ServerName *www.hostname.com:80*

Sintassi:	`ServerName` *`fully-qualified-domain-name`*`[:`*`port`*`]`

Se non viene specificato nessun ServerName,il server tenta di dedurre il nome dell' host su cui sta girando facendo un'operazione di lookup inverso sul suo indirizzo IP. Se invece è solo la porta a non essere specificata, il server utilizzerà la stessa porta da cui è arrivata la richiesta.

UseCanonicalName: Può capitare in molte situazioni che Apache debba costruire un URL con "autoriferimento",cioè un URL che si riferisce al server stesso. A seconda di come impostiamo la direttiva in questione, il server ricostruisce in modo differente il nome canonico e la porta da usare per l'autoriferimento:

UseCanonicalName *On* : Apache userà il nome e la porta specificate da ServerName per ricostruire il nome canonico del server;

UseCanonicalName *Off:* Apache userà il valore che era presente nella query string proveniente dal client per ricostruire il nome canonico.

E' possibile anche una terza opzione, *double,* della quale non verrà affrontata l'analisi per non scendere in un livello di dettaglio troppo approfondito per questo tipo di documentazione.

Sintassi:	`UseCanonicalName` *`On`* `|` *`Off`* `|` *`Double`*

Di default:	`UseCanonicalName` *`On`*

DocumentRoot: specifica la directory a partire dalla quale si trovano tutti i documenti HTML e che il server utilizza per mappare le richieste Ad esempio: se

DocumentRoot */usr / local / www / data*

e un client mi invia la richiesta

GET / ricerca/ rmi.html http 1.0 accept text/html

a questo punto, il server può ricostruire il path assoluto del file richiesto in questo modo:

/usr / local / www /data /ricerca / rmi.html

Sintassi:	`DocumentRoot` *`directory-path`*
Di default:	`DocumentRoot` */usr/local/apache/htdocs*

ErrorLog: Per avere la possibilità di monitorare effettivamente ilcomportamento del server web è necessario disporre di strumenti che ritornino all'utente informazioni sulle attività, le prestazioni e i problemi che si potrebbero verificare durante l'esecuzione del programma. Per questa finalità Apache dispone di apposite direttive che ci permettono di configurare le risorse di logging del server e di capire cosa contengono i file di log.
La direttiva ErrorLog definisce qual è il nome e la locazione del più importante file di log del server. In questo punto del file system infatti,Apache invierà le informazioni diagnostiche e registrerà ogni errore incontrato nel processare una richiesta. Per queste sue caratteristiche il file 'server error log ' è il primo posto utile da controllare quando sorge un problema che riguarda o l'avvio del server o la sua operatività, perché spesso contiene con precisione i dettagli di ciò che è andato storto e ci permette quindi di risalire alla fonte del problema. Un esempio di specifica della direttiva in questione potrebbe essere:

ErrorLog */ var/ log/ httpd-error.log*

Addentriamoci,seppur brevemente nell'analisi di questo file, per capirne meglio l'uso e le possibilità che ci offre.Il seguente messaggio,ad esempio, potrebbe essere parte del contenuto del nostro error log file:

[Mon May 31 08:30:00 1979] [warn] [client 135.6.6.91] cient denied by server access:

/ università/ home/ docenti/ esami/ test

Nella prima parte del messaggio il server registra ora e data completa di quando si è verificato l'evento indesiderato.
Nella seconda parte è specificato " il livello " di gravità dell'errore che è stato riscontrato (chiariremo meglio questo concetto quando affronteremo la direttiva LogLevel).

La terza parte riporta l'indirizzo IP del client che ha generato l'errore; di seguito c'è il messaggio vero e proprio che,in questo caso, ci notifica il rifiuto del server ad accettare la richiesta di quel client come previsto nella sua configurazione. Viene inoltre memorizzato anche il file-system path del documento richiesto.
Come vediamo dall'esempio il formato di questo file è descrittivo e di facile consultazione, nonché relativamente libero, sarà infatti possibile per l'utente stesso definire dei formati personalizzati sulle sue esigenze. Solitamente questi messaggi di log vengono memorizzati in un file situato nelladirectory / log; è quindi necessario che,chi lancia il server abbia i diritti in scrittura su tale directory. Nel caso che tale diritto non appartenga solo a root è necessario prestare molta attenzione per evitare che un utente o un cliente malizioso, con un livello di privilegio sufficiente, alteri il contenuto dell'error log file nascondendo informazioni preziose soprattutto per l'amministratore del sistema.
Un'ultima nota su questa direttiva riguarda la sintassi: in alternativa al path dell'error log file è possibile usare l'opzione *syslog[:facility]*,in questo modo si passa la gestione del logging al processo syslogd (per maggiori dettagli consultare la prima parte delle dispense relativa alla gestione dei files di log). Facility è una specifica opzionale la cui descrizione non viene riportata per non scendere in un livello di dettaglio troppo approfondito per questo tipo di documentazione.

Sintassi:	`ErrorLog`	*`file-path | syslog[:facility]`*
Di default:	`ErrorLog`	*`logs/error_log`*

LogLevel: Come abbiamo già accennato precedentemente, gli errori riscontrati dal server durante la sua esecuzione possono avere livelli diversi di priorità, Apache distingue tra otto possibili livelli di errore. La direttiva LogLevel permette stabilire una soglia di priorità (scelta tra uno degli otto valori possibili) che seleziona i messaggi da registrare nell'error log; in questo modo solo i messaggi che hanno livello di priorità superiore o uguale a tale soglia sono memorizzati.
Gli otto possibili livelli di priorità, in ordine crescente di priorità, sono:
- debug
- info
- warn
- error
- crit
- alert
- emerg

Esempio:

LogLevel *crit*

in questo modo il server scriverà solo quei messaggi di errore che hanno una priorità: crit,alert, emerg.

Sintassi:	`LogLevel`	*`level`*
Di default:	`LogLevel`	*`warn`*

LogFormat: parlando della direttiva ErrorLog abbiamo visto la possibile struttura con cui un messaggio viene registrato all'interno dell'error log e abbiamo detto che tale struttura poteva essere modificata.
In generale tutti i file di log del server Apache sono conformi al Common Log Format (CLF).CLF è uno degli standard più sfruttati dai produttori di server Web e usarlo significa che i log generati dal nostro server potrebbero essere elaborati da qualsiasi strumento di analisi di log, a patto che anch'esso sia conforme allo standard. La direttiva LogFormat ci permette creare dei ' tipi ' di log personalizzati. Il formato da usare nella dichiarazione di questi nuovi log è appunto CFL.
Vediamo un esempio generico:

LogFormat " *%h %l %u %t %r* %s> %b " *common*

Secondo questa direttiva,per ogni richiesta che arriva al server,esso crea una riga di log che contiene queste informazioni:

- %h : registra l' host name del client;
- %l : registra il nome che l'utente ha usato per effettuare il log-in al client. Visto che tale nome proverrebbe dal file passwd, molti client evitano di fornire questa informazione.
- %u : registra il nome utente usato per l'autenticazione del server Web;
- %t : registra l'ora e la data;
- %r : registra la prima riga della richiesta;
- %s> : registra lo stato dell'ultima richiesta;
- %b : registra il numero di byte inviati.

Se il valore di un campo non è disponibile nella richiesta,il server registra un trattino nel campo.
Il formato del log è quello racchiuso tra virgolette mentre *common* è la sua etichetta e non fa parte del formato, nel senso che non viene registrata nel file di log.
Lo scopo dell'etichetta è quello di associare il messaggio prodotto da un comando LogFormat al file specificato dal comando CustomLog.
In questo particolare esempio i logs di LogFormat sono legati al file / var/ log/ httpd/ apache/ access_log definito dal comando:

```
CustomLog / var/ log/ httpd/ apache/ access_log common
```

Un ultima nota sull'argomento dei log per sapere che Apache supporta anche il logging condizionale,che permette di selezionare i campi da registrare solo quando si verificano particolari condizioni. Queste condizioni possono essere anche i codici di stato restituiti dal server,come ad esempio:

200: Ok, la richiesta è valida;
401: Unauthorized, al client o all'utente è negato l'accesso;
500: Server Error, un errore non specificato si è verificato nel server;

e molti altri.

Sintassi:	`LogFormat`	*`format [nickname]`*
Di default:	`LogFormat`	*`"%h %l %u %t \"%r\" %>s %b"`*

CustomLog: stabilisce, tramite il suo primo argomento, il nome del file di log e, tramite il secondo argomento, il formato di tale file.Questo secondo argomento può essere rappresentato sia da un'etichetta definita tramite la direttiva LogFormat sia da una stringa completa che descrive il formato come spiegato nella direttiva precedente.
Supponiamo di avere definito LogFormat in questo modo:

LogFormat " %h %l %u %t \"%r\" %>s %b " common

allora

CustomLog CustomLog logs/access_log common

Sintassi:	`CustomLog`	*`nome file |[nickname]`*

TransferLog: Questa direttiva ha gli stessi parametri e produce gli stessi risultati della direttiva CustomLog, con la differenza che non permette di specificare esplicitamente il formato del log e usa l'ultimo LogFormat specificato prima di lui.

Sintassi:	`TransferLog`	*`file-path`*

HostNameLookups: permette di decidere se si vuole registrare il nome simbolico o l'indirizzo numerico dei client che cercano di accedere al servizio:
HostNameLookups *On:* si annota il nome simbolico di chi ha fatto la richiesta. In questo caso il nome è più leggibile ma è necessario l'utilizzo del DNS per la traduzione; visto che questa operazione richiede del tempo si preferisce di solito impostare la direttiva su off;
HostNameLookups *Off*: si annota l'indirizzo IP numerico.

E' possibile anche una terza opzione, *double*, della quale non verrà affrontata l'analisi per non scendere in un livello di dettaglio troppo approfondito per questo tipo di documentazione.

Sintassi:	`HostNameLookups` *`On | Off | Double`*	
Di default:	`HostnameLookups` *`Off`*	

Ulteriori direttive specifiche per l'accesso ai dati su server sono:
AccessFileName: mi permette di definire il nome di un file che, se presente all'interno di una directory, serve a controllare l'accesso ai dati interni secondo le restrizioni in esso definite. Il nome predefinito per questo file è.htaccess:

AccessFileName . *htaccess*

Questo comando ci permette di distribuire il controllo di accesso a coloro che creano e mettono a disposizione pagine Web personali, senza che debba essere l'amministratore a specificare in httpd.conf quali sono i diritti su tali files.

Sintassi:	`AccessFileName` *`filename`*
Di default:	`AccessFileName.`*`htaccess`*

< Directory *nome-dir* >

Options Indexes

< /Directory>:

In questo caso non ci troviamo di fronte ad una direttiva ma si tratta di una sezione del file di configurazione. Le sezioni ' Directory ' servono per raccogliere tutte quelle direttive (Options, AllowOverride, direttive di autorizzazioni ed altre) che fanno riferimento ad una particolare directory (*nome-dir*). Esempio:

```
< Directory  / home / paola  >
        Options   Indexes
< /Directory >
```

in questo modo consento che sia possibile visualizzare via Web il contenuto della directory specificata.
Altri valori possibili per il comando Options sono:

- All : autorizza l'uso di tutte le opzioni del server;
- ExeCGI : autorizza l'esecuzione degli script CGI in questa directory;
- FollowSymLinks : autorizza l'uso di link simbolici;
- Includes : autorizza l'uso di Server Side Include (SSI);
- MultiViews : consente di negoziare il linguaggio del documento(viene usato insieme ai comandi AddLanguage e LanguagePriority);
- None : non consente alcuna opzione del server;

E' consigliabile usare queste opzioni con prudenza perché tutte,eccetto MultiViews potrebbero causare problemi di sicurezza e tutte occupano risorse del Web. Per l'opzione Indexes vedi la direttiva Directory Index.
All'interno della sezione Directory possono essere usati ancora molti altri comandi dei quali non affronteremo l'analisi in queste dispense.

Sintassi:	`<Directory` *`directory-path`*`>...</Directory>`

UserDir: Come abbiamo precedentemente spiegato il server Apache ricerca tutti i documenti html che gli vengono richiesti dai client a partire da una determinata locazione del disco definita dalla direttiva DocumentRoot. Questo metodo è accettabile nel caso in cui la gestione dei documenti che devono essere resi disponibili in rete sia affidata esclusivamente all'amministratore; ma se volessimo offrire la possibilità anche agli utenti non privilegiati del sistema di condividere loro documenti ? Una possibile soluzione sarebbe quella di dare a tutti gli utenti, che eventualmente lo desiderano, l'accesso in scrittura alla directory che è stata definita come DocumentRoot in modo che possano autonomamente trasferire in quella parte del file system i documenti che intendono condividere. E' facile accorgersi che questa soluzione rappresenta un considerevole rischio per la sicurezza dei dati contenuti all'interno della directory in quanto uno qualunque degli utenti abilitati potrebbe, con i diritti in scrittura, modificare o addirittura cancellare interamente il contenuto di DocumentRoot. Una soluzione certamente più valida ci viene offerta dalla direttiva UserDir, che permette agli utenti di gestire le proprie pagine direttamente nelle loro home directory. Gli utenti creano una determinata sottodirectory all'interno della loro home e vi depositano le loro pagine html,le immagini, gli script e tutti gli altri files che vogliono mettere a disposizione. Il nome di tale sottodirectory deve essere lo stesso per tutti gli utenti e deve coincidere con quello riportato dalla direttiva UserDir. Di default questa direttiva è attiva e impostata a:

UserDir *public_html*

Tutto quello che gli utenti depositeranno in queste sottodirectory sarà automaticamente accessibile da web all' URL:

http: // < *indirizzo*> / ~ <*nome utente*>

Ad esempio tutti i files che l'utente *marzia* metterà nella directory:

/ home / marzia / public_html

saranno disponibili in rete all'indirizzo:

http: // localhost / ~ marzia /

Affinché questo meccanismo funzioni a dovere è necessario che il server Apache possieda i diritti necessari per accedere alle sottodirectory delle home personali, altrimenti si potrebbe verificare un errore del tipo *403: Forbidden.*
Riferendoci all'esempio precedente:

chmod *-R 755 / home / marzia / public_html*

Questa direttiva può essere totalmente disabilitata con l'opzione:

UserDir *DISABLED*

Sintassi:	`UserDir` *`directory-filename`*

Di default:	`UserDir` *`public_html`*

DirectoryIndex: Quando all'interno dell'URL richiesto dal client non viene specificato il nome di un file ma quello di una directory,il server restituirebbe l'elenco del suo contenuto se per quella directory è specificata l'opzione Indexes. E' possibile, sia per motivi di sicurezza sia per evitare che l'utente si perda tra un'infinità di files, mascherare tale contenuto grazie alla direttiva DirectoryIndex:

DirectoryIndex *index_html*

in questo modo ogni volta che all'interno dell'URL compare solo il nome di una directory,il server cerca se al suo interno esiste il file index_html,se lo trova restituisce al client tale file.

Esiste la possibilità di specificare nella direttiva più di un file,in questo caso il server restituirà il primo che trova. Ripetiamo che se non esiste nessuno dei file specificati e l'opzione Indexes per quella directory è attiva,il server genererà da solo la lista del contenuto della directory.

Sintassi:	`DirectoryIndex` *`local_URL [local_URL]...`*
Di default:	`DirectoryIndex` *`index_html`*

Alias,ScriptAlias: si rivelano utili per mappare gli URL in modo alternativo rispetto al meccanismo definito da DocumentRoot. Tali direttive si trovano nella forma:

Alias < *directory passata* > < *directory reale* >
ScriptAlias < *directory passata* > <*directory reale* >

dove la directory passata è quella che il client scrive nell' URL all'atto della richiesta mentre la directory reale è quella che il server le sostituisce nel momento in cui deve cercare il documento all'interno del file system. Ad esempio,con la dichiarazione:

Alias */ icons /* *" usr / local / www / icons "*

quando il server incontra un URL del tipo

GET */ icons / pagina* *http 1.0*

estrae la URI dalla GET, esegue la sostituzione e cerca il file

/ usr /local / www / icons / pagina

Allo stesso per ScriptAlias con una differenza: si suppone che all'interna delle directory specificate vi siano dei file eseguibili.

15.3 Il processo server

Una volta ‘settato’il comportamento che deve avere il nostro server web lo possiamo attivare e rendere pronto a ricevere richieste da client.
Se tra le varie modifiche avessimo anche deciso di spostare o sostituire il file httpd.conf, è necessario usare il comando httpd con l’opzione –f,nel seguente modo:

httpd -f < Path assoluto del file httpd >

in questo modo indico esplicitamente quale file di configurazione il demone httpd deve leggere ed eseguire prima di iniziare a gestire il suo servizio.
Se successivamente vengono apportate modifiche al file di configurazione,posso renderle effettive con il comando:

Kill -HUP < PID di httpd >

Da questo momento in poi il server è attivo con la nostra configurazione e si mette in attesa.

15.3.1 Processo padre e processi figli

In realtà quello che ho attivato è un processo httpd PADRE che è sempre in ascolto di eventuali richieste per essere pronto a riceverle ma che non si occupa di gestirle. A tale scopo il processo padre genera dei processi FIGLI (uguali a lui nel codice) ai quali affida l’ effettiva gestione della richiesta.
Come già accennato quando parlavamo delle direttive MinSpareServers e StartServers il processo padre genera, appena lanciato,un certo numero di figli prima che arrivino le richieste e fa in modo che ci siano sempre almeno un determinato numero di figli liberi.
Per questo motivo se osserviamo i processi attivi su una certa macchina subito dopo aver lanciato il server non vedremo un unico processo httpd ma una serie di processi httpd,che sono stati lanciati dal padre,cioè quello con PID più basso.

Vediamo un possibile esempio usando il comando ‘ ps -aux ‘:

PID	TTL	STAT	TIME	COMMAND
< ….tralasciamo le parti precedenti … >				
435	1	R	0:05	httpd
436	2	R	0:03	httpd
440	3	S	0:01	httpd
443	4	S	0:00	sbin/ mingetty / tty 2
449	5	R	0:01	httpd

privilegiato vediamo come nell’esercitazione seguente.
Per conoscere anche l’utente che ha lanciato il comando, possiamo usare ‘ top ’.
Inoltre l’httpd primario si occupa di modificare, in base alle specifiche, l’UID e il GID dei processi secondari che hanno il compito di soddisfare direttamente le richieste esterne.

In questo modo abbiamo la possibilità di stabilire i privilegi che devono avere quegli utenti che accedono al sistema tramite Internet.
Si può osservare che il processo padre rimane sempre attivo mentre i figli vengono terminati e riavviati secondo quanto specificato dalle direttive MaxRequestsPerChild, MaxClients, MinSpareServers e MaxSpareServers; per accorgersi di questi cambiamenti è sufficiente guardare il PID dei processi.
Se volessi modificare il file di configurazione dopo che ho già lanciato il demone httpd sarebbe sufficiente inviare il comando

Kill -HUP < Pid di httpd >

al processo padre, in questo modo anche tutti i processi figli andranno a rileggere il file di configurazione.
Evidenziamo nuovamente che tutto quello fatto finora, dall'avvio del demone httpd alle modifiche sul file httpd.conf è possibile solo per un utente con livello di privilegio root.
Abbiamo però la possibilità di rendere accessibili tutte queste operazioni anche a un utente non privilegiato vediamo come nell'esercitazione seguente.

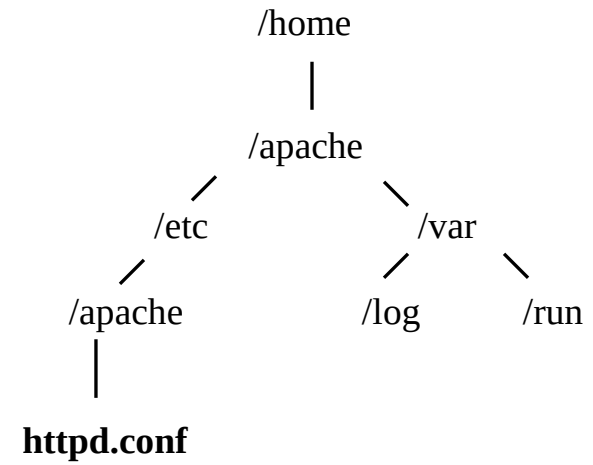
ESERCITAZIONI

ESERCITAZIONE 1

Far partire il server come utente non privilegiato.

Svolgimento

Per prima cosa è necessario ricostruire a partire dalla nostra directory / home la struttura di file system che abbiamo visto nel paragrafo 9.1:



Solo in questo modo avremo la possibilità di:

- intervenire sul file di configurazione. Prima questa operazione non era possibile per un utente qualsiasi perché il file httpd.conf era contenuto in una sottodirectory di /etc sulla quale non aveva i diritti necessari per compiere delle modifiche;
- intervenire,con i maggiori diritti possibili, sulle nuove directory /home/var/log e /home/var/run, che abbiamo ricostruito perché possano essere utilizzate da utente non privilegiato così come venivano usate da root;

Ora che possiamo intervenire sul file di configurazione di Apache, prima di poter lanciare il server, dobbiamo modificare:

- PidFile: cambia la posizione in cui il server deve andare a cercare il file:
  da / var /run a
  /home /var /run /httpd.pid;
- ScoreBoardFile: anche per questo file cambia la locazione:
  /home /var /run /statoServer;
- il numero della porta su cui gira il server web: la porta 80 non è più utilizzabile in quanto, essendo tra le prime 1024, è riservata a quei processi lanciati dall'utente privilegiato. Devo allora modificare il valore assegnato alla direttiva Port con un numero maggiore di 1024 ad esempio:
  Port 8080;

- tutti gli indirizzi relativi ai files di log; quindi le direttive ErrorLog, LogLevel,LogFormat,CustomLog e TransferLog devono specificare path del tipo:
  ErrorLog /home /var /log /httpd-error.log

Come accennato quando parlavamo delle direttive User e Group, qualora il processo httpd non venga lanciato da root ma da un utente con privilegi inferiori, le due direttive non vengono prese in considerazione.
Quando anche il file di configurazione è pronto, dobbiamo notificare al demone il suo cambiamento di posizione con il comando:

httpd -f /home /etc /apache / httpd.conf

in questo modo il demone legge le nuove impostazioni e l'utente non privilegiato riesce a lanciare il server.
Se volessimo apportare nuove modifiche dopo che il server è già stato avviato, possiamo usare il comando:

kill -HUP < PID di httpd >

Per verificare il successo del nostro intervento possiamo:

a) aprire l'interfaccia grafica (startx)
b) aprire netscape
c) dare come URL quello del nostro stesso terminale, specificando la porta che abbiamo associato al server
   127.0.0.1: 8080

A questo punto dovrebbe vedersi visualizzata una pagina in cui il server Apache avvisa che sta girando correttamente.

ESERCITAZIONE 2

Modificare le impostazioni SpareServers di default del server Apache.

Svolgimento

Le direttive in questione sono contenute nel file di configurazione httpd.conf; per modificarle è quindi necessario che l'utente possa accedere al file con i diritti di scrittura.
Una volta aperto httpd.conf cerchiamo al suo interno:

- MinSpareServers per stabilire il numero minimo di figli che il server deve tenere liberi in attesa di risposta;
- StartServers per stabilire il numero di processi secondari che httpd lancia al suo avvio;
- MaxSpareServers per stabilire il numero massimo di figli che possono rimanere inattivi;

quindi andiamo a sostituire i valori predefiniti con quelli che riteniamo più opportuno e infine salviamo le modiche apportate.
Se,come ipotizzato,abbiamo i diritti necessari per intervenire sul file di configurazione possiamo lanciare il demone httpd che si comporterà secondo le nuove direttive.Se invece non possediamo sufficienti diritti sarà necessario ripercorrere tutti i passaggi svolti nell'esercitazione1 prima di poter modificare le direttive che ci interessano.In quest'ultimo caso inoltre prima di lanciare nuovamente il server Web è necessario notificare al processo l'avvenuto cambiamento di locazione del file di configurazione.

Per verificare che le modifiche siano state accettate posso eseguire il comando:

```
ps -aux
```

e controllare quali e quanti sono i processi httpd attivi.

ESERCITAZIONE 3

Fare in modo che il server accetti la connessione solo da due specifici indirizzi IP e numeri di porta.

Svolgimento

Quando abbiamo parlato della direttiva Listen,avevamo visto come,nell'uso più comune, si preferiva specificare solo il numero di porta sulla quale il server doveva rimanere in ascolto senza precisare l'indirizzo IP così da permettere il collegamento a qualunque host. Se però riguardiamo la sintassi della direttiva è chiaro come oltre al numero di porta, obbligatorio, è possibile specificare anche un preciso indirizzo IP;in questo caso il server accetterà richieste solo dal client con quell'indirizzo e sulla quella porta. Esiste anche la possibilità di specificare più indirizzi è sufficiente ripetere la direttiva Listen per ognuno degli indirizzi che intendiamo ascoltare.
Supponiamo di scegliere l'indirizzo IP 129.180.3.1 e la porta 80 e l'indirizzo IP 131.124.4.5 e la porta 8080:

- per prima cosa dobbiamo aprire il file di configurazione di Apache e modificare la direttiva Listen nel seguente modo:

```
Listen          129.180.3.1:80
Listen          131.124.4.5:8080
```

- di seguito è necessario informare il server,che supponiamo già attivo,delle modifiche effettuate sul file di configurazione,con il comando:

```
kill      -HUP   < PID di httpd>
```

La sequenza descritta consiste di diverse operazioni; cerchiamo di capire,anche se non richiesto esplicitamente dall'esercizio, quale deve essere il livello di privilegio di un utente per eseguirle con successo.Per semplicità supponiamo di aver specificato un solo indirizzo IP,il primo;considerazioni analoghe possono essere fatte nel caso siano presenti entrambi gli indirizzi.
Il primo intervento è la modifica del file di configurazione httpd.conf: come detto più volte, in una situazione di default, solo l'amministratore del sistema,root, può manipolare questo file. Esiste comunque la possibilità per un utente non privilegiato,che esegua le modifiche spiegate nell'esercitazione 1,di eseguire questo primo intervento.
La seconda operazione è l'assegnamento alla direttiva Listen dello specifico valore 129.180.3.1:80:mentre la scelta dell'indirizzo IP non influisce sul livello di privilegio che l'utente deve avere per rendere effettiva tale modifica, il numero di porta impone invece dei vincoli. Come detto più volte,la porta 80 fa parte delle prime 1024 porte privilegiate riservate a quei processi lanciati da root. Di conseguenza tale modifica può avere successo solo se apportata da root.
Infine l'ultima operazione che consiste nel lanciare il comando kill al processo httpd non prevede particolari restrizioni di privilegio.

ESERCITAZIONE 4

Se ci accorgiamo dal degrado delle prestazioni che il nostro server sta gestendo contemporaneamente un numero troppo elevato di richieste, su quale parte del file di configurazione è necessario intervenire ?

Svolgimento

Se vogliamo forzare il server a gestire contemporaneamente un minor numero di richieste dobbiamo intervenire su una delle direttive che riguardano il numero di processi figli attivi sulla macchina.
Sono infatti questi processi secondari che si occupano di gestire effettivamente le richieste; inoltre intervenendo su di loro permettiamo al processo padre e quindi al server stesso di continuare a svolgere le sue funzionalità.
La direttiva che dobbiamo modificare è

MaxClients *100*

Quest'ultima infatti specifica il numero di figli che il server tiene attivi contemporaneamente, cioè il numero massimo di clienti che possono essere serviti contemporaneamente.
Se la nostra macchina sta incontrando difficoltà, a livello di risorse disponibili,nella gestione del richieste probabilmente il numero impostato per MaxClients è troppo alto, non ci resta quindi che diminuirlo di una quantità opportuna:

MaxClients *80*

ESERCIZI

Riportiamo adesso il testo di alcuni esercizi che lasciamo svolgere allo studente.

Esercizio 1

Supponiamo che l'utente matteo voglia mettere a disposizione su Internet i suoi documenti contenuti nella sottodirectory locale:

```
/usr /matteo / home /pghtml
```

Qual è la sequenza di operazioni che deve svolgere?

Esercizio 2

Intervenire sulla direttiva LogFormat per definire un nuovo formato di log che mi dica:

- l'host name del client che ha fatto la richiesta;
- il nome utente usato per l'autenticazione del server Web;
- la data e l'ora in cui è stata effettuata la richiesta;
- la prima riga della richiesta.

Esercizio 3

Cambiare il livello minimo di priorità dei messaggi di log che il server registra.

Esercizio 4

Sia data una directory con un path particolarmente lungo e difficoltoso, ad esempio:
Come potremo rendere più agevole l'identificazione di tale directory da parte del client? (direttiva Alias)

CAPITOLO 16

GESTIONE DI UN FIREWALL

16.1 Il problema della sicurezza

L'apertura di Internet al vasto pubblico ha portato con sé il problema delle intrusioni da parte di estranei a documenti privati. Sono state elaborate diverse soluzioni che permettessero in qualche modo di proteggersi da tali intrusi: una di queste è il Firewall.
Solitamente il Firewall è qualcosa che si interpone tra una rete esterna (ad esempio Internet) e una rete interna (ad esempio una LAN) per evitare un accesso indiscriminato a quest'ultima da parte di host esterni.
Il Firewall è un programma con una funzione simile a quella di un filtro per il quale possiamo stabilire la politica di accesso a seconda delle nostre esigenze.
Supponiamo di voler proteggere la rete locale (che chiamiamo genericamente LAN) dalla rete esterna a cui è collegata (Internet):

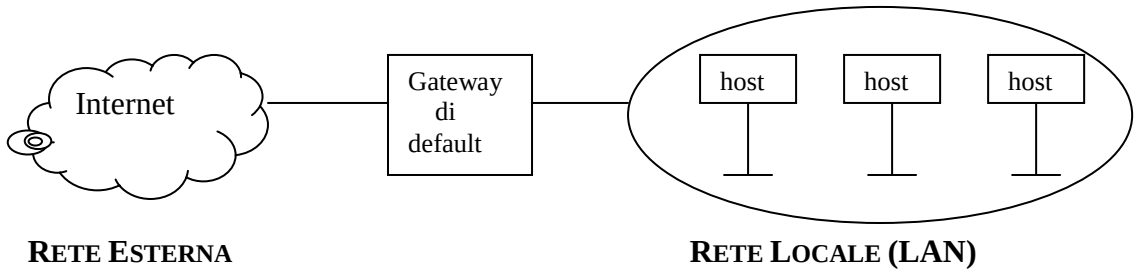


Figura 13: rete esterna e rete locale

allora il Firewall sarà collocato sul gateway di default e controllerà tutto il traffico che attraversa il gateway bloccando quei determinati messaggi che non soddisfano le regole del filtro.

16.2 Tipi di firewall

Esistono due tipologie fondamentali di Firewall:

- filtro di pacchetti IP;
- proxy;

Il primo tipo opera a livello di trasporto e permette di bloccare o abilitare il traffico che attraversa la rete,definendo i tipi di pacchetto (stabiliti in base al protocollo), gli indirizzi IP e le porte utilizzate.Tale Firewall è in grado di controllare i tipi di servizio che vengono utilizzati e gli indirizzi IP coinvolti nella comunicazione ma,solitamente, non ha la possibilità di registrare i collegamenti che sono stati fatti,né può identificare gli utenti che li utilizzano; è come se avessimo un router che si occupa solo di filtrare i pacchetti in base alle nostre regole.
Il proxy invece è una sorta di intermediario che opera a livello applicazione e si occupa di sostenere le connessioni per conto di un altro host della rete interna. Un utente esterno che cercasse di stabilire una collegamento verso un host interno si troverebbe, in realtà,connesso

con il proxy. Tale Firewall ha un ruolo attivo e può mantenere memoria delle azioni compiute ed,eventualmente, può identificare gli utenti che lo utilizzano.

16.3 IPFW

Il Firewall che analizzeremo sotto FreeBSD prende il nome IPFW, appartiene alla prima tipologia e come tale può intervenire solo fino al terzo livello dell'architettura OSI.
Il programma IPFW definisce le regole di accesso per il firewall elencandole in una lista in cui ad ogni regola è associato un numero di linea compreso tra 1 e 65534. Questo elenco è contenuto all'interno del principale file di configurazione del firewall: e può essere facilmente visualizzato con il comando:

```
ipfw show
```

Vediamo un esempio di quale potrebbe essere il risultato di questa operazione:

```
[…..]
21566     allow     TCP   from   any  to   me  8080      setup
21567     deny      TCP   from   any  to   any            setup
21568     allow     TCP   from   any  to   me             estabilished
[…..]
```

Come si può vedere ogni regola ha una struttura generale del tipo:

```
<numero.linea>  <azione>  <pattern> <flag>
```

In cui il primo valore è il numero di linea di cui abbiamo già parlato.
Il secondo parametro rappresenta l'azione che intende svolgere quella regola. Le possibili azioni sono:

- allow: permette al pacchetto di passare e smette di processare le regole successive;
- deny: blocca il pacchetto e lo scarta senza inviare alcun messaggio di notifica.Smette di processare le regole successive;
- unreach: blocca il pacchetto e lo scarta ma invia un messaggio ICMP di host irraggiungibile al mittente. Smette di processare le regole successive;
- reset: scarta il pacchetto e invia un messaggio TCP al mittente.Può essere applicato solo a pacchetti TCP. Smette di processare le regole successive;

Le azioni dette finora sono le più ricorrenti ma ne esistono anche altre:

- count: conta il pacchetto e procede nel processare la regola successiva;
- skipto *rule*: continua a processare le regole da quella con numero di riga rule.

Il terzo parametro "pattern" è a sua volta strutturato in questo modo:

```
<protocollo>  FROM <mittente> TO <destinatario>
```

dove il primo campo identifica, specificandone il protocollo, il tipo di pacchetto su cui compiere l'azione e i protocolli possibili sono:

IP TCP UDP ICMP

Di seguito devo specificare da dove questi pacchetti devono arrivare;per farlo uso la parola FROM seguita da un mittente che può essere:

- un indirizzo IP singolo o una classe;
- un indirizzo di rete;
- la parola " any " che identifica qualunque host;
- la parola " me " che identifica il mio host;
-

infine le stesse opzioni sono possibili per specificare, dopo la parola TO, il destinatario di questi pacchetti.
L'ultimo parametro che può essere presente solo se il protocollo è TCP è uno dei due possibili flag:

established: mi dice che la regola è valida solo se la connessione è già stata stabilita;
setup: mi dice che la regola è valida solo quando la connessione è in fase di creazione.

Ogni volta che voglio aggiungere una nuova regola all'elenco uso il comando

ipfw add <N° di linea > <action> <pattern> {<flag> }

Se non specifico il numero il numero di linea le viene direttamente assegnato quello della regola precedente più cento.

Nota)E' importante fare attenzione al numero scelto perché nel momento in cui il firewall è attivo e arriva un pacchetto dalla rete esterna, IPFW inizia a scorrere l'elenco delle regole partendo da quella con il numero di riga più basso e via via salendo. Appena incontra una direttiva che interessa quel determinato pacchetto, esegue l'azione impostata e smette di scorrere l'elenco, anche se più avanti ci sono altre regole che lo riguardano; passa poi a controllare altri pacchetti.
Perciò se,ad esempio, arrivasse un pacchetto IP per l'indirizzo 154.56.0.98 sulla porta 80 e il firewall fosse impostato in questo modo:

34552 deny IP from any to any
34555 allow IP from any to 154.56.0.98 80

anche se c'è una regola (34555) che prevede che quell'indirizzo può ricevere pacchetti IP, sarebbe scartato comunque perché IPFW leggere prima la regola 34552 che gli proibisce l'accesso.
Anche se alcuni azioni (come count) non bloccano IPFW nel suo scorrere l'elenco delle regole,questo è vero per tutte quelle azioni che decidono se un pacchetto deve passare o meno alla rete locale, per questo si è ritenuto valido fare questa nota.

Abbiamo preferito dare prima la visione di un firewall già configurato per capire come vengono tradotti i nostri vincoli di accesso per il programma.
Vediamo però ora quale sono le specifiche di base di un firewall sul quale non siamo ancora intervenuti.
Di default non viene permesso il passaggio di nessun tipo di traffico.
Infatti,prima del nostro intervento, l'elenco di regole del file di configurazione contiene solo quella corrispondente al numero di linea più alto, 65535, che impone:

65535 deny all from any to any

Prima dicevamo di poter associare ad ogni regola un numero da 1 a 65534;questa affermazione continua ad essere valida perché all'utente non è concesso intervenire su quest'ultima regola.Essa viene scritta in modo indelebile quando si costruisce il kernel del sistema.
Se però si costruisce il kernel con l'opzione IPFIREWALL_DEFAULT_TO_ACCEPT, questa regola cambia nel suo opposto:

65535 allow all form any to any

anche in questo caso non è modificabile né eliminabile.
Queste differenti scelte illustrano le due strategie di base possibili a seconda della politica di sicurezza che abbiamo scelto:

- la prima stabilisce che ' niente può passare a meno che non gli sia esplicitamente permesso ';
- la seconda invece stabilisce che ' tutto può passare a meno che non gli sia esplicitamente proibito ';

come è facile intuire, la prima scelta è quella più sicura perché, in caso di errori nella configurazione da parte dell'utente, si rischia di bloccare l'accesso a chiunque mentre nel secondo caso si permetterebbe l'accesso a chiunque.

16.3.1 Profili definiti

Nel file rc.firewall sono già disponibili tre profili tra cui scegliere quello che più si adatta alle nostre esigenze.
Ogni profilo contiene al suo interno un set di regole selezionate in base alla sua politica; il processo che avvia il firewall una volta scelto il profilo provvede a scrivere le sue regole nel file di configurazione di ipfw nell'ordine corretto.
I profili risparmiano ad un utente inesperto il compito di modificare manualmente le impostazioni di default di ipfw; lasciandogli la possibilità di intervenire in seguito per aggiungere o rimuovere dei vincoli.
Chi,invece, si appresta a configurare il proprio firewall con più esperienza, può subito scegliere singolarmente quali regole aggiungere all'impostazione base per ottenere i propri scopi.
I comandi usati per compiere questi interventi sono:

- ipfw add <action> <pattern> {<flag>}: per aggiungere una nuova regola;
- ipfw delete <n> : per cancellare la regola con numero di linea n;
- ipfw flush : per resettare l'intero elenco delle regole cancellando tutte le righe che vi erano scritte;

Nessuno di questi tre comandi può comunque intervenire sull'ultima regola,quella con numero di linea 65535.

RC.FIREWALL:

Il file rc.firewall parte abilitando tutto il traffico locale:

/sbin/ipfw add 1000 pass all from 127.0.01 to 127.0.0.1

Dopo questo punto inizia la descrizione dei tre profili disponibili:

aperto: è come se il firewall fosse disabilitato,tutti pacchetti possono passare indisturbati.

```
if [ " $ { firewall} " = " open " ]; then
    /sbin /ipfw add 65000 pass all from any to any
```

cliente: è un profilo che protegge il sistema contro persone che tentano di accedere dall'esterno della propria rete

```
if [ " $ { firewall} " = " client " ]; then

# scriviamo manualmente i parametri della nostra rete: ip di rete,
# netmask e indirizzo IP del nostro host: ad esempio:

    net = "233.147.37.0"
    mask = "255.255.255.0"
    ip = "233.247.37.1"

# permette il passaggio di tutto il traffico da o verso la mia rete locale:
    /sbin /ipfw add pass all from ${ip} to ${net}: ${mask}
    /sbin /ipfw add pass all from ${net}:${mask} to ${ip}
# ${net}: ${mask} esegue queste operazioni: and bit a bit tra l'IP
# destinatario e mask,confronto del risultato con net per verificare
# l'uguaglianza.
# quando una connessione TCP è stata stabilita,le è permesso continuare:
    /sbin /ipfw add pass TCP from any to any estabilished
```

```
# per stabilire una connessione TCP è invece necessario che siano
# rispettate altre regole:
        # permetto di stabilire connessioni TCP solo  in ingresso sulla
        porta 25, cioè autorizzo l'ingresso di email:
        /sbin /ipfw   add    pass  TCP   from  any   to  ${ip} 25 setup
        # permetto di stabilire connessioni TCP solo in uscita
        /sbin /ipfw   add    pass  TCP    from  ${ip}   to any  setup
        # disabilito la possibilità di stabilire altri tipi di connessioni TCP
        /sbin /ipfw   add    deny  TCP    from  any   to   any    setup

# le due regole che seguono servono per regolare le richieste DNS/NTP:
        # permette di inoltrare richieste DNS  verso il mondo esterno
        /sbin /ipfw   add    pass  UDP   from  any  53  to  ${ip}
        /sbin /ipfw   add    pass  UDP   from  ${ip}   to any   53
        # permette di inoltrare richieste NTP verso il mondo esterno:
        /sbin /ipfw   add    pass  UDP   from  any  123  to  ${ip}
        /sbin /ipfw   add    pass  UDP   from  ${ip}   to any   123
```

Per default è negato l'accesso e l'uscita di qualunque altro tipo di traffico.

semplice: è la configurazione per un semplice server che svolga funzione di gateway di una rete con IP privati, servizi DNS e NTP per il mondo esterno e protezione dallo spoofing

```
    if  [    " $ { firewall} "  =  " simple  "   ];   then

#un gateway ha due interfacce di rete: quella esterna rivolta alla rete #esterna e
quella interna #rivolta alla rete locale.
# stabiliamo questi parametri per l' interfaccia di rete esterna:IP
#della rete, netmask e indirizzo Ip dell'host:
            oif = "turn0"
            onet = "139.130.136.0"
            omask = "255.255.255.0"
            oip = "139.130.136.133"
```

```
# stabiliamo questi parametri per l'  interfaccia di rete interna::IP
# della rete, netmask e indirizzo Ip dell'host:

        inet = "233.147.37.0"
        imask = "255.255.255.0"
        iip = "233.147.37.0"

# impedisco lo spoofing.
        #Disabilito i pacchetti che arrivano all'interfaccia esterna con
        #mittente un utente della rete interna
        /sbin /ipfw  add  deny  all  from ${inet}: ${imask}  to  any  in via
        ${oif}
        #Disabilito i pacchetti che arrivano all'interfaccia interna con #mittente
        un utente della rete esterna
        /sbin /ipfw  add  deny  all  from ${onet}:${omask}  to  any  in via
        ${iif}

#Fermo gli indirizzi privati definiti dall' RFC1918 all'interno della rete
#locale
        /sbin /ipfw  add  deny  all  from  192.168.0.0:255.255.0.0 to  any  in
        via  ${oif}
        /sbin /ipfw  add  deny  all  from  172.16.0.0:255.240.0.0  to  any  in
        via ${oif}
        /sbin /ipfw  add  deny  all  from  10.0.0.0:255.0.0.0  to  any  in via
        ${oif}

#Permetto il  passaggio di traffico TCP una volta stabilita con successo la
#connessione:
        /sbin /ipfw  add  pass  TCP  from  any  to  any  estabilished

#Permetto ricezione di email, DNS e WWW.
        # permetto di stabilire connessioni TCP  per la ricezione di email:
        /sbin /ipfw  add  pass  TCP  from  any  to  ${oip}  25  setup
        # permetto  l'accesso al mio DNS server:
```

```
/sbin /ipfw  add  pass TCP  from any to ${oip} 53 setup
# permetto  l'accesso al mio servizio WWW:
/sbin /ipfw  add  pass TCP  from any to ${oip} 80 setup

#A questo punto rifiuto altri tipi di connessione da parte della rete esterna
#loggando i tentativi effettuati.
/sbin /ipfw  add  deny log TCP  from any to any via ${oif}
setup

#Permetto  a qualunque altro tipo di connessione TCP di essere stabilita
#(ormai le rete esterna è già stata bloccata ):
/sbin /ipfw  add  pass TCP  from any to any setup

#  altre due  regole che riguardano le richieste DNS e NTP:
# permette di inoltrare richieste DNS  verso il mondo esterno
/sbin /ipfw  add  pass UDP  from any 53 to ${oip}
/sbin /ipfw  add  pass UDP  from ${oip}  to any  53
# permette di inoltrare richieste NTP verso il mondo esterno:
/sbin /ipfw  add  pass UDP  from any 123 to ${oip}
/sbin /ipfw  add  pass UDP  from ${oip}  to any  123
```

Per default è negato l'accesso e l'uscita di qualunque altro tipo di traffico.

Una volta scelto il profilo che meglio si adatta alla nostra rete, lo dobbiamo annotare nel file di configurazione /etc/rc.conf e,sempre nello stesso file,dobbiamo scrivere che intendiamo abilitare il firewall.Se pensiamo che il profilo più adatto a noi sia ad esempio client dovremo scrivere:

```
firewall_enable = "YES"          // per l'abiltazione;
firewall_type = "client"          //  per il profilo
```

IPFW è soggetto alle limitazioni già citate di tutti i firewall a pacchetti; presenta quindi il significativo problema di riuscire a ottenere una configurazione corretta in base agli scopi che vogliamo raggiungere. Per chiarire questo limite pensiamo ad esempio all'incapacità dei firewall a pacchetti di poter intervenire solo ed esclusivamente sul 'protocollo HTTP'; al massimo potrebbero intercettare i pacchetti TCP in arrivo sulla porta 80 per impedire la connessione con un servizio http locale,oppure per impedire lo stesso tentativo ma in uscita. Intervenendo in questo modo arrivo al risultato voluto ma questo non vuol dire che si sia bloccato il protocollo http.

16.4 Configurazione di natd per l'IP Masquerading

Nel capitolo 2 abbiamo spiegato in cosa consiste la tecnica NAT per il mascheramento degli indirizzi privati.
Vediamo adesso i passi da compiere per predisporre il gateway di default,sul quale abbiamo montato il nostro firewall, alla conversione degli indirizzi:

1) Includiamo all'interno del nostro file rc.firewall la seguente opzione:

   options IPDIVERT

2) Assicuriamoci che la macchina su cui stiamo lavorando possieda le funzionalità di un gateway.
3) Assicuriamoci che l'interfaccia di rete sia attiva.
4) Definiamo nel file etc/service, la seguente entrata:

   natd 6668/divert # Network Address Translation socket

5) Aggiungiamo al file /etc/rc.conf queste righe:

   natd_enabled = "YES" # abilito natd
   natd_interface = "tun0" # specifico l'interfaccia pubblica o l'indirizzo IP usato.

ESERCITAZIONI

ESERCITAZIONE 1

Configurare il firewall in modo che blocchi tutto il traffico meno quello per il server Web.

Svolgimento

Per ottenere ciò che viene richiesto è necessario aggiungere le seguenti regole alla lista del file di configurazione del firewall (se sono già presenti altre regole è necessario anteporvi queste):

```
allow    TCP  from  any  to  me 8080   setup
        //in questo modo permetto che venga stabilita una connessione solo sulla porta
        //relativa al server Web 8080;
allow    TCP  from  any  to  me          estabilished
        // così permetto  che vengano fatti passare i pacchetti solo dove la connessione //è già
        stabilita;
deny     TCP  from  any  to  me   setup
         // impedisco che vengano stabilite connessioni su altre porte,cioè con processi  //
        diversi dal server Web .
```

ESERCITAZIONE 2

Configurare il firewall in modo che blocchi il traffico ICMP. Quando questa limitazione potrebbe essere utile?

Svolgimento

La regola da aggiungere per ottenere quanto richiesto è la seguente:

```
deny        ICMP  from  any  to  any
```

Questa limitazione potrebbe rivelarsi utile ad una macchina server per evitare attacchi del tipo denial-of-service,
cioè tutte le volte che utenti malintenzionati tentano di bloccare il servizio sovraccaricando il server della rete con richieste di tipo ICMP.
Se volessi notificare al mittente il mio rifiuto dovrei sostituire il tipo di azione compiuta dal firewall:

```
unreach     ICMP  from  any  to  any
```

ESERCITAZIONE 3

Configurare il firewall in modo che un determinato host della rete possa solo ricevere ma non inviare messaggi di posta elettronica

Svolgimento

Facciamo l'ipotesi che la politica del firewall che vogliamo configurare sia tre le due possibili quella meno restrittiva. Non c'è quindi bisogno di definire una nuova regola per permettere al calcolatore in questione di inviare messaggi di posta elettronica.
Dobbiamo aggiungere alla liste delle regole di IPFW, solo la regola che blocca i messaggi di posta elettronica in egresso verso quell' host.
Usiamo allora il comando:

/ sbin / add <n. linea > <action> <pattern> <flag>

Il numero di linea potrebbe essere alquanto significativo,in relazione alle regole che precederanno questa nell'elenco.
Ricordiamoci infatti che, quando riceve una richiesta, IPWF inizia a scorrere l'elenco delle regole partendo da quella con il numero di riga più basso e via via salendo.
Se incontra una direttiva che interessa quel determinato pacchetto, esegue l'azione imposta e smette di scorrere l'elenco, anche se più avanti ci sono altre regole che lo riguardano; passa poi a controllare altri pacchetti.
Per questo motivo se la nostra nuova regola fosse preceduta da una regola che permette il passaggio del tipo di pacchetto che vogliamo ostacolare, la modifica fatta risulterebbe inutile.
L'azione da compiere sarà quella di fermare (deny) tutti i messaggi di posta elettronica rivolti a quel determinato host:

/ sbin / add <n. linea > deny <pattern> <flag>

Per il pattern dobbiamo determinare:

<protocollo> from <mittente> to <destinatario>

- il protocollo: TCP
- il mittente: non è specificato un mittente particolare ma tutti i messaggi di posta elettronica qualunque sia il mittente devono essere bloccati; quindi useremo any;
- il destinatario: l'indirizzo IP e la porta del calcolatore in questione (supponiamo che sia 131.5.0.79). Nel caso di posta elettronica la porta utilizzata è la 25.

otterremo:

/ sbin / add <n. linea > deny TCP from any to 131.5.0.79:25 setup

ESERCIZI

Riportiamo adesso il testo di alcuni esercizi che lasciamo svolgere allo studente

Esercizio 1

Aggiungere al firewall una regola che impedisca l'attraversamento di tutti i pacchetti provenienti da host con indirizzo IP appartenente alle classi D ed E.

Esercizio 2

Configurare il firewall in modo che due determinati host della rete non possano scambiarsi messaggi di posta elettronica.

Esercizio 3

Impedire l'accesso al DNS server della mia rete.

CAPITOLO 17

MONITORAGGIO DELLE PRESTAZIONI

17.1 Introduzione

I vari tipi di firewall , i protocolli sicuri SHTTP e SSL , le tecniche di backup e di ripristino dei dati sono tutti strumenti dedicati alla protezione e sicurezza del sistema ; il loro impiego assicura al sistema una maggior robustezza nei confronti di attacchi di virus o di eventuali intrusioni ; tuttavia non possono garantire l'assoluta immunità da questo tipo di fattori indesiderati.
Avremo quindi bisogno di strumenti che ci permettano di rilevare la presenza di virus all'interno del nostro sistema , di scoprire intrusioni indesiderate e , ancor prima ,di identificare le cause di un malfunzionamento relativo alla rete.
Partiamo da quest'ultimo punto e supponiamo di dover affrontare il seguente problema : il nostro PC appartiene alla rete locale di un'azienda e può inoltre collegarsi ad internet attraverso il gateway di default dell'azienda stessa; dopo averlo avviato abbiamo lanciato il consueto programma di posta elettronica (vedi cap.1 :user agent) per scaricare la posta ; l'operazione non è andata a buon fine ed è comparsa sullo schermo una finestra che annunciava l'impossibilità di collegarsi all' host remoto (in questo caso il server di posta elettronica) .
Questo problema è rappresentativo di tutta una classe di problemi che possono verificarsi frequentemente e che sono legati o ad una mancata connessione tra client e server o più semplicemente a disturbi su tale connessione ; nell'analizzare alcuni dei mezzi e delle possibilità che abbiamo a disposizione cerchiamo quindi di proporre soluzioni per risolvere non solo lo specifico caso proposto ma tutta la classe di problematiche simili.
L'ordine con cui valuteremo le possibili cause di malfunzionamento prevede di partire dall'analisi del nostro sistema allontanandoci via via dalla rete locale verso la destinazione prevista sulla rete di reti .

17.2 IFCONFIG

Partiamo dal presupposto che il nostro cavo di rete sia integro e correttamente connesso da entrambe le parti ; la causa del malfunzionamento potrebbe allora essere la nostra interfaccia di rete . Lo strumento che ci viene in aiuto per verificare questa ipotesi è il programma IFCONFIG . Con ifconfig possiamo innanzi tutto configurare l'interfaccia di rete e poi monitorarne successivamente il funzionamento.
Per usare questo programma è sufficiente lanciare da shell il comando ifconfig (per il momento senza opzioni), sullo schermo appariranno delle informazioni simili a queste (vedi cap.2 per spiegazioni più approfondite) :

```
Eth0    link    encap : Ethernet      Hwaddr    00:A0:24 :77:49:97
inet    addr : 192.144.9.1    Bcast :192.168.1.255      Mask:255.255.255.0
UP    BROADCAST    RUNNING    MTU:1500      Metric:1
RX    packets:0    errors:0    dropped:0    overruns:0
TX    packets:42    errors:0    dropped:0    overruns:0
          Interrupt :8  Base address : 0xff80
```

Tenendo conto delle informazioni che ci interessa ricavare da questi dati , concentriamo la nostra attenzione sul primo parametro della terza riga : l'opzione UP indica che la scheda è attiva e funzionante.
Qualora invece di UP fosse presente DOWN ,possiamo provare a lanciare nuovamente il comando ifconfig seguito stavolta dall'opzione UP :

ifconfig < nome dell'interfaccia > - *up*

nel tentativo di attivare la scheda.
Se il comando fallisce abbiamo localizzato la causa del problema : il malfunzionamento della scheda di rete . Se invece la scheda è attiva e gli altri parametri di configurazione sono corretti , dobbiamo ricercare altrove le cause.

17.3 Controllo dell'interfaccia di rete tramite PING

Se l'interfaccia di rete funziona il problema potrebbe essere sulla nostra connessione alla rete locale . Il programma PING, che useremo più volte in questa trattazione, permette di inviare una richiesta di eco a un determinato indirizzo , utilizzando il protocollo ICMP. Si riesce a ottenere l'eco solo se l'instradamento verso quell'indirizzo è funzionante e, contemporaneamente, se è attivo quello di ritorno gestito a partire dall'indirizzo di destinazione.
In generale quindi se l' host A lancia il comando:

ping [opzioni] <indirizzo host B >

se l' host B è attivo e la connessione tra A e B è agibile ,il ping ritornerà correttamente facendo l'eco a video. Il comando ping una volta lanciato non invia una singola richiesta ma una serie di richieste distanziate l'una dall'altra da una pausa di tempo la cui durata può essere stabilita all'avvio del programma con l'opzione –i seguita dal numero di secondi , ad esempio con

ping -i 3 129.156.3.7

verrà inviata una richiesta ogni tre secondi . Di default questa opzione è impostata a 1.
L'output a video sarà ad esempio :

```
PING  129.156.3.7 (129.156.3.7 )  : 56  data  bytes
64   bytes   from   129.156.3.7: icmp_seq = 0   ttl = 64  time = 53 ms
64   bytes   from   129.156.3.7: icmp_seq = 1   ttl = 64  time = 40 ms
64   bytes   from   129.156.3.7: icmp_seq = 2   ttl = 64  time = 35 ms
64   bytes   from   129.156.3.7: icmp_seq = 3   ttl = 64  time = 40 ms
. . . . .

 [crtl +c]
```

--- 129.156.3.7 ping statistic ---
4 packets transmitted , 4 packets received , 0 % packet loss
round-trip (ms) min /avg /max = 35 / 42 / 53

Ogni volta che riceve una risposta , il programma specifica :

- il numero di bytes ricevuti ;
- l'indirizzo dal quale sono stati ricevuti ;
- il numero di sequenza di quel pacchetto;
- il ' *time to live* ' (ttl) del pacchetto cioè il numero di nodi che il pacchetto può attraversare prima di essere automaticamente scartato;
- infine il tempo di percorrenza ,cioè il tempo che passa dal momento in cui si inoltra il messaggio al momento in cui si riceve una risposta , detto anche '*round trip time* '.
-

Quando l'utente ferma il programma , ping fornisce un riassunto che specifica:

- il numero totale di pacchetti spediti e ricevuti ;
- la perdita percentuale di pacchetti ;
- il round trip time minimo ,medio e massimo.
-

Grazie a queste sue caratteristiche ping si rivela uno strumento molto utile in fase diagnostica. Sebbene i tempi di percorrenza forniscano scarse informazioni all'utente, in quanto non indicano il motivo per cui il tempo necessario a raggiungere una postazione 'vicina' sia più lungo del tempo impiegato per raggiungere località distanti , e sebbene ping non sia in grado di determinare il motivo di una mancata risposta ma lasci aperte diverse possibilità che vedremo più avanti , questo programma si rivela essere uno degli strumenti usati più frequentemente dagli amministratori di rete. Questi sono soliti ricorrere a ping appena hanno notizia di un guasto per capire quali parti della rete sono ancora in funzione e quali no, individuando rapidamente i malfunzionamenti .

Torniamo alla nostra analisi: a noi interessa ottenere una singola risposta al fine di verificare l'effettiva integrità della connessione tra i due host . Possiamo usare il seguente comando , evitando di concludere forzatamente l'esecuzione del programma (ctrl +c) :

ping -c 1 129.156.3.7

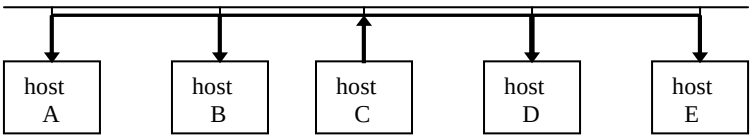
il comando invia una richiesta di eco all'indirizzo specificato e termina di funzionare quando riceve la prima risposta di eco. L'opzione -c < quantità > serve infatti a concludere il funzionamento di ping dopo aver ricevuto il numero indicato di risposte . Esistono molte altre possibili opzioni per questo comando ma non è nostro interesse elencarle ora .
Adesso che abbiamo compreso almeno superficialmente le potenzialità di questo programma , vediamo come sfruttarle ai fini della nostra analisi .
L'intento è quello di verificare che la nostra connessione alla rete sia integra e funzionante : possiamo allora usare il comando ping specificando come indirizzo destinatario l'indirizzo IP della nostra stessa interfaccia di rete :

ping -c 1 192. 168.1.1

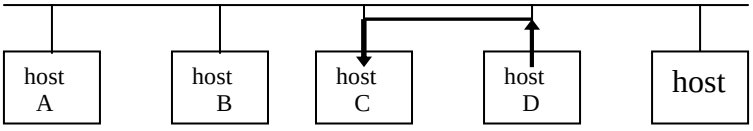
in questo modo la scheda di rete si attiva per inviare il messaggio , il messaggio viene effettivamente immesso sulla rete e ricatturato dal nostro host.
Se otteniamo l'eco della risposta abbiamo provato che sulla nostra connessione non sussistono problemi.
Notiamo in particolare che , nel lanciare il comando ping , abbiamo specificato l'effettivo indirizzo IP della macchina e non l'indirizzo di loopback . Usando quest'ultimo non saremmo scesi fino al livello fisico (connessione fisica di rete) fermandoci al livello IP.

17.3.1 ARP

La famiglia TCP/IP contiene al suo interno un protocollo per la risoluzione degli indirizzi che garantisce che non vi siano incongruenze tra i formati dei messaggi di richiesta di risoluzione: tale protocollo prende appunto il nome di ARP (Address Resolution Protocol) .
ARP prevede l' impiego di due tipi fondamentali di messaggio : richiesta e risposta .
La richiesta contiene un indirizzo IP e richiede il corrispondente indirizzo fisico , mentre il messaggio di risposta contiene sia l'indirizzo IP ricevuto sia la sua traduzione .
Le specifiche di ARP stabiliscono inoltre l'esatta modalità di trasmissione dei messaggi ARP.
Un messaggio di richiesta , dopo essere stato correttamente inserito in un frame conforme alla specifica rete fisica , viene inviato in modalità broadcast ; in questo modo tutte le stazioni ricevono il messaggio ed esaminano l'indirizzo IP in esso contenuto. Il calcolatore che si riconosce nell'indirizzo specificato risponde alla richiesta mentre tutti gli altri la ignorano senza inviare risposta .
Notiamo che a differenza della richiesta la trasmissione di un messaggio ARP di risposta non avviene in modalità broadcast ; la risposta , anch'essa correttamente impacchettata in un frame, viene spedita direttamente al calcolatore che aveva trasmesso la richiesta , senza pertanto raggiungere tutti le altre postazioni .
Quello che abbiamo detto può essere riassunto nello schema sottostante :



L' host C trasmette in broadcast il suo messaggio di richiesta che raggiunge così tutte le postazioni ; supponiamo che il messaggio contenga l'indirizzo IP dell' host D.



L' host D riconosce nella richiesta il proprio indirizzo IP e risponde a C inviandogli un messaggio di risposta contenente il suo indirizzo fisico.
Dedichiamo adesso poche righe all'analisi del formato dei messaggi ARP al fine di comprendere meglio il funzionamento di questo protocollo .
Il protocollo per la risoluzione degli indirizzi stabilisce solo la struttura generale dei messaggi; fornisce poi una serie di istruzioni che permettono di determinare in funzione della rete fisica sottostante , quale deve essere, nel dettaglio, il formato del messaggio. In questo modo i messaggi ARP possono contenere indirizzi fisici il cui formato varia da caso a caso .

Per questo motivo all'inizio del messaggio esiste un campo di lunghezza fissa che specifica la dimensione dell'indirizzo fisico contenuto nel messaggio.
Ogni messaggio ARP sia esso di richiesta o di risposta sarà inoltre composto da un campo che contiene l'indirizzo fisico del mittente , un campo con l'indirizzo fisico dell'obiettivo e un campo con l'indirizzo IP dell'obiettivo . I campi ignoti al mittente saranno impostati a zero nel messaggio di richiesta .
Il messaggio completo è formato da 7 righe di 32 bit ciascuna ; non è per noi di particolare interesse in questa trattazione analizzare gli altri campi che lo compongono.
E' facile intuire come la trasmissione di una richiesta per ogni singola traduzione possa facilmente generare una notevole quantità di traffico sulla rete ,rivelandosi per questo una soluzione assolutamente inefficiente .
Nel tentativo di alleggerire il traffico generato da questi continui scambi , ARP registra le informazioni ottenute dalle risposte precedenti per riutilizzarle nelle trasmissioni successive .
Il protocollo mantiene in memoria una piccola tabella di indirizzi risolti che viene gestita come una memoria cache : appena arriva una nuova soluzione viene rimpiazzato l'elemento più vecchio o non usato da più di un certo tempo . In questo modo le risoluzioni degli indirizzi non sono memorizzate permanentemente , ma sfruttando solo una piccola parte di memoria si migliorano concretamente le prestazioni del protocollo e le condizioni di traffico sulla rete.
E' proprio questa sorta di memoria cache che rappresenta per noi uno strumento utile al fine di riconoscere altre possibili cause di malfunzionamento.
Avevamo finito di verificare grazie al programma ping che la nostra connessione alla rete non presentava problemi , controlliamo adesso la memoria cache del protocollo ARP con il comando dato da shell

```
arp -a
```

che ci permette di consultare la tabella delle risoluzioni.
Nello scorrere le coppie indirizzo IP/ indirizzo fisico può capitare di accorgersi che due host con indirizzo fisico diverso abbiano lo stesso indirizzo IP. Una situazione di questo tipo potrebbe essersi verificata per due motivi : nel primo caso due host hanno fatto il boot contemporaneamente ed è stato assegnato loro per errore lo stesso IP.
La soluzione in questo caso è semplice: è sufficiente spengere il calcolatore e riavviarlo, la probabilità che si verifichi di nuovo lo stesso inconveniente è estremamente ridotta.
Nel secondo caso l'uguaglianza degli indirizzi IP potrebbe non essere un'involontaria coincidenza ma un premeditato attacco di spoofing , cioè un utente malintenzionato sta usando di proposito il nostro indirizzo IP ,ad esempio per accedere ad una parte riservata della rete locale.
ARP rappresenta quindi uno strumento utile non solo per scoprire se c'è stato un errore nell'assegnamento degli indirizzi IP , ma soprattutto per rilevare eventuali intrusioni sfuggite agli altri sistemi di sicurezza .
Poniamoci , ad esempio , nel caso di aver protetto precedentemente la parte riservata della rete locale dell'azienda con un firewall che vietava esplicitamente l'accesso a tutti gli host eccetto quelli con un determinato indirizzo IP, l'intruso avrebbe potuto tranquillamente violare la nostra barriera dal momento che utilizzava uno degli indirizzi IP privilegiati.
Infine , grazie al comando arpwatch , la tabella delle corrispondenze può essere aggiornata dinamicamente ed è possibile inviare messaggi informativi in merito ad ARP all'utente root .
Il database delle mappature degli indirizzi , costruito mimicamente, viene memorizzato (per Linux Red Hat) nel file arp.dat all'indirizzo /var/arpwatch.

Ogni riga del file contiene una corrispondenza tra indirizzo fisico è indirizzo IP. Le righe presenti in questo file contengono quattro campi :

< indirizzo fisico > < indirizzo IP > < time stamp > < hostname >

Il terzo parametro è un numero del quale non analizziamo il significato per non scendere in un livello di dettaglio troppo approfondito per gli scopi di queste dispense .
Per scoprire quando vengono aggiunte nuove corrispondenze "indirizzo fisico – indirizzo IP " o per controllare che due indirizzi fisici non stiano utilizzando lo stesso indirizzo IP non abbiamo bisogno di aprire direttamente il file arp.dat perché , come abbiamo accennato, arpwacth monitora l'attività di ARP e trasmette automaticamente via email i suoi messaggi all'utente root quando accade qualcosa di significativo .
Arpwacth può inviare 5 tipi di messaggi all'amministratore:

- Changed ethernet address : l'indirizzo IP sta utilizzando un indirizzo fisico diverso da quello precedentemente memorizzato nel database;
- Flip Flop : l'indirizzo IP è stato riconvertito nell'indirizzo fisico che usava precedentemente;
- New Activity : una vecchia coppia indirizzo fisico-indirizzo IP che non era stata utilizzata da molto tempo (più di sei mesi) è tornata ad essere utilizzata;
- New Station : è appena stata individuata una nuova coppia indirizzo fisico-indirizzo IP;
- Resued old ethernet address : un indirizzo IP è associato ad un indirizzo fisico (ethernet) che è vecchio di almeno tre generazioni ; cioè il file arp.dat ha almeno tre indirizzi fisici mappati sullo stesso indirizzo IP.

17.4 Controllo del gateway di default tramite PING

Adesso che abbiamo verificato che il problema non risiede né sulla nostra postazione né sul nostro collegamento , seguiamo il percorso fatto verso l'esterno dalla nostra richiesta di connessione . Dal nostro calcolatore la richiesta viene immessa sulla rete locale e da questa arriva al gateway di default che si occuperà di inoltrarla in Internet.
L'anello debole della catena potrebbe essere l'arresto (o l'inattività) del gateway o il collegamento con il gateway stesso.
Di nuovo ci viene in aiuto il programma PING che offre la possibilità di verificare contemporaneamente entrambe le circostanze . Se lanciando il comando :

ping - c 1 < indirizzo del gateway di default >

otteniamo la risposta prevista , la connessione non presenta ostacoli e il gateway di default è attivo .
Questo non è tuttavia sufficiente a garantire che il problema non risieda sul gateway . Infatti quando ,nel comando ping , indichiamo l'indirizzo del gateway di default ci stiamo riferendo all'indirizzo fisico della sua interfaccia di rete interna . Il gateway possiede due interfacce di rete una verso l'interno con la quale dialoga con le stazioni della rete locale , una rivolta verso l'esterno per il collegamento a Internet .
La verifica fatta non è in grado di fornire alcun tipo di informazione né sullo stato dell'interfaccia esterna né sulla connessione verso Internet del gateway di default .

E’ infine necessario fare una nota sull’affermazione implicita che è stata fatta riguardo lo stato del gateway se il comando ping fallisce. Qualora infatti il comando che abbiamo lanciato non producesse alcuna risposta (e fossimo certi dell’integrità della connessione tra i due punti) non potremmo affermare con certezza che il gateway ha dei problemi.
Come già accennato in precedenza , nel caso in cui non si riceva risposta , ping non è in grado di determinarne il motivo : la macchina remota potrebbe essere spenta o scollegata dalla rete , la sua interfaccia di rete potrebbe essere guasta e anche qualora la macchina fosse perfettamente in grado di svolgere i suoi compiti , potrebbe essere controllata da software che non risponde a messaggi di ping .
Di fatto accade spesso che l’amministratore della rete disabiliti il gateway a ricevere messaggi ICMP per evitare attacchi del tipo ‘ denial of service ’(vedi cap.3 / esercitazione 2) . Poiché ping utilizza messaggi ICMP , il gateway potrebbe essere perfettamente funzionante ed evitare comunque di rispondere alla richiesta viste le sue specifiche.

17.5 TRACEROUTE

Potremo procedere a questo punto utilizzando nuovamente il comando ping verso una postazione che si trovi all’esterno della rete locale e sia sul percorso per raggiungere il server di posta elettronica che stiamo cercando di contattare .
Ma come facciamo a sapere quali nodi attraverserà la mia richiesta per giungere a destinazione ? E come posso scoprire quali sono gli indirizzi IP di queste postazioni ?
Ci accorgiamo subito arrivati a questo punto che il programma ping non è più sufficiente a portare avanti la nostra analisi ; infatti l’unico indirizzo certo che conosciamo al di fuori della rete è quello del destinatario , ma abbiamo già testato l’impossibilità di connettersi con questo host e sarebbe quindi scontato il fallimento del comando ping .
Peraltro questo fallimento non aggiungerebbe nessuna informazione utile alle verifiche che già abbiamo portato .
Nota) Quest’ultima affermazione serve ad analizzare lo sfortunato caso in cui non conosciamo l’indirizzo dell’interfaccia del gateway verso l’esterno , altrimenti potrebbe essere utile accertarsi che anch’essa sia attiva e funzioni correttamente , semplicemente usando ping come abbiamo visto finora .
Uno strumento che gli amministratori di rete usano spesso di fronte a questo tipo di problema è il programma TRACEROUTE , che permette di verificare il percorso verso una destinazione , dando delle indicazioni che possono aiutare a comprendere in quale punto ci siano delle difficoltà .
Al pari di ping , traceroute richiede come argomento il nome o l’indirizzo di una macchina remota :

```
traceroute      [opzioni]    < destinazione >
```

Lanciando ad esempio il comando :

```
traceroute      serverdipostaelettronica.plip.dg
```

traceroute inizia la trasmissione di pacchetti (utilizzando il protocollo UDP) con un valore TTL molto basso e attende di ricevere un messaggio di errore (protocollo ICMP) dal nodo in cui il valore di TTL raggiunge lo zero.
Incrementando lentamente il valore di TTL ,traceroute riesce a conoscere gli indirizzi dei nodi attraversati , purché tutto funzioni come previsto (cioè che i vari nodi generino correttamente i pacchetti ICMP di errore).
Dunque un possibile output (se la connessione con il server funzionasse a dovere) sarebbe:

traceroute to serverdipostaelettronica.plip.dg (181.135.254.1), 30 hops max , 40 byte packets
1 dinkel.brot.dg (182.135.1.1) 0.433 ms 0.278 ms 0.216 ms
2 router.brot.dg (182.135.1.254) 0.433 ms 0.278 ms 0.216 ms
3 ihtpf.mit.dg (182.135.28.1) 0.433 ms 0.278 ms 0.216 ms
4 * * *
5 * * *
6 * * *
7 serverdipostaelettronica.plip.dg (181.135.254.1) 10.654 ms 13,341 ms 11.111 ms

Traceroute individua le macchine che rappresentano le tappe intermedie lungo il cammino che porta alla destinazione desiderata , e stampa una riga per ciascuna di esse.
Nel nostro caso abbiamo 7 righe di output : una per ciascuna dei 6 nodi attraversati lungo il cammino, più una che rappresenta la destinazione : si direbbe in gergo che la destinazione è a 7 salti (hop).
Se in alcuni salti non si ottiene risposta , i nodi ipotizzati vengono segnalati con asterischi.Su tali nodi non si può dire nulla di certo , ma solo fare delle congetture .
Tra le varie opzioni del comando traceroute ,sicuramente quella per noi più interessante è :

traceroute -m *TTL_massimo*

che stabilisce il numero massimo di nodi da attraversare , definendo il valore massimo che TTL può raggiungere.
Il programma inizierà normalmente impostando a 1 il valore di TTL e incrementa gradualmente tale valore (per ogni messaggio UDP spedito) , fino a quanto specificato dall'opzione –m .
Di default il TTL massimo è impostato a 30 salti.
Se da questa breve spiegazione abbiamo capito le linee generali del funzionamento di traceroute , passiamo a illustrare come utilizzare questo strumento per i nostri scopi.
Il primo passo da compiere è verificare che l'interfaccia esterna del gateway di default sia attiva . Per farlo è sufficiente lanciare il comando :

traceroute - m 0 serverdipostaelettronica.plip.dg

se l'interfaccia esterna del gateway è attiva , traceroute riceve un messaggio ICMP di risposta che notifica lo scarto del pacchetto e stampa una riga relativa al primo salto effettuato.
Incremento il valore di TTL e lo porto ad 1 :

traceroute - m 1 serverdipostaelettronica.plip.dg

verifico se funziona il router successivo e implicitamente verifico anche che la connessione tra gateway di default e il successivo hop è attiva.
Procedo incrementando TTL finché o raggiungo la destinazione specificata , e in questo caso avrei accertato che il collegamento fisico tra il mio calcolatore di partenza e quello di arrivo (serverdipostaelettronica.plip.dg) è perfettamente funzionante , il destinatario è attivo e la sua scheda di rete funzionante, oppure, dopo un certo hop, traceroute non riuscirà più ad andare avanti perché il collegamento fisico è interrotto.
Nel primo caso, devo ricercare , per il malfunzionamento, altre cause che non siano imputabili alla connessione fisica ; nel secondo caso ho individuato in quale punto della rete si trova il guasto (a patto che ne conosca la topologia).
In realtà l'uso di traceroute non è così semplice né è tanto immediata l'interpretazione dei risultati ottenuti dalla sua esecuzione.
Traceroute deve tener conto di molti dettagli , tra cui il problema del cambiamento dinamico dei percorsi. Non vi è alcuna garanzia che due datagram spediti dallo stesso mittente raggiungano il destinatario seguendo lo stesso percorso . Ne segue che traceroute potrebbe indicare una lista di router che corrispondono a tappe di diversi percorsi verso la stessa destinazione.
Per questo motivo , l'utilità di traceroute è limitata soprattutto a inter-reti relativamente stabili.

17.6 NETSTAT

Se la connessione fisica non presenta alcun tipo di problema , non rimane che controllare la connessione logica .
Come abbiamo illustrato nel capitolo 7 quando parlavamo della connessione tra client e server, la connessione logica utilizza i socket , oggetti creati da system call del sistema operativo che permettono ad un processo di inviare e ricevere dati da un altro processo su una macchina remota .
NETSTAT è un programma in grado di mostrare in modo agevole alcune delle informazioni contenute nella directory ' /proc /net ' (per GNU/Linux).
Le informazioni disponibili sono davvero molte ; in questa trattazione mostreremo solo quelle che sono utili ai nostri scopi.
Il comando :

```
netstat [opzioni]
```

emette attraverso lo standard output una serie di notizie che si riferiscono a tutti i tipi di connessioni disponibili, traendo le informazioni dai file della directory '/proc /net'.
Usato senza opzioni , netstat mostra la situazione di tutti i tipi di connessione attive , elencando i socket aperti.
Se tra le opzioni appare l'indicazione di uno o più protocolli , le informazioni ottenute si limitano alle connessioni di quel tipo di protocolli. Ad esempio:

```
netstat -t | --tcp (1)
```

mostra solo lo stato delle connessioni TCP.
Mentre :

netstat -u | --udp : mostra lo stato delle connessioni UDP ;
netstat --inet | --ip : mostra lo stato delle connessioni TCP/IP ;

altre opzioni utili potrebbero essere :

netstat -e : aggiunge l'indicazione dell'utente proprietario del processo che attua la connessione;
netstat -a : elenca la situazione di tutte le porte , incluse quelle dei server in ascolto ;

Un possibile output alla richiesta (1) potrebbe essere :

Active Internet connections (w/o servers)

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	dikken .mit.ye:1403	sunny.mit.ye :telnet	ESTABILISHED
tcp	0	0	sunny.mit.yet :telnet	dinkken.mit.yet:1403	ESTABILISHED

L'elenco che netstat restituisce ha la forma di una tabella . Descriviamo cosa rappresenta e il significato di ogni colonna partendo da sinistra verso destra:

- **Proto :**
rappresenta il protocollo usata in ogni porta attiva.Le possibilità sono:TCP,UDP , RAW;

- **Recv-Q , Send-Q :**
rappresenta la coda di byte che sono stati ricevuti ma non ancora prelevati dal programma che utilizza la connessione , o che sono stati trasmessi ma per i quali non è ancora stata ricevuta la conferma (ack) dall' host remoto.

- **Local Address , Foreign Address :**
rappresentano rispettivamente l'indirizzo locale e quello remoto , completi della porta relativa;

- **State :**
rappresenta lo stato dei socket TCP ,indicato attraverso una delle parole chiave che sono elencate di seguito . Questa informazione è quella che interessa i nostri scopi in quanto ci fa capire le condizioni di ognuno dei socket aperti .

_ *ESTABILISHED* :
la porta ha una connessione in corso ;
_ *TIME WAIT*:
la porta è in attesa della conferma dall'altra parte ;
_ *CLOSED* :
la porta non è in uso ;
_ *LISTEN* :
la porta è in ascolto di connessioni in arrivo ;
_ *SYN SENT* :
la porta sta tentando di instaurare una connessione ;

_ *CLOSE WAIT* :

la porta remota conclude la connessione ed è in attesa di conferma

dall'altra parte;

_ *UNKNOW* :

lo stato della porta è sconosciuto;

ed altre ancora che non riportiamo per non appesantire troppo la trattazione.

- **User :**
  specifica il nome o il numero UID dell'utente proprietario del processo che sta girando su quella porta . Si ottiene questa informazione con l'opzione –e.

17.7 TCPDUMP

L'ultimo strumento che esamineremo per il monitoraggio della rete è TCPDUMP.
Configurando la scheda di rete in modalità promiscua , tcpdump ci offre una visione sintetica dei tutti i pacchetti che attraversano la rete fisica . Il comando da utilizzare è :

tcpdump [opzioni] [espressione]

I pacchetti vengono analizzati solo nella prima parte , solitamente i primi 68 byte,senza tenere memoria delle informazioni successive. L'opzione :

tcpdump - s < n_byte>

mi permette di modificare questa dimensione , definendo esplicitamente la quantità di byte da prendere in considerazione . In generale è però sconsigliabile aumentare questo numero dal momento che richiederebbe un tempo di elaborazione più lungo che potrebbe anche portare alla perdita di pacchetti.
Il risultato generato può essere in esadecimale oppure può riportare i pacchetti così come sono, per questo motivo è necessario conoscere la struttura dei pacchetti in base al protocollo relativo per interpretare efficacemente il risultato. A tal fine è possibile definire un programmino filtro che faccia da interprete .
Alcune opzioni interessanti , oltre quella già vista , sono :

- -i <interfaccia>
  definisce l'interfaccia di rete attraverso cui tcpdump deve porsi in ascolto del traffico sulla rete . Se non viene specificato nulla , tcpdump sceglie la prima;
- -w <file>
  memorizza i pacchetti grezzi all'interno del file specificato ,invece di analizzarli ed emettere il risultato . Il contenuto di questo file può essere esaminato in seguito con l'opzione –r ;
- -r <file>
  legge i pacchetti da quanto accumulato precedentemente nel file dichiarato ; ne permette in sostanza l'analisi ;
- -F <file>
  permette di fornire l'espressione del filtro dei pacchetti attraverso il file indicato.

Ecco due esempi :

```
tcpdump      -i   eth0
tcpdump      -i   eth0     - F    /home/mitt/filtro
```

con il primo commando il programma emette sull'output tutto il traffico che intercetta attraverso l'interfaccia eth0 , con il secondo lo stesso output è filtrato con il programma contenuto nel file specificato.

17.8 Conclusioni

Di tutti gli strumenti che abbiamo analizzato ARP e TCPDUMP possono rivelarsi utili non solo per identificare malfunzionamenti ,ma soprattutto per scoprire la presenza di intrusi indesiderati all'interno della rete.

Appendice A

Le distribuzioni Linux

Linux è disponibile gratuitamente sulla rete, ma molte società assemblano le cosiddette *distribuzioni*, che sono delle versioni di Linux pronte per l'uso. Queste comprendono un ambiente Linux completo, un programma di installazione guidata, qualche programma aggiuntivo e/o commerciale ed il supporto per l'assistenza tecnica. In generale le differenze tra una distribuzione Linux rispetto ad un'altra sono da individuare nei tool di installazione/configurazione e nel numero e tipo di pacchetti software che accompagnano il sistema base. Infatti il kernel da solo non basta: occorrono almeno un compilatore ed una serie di applicativi per poter sfruttare il sistema operativo Linux. In questo senso chiunque si occupi di rendere disponibile una distribuzione, cerca nel modo che ritiene migliore di rendere Linux più "amichevole", aggiungendo tool e utility di vario genere per raggiungere lo scopo.
Non esiste una sola distribuzione di Linux ma diverse distribuzioni che sono disponibili su siti FTP o in vendita su CD-ROM.
La differenza fondamentale tra le distribuzioni scaricabili da siti ftp o distribuite con le riviste e le distribuzioni commerciali ufficiali riguarda, a vantaggio di queste ultime, la presenza di eventuale software commerciale, il supporto tecnico all'installazione, la presenza di manuali cartacei spesso molto corposi. Tale differenza naturalmente si paga.
Non è possibile stabilire razionalmente quale sia la distribuzione migliore. Dipende molto dall'esperienza, il modo migliore sarebbe quello di provarne più di una e poi scegliere quella che meglio si adatta alle proprie esigenze. Alcune distribuzioni inoltre sono ottimizzate per usi diversi. Per esempio la scelta può dipendere dal tipo di utente (principiante proveniente dal mondo DOS o esperto) o dall'uso che si farà del sistema su cui andrà installata (computer di casa, server Internet, ecc.).

Per quanto riguarda la scelta della distribuzione è utile citare il documento che raccoglie le FAQ in italiano su Linux curato da Marco Iannacone:

- se tutti coloro che conoscete utilizzano una particolare distribuzione e voi siete alle prime armi, scegliete quella stessa stessa distribuzione.
- se vi piace fare le cose direttamente e cioè compilare ed installare tutto voi stessi, probabilmente Slackware è ciò che fa per voi.
- se volete fare quello che fa la maggioranza, scegliete RedHat.
- se volete *tutto*, installate Suse.
- se volete la distribuzione più commerciale, usate Caldera.
- se per voi la filosofia del free software è la cosa più importante, o volete essere coinvolti nello sviluppo di una distribuzione scegliete Debian.

Di seguito è riportata una lista delle distribuzioni principali:

A.1 Red Hat

Alcune caratteristiche:

- è adatta agli utenti con poca conoscenza dei sistemi UNIX -- eventualmente può essere scelta una modalità di installazione semiautomatica, in cui l'utente è sollevato dall'onere di scegliere i pacchetti applicativi;
- gli archivi dei pacchetti che compongono la distribuzione sono in formato RPM (*Red Hat package manager*) e la loro gestione è relativamente semplice;
- al termine dell'installazione, il sistema ha già una buona configurazione di partenza, completa di piccoli accorgimenti per gli utenti meno esperti.

Più sinteticamente:

```
Sicurezza:  **
Stabilità:  ***
Semplicità: ****
```

A.2 Mandrake

E’ derivata dalla Red Hat e quindi è consigliata ad utenti principianti. La novità principale è comunque la possibilità di farsi le partizioni utilizzando solo un tool con interfaccia grafica.

Più sinteticamente:

```
Sicurezza:  ****
Stabilità:  ****
Semplicità: ****
```

A.3 Slackware

Si tratta della prima distribuzione Linux relativamente “facile” da installare e in ciò ha il merito di avere contribuito alla sua diffusione nei primi anni di vita di questo sistema operativo. Alcune caratteristiche:

- è una distribuzione adatta agli utenti che hanno una buona conoscenza dei sistemi UNIX, tuttavia, l'installazione è un po' complicata e tende a scoraggiare l'utente inesperto;

Più sinteticamente:

```
Sicurezza:  ****+
Stabilità:  ****
Semplicità: **
```

A.4 SuSE

La distribuzione SuSE è nata come una variante tedesca della distribuzione Slackware, tanto che oggi ci sono ancora alcune affinità con quella distribuzione, anche se utilizza attualmente il sitema RPM per la gestione dei pacchetti software. Alcune caratteristiche:

- è adatta agli utenti con poca conoscenza dei sistemi UNIX;
- il programma di installazione e di configurazione è molto raffinato;
- gli archivi dei pacchetti che compongono la distribuzione sono in formato RPM;
- al termine dell'installazione, il sistema ha già una buona configurazione di partenza;

Più sinteticamente:

```
Sicurezza:  ****
Stabilità:  ****
Semplicità: ***
```

A.5 Debian

Questa distribuzione è realizzata da un gran numero di volontari e tutto è organizzato per facilitare la coordinazione di queste persone. Un aspetto molto importante della politica della distribuzione è l'attenzione alle licenze e ad altre restrizioni legali. Alcune caratteristiche:

- è adatta agli utenti che hanno una buona conoscenza dei sistemi UNIX, mentre il principiante può avere difficoltà a installarla;
- gli archivi dei pacchetti che compongono la distribuzione sono in formato Debian (`.deb`) e il sistema di gestione relativo è molto efficace;
- data la complessità del sistema di gestione dei pacchetti Debian, il programma che guida nell'installazione dei pacchetti è altrettanto complicato da utilizzare.

Più sinteticamente:

```
Sicurezza:  ****+
Stabilità:  ****
Semplicità: **
```

A.6 Caldera

La distribuzione Caldera è una vera workstation UNIX, stabile e paragonabile a prodotti enormemente più costosi. L'installazione è rapida e guidata da menu, con auto-riconoscimento delle periferiche. In pratica è lo standard utilizzato nelle applicazioni di e-commerce.

Appendice B

Installare RedHat 7.2

La procedura che verrà descritta nelle prossime pagine non è affatto l'unica: infatti si può installare Linux in molti modi. Comunque si è cercato di scegliere il modo più universale e generico possibile.
Prima di iniziare l'installazione, creare un dischetto di ripristino di Windows e controllare che contenga il programma FDISK. Questo perché l'installazione di Linux sovrascrive l'MBR (*Master Boot Record*). Se la sovrascrittura non funziona correttamente, non si riuscirà più ad riavviare Windows. Invece attraverso il dischetto di ripristino, si può rimettere a posto l'MBR dal dischetto, usando il programma FDISK con il comando `fdisk /mbr`.

B.1 Fare spazio per Linux

Ci sono due metodi di base per installare Linux su un computer sul quale già risiede Windows:

- installarlo all'interno di un enorme file compatibile con Windows (si tratta di un file in formato *FAT32*). Questo rallenta leggermente le prestazioni di Linux, ma rende l'installazione molto semplice;
- installarlo in una partizione apposita, ricavata ridimensionando quella di Windows. Tale operazione restituisce un Linux vero, indipendente e incontaminato da Windows. Solo questo tipo di installazione consente di usufruire delle vere prestazioni di Linux sia in fatto di velocità, sia in fatto di sicurezza.

Si prenderà in considerazione il secondo punto in quanto il primo è poco utilizzato e non consente di avere tutte le caratteristiche fornite da Linux. Per fare ciò dobbiamo creare spazio per il nuovo sistema sul disco rigido accanto a Windows.
Purtroppo fare spazio per Linux non è soltanto una questione di avere spazio libero su disco sotto Windows. Linux usa un sistema di gestione del disco (un *filesystem*) completamente diverso da quello di Windows. Quindi, far “convivere” Linux e Windows su un unico disco rigido significa che una parte del disco deve essere formattata nel formato di Windows e un'altra parte deve essere formattata nel formato di Linux: bisogna effettuare un *partizionamento* del disco rigido.
Prima di cominciare si consiglia vivamente di fare una copia di sicurezza di tutti i dati e programmi prima di cominciare.
Nel descrivere la procedura di partizionamento, si presume che la macchina su cui si sta per installare Linux abbia un solo disco rigido, sul quale è già installato Windows e sul quale si vuole installare anche Linux.

B.1.1 Pulizie preliminari

Linux richiede che ci siano circa 1,5 gigabyte di spazio libero *contiguo,* dopo l'ultimo file di Windows, sul disco sul quale si vuole installarlo (la quantità di spazio libero necessaria si può ridurre a circa 600 megabyte settando in modo appropriato la configurazione del sistema). Questo spazio contiguo viene sottratto a Windows e, una volta formattato in formato Linux attraverso il partizionamento, diventerà la partizione di residenza di Linux.

Da notare che si è sottolineato il termine *spazio contiguo*. Infatti Windows normalmente non scrive sempre i file uno dopo l'altro sul disco rigido: li mette in modo disordinato. In particolare più si usa il disco rigido, più aumenta questa frammentazione di file.
A questo problema si rimedia con un procedimento chiamato *deframmentazione*, che riordina i file presenti sul disco e li accumula, senza lasciare buchi, nella zona iniziale del disco. Sotto Windows, il programma di deframmentazione si chiama *Defrag*. Dopo la deframmentazione lo spazio contiguo situato dopo l'ultimo file di Windows diventa utilizzabile per un partizionamento non distruttivo. Naturalmente sarà meglio non prendere tutto lo spazio libero, ma lasciarne un po' come margine per la crescita di Windows (richiesta ad esempio dall'aggiunta di nuovi programmi).
Dopo il partizionamento, eseguito con *fips* fornito con Linux, il disco rigido è diviso in due parti. Ne consegue che Windows e Linux possono scrivere soltanto nelle proprie partizioni. Il pregio di fips è che non altera il contenuto della zona Windows del disco, e per questo si parla di partizionamento *non distruttivo*.
Riassumendo, creare la partizione di Linux richiede due operazioni:

I. deframmentare il disco rigido in modo da ottenere uno spazio contiguo di almeno 1,5 gigabyte, situato dopo l'ultimo file di Windows, usando il programma *Defrag*;

II. trasformare lo spazio contiguo in una partizione, senza alterare il contenuto di dati del disco rigido, usando il programma *fips*.

Tutte le operazioni descritte da qui in avanti devono essere eseguite una dopo l'altra senza interruzioni.

Prima di cominciare, bisogna procurarsi un programma di partizionamento e installarlo su un dischetto di avvio. Il top tra i programmi creati per lo scopo è *Partition Magic* ma è a pagamento. Utilizzando invece il software libero, si farà riferimento al programma *fips*, che è gratuito e liberamente distribuibile. Lo si può trovare su Internet nei siti Linux e nel CD di installazione di Linux (in quasi tutte le distribuzioni è nella directory `dosutils`). Inoltre si consiglia di crearsi un dischetto di avvio per Windows su cui copiare i programmi `fips.exe` e `restorrb.exe`, entrambi presenti, nella directory `dosutils` del CD di Red Hat.

Nell'ambito dei preparativi è consigliabile eseguire anche un controllo dell'integrità del disco rigido utilizzando *Scandisk* di Windows

Superata la verifica di Scandisk, si può passare alla deframmentazione vera e propria, con il programma *Defrag*. La deframmentazione è un procedimento lungo e metodico in cui la macchina va lasciata assolutamente indisturbata durante tutto il procedimento, altrimenti ricomincerà da capo.

B.1.2 Prove di partizionamento

Subito dopo il termine della deframmentazione, bisogna aprire una finestra DOS. Andare alla directory radice del dischetto (`A:\`) e lanciare *fips*. Bisogna fare attenzione a non dirgli di eseguire modifiche alle partizioni esistenti: l'obiettivo, per ora, è soltanto sapere quanto spazio si ha a disposizione per Linux.

Per questa volta ignorare l'avvertimento di fips e premere *Y* per proseguire. Questo porterà ad una schermata che presenta la *tabella delle partizioni*, ossia l'elenco delle partizioni, appunto, in cui è attualmente diviso il disco rigido.

Vediamo come leggere la tabella delle partizioni. Le colonne più importanti della tabella elencano:

- il numero progressivo assegnato a ciascuna partizione (*Part*);
- se la partizione è in grado di avviarsi da sola o no (*bootable*);
- la testina, il cilindro e il settore in cui inizia la partizione (*Start Head Cyl. Sector*);
- testina, cilindro e settore in cui la partizione finisce (*End Head Cyl. Sector*);
- il numero progressivo del settore in cui inizia la partizione (*Start Sector*);
- il numero di megabyte assegnati alla partizione (*MB*).

Premere poi un tasto qualsiasi per proseguire: comparirà una serie di informazioni sulla struttura del disco rigido. Poi inizierà una serie di verifiche, indicate da *Checking boot sector… Checking FAT…*ed infine inizierà la ricerca di spazio libero, come segnalato dal messaggio *Searching for free space*. Queste operazioni possono richiedere qualche minuto di attesa.
Alla fine compare la richiesta *Do you want to make a backup copy…*, che vuole sapere se creare una copia di backup dei settori di root e di boot (due porzioni assolutamente vitali del disco rigido) prima di proseguire. *Per questa volta*, rispondere *N*: si sta ancora conducendo una prova e non si ha ancora intenzione di modificare in alcun modo il disco rigido.
Nella schermata successiva, utilizzare la tabella *Old partition…cylinder… New partition*. Usando i tasti freccia destro e sinistro, si possono regolare i valori della riga immediatamente successiva, che indicano le dimensioni in megabyte della partizione Windows (sotto *Old partition*) e di quella Linux (sotto *New partition*).
Regolare per prova questi due valori e verificare che si possa lasciare un po' di spazio per la crescita di Windows (almeno 200-500 megabyte) ed al tempo stesso lasciare almeno un gigabyte e mezzo a disposizione della nuova partizione, da assegnare a Linux.
Qualunque sia l'esito di questa schermata, non è ancora il momento di attivare le modifiche proposte, per cui premere Invio. Comparirà una tabella delle partizioni aggiornata, seguita dalla domanda *Do you want to continue or reedit the partition table (c/r)?*, a cui si dovrà rispondere *C* per continuare.
Alla domanda successiva, *Ready to write new partition scheme to disk - Do you want to proceed?* (sono pronto a scrivere sul disco la nuova struttura delle partizioni - vuoi che lo faccia?), **fare molta attenzione**. Bisognerà assolutamente rispondere *N*. Se si risponde *Y*, le modifiche proposte, e fin qui simulate, diverranno reali, e lo faranno mentre Windows è in funzione. Il risultato potrebbe essere molto, molto spiacevole.
Rispondendo *N*, fips termina e si può chiudere la finestra DOS (probabilmente si chiuderà da sola).
Se in questa simulazione si è riusciti a ridimensionare le partizioni esistenti in modo da poterne creare una nuova che occupa almeno 1,5 gigabyte si può procedere con il partizionamento vero e proprio.

B.1.3 Partizionamento vero e proprio

Fin qui non si è ancora alterato in alcun modo la struttura del disco rigido. È quello che si farà adesso: questo è un punto di non ritorno. Si ricorda di fare sempre un backup prima di procedere a modificare in modo permanente la struttura del disco rigido.

Lasciare il dischetto di fips nell'unità e chiudere Windows, selezionando *riavvia il sistema*. Il computer avvia poi l'MS-DOS presente sul dischetto invece di avviare Windows come al solito.
Quando compare il messaggio *A:*, immettere `fips` per lanciare fips dal dischetto.
A questo punto si può eseguire il partizionamento del disco rigido. La procedura è esattamente la stessa di prima, solo che alla domanda *Do you want to make a backup copy of your root and boot sector before proceeding (y/n)?* si dovrà rispondere *Y*.
Questo fa comparire la richiesta *Do you have a bootable floppy disk in drive A: as described in the documentation (y/n)?* Cioè fips vuole sapere se si possiede un dischetto avviabile (contenente un sistema operativo) nel drive. Questa volta premere *Y*. Fips scriverà sul dischetto una copia dei settori di root e di boot.

Attenzione: conservare il dischetto di fips! Infatti può darsi che si decida, anche a distanza di tempo, di voler togliere Linux dal computer. In tal caso, per riassegnare a Windows tutto il disco rigido sarà utilissimo il dischetto di fips, perché contiene una copia della tabella dei settori di root e di boot che si può utilizzare per ripristinare la situazione originale senza perdere il contenuto della partizione Windows.
Rimuovere poi il dischetto. Fatto questo, si procede come prima: regolare le dimensioni delle partizioni, usando i tasti freccia, su valori adatti alla configurazione, poi premere Invio. All'ultima domanda (*Ready to write new partition scheme to disk - Do you want to proceed?*), quando fips chiede se scrivere su disco la nuova struttura delle partizioni, rispondere *Y*.
Questo in genere causa il riavvio praticamente immediato del computer e di Windows. Una volta riavviato, Risorse del computer in Windows probabilmente rileverà due dischi rigidi anziché uno. Anche la lettera assegnata al lettore di CD-ROM sarà cambiata, ma è soltanto un effetto temporaneo che scomparirà con l'andare avanti dell'installazione.
A questo punto si è pronti per l'installazione di Linux.

B.1.4 Problemi diffusi

Se non si riesce a liberare spazio a sufficienza o non si riesce a deframmentare i file di Windows in modo che stiano tutti all'inizio del disco significa che qualcosa non va. Si dovrà tentare uno dei seguenti rimedi e poi ripetere tutta la procedura descritta.

Windows è esteso oltre il 1024° cilindro
Questo è in realtà ormai un falso problema: si accenna soltanto perché nella documentazione di Linux si possono trovare moltissimi accenni alla questione del 1024° cilindro. Le vecchie versioni di Linux non potevano essere installate nella zona del disco situata al di là di questo cilindro a causa di una limitazione del programma di avvio. Con le versioni recenti di Linux (per le distribuzioni Red Hat, dalla 6.2 in poi), questo problema non si pone più.

File inamovibili
Capita spesso che fips si rifiuti di creare una nuova partizione di dimensioni sufficienti per ospitare Linux, anche se il totale dello spazio occupato dai file sembrerebbe consentirlo. Questo rifiuto deriva dalla presenza di file *inamovibili*, cioè di file che la deframmentazione di Defrag non può spostare. Spesso, infatti, la deframmentazione riesce a spostare verso l'inizio del disco praticamente tutti i file tranne alcuni, che rimangono isolati nel disco. Siccome fips può creare una partizione soltanto usando spazio contiguo (cioè ininterrotto), se un file si piazza in questo modo, può avere centinaia di megabyte prima e dopo di sé, ma fips potrà utilizzare soltanto lo spazio disponibile *dopo* quel file. L'unico rimedio al problema è scoprire di che file si tratta e trovare la maniera di rimuoverlo. L'ideale sarebbe avere un programma che riveli il nome del file, cosa che Defrag non fa, ma ci sono programmi gratuiti che lo fanno

oppure occorre ricorrere a soluzioni a pagamento come le Norton Utilities. Si può anche provare l'opzione gratuita: tentare una per una ciascuna delle seguenti possibilità, usando fips al termine di ogni tentativo per vedere se lo spazio utilizzabile su disco è aumentato a sufficienza.

B.2 Tipi di installazione

La distribuzione 7.2 di Linux offerta da Red Hat può essere installata avviandola in due modi fondamentali:

- creando un dischetto di avvio, dal quale avviare il computer e procedere con l'installazione attingendo ai file presenti sul primo CD di installazione (applicabile a qualsiasi computer);

- avviando il computer direttamente dal primo dei due CD di installazione, se il computer lo consente.

La seconda soluzione è quella più utilizzata in quanto la maggior parte dei computer consente di estendere il *boot sequence* anche al lettore CD-ROM. Ovviamente questa soluzione funziona se il primo dei CD-ROM di installazione di Red Hat Linux 7.2 è un CD *bootable*. Attenzione alle copie perché se si copia il contenuto file per file, il CD risultante dalla copia non è *bootable* e quindi non si potrà fare boot da quel CD. Bisognerà utilizzare l'installazione da dischetto.

B.3 Installazione con avvio da dischetto

Questa tecnica di installazione crea un *dischetto di boot* che contiene un Linux ridotto all'osso, sufficiente ad eseguire il programma di installazione. Il dischetto di boot servirà anche in caso di problemi di configurazione o se si commette qualche errore di impostazione della macchina che impedisce di avviare Linux.

Avviare Windows e aprire una finestra DOS. Inserire il primo CD di Red Hat Linux 7.2 nel lettore, rendere corrente la directory *dosutils* del CD Red Hat.

Procurarsi un dischetto vuoto e formattato da 1,44 MB.

A questo punto lanciare il programma `rawrite` contenuto nella directory `/dosutils`. Alla domanda *Enter disk image source file name*, rispondere **`\images\it\boot.img`**.

Alla domanda *Enter target diskette* `drive`, rispondere **`A:`** (o in generale con la lettera del drive in cui si è inserito il floppy). Rawrite chiederà di inserire un floppy nel drive e di premere Invio (*Please insert a formatted diskette into drive A: and press ENTER*).

Sul primo CD di installazione di Linux ci sono dei file che contengono una *immagine* di un dischetto di avvio di Linux: una sorta di copia integrale bit per bit. Rawrite prende una di queste immagini (nel caso specifico, il file `boot.img` in versione italiana contenuto nella sottocartella `\images\it` del primo CD) e la scrive sul dischetto inserito nell'unità. In altre parole si sta creando un dischetto di avvio formattato in formato Linux anche se si sta lavorando sotto Windows: è uno dei pochissimi modi in cui Windows può scrivere in un formato di un altro sistema operativo.

B.4 Installazione con avvio da CD-ROM

Assicurarsi di aver impostato la sequenza di avvio in modo che il computer cerchi il sistema operativo prima nel lettore di CD-ROM e poi negli altri drive.

B.4.1 Avvio dell’installazione

A questo punto si è pronti per procedere all'installazione vera e propria di Linux:

- se si è creato il dischetto di avvio, lasciarlo nel drive; lasciare anche il primo CD di installazione di Linux nel lettore e chiudere la finestra DOS;
- se non si è avete creato il dischetto perché si può avviare dal CD, inserire il primo CD di installazione di Linux nel lettore;
- in entrambi i casi, uscire da Windows dicendogli di riavviare il sistema.

A questo punto, invece dell’avvio di Windows, si otterrà una schermata di benvenuto che è in italiano se si avvia dal dischetto e in inglese se si avvia dal CD. Premere Invio per proseguire. Notare le altre opzioni della schermata di avvio, che consentono l'installazione in modalità non grafica, il ripristino di un Linux già installato e altro ancora. Dopo alcuni secondi l'installazione parte automaticamente.
Dopo il messaggio *Loading initrd.img*, compare la dicitura *Loading vmlinuz* seguita da una lunghissima serie di scritte che scorrono sullo schermo: è Linux che sta identificando e riconoscendo i componenti del computer. Questo è uno dei punti critici dell'installazione di Linux. La serie di messaggi potrebbe bloccarsi improvvisamente e dare una schermata di informazioni di errore indecifrabili. Linux ha incontrato un errore: in tal caso, trascrivere fedelmente quello che vedete sullo schermo e farlo leggere a qualcuno con più esperienza oppure cercare in Rete per vedere se qualcuno ha avuto lo stesso problema.
Dopo il messaggio che parla di `/sbin/loader` sarà presente una lunga pausa di inattività: è normale. Dopo qualche altro messaggio, improvvisamente la schermata si oscura completamente e rimane nera per diversi secondi, mentre il CD continua a girare a lungo. È Linux che sta identificando la scheda video e testando il monitor per usare le schermate grafiche più adatte. Infatti dopo qualche istante di attesa e una schermata grigia con una grossa X in mezzo, dovrebbe comparire il logo della Red Hat (l’uomo con il cappello rosso).
Il fatto di vedere il logo della Red Hat dimostra che Linux è in grado di gestire in modo autonomo attraverso i suoi driver la scheda grafica e il monitor del computer su cui si sta effettuando l’installazione.
Se non compare la schermata grafica, interrompere la procedura con `Ctrl-Alt-Canc` e riavviare il computer. L'installazione non ha ancora scritto niente sul computer, per cui si può riprendere a lavorare con Windows esattamente come prima.

Scelta della lingua

A questo punto compare la schermata in cui scegliere la lingua da usare per l'installazione. Cliccare su *Italian* e poi su *Next*. Da questo momento in poi, il resto dell'installazione prosegue in Italiano.

B.5 Proseguimento dell’installazione

L'installazione di Linux non scrive sul disco rigido fino al momento in cui ha raccolto tutte le informazioni necessarie per installarsi. Questo significa che se si deve interrompere

l'installazione per qualsiasi motivo, si può farlo senza alcun problema. Il disco rigido non resterà "sporco" in alcun modo. Per interrompere l'installazione è sufficiente fare reset (tramite il pulsante omonimo o premendo contemporaneamente i tasti `Ctrl+Alt+Canc`) oppure spegnere il computer.
Durante la procedura di installazione verrà chiesto quale bootloader installare tra LILO e GRUB. La differenza tra i due sta nel fatto che GRUB è migliore rispetto a LILO dal punto di vista grafico. Per chi non ha mai utilizzato Linux si consiglia di installare LILO in quanto rappresenta lo standard dei bootloader nei sistemi UNIX ed inoltre è molto facile da configurare.

Tastiera
La prima cosa da impostare è la tastiera. Solitamente il modello da scegliere è il *Generic 102-key (Intl) PC* se si possiede una tastiera recente con i tasti per Windows (quelli fra Ctrl e Alt e fra Ctrl e AltGr); il *layout*, cioè la disposizione dei simboli sui tasti, è *Italian*. I `dead keys` ("tasti morti", letteralmente) servono per la composizione di caratteri tramite combinazioni di tasti (ad esempio la parentesi quadra o il simbolo @): è consigliabile abilitarli. Nel riquadro etichettato come *Verifica della configurazione*, si può provare a digitare dei caratteri per vedere se il risultato a video corrisponde al tasto premuto. Provare in particolare le lettere accentate, le parentesi quadre, il simbolo @ e le parentesi graffe. Se non si riesce ad ottenere tutti i caratteri, provare a selezionare un altro modello di tastiera. In particolare è probabile che le parentesi graffe non funzionino: in realtà non è che non funzionino, è che si producono con una combinazione di tasti diversa da quella di Windows. Sotto Windows italiano, infatti, le parentesi graffe si generano digitando e tenendo premuti tre tasti: `Maiusc`, `AltGr` e la `è` accentata grave (è) per la graffa aperta e `Maiusc`, `AltGr` e `+` per la graffa chiusa. Sotto Linux, invece, a seconda della versione, le graffe si ottengono digitando **`AltGr`** insieme a un numero da **`7`** a **`0`**. Cliccare su *Avanti* per proseguire.

Mouse
Leggere le istruzioni della schermata *Configurazione del mouse* e scegliere la descrizione che più si avvicina alla marca e al modello del mouse in possesso. L'ideale per Linux è avere un mouse con tre pulsanti, ma se si possiede uno con due soli pulsanti, è possibile attivare l'emulazione del terzo cliccando su *Emulazione 3 pulsanti*: quando si cliccherà premendo contemporaneamente entrambi i pulsanti del mouse, Linux crederà che si sia premuto il terzo pulsante e si comporterà di conseguenza. Di solito la scelta comune che attiva anche la "rotellina" del mouse è: *MS IntelliMouse*. Cliccare su *Avanti* per proseguire; quando compare il logo di Red Hat e la scritta *System installer*, cliccare di nuovo su `Avanti`.

Tipo di installazione
In questa schermata c'è da scegliere il tipo di macchina Linux che si vuole ottenere: la scelta standard è l'opzione *Workstation*. Le altre opzioni corrispondono ad altrettanti modi diversi di installare o reinstallare Linux. *C*liccare su *Avanti*. L'opzione *Workstation* installa circa 1050 megabyte di software.

B.6 Partizionamento manuale

Cliccando su *Avanti*, la distribuzione 7.2 di Red Hat Linux ha un'opzione di partizionamento automatico, che consente di lasciar decidere al programma di installazione se e come partizionare il disco rigido. Si consiglia di utilizzare il partizionamento manuale. Quindi quello qui descritto è il partizionamento manuale con Disk Druid: questa non è solo la soluzione più cauta, ma anche quella che consente di decidere in dettaglio come si vuole

partizionare lo spazio su disco. Scegliere dunque il partizionamento manuale con Disk Druid e cliccare su *Avanti*.

B.6.1 Disk Druid

Red Hat Linux può essere installato in due modi: all'interno di due enormi file compatibili con Windows e senza partizionare, oppure in una partizione apposita, ricavata ridimensionando quella di Windows. Per vedere davvero come lavora Linux, bisogna affidargli una partizione propria invece di farlo convivere con il file system di Windows. Di conseguenza si descriverà soltanto il secondo di questi modi di installazione. Da notare che per installare Linux in una partizione apposita, quella partizione deve essere già pronta per l'uso quando si lancia il programma di installazione. Se non è pronta, bisognerà crearla come descritto in precedenza e poi ricominciate l'installazione.

La schermata di Disk Druid presenta la situazione delle partizioni del disco rigido e chiede di definire come e dove installare Linux. Normalmente si troverà elencato il disco rigido, suddiviso in due partizioni, che Linux chiama rispettivamente *hda1* (quella che contiene Windows) e *hda2* (quella che tra poco conterrà Linux). Nella zona superiore della schermata, quella etichettata *Partizioni*, fare clic sulla riga che rappresenta la partizione che si vuole assegnare a Linux: dovrebbe essere la seconda. Fare attenzione a non sbagliare riga, altrimenti verrà cancellato il contenuto della partizione di Windows! Non preoccuparsi troppo, comunque: per ora non viene scritto nulla sul disco, per cui eventuali errori non lasceranno traccia.

La prima cosa da fare è cancellare la seconda partizione. Infatti bisogna ridefinirla ad uso e consumo di Linux: fips è servito semplicemente per ridurre le dimensioni della partizione di Windows. Assicurarsi ancora una volta di *non* avere scelto la prima partizione, quella di Windows, e poi cliccare su *Cancella*. Cliccare su *Sì* in risposta alla richiesta di conferma. Da notare che la situazione della metà inferiore della schermata è cambiata per rispecchiare questa modifica.

A questo punto bisogna aggiungere le partizioni necessarie al funzionamento di Linux. Infatti a Linux si devono assegnare normalmente *due* partizioni (gli utenti più evoluti ne creano anche più di due). Cliccare su *Aggiungi*. Nella finestra di dialogo cliccare sul menu a tendina *Linux Native* e scegliere *Linux swap*.

Le dimensioni di questa partizione di swap sono regolabili a piacimento, ma il criterio generale è questo:

> *assegnare, se possibile, il doppio della dimensione della RAM alla partizione di swap.*

Cliccare su *OK*. Scegliere di nuovo *Aggiungi* per aggiungere la *partizione nativa* di Linux, ossia quella in cui risiederanno il sistema operativo, i dati ed i programmi. Stavolta si può accettare il tipo di partizione proposto (*Linux native*) ma si dovrà solo specificare il *mount point* della partizione. Digitare uno *slash* (*/*) nella casella *Mount Point*, poi cliccare su Usa spazio rimanente: questo dice a Linux di usare tutto lo spazio rimanente sul disco per questa partizione. Poi cliccare su *OK*.

A questo punto si dovrebbero avere tre partizioni:

- la prima, identificata come `<non impostato>` e `hda1`, che è quella di Windows, come indicato dalla dicitura `Win95 FAT32` (o `FAT16`);
- la seconda, contrassegnata da `/` e `hda2`, di tipo *Linux native*;

- la terza, definita come `<Swap>` e `hda5`, di tipo *Linux swap.*

A seconda della configurazione del proprio computer, le assegnazioni *hda2* e *hda5* potrebbero ricevere numeri diversi; l'importante è che si abbiano una partizione Linux nativa e una partizione Linux swap. Prendere nota, comunque, dei numeri assegnati a queste due partizioni. Cliccare su *Avanti* per proseguire.

B.6.2 Scelta delle partizioni da formattare

Il programma di installazione propone di formattare la partizione `/dev/hda2` (o un `hda` seguito da un altro numero). Per sapere di che partizione si tratta, guardare gli appunti: dovrebbe essere quella Linux nativa. In questo caso può essere di aiuto ricordarsi la dimensione scelta per la partizione di Linux e la scomposizione in partizione nativa e swap. Cliccare sul pulsante *Controllo dei blocchi danneggiati durante la formattazione* in modo da far comparire un segno di spunta: in questo modo il programma di installazione verifica l'integrità del disco rigido, e se trova porzioni di disco danneggiate le contrassegna in modo che non vengano più utilizzate. Cliccare su *Avanti*.

B.7 Altre configurazioni

Configurazione della rete

Se avete si possiede nel computer una scheda di rete che viene riconosciuta dal programma di installazione, a questo punto compare sullo schermo la richiesta di immettere i parametri di configurazione della connessione di rete. Se non si sa cosa mettere, cliccare su *Avanti*; si può sempre configurare la connessione di rete in seguito. Se invece si conoscono quali parametri immettere, immetterli e proseguire cliccando su *Avanti*.

Configurazione del firewall

Un *firewall* è una sorta di “cinta muraria” digitale eretta a protezione del sistema. Leggere attentamente le spiegazioni nella colonna di sinistra per capire il significato delle varie opzioni. Si suggerisce di adottare l'opzione *Alto* per il livello di sicurezza, attivando il pulsante *Personalizza* e attivando la casella *eth0* nella sezione *Periferiche fidate*. Poi cliccare su *Avanti*.

Selezione delle lingue di supporto

Questa sezione dell'installazione vi consente di scegliere quali lingue volete installare, evitando di installare le altre. In genere, però, si installa una sola lingua (l'italiano, in questo caso), per cui lasciare il segno di spunta su *Italian (Italy)* e cliccare su *Avanti*.

Scelta del fuso orario

Nel menu a tendina del pulsante `Vista` si può scegliere la zona del mondo da visualizzare nella mappa. Se si vuole usare l'ora italiana, cliccare sul puntino che indica Roma.

Sarebbe bene però non usare l'ora italiana, ma l'ora standard di Greenwich o GMT, che equivale all'ora UTC presentata fra le opzioni di Linux. Questo perché in questo modo non si dovrà pensare ai cambi imposti dall'ora legale. L'ora di Greenwich è uno standard di riferimento internazionale in moltissimi settori scientifici, tecnologici, industriali e commerciali, e non fa mai salti per cambi d'ora legale o solare. Inoltre praticamente tutti i computer di Internet usano GMT o UTC: questo dà a tutti gli utenti la certezza dell'ora in cui, ad esempio, è stato aggiornato un documento o spedito un messaggio, a prescindere da fusi orari o ore legali.

- Se si sceglie di adottare l'ora di Greenwich, cliccare su *L'orologio di sistema usa UTC*, poi scegliere la scheda *Offset UTC* e selezionare la riga *UTC* (senza numeri positivi o negativi).
- Se si sceglie l'ora italiana, cliccare sul puntino che rappresenta Roma sulla mappa oppure selezionare la località dal menu sottostante.

Infine cliccare su *Avanti*.

Configurazione account

Si consiglia vivamente di prendere carta e penna e prendere nota delle password e dei nomi di account (nomi di login) che si immettono in questa schermata. Uno degli errori più comuni delle installazioni di Linux è fare tutto giusto e trovarsi con un Linux correttamente configurato, ma non ricordarsi più la password di root. Fare anche attenzione che non sia attivato il blocco delle maiuscole sulla tastiera altrimenti la password verrà digitata in maiuscolo credendo di farlo in minuscolo. Linux è *case-sensitive* e quindi in seguito considererà sbagliata la digitazione e non permetterà l'accesso al computer.

Nella casella *Password di root* specificare la password che verrà usata dal superutente *root*. Per evitare che si sbagli a scriverla, bisognerà digitarla due volte, una qui e una nella casella sottostante. Inoltre la password deve essere lunga almeno sei caratteri, altrimenti non verrà accettata.

E' consigliabile scegliere una password non troppo ovvia e che contenga almeno sei caratteri (è il minimo obbligatorio).

La casella *Nome account* serve per definire il nome, o *userid*, di un utente *comune*, cioè che non ha i privilegi e gli accessi assegnati al superutente amministratore. Nelle caselle sottostanti definire la sua password (due volte, come prima) e immettere il suo nome e cognome (facoltativo o di fantasia, se si preferisce).

Anche se si è la sola persona che usa il computer su cui si sta installando Linux, è meglio definire anche un utente non privilegiato (spesso lo si chiama "utente comune" o "utente normale"). Si userà *root* soltanto per la manutenzione, e l'utente non privilegiato per il lavoro normale. È una semplice ma efficace forma di autodifesa da errori e distrazioni.

Ciccare quindi su *Aggiungi*. I dati dell'utente non privilegiato compariranno nella parte inferiore della schermata. A questo punto si può anche creare altri utenti, cliccando su *Nuovo* e immettendone i dati. Quando si è terminato, cliccare su *Avanti* per proseguire.

Scelta dei pacchetti

Questa schermata consente di scegliere quale tipo di interfaccia grafica si vuole usare (Linux ne offre due, *KDE* e *GNOME*) e di decidere se si vuole dedicare una parte dello spazio disponibile ai giochi allegati a Red Hat Linux. Volendo, è qui che l'utente esperto può scegliere esattamente quali programmi (o *pacchetti*) installare e quali no e quindi crearsi un'installazione più compatta e snella.

Si consiglia di scegliere *KDE* (lasciando attivato soltanto il suo pulsante di scelta e disattivando gli altri, si risparmiano circa 120 megabyte) e cliccare su *Avanti*.

B.7.1 Configurazione di X

Configurazione di X: rilevamento della scheda video

Il rilevamento automatico di Linux è in grado di identificare automaticamente la scheda video nella maggior parte dei casi. Controllare, in particolare, che la quantità di memoria RAM video che si possiede sulla scheda video corrisponda a quella rilevata da Linux. Se Linux non riesce a trovare nessuna scheda, si consiglia di selezionare *Generic VGA compatible*. Una volta scelta la scheda, cliccare su *Avanti* per proseguire.

Rilevamento del monitor
Questa schermata presenta il risultato del tentativo di Linux di rilevare automaticamente il tipo di monitor. Se questo risultato corrisponde alle caratteristiche effettive del monitor posseduto, si può accettarlo cliccando su *Avanti*.
Se non viene rilevato nulla oppure viene indicato un modello completamente diverso da quello che si possiede, vuol dire che Linux non riesce a capire autonomamente il tipo di monitor utilizzato. Quindi viene presentata una lista dalla quale scegliere le voci che più si avvicinano, come caratteristiche e definizioni, al monitor in questione. Nel dubbio, scegliere la sezione *Generic* e la sottosezione *Generic VGA compatible*.

Test della configurazione grafica
Nella schermata *Configurazione personalizzata* cliccare sul pulsante *Verifica dell'impostazione*. Se dopo qualche secondo compare quella che sembra una schermata di Windows un po' strana (in realtà è un'immagine di una schermata grafica di Linux) e la domanda *Potete vedere questo messaggio?*, cliccare su *Sì* e attivare l'opzione *Grafico* nella sezione *Scegliete il tipo di login*.
Se non si vede nulla, attendere una ventina di secondi per tornare automaticamente alla schermata di configurazione grafica. Tentare con altri parametri per la risoluzione dello schermo e la profondità di colore. E' probabile che almeno una delle varie combinazioni di queste due caratteristiche faccia funzionare correttamente il monitor che si sta utilizzando.
Se non c'è modo di ottenere una configurazione grafica funzionante, cliccare su *Indietro* fino a tornare alla schermata di riconoscimento della scheda video e attivare il segno di spunta nella casella *Salta la configurazione di X*: a questo punto si consiglia di andare in Internet per vedere se qualche altro utente ha avuto lo stesso problema.
In ogni caso, cliccare su *Avanti* per proseguire finché compare una schermata intitolata *Processo di installazione*.

B.8 Modifica effettiva del disco

Attenzione: Se si clicca su *Avanti* in questa schermata, le modifiche e le impostazioni che sono state fatte cesseranno di essere virtuali e diverranno reali: il programma di installazione comincerà a scrivere sul disco rigido e continuerà a scrivervi per diverso tempo. Fino a questo momento si può interrompere l'installazione brutalmente, togliendo il CD di Linux o il floppy e premendo il pulsante di reset o la combinazione di tasti `Ctrl-Alt-Canc`: il computer non serberà traccia dell'installazione interrotta. Dopo questo momento, invece, il disco rigido verrà "sporcato" dall'installazione incompleta.
Cliccando su *Avanti*, sullo schermo compare la scritta *Formattazione filesystem /*. Significa che il programma di installazione di Linux sta formattando, secondo il formato di Linux, la partizione che gli è stata indicata nei passi precedenti. La scritta rimarrà sullo schermo per qualche minuto.
A questo punto il programma di installazione inizia ad installare il sistema operativo e i *pacchetti* di Linux. Ad un certo punto dell'installazione dei pacchetti verrà chiesto di inserire il secondo CD di Red Hat Linux. Inserirlo e cliccare su *OK* per proseguire.

B.9 Dischetto di boot

Al termine dell'installazione dei pacchetti, il programma di installazione chiederà un dischetto da usare come *boot disk* (dischetto di avvio): inserirlo e ciccare su *Avanti*. Il programma di installazione creerà un dischetto da usare per avviare Linux in caso d'emergenza.

Finita la creazione del dischetto, comparirà un messaggio che annuncia che l'installazione è terminata. Cliccare su *Esci*. Togliere dal drive il CD-ROM non appena Linux lo espelle e rimuovere il floppy che è stato appena scritto.
A questo punto lo schermo si oscura per qualche decina di secondi, poi il computer si riavvia. Se tutto è andato liscio, compare una schermata grafica con due opzioni (*linux* e *dos*): selezionare *dos* e premete Invio se si vuole avviare Windows. Se si seleziona *linux* dopo una breve attesa parte Linux. Se si vuole uscire da Linux e tornare alla schermata di avvio in cui scegliere fra Windows e Linux, cliccare su *Shutdown* e selezionate *Shutdown*; cliccare su OK.

B.10 Primo utilizzo

A questo punto se si è ritornati in Windows si consiglia di riavviare il computer, senza inserire floppy. Selezionare Linux dal menu di avvio e premere Invio.
Comparirà il messaggio *Loading linux…*: da qui inizia un'elencazione dei componenti hardware rilevati. Inoltre comparirà una lista in cui ogni singolo elemento del software e dell'hardware viene verificato e, se tutto è in ordine, attivato con un *OK*. Tutto questo serve per localizzare la causa di eventuali problemi di avvio della macchina.
Da notare che Linux impiega molto più tempo di Windows a partire. Il motivo è semplice: Linux non è concepito per macchine che si accendono e spengono frequentemente. È stato progettato per funzionare in continuazione come ad esempio sui server.
Una volta che Linux ha verificato i vari componenti, compare la richiesta di accesso:

```
localhost login:
```

Ciò indica che Linux si è avviato. Se si ignora la richiesta e si aspetta qualche secondo compare la schermata grafica di login, nella quale si può immettere il nome dell'utente comune (*non* quello di *root*) e la sua password, che non viene visualizzata, nemmeno con asterischi (sicurezza). Cliccare su *Go!* e l'interfaccia grafica KDE di Linux verrà avviata.
Se la schermata grafica di login non compare, digitare nome e password dell'utente comune e poi digitare **startx** per avviare l'interfaccia grafica di Linux. Per completare l'avvio Linux impiegherà una decina di secondi circa. Comparirà poi un draghetto (Kandalf) che propone il suggerimento del giorno. Cliccare su *Chiudi* per chiuderlo.
A questo punto si è liberi di fare un "giro" all'interno dell'interfaccia grafica del nuovo sistema operativo. Ad esempio si può cliccare sulla *K* della "barra delle applicazioni" e scegliere *Accessori* > *Calcolatrice*. Comparirà sullo schermo la calcolatrice di Linux. Oppure se si è un utente medio-esperto si può aprire la finestra di Terminale e digitare dei comandi…
Per terminare la sessione di lavoro, basta cliccare sulla *K* e scegliere *Termina sessione*. Da qui si ritorna alla schermata di login: a questo punto si può cliccare su *Shutdown*. Per spegnere il computer, selezionare l'opzione *Shutdown* e cliccare su *OK*.
Se l'interfaccia grafica non è partita automaticamente durante l'avvio di Linux, bisognerà digitare il comando di chiusura di Linux: **`shutdown -r now`** (che causa un reboot della macchina). Per spegnerlo definitivamente bisogna utilizzare: **`shutdown -h now`**. Da notare che per eseguire questi comandi bisogna essere loggiati come *root*, altrimenti l'alternativa è di spegnere manualmente il computer.
Durante la chiusura di Linux comparirà un'altra lista di messaggi diagnostici, al termine della quale ci sarà il messaggio *Power down*, che conferma che si può spegnere manualmente il computer, oppure si assisterà allo spegnimento automatico se il computer supporta questa funzione.

B.11 Nascondere Linux

Ora che si è installato Linux, ad ogni avvio si avrà la richiesta di scelta del sistema operativo. Su un computer condiviso con altre persone, questo può dar fastidio o disorientare. C'è un modo molto semplice per far sparire questa richiesta e far avviare automaticamente Windows esattamente come prima dell'installazione di Linux.

- Avviare il computer usando il dischetto di ripristino di Windows.
- Digitare **`fdisk /mbr`** e seguite le istruzioni.
- Riavviare il computer.

Fatto questo, l'unico modo per avviare Linux è utilizzare il dischetto di boot, che diventerà una sorta di "chiave" personale di avviamento. L'avviamento sarà un po' più lento, ma perlomeno chiunque altro accenda il computer avvierà Windows e non si accorgerà dell'esistenza di Linux.

B.12 Terminale all'avvio

Per coloro che vogliono utilizzare Linux per l'ufficio oppure per vedere solamente come funziona, si consiglia di non modificare le impostazioni utilizzate durante la descrizione della procedura di installazione del sistema operativo. In questo caso infatti, ogni volta che si sceglie di utilizzare Linux, partirà automaticamente l'interfaccia grafica KDE da cui si possono utilizzare dei programmi di videoscrittura più evoluti rispetto ai programmi utilizzati di solito nel terminale. Per chi volesse comunque utilizzare il terminale per i comandi più comuni, basta andare sulla *K* > *Sistema* > *Shell* o *Terminale (in modalità superutente)*.
Comunque si consiglia, per gli utenti desiderosi di imparare le basi di Linux, di modificare la propria configurazione nel caso in cui si verifica che KDE parte ogni volta che si accede al sistema operativo. Per fare ciò basta eseguire i seguenti comandi (bisogna essere loggiati come root):

```
# cd /etc
# pico inittab
```

A questo punto si avrà una schermata in cui sono riportate le seguenti scritte:

```
#
# inittab     This file describes how the INIT process should set up
#             the system in a certain run-level.
…
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have
networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:5:initdefault:

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

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
```

...

A questo punto basterà modificare la riga :

`id:5:initdefault:` con: `id:3:initdefault:`

e salvare il documento premendo la combinazione di tasti `Ctrl+X` e rispondendo affermativamente alla richiesta di salvataggio.
Da ora in poi ogni volta che si accede a Linux, si accederà tramite terminale. Quindi si verrà accolti dalle scritte:

```
localhost login:
Password:
```

in cui inserire i propri dati settati durante l'installazione. A questo punto comparirà la riga di comando del tipo (Bourne Shell):

`[root@localhost root]#` (ovviamente se ci si è loggiati come root)

dove, root@localhost indica chi è la persona che è entrata (*root*) e su quale computer si trova (*localhost*). Inoltre `root` specifica la directory corrente in cui ci si trova. Il prompt si conclude con un carattere finale: se è *#* (cancelletto) indica che si hanno poteri da amministratore (root); se invece è *$* indica che si hanno i privilegi di un utente normale.
Per accedere alla modalità grafica di Linux si può digitare il comando:

```
[root@localhost root]# startx
```

mentre se ci si trova in modalità di utente normale:

```
[username@localhost username]$ startx
```

Appendice C

Installare Linux Mandrake 8.2

L’installazione di Linux Mandrake 8.2 è totalmente grafica e non si discosta molto da quella della RedHat 7.2.
Poiché i CD sono bootable, il metodo più semplice per avviare l’installazione è quello di impostare il boot da CDROM e inserire il primo CD. Altra strada percorribile rimane sempre quella del boot da floppy (utilizzando rawrite per la creazione del floppy di avvio).
Analizziamo, in modo schematico, i passi necessari per l’installazione di questa distribuzione:

1. Scelta della lingua
Sono a disposizione praticamente tutte le lingue in uso: ovviamente la scelta consigliata è quella dell’italiano. A fine installazione, tutta la distribuzione "parla" italiano (pagine di man, localizzazione delle applicazioni, ecc.). Per proseguire con l’installazione sarà prima necessario accettare le condizioni di licenza.

2. Classe di installazione
Qui si sceglie quale procedimento utilizzare per l’installazione della distribuzione: si può optare tra quella "Raccomandata", che consentirà a Mandrake di fare alcune scelte al posto dell’utente (come il tipo di mouse, tastiera, ecc.) o quella "Esperto" che fornisce la libera scelta per quel che riguarda la personalizzazione e configurazione. Se si ha un minimo di esperienza con l’installazione di un sistema operativo si consiglia la modalità "Esperto". Da notare che in questo punto è anche possibile indirizzare l’installazione verso l’aggiornamento di una versione precedente di Mandrake.

3. Ricerca del disco fisso
Ora la procedura di installazione provvederà a riconoscere ed analizzare l’hard disk del computer su cui si sta installando il sistema.

4. Configurare il mouse
5. Scelta della tastiera
Qui verrà chiesto il tipo di mouse presente. Una volta selezionato sarà possibile testare la corretta configurazione grazie alla apposita interfaccia. Segue poi la scelta del tipo di tastiera da utilizzare.

6. Sicurezza
Si può scegliere tra quattro livelli di sicurezza differenti: Normale, Alto, Più Alto e Paranoico. Per un’utenza normale si consiglia "Normale", rimandando ad una successiva (e minuziosa) configurazione di un firewall.

7. Configurazione del filesystem
8. Formattazione partizioni
Questa è una delle fasi più delicate di tutta l’installazione. Verrà lanciato DiskDrake, l’utility per il partizionamento dell’hard disk. Tra le opzioni disponibili c’è la presenza di un filesystem crittato (abilitando l’apposita voce tra le opzioni della partizione). Questa maggiore sicurezza dei dati si paga ovviamente con una maggiore lentezza e la necessità di dover scrivere la relativa pass phrase (una password di almeno 20 caratteri) ad ogni boot. Grazie a DiskDrake è anche possibile ridurre la dimensione delle partizioni di tipo FAT per far posto a partizioni create.

9. Scelta dei pacchetti
10. Installazione del sistema
Dopo che il programma ha chiesto di quali CD si dispone, si procederà con quelle di Linux. Una volta scelta la configurazione per il file system, verranno formattate le la scelta dei pacchetti. È possibile selezionare i programmi da installare tramite 19 gruppi ed eventualmente raffinare la scelta usando l'opzione "Selezione individuale dei pacchetti". Cliccare poi su "Installa" per avviare l'installazione vera e propria. In questa fase avverrà l'installazione dei pacchetti scelti. Per un'installazione quasi totale, Mandrake ha impiegato circa un'ora, mentre un'installazione minimale di 600 MB si è conclusa in un quarto d'ora circa.

11. Scelta password di root
Qui bisogna inserire la password scelta per l'utente root, l'amministratore di sistema.

12. Aggiungere un utente
Ora è possibile creare degli utenti. È bene ricordare l'importanza dell'utilizzare l'account di root solo occasionalmente per le operazioni di manutenzione e amministrazione. Per un uso "comune" del sistema operativo è molto più sicuro utilizzare un proprio e personale account.

13. Configurazione rete
Da qui si può configurare la propria connessione ad Internet. Già in fase di installazione ad esempio è possibile impostare il nostro collegamento tramite ADSL o modem, oppure inserire il proprio PC in una LAN (con il riconoscimento automatico della scheda di rete) tutto con pochi click utilizzando un'interfaccia estremamente intuitiva.

14. Riepilogo
In questo punto si possono rivedere alcuni punti della configurazione come la scelta del mouse, della tastiera o del fuso orario, oppure configurare una eventuale stampante.

15. Configurazione servizi
Da qui è possibile scegliere i "demoni" da attivare o disattivare al boot. Mandrake tende di solito ad abbondare con i "servizi" (termine più appropriato per i sistemi operativi Microsoft), quindi è consigliabile dare almeno uno sguardo a quello che verrà caricato.

16. Installazione bootloader
In questa fase verrà configurato e installato il bootloader. Di default viene scelto LILO nella sua versione "grafica", ma è possibile scegliere anche la versione testuale oppure il più recente GRUB. Da qui è possibile anche impostare il sistema operativo da caricare di default all'avvio.

17. Creazione disco di boot
Verrà chiesto se creare un floppy di boot qualora qualcosa vada storto e ci si trovi ad essere impossibilitati ad avviare il sistema operativo dal disco fisso. Si consiglia di crearlo: potrebbe sempre tornare utile.

18. Configurazione
Ora è possibile scegliere il server X da utilizzare. Si consiglia di scegliere XFree86 4.2.0 (invece della più collaudata ma anche più vecchia 3.3.6), l'ultima versione di X disponibile, in quanto supporta un maggior numero di schede con prestazioni nettamente superiori. Una volta installati i pacchetti necessari verrà chiesto il tipo di monitor utilizzato (se non si dovesse

trovare il proprio modello tra la lista di quelli supportati è sempre possibile scegliere uno di quelli generici). Poi bisogna scegliere la risoluzione video preferita ed eventualmente provarla (consigliato). Fatto questo verrà chiesto se avviare o meno il server X al boot (per il login grafico).

19. Installazione aggiornamenti di sistema
Da qui è possibile controllare su Internet la presenza di aggiornamenti. Se si dispone di una connessione veloce si consiglia di rispondere affermativamente alla domanda relativa a questo punto.

20. Terminazione dell'installazione
Si è finalmente conclusa l'installazione. Cliccando su OK è possibile riavviare il sistema e iniziare ad utilizzare già da subito il nuovo sistema operativo.

Appendice D

Installare UNIX FreeBSD 4.x

Prima di passare all'installazione vera e propria bisogna stabilire lo spazio su disco da riservare al sistema FreeBSD. Di solito si ridimensiona una partizione dos o ext2 esistente utilizzando fips. Anche se fips è un programma molto semplice, prima di utilizzarlo è consigliabile fare un backup dei dati della partizione esistente. Una volta stabilita la partizione primaria o lo spazio libero che sarà destinata a contenere FreeBSD, si procede a far bootare il CDROM. In alternativa si possono creare i 2 floppy di boot e di installazione seguendo un procedimento simile a quello utilizzato in Linux.

D.1 Avvio dell'installazione

Dopo i primi messaggi di boot il tool di installazione presenta 3 opzioni relative al kernel: servono per passare eventuali parametri al kernel direttamente in fase di installazione. Per macchine standard si sceglierà:

"*Skip kernel configuration and continue with installation*"

Apparirà la schermata del tool di installazione chiamato "*sysinstall*" che si trova normalmente in ogni momento anche a Sistema Operativo installato in `/stand/sysinstall`.

```
??????????????????????? /stand/sysinstall Main Menu ???????????????????????
? Welcome to the FreeBSD installation and configuration tool.  Please   ?
? select one of the options below by using the arrow keys or typing the ?
? first character of the option name you're interested in.  Invoke an   ?
? option with [SPACE] or [ENTER].  To exit, use [TAB] to move to Exit.  ?
? ??????????????????????????????????????????????????????????????????? ?
? ?   Usage       Quick start - How to use this menu system           ? ?
? ?  Standard     Begin a standard installation (recommended)         ? ?
? ?  Express      Begin a quick installation (for the impatient)      ? ?
? ?   Custom      Begin a custom installation (for experts)           ? ?
? ?  Configure    Do post-install configuration of FreeBSD            ? ?
? ?  Doc          Installation instructions, README, etc.             ? ?
? ?  Keymap       Select keyboard type                                ? ?
? ?  Options      View/Set various installation options               ? ?
? ?  Fixit        Enter repair mode with CDROM/floppy or start shell  ? ?
? ?  Upgrade      Upgrade an existing system                          ? ?
? ?  Load Config  Load default install configuration                  ? ?
? ?  Index        Glossary of functions                               ? ?
? ??????????????????????????????????????????????????????????????????? ?
????????????????????????????????????????????????????????????????????????
?                         [Select]    X Exit Install                    ?
??????????????????[ Press F1 for Installation Guide ]???????????????????
```

FIG. A1

Si procederà ora con i seguenti passi:

1. Settare la tastiera nel sottomenu "*keymap*": solitamente *Italian Italian ISO keymap* .

2. A questo punto si può iniziare con l'installazione vera e propria selezionando il menu "*custom*", dopo essere usciti da "*keymap*" ed essere tornati nella videata principale di sysinstall. Il sottomenu "*custom*" (installazione custom) si presenta come segue:

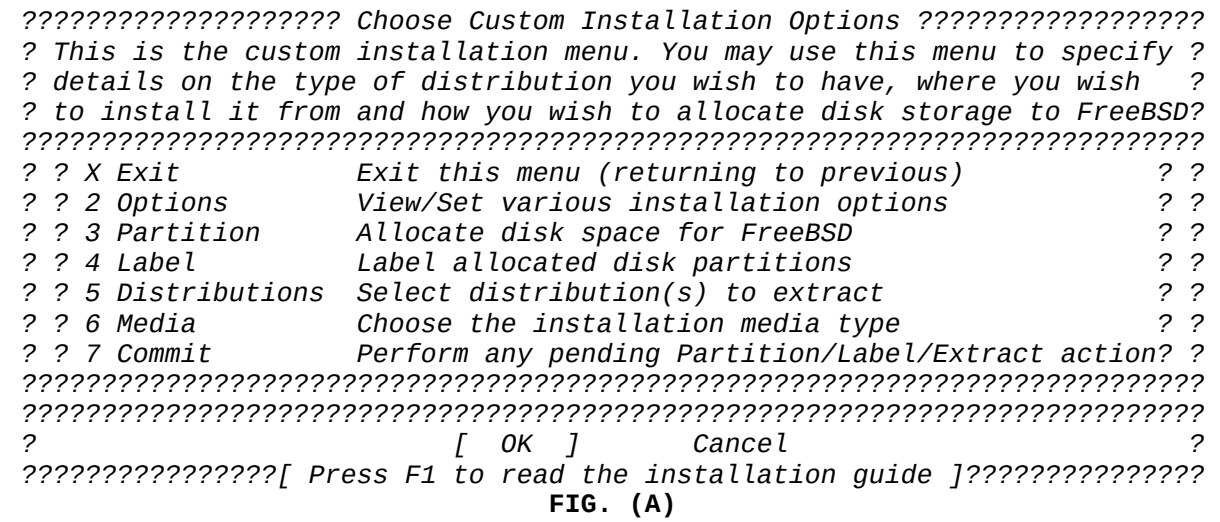


FIG. (A)

Si procederà in ordine, dalla prima opzione all'ultima:

3. In "*options*" si possono modificare le varie opzioni di installazione: il consiglio è di non modificare nulla per il momento. Premere il tasto "*Q*" ed uscire da "*options*".

4. Selezionare *Partition* per andare a creare la "SLICE" BSD.

Attenzione: bisogna sottolineare la differenza e la particolarità del sistema di partizionamento di *BSD (FreeBSD in questo caso). Verrà creato lo "**SLICE**" che per l'hard disk sarà una partizione primaria e, dentro di esso verrano create le "LABELS" , ossia una sorta di sottopartizioni (praticamente NON viste da tools tipo fdisk del dos o di linux) che formeranno il file system vero e proprio del FreeBSD (`/`, `/var`, `/home`, `/usr` ad esempio).

Da notare che:

- nell'hard disk NON potranno esistere più di 4 partizioni primarie e lo SLICE è visto proprio come una partizione primaria a tutti gli effetti;

- NON si possono utilizzare partizioni formattate ext2 di Linux per metterci il filesystem BSD o parte di esso, ma si può solo utilizzare lo spazio precedentemente occupato da una partizione ext2 o dos o altro cancellandola e poi ricrearla come SLICE BSD (flag 165);

- NON si possono nemmeno utilizzare eventuali partizioni di SWAP di Linux che siano esse primarie o estese.

Praticamente si può immaginare lo SLICE come un guscio ermetico al cui interno esistono partizioni di filesystem e swap (di qualsiasi numero, non c'è limite).

D.2 Prove di partizionamento

Si è appena selezionata la voce "*Partitions*" dal sottomenu "*custom*" del sysinstall e qui dipende se si posseggono uno e più hard disk all'interno della macchina. Se si possiede un solo HD, il tool provvederà automaticamente a lanciare fdisk, altrimenti farà scegliere quale disco si vuole partizionare. Da notare la differenza di sintassi usata per i device fra DOS, Linux e FreeBSD:

```
-Hard disk 1, primo canale IDE (master):
       DOS= C:\              Linux=   /dev/hda        FreeBSD=   /dev/ad0(o
/dev/wd0)

-Hard disk 2, secondo canale IDE (master):
       DOS= D:\              Linux= /dev/hdb FreeBSD= /dev/adl
```

Quindi, se nella macchina ci sono 2 HD, la schermata che si presenta è la seguente:

```
  ??????????????????????????? Select Drive(s) ???????????????????????????????
  ? Please select the drive, or drives, on which you wish to perform    ?
  ? this operation.  If you are attempting to install a boot partition  ?
  ? on a drive other than the first one or have multiple operating      ?
  ? systems on your machine, you will have the option to install a boot ?
  ? manager later.  To select a drive, use the arrow keys to move to it ?
  ? and press [SPACE] or [ENTER].  To de-select it, press it again.     ?
  ?                                                                     ?
  ? Use [TAB] to get to the buttons and leave this menu.                ?
  ? ????????????????????????????????????????????????????????????????? ?
  ? ?                          [ ] ad0  ad0                           ? ?
  ? ?                          [ ] ad1  ad1                           ? ?
  ? ????????????????????????????????????????????????????????????????? ?
  ???????????????????????????????????????????????????????????????????
  ?                       [  OK  ]       Cancel                         ?
  ????[ Press F1 for important information regarding disk geometry! ]????
```

FIG: (B)

Selezionando "*ad0*", si entra in fdisk per creare lo slice nel primo disco IDE. Con fdisk si creerà SOLO lo slice, ossia il contenitore. La schermata dell'fdisk sarà simile alla seguente:

```
-------------------------------------------------------------------------
 Disk name:      ad0                                    FDISK Partition
Editor
DISK Geometry:  4866 cyls/255 heads/63 sectors = 78172290 sectors (38170MB)

Offset       Size(MB)        End     Name  PType       Desc  Subtype  Flags

         0          0         62        -      6     unused        0
        63       7004   14346044    ad0s1      1 NTFS/HPFS/QNX           7
  14346045      11005   36885239    ad0s2      3     ext2fs        0
  36885240      11993   61448624    ad0s3      3     ext2fs        0
  61448625       8165   78172289    ad0s4      4   extended        5
  78172290          2   78177791        -      6     unused        0

The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry   C = Create Slice
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable
T = Change Type        U = Undo All Changes     W = Write Changes

Use F1 or ? to get more help, arrow keys to select.
-------------------------------------------------------------------------
```

FIG: (C)

Per comodità utilizzare il tasto "*Z*" per variare l'unita' di misura nella colonna "*Size*" in modo da far comparire i MB. Se si vuole creare uno slice al posto della attuale partizione "*ad0s2*" (ossia primo disco, seconda partizione primaria) con flag ext2fs (Linux) basta portarsi con i tasti freccia (cursore) sulla partizione in oggetto e premere "*D*" (delete slice): la partizione che prima era `ad0s2=ext2fs` ora è diventata "*unused*".

Subito dopo premere "*C*" (create slice) in cui viene proposto un numero (la grandezza delle spazio libero). Non rimane che accettarlo con "Ok" (tasto Invio) , accettare il flag che mi propone (165 = slice BSD) e la partizione che era "*unused*" ora diventa:

```
14346045       11005       36885239       ad0s2       3       freebsd       165
```

Si è creato uno slice di circa 11GB (11000 Mb circa) di grandezza. Se si avesse avuto solo spazio vuoto, o addirittura un disco vergine, per specificare lo spazio da destinare allo slice FreeBSD, bastava scrivere la quantità in Mb da destinare allo slice al posto del numero proposto: ad esempio si sarebbe potuto scrivere 11000MB (anche se una FreeBSD si può installare in molto meno, 1 Gb è più che sufficiente). Lo slice a questo punto è creato. Premere "*Q*" per uscire da fdisk.

Il tool ora propone come far bootare lo slice, ossia il nuovo sistema operativo (anche se non ancora installato), per cui viene proposta questa schermata:

```
????????????????? Install Boot Manager for drive ad0? ?????????????????
? FreeBSD comes with a boot selector that allows you to easily          ?
? select between FreeBSD and any other operating systems on your machine?
? at boot time.  If you have more than one drive and want to boot       ?
? from the second one, the boot selector will also make it possible     ?
? to do so (limitations in the PC BIOS usually prevent this otherwise)  ?
? If you do not want a boot selector, or wish to replace an existing    ?
? one, select "standard".  If you would prefer your Master Boot         ?
? Record to remain untouched then select "None".                        ?
?                                                                       ?
?   NOTE:  PC-DOS users will almost certainly require "None"!           ?
? ??????????????????????????????????????????????????????????????????? ?
? ?         BootMgr   Install the FreeBSD Boot Manager                ? ?
? ?         Standard  Install a standard MBR (no boot manager)        ? ?
? ?         None      Leave the Master Boot Record untouched          ? ?
? ??????????????????????????????????????????????????????????????????? ?
???????????????????????????????????????????????????????????????????????
?                          [  OK  ]       Cancel                        ?
?????????????????[ Press F1 to read about drive setup]?????????????????
```

FIG: (D)

Si hanno 3 opzioni. **NOTARE BENE** che se già sulla macchina fosse presente una distribuzione Linux con un LILO installato nel MBR (*Master Boot Record*), l'opzione da scegliere sarà "*none*", ossia NON installare alcun BSD-loader. Invece se sulla macchina esistesse solo la FreeBSD nuova oppure FreeBSD + Windows, si dovrà installare il boot loader che viene proposto dalla FreeBSD stessa: quindi a tale scopo si potrà scegliere la prima opzione "`BootMgr`" (`Install the FreeBSD Boot Manager`). Andare su "OK" con il tasto tab oppure con le frecce e premere Invio.

Si è tornati nuovamente al menu di "*custom*" (FIG: (A)). A questo punto quindi si possono creare le "LABELS" ossia le vere partizioni che formano il file system, e lo swap. Selezionare

in ordine di elenco quindi "*Label*" dalla schermata del menu ed entrare nel programma "*disklabels*".
La schermata che apparirà sarà la seguente:

```
--------------------------------------------------------------------------
                    FreeBSD Disklabel Editor

Disk: ad0       Partition name: ad0s2   Free: 22125000 blocks (11000MB)


Part      Mount          Size Newfs   Part      Mount          Size Newfs
----      -----          ---- -----   ----      -----          ---- -----



The following commands are valid here (upper or lower case):
C = Create        D = Delete   M = Mount pt.   W = Write
N = Newfs Opts    Q = Finish   S = Toggle SoftUpdates
T = Toggle Newfs  U = Undo     A = Auto Defaults

Use F1 or ? to get more help, arrow keys to select.

--------------------------------------------------------------------------
```

FIG: (E)

Per capire cosa si sta facendo, basta vedere l'elenco delle partizioni che si andranno a creare:

`/` = 120Mb
`SWAP` = 256Mb
`/var` = 256Mb
`/home` = 800Mb
`/usr` = tutto lo spazio rimanente (9309 Mb circa)

Mantenendo selezionata la prima riga in alto, ossia lo slice appena creato: "*Disk: ad0 Partition name: ad0s2 Free: 0 blocks (0MB)*" si possono creare le labels premendo "*C*" (create). Viene proposto un numero, che equivale allo spazio totale dello slice stesso. A questo punto basta cancellare il numero proposto e scrivere "*120M*", premere Invio; vengono poi proposte 2 opzioni: "*FS A file system*" oppure "*Swap A swap partition*" in questo modo:

```
??????????????????? Please choose a partition type ???????????????????
? If you want to use this partition for swap space, select Swap. ?
? If you want to put a filesystem on it, choose FS.              ?
? ??????????????????????????????????????????????????????????? ?
? ?                  FS    A file system                     ? ?
? ?                  Swap  A swap partition.                 ? ?
? ??????????????????????????????????????????????????????????? ?
?????????????????????????????????????????????????????????????????
?                      [  OK  ]       Cancel                     ?
?????????????????????????????????????????????????????????????????
```

FIG: (F)

Selezionare "***FS***"; verrà proposta una finestra con la domanda: "*Please specify a mount point for the partition*". Inserire qui il simbolo "/" (ossia la root di inizio del filesystem). Ora , sempre con la prima riga in alto selezionata, premere nuovamente "*C*", viene proposto il solito numero in blocchi (meno i 120 MB creati poco fa della `/`): basta cancellarlo e scriverci al suo

interno "*256M*". Dare Invio, e il programma chiederà ancora se è FS oppure swap. Questa volta bisogna inserire "***SWAP***".
N.B. Da notare che lo swap dovrebbe essere circa il doppio della RAM. Premere ancora "*C*", cancellare il numero, inserire "*256M*", settare "*FS*" e scriverci "**/var**". Premiere ancora "*C*", cancellare il numero, inserire "*800M*", settare "*FS*" e scriverci che è "**/home**". Premere ancora "*C*", tenere stavolta tutto il numero proposto, settare "*FS*" e scriverci che è "**/usr**".

Ora, restando ancora in "*disklabels*", selezionare con il cursore (tasti freccia) le labels (partizioni) relative ad "*FS*" appena create (swap esclusa) e premere su di esse ad una ad una il tasto "*S*". In questo modo si settano le *SOFTUPDATES*, una sorta di journaled filesystem (non è esatto chiamarlo così, ma è un software per "aiutare" il sistema in caso di crash o di mancanza di corrente). Premere "*Q*" ed uscire da disklabel.

Qui si è settato parecchio spazio per la FreeBSD, ma un Gigabyte o anche meno (600-800 Mb) possono bastare ad installare una FreeBSD (anche se 1.5 - 2 Gb sono ottimali). In questo caso si consiglia di NON creare tutte le partizioni appena fatte (cioè `/`, `/var` , `/home` , `/usr`), ma un unico filesystem in "/" (solo la / root) più lo swap. Quindi in totale si consiglia di creare 2 labels: / + swap.

D.2.1 Cosa installare

Si è ritornati nuovamente nel menu "*custom*" (FIG. (A)): l'opzione seguente a "*label*" è "*Distributions*" (*Select distribution(s) to extract*). Premere Invio per entrarci. La schermata è la seguente:

```
??????????????????????????????Choose Distributions??????????????????????????????
? As a convenience, we provide several "canned" distribution sets.         ?
? These select what we consider to be the most reasonable defaults         ?
? for the type of system in question.  If you would prefer to pick and     ?
? choose the list of distributions yourself, simply select "Custom".  You ?
? can also pick a canned distribution set and then fine-tune it with the   ?
? Custom item.                                                             ?
?                                                                          ?
? Choose an item by pressing [SPACE] or [ENTER].  When finished,           ?
? choose the Exit item or move to the OK button with [TAB].                ?
????????????????????????????????????????????????????????????????????????????
? ? <<< X Exit              Exit this menu (returning to previous)      ? ?
? ?     All                All system sources, binaries and X Window    ? ?
? ?     Reset               Reset selected distribution list to nothing ? ?
? ? [ ] 4 Developer         Full sources, binaries and doc but no games ? ?
? ? [ ] 5 X-Developer       Same as above + X Window System             ? ?
? ? [ ] 6 Kern-Developer    Full binaries and doc, kernel sources only  ? ?
? ? [ ] 7 X-Kern-Developer  Same as above + X Window System             ? ?
? ? [ ] 8 User              Average user - binaries and doc only        ? ?
??????v(+)??????????????????????????????????????????????????????????????????
????????????????????????????????????????????????????????????????????????????
?                            [  OK  ]       Cancel                         ?
??????????????[ Press F1 for more information on these options.]?????????
```

FIG: (G)

Se si possiede tutto lo spazio selezionare "*All*" e premere la barra spaziatrice. Se non se ne possiede moltissimo (meno di 1 - 1.5 Giga Byte) selezionare ugualmente "*All*", poi entrare in :

```
    > > B Custom            Specify your own distribution set
```

A questo punto è consigliabile deselezionare i seguenti *item*:

```
[X]compat1x  FreeBSD 1.x binary compatibility
[X]compat20  FreeBSD 2.0 binary compatibility
[X]compat21  FreeBSD 2.1 binary compatibility
[X]compat22  FreeBSD 2.2.x and 3.0 a.out binary compatibility
[X]compat3x  FreeBSD 3.x binary compatibility
[X]crypto    Basic encryption services
[X]krb4      KerberosIV authentication services
[X]dict      Spelling checker dictionary files
[X]games     Games (non-commercial)
```

Ora si può tornare nuovamente al menu "*custom*" come da (FIG. (A)) selezionando per due volte "*Exit*". Probabilmente il tool di installazione chiederà se si vogliono installare i "PORTS": rispondere "*Yes*". In questo modo si è selezionato il software di "base" della FreeBSD da installare. Il software extra come windows managers, programmi vari, browser web ecc... lo si può installare in seguito.

Selezionare "*Media*" (*Choose the installation media type*) e si aprirà la schermata:

```
??????????????????????? Choose Installation Media ?????????????????????
? FreeBSD can be installed from a variety of different installation    ?
? media, ranging from floppies to an Internet FTP server.  If you're   ?
? installing FreeBSD from a supported CD/DVD drive then this is        ?
? generally the best media to use if you have no overriding reason for ?
? using other  media.                                                  ?
?                                                                      ?
????????????????????????????????????????????????????????????????????????
? ?   1 CD/DVD       Install from a FreeBSD CD/DVD                  ? ?
? ?   2 FTP          Install from an FTP server                     ? ?
? ?   3 FTP Passive  Install from an FTP server through a firewall   ? ?
? ?   4 HTTP         Install from an FTP server through a http proxy ? ?
? ?   5 DOS          Install from a DOS partition                   ? ?
? ?   6 NFS          Install over NFS                               ? ?
? ?   7 File System  Install from an existing filesystem            ? ?
? ?   8 Floppy       Install from a floppy disk set                 ? ?
? ?   9 Tape         Install from SCSI or QIC tape                  ? ?
? ?   X Options      Go to the Options screen                       ? ?
? ??????????????????????????????????????????????????????????????????? ?
????????????????????????????????????????????????????????????????????????
?                          [  OK  ]       Cancel                      ?
??????[ Press F1 for more information on the various media types ]??????
```

FIG (H)

Selezionare la prima opzione tramite Invio:

```
1 CD/DVD Install from a FreeBSD CD/DVD
```

Se si possiede un solo lettore di CDROM, nessun problema; se ne sono presenti 2 si presenta ancora una videata:

```
??????????????????????????Choose a CD/DVD type????????????????????????????
? FreeBSD can be installed directly from a CD/DVD containing a valid    ?
? FreeBSD distribution.  If you are seeing this menu it is because      ?
? more than one CD/DVD drive was found on your system.  Please select   ?
? one of the following CD/DVD drives as your installation drive.        ?
? ?????????????????????????????????????????????????????????????????? ?
? ?                         acd0c  ATAPI/IDE CDROM                    ? ?
? ?                         acd1c  ATAPI/IDE CDROM                    ? ?
? ?????????????????????????????????????????????????????????????????? ?
????????????????????????????????????????????????????????????????????????
?                           [  OK  ]       Cancel                       ?
??????????????[ Press F1 to read the installation guide ]???????????????
```

FIG (I)

che fa scegliere il lettore di cdrom dove al momento è presente il disco della FreeBSD che si sta installando:

```
/dev/acd0c       ATAPI/IDE CDROM disco slave sul primo canale IDE
/dev/acd1c       ATAPI/IDE CDROM disco slave sul secondo canale IDE
```

Selezionare (solitamente è il primo disco che fa bootare il bios) "*acd0c*" e premere Invio. Si è ancora nel menu "*custom*" come da (FIG (A)).

D.2.2 Punto di non ritorno

Entrare in "*Commit*" (*Perform any pending Partition/Label/Extract actions*) e premere Invio: il tool di installazione farà la domanda "*ARE YOU SURE* ecc.." , ossia chiede se si è sicuri di installare. Prima di premere Invio in questa schermata nessuna modifica è stata scritta sull'Hard Disk! Premere il tasto Invio. Se tutto è stato fatto correttamente partirà l'installazione, il disco verrà formattato, il bootloader eventuale verrà collocato sull'MBR, e il software viene installato.

D.2.3 Configurazione

Una volta che il software ha terminato la sua installazione, il cdrom è fermo ed è tornata la schermata di installazione: tornare alla primissima videata, quella di (FIG A1).
Qui bisogna settare le opzioni di installazione; quindi nella schermata di FIG. A1 selezionare il menu "*Configure*" (*Do post-install configuration of FreeBSD*) e si entrerà nella schermata seguente:

```
??????????????????????? FreeBSD Configuration Menu ?????????????????????????
? If you've already installed FreeBSD, you may use this menu to customize ?
? it somewhat to suit your particular configuration.  Most importantly,   ?
? you can use the Packages utility to load extra "3rd party"              ?
? software not provided in the base distributions.                        ?
? ?????????????????????????????????????????????????????????????????????? ?
? ? X Exit            Exit this menu (returning to previous)           ? ?
? ?  Distributions    Install additional distribution sets             ? ?
? ?  Packages         Install pre-packaged software for FreeBSD        ? ?
? ?  Root Password    Set the system manager's password                ? ?
? ?  Fdisk            The disk Slice (PC-style partition) Editor       ? ?
? ?  Label            The disk Label editor                            ? ?
? ?  User Management  Add user and group information                   ? ?
? ?  Console          Customize system console behavior                ? ?
? ?  Time Zone        Set which time zone you're in                    ? ?
? ?  Media            Change the installation media type               ? ?
? ?  Mouse            Configure your mouse                             ? ?
? ?  Networking       Configure additional network services            ? ?
? ??v(+)???????????????????????????????????????????????????????????? ?
????????????????????????????????????????????????????????????????????????
?                          [  OK  ]       Cancel                         ?
???????????[ Press F1 for more information on these options ]?????????????
```

FIG A2

Per cominciare settare "*Root password*": il sistema la farà scrivere 2 volte e NON visualizzerà (per sicurezza) nulla a video.

Selezionare "*Console*" (*Customize system console behavior*), entrare nella schermata di sottomenu e settare nuovamente "*keymap*" con "*italian*", "*Repeat*" (si consiglia "*fast*" o "*normal*"), "*Saver*" *6 Logo* "*BSD Daemon" animated screen saver (graphics)*. Andare su "*Exit*" e tornare al menu di (FIG A2).

Selezionare "*Time zone*". Dare Invio su "*NO*" alla prima domanda, poi settare "*Europe*" e quindi "*Italy*",e all' ultima domanda: "*Does the abbreviation CET look reasonable??*" dare Invio su "*YES*".

Selezionare "*Mouse*" (*Configure your mouse*), entrare nella schermata di sottomenu e (esempio per un mouse PS2 a 2 tasti) settare "*Type*" su "*Auto*", "*Port*" su "*PS/2*", dentro "*Flags*" scrivere " *-3* " (questo solo se è a 2 tasti; se fosse a 3 tasti non si scriverebbe nulla). Andare su "*Enable*", dove si può testare e poi andare su "*Exit*". Si è ancora alla schermata di (FIG: A2).

"*Networking*": E’ consigliabile lasciare tutto come è, ed in più se si vuole si può selezionare:

```
Sendmail   (dovrebbe essere già settato)
TCP-Extension
AMC Flags (dovrebbe essere già settato)
Inetd   (dovrebbe essere già settato)
Interfaces : se si hanno schede di rete da settare bisogna farlo QUI!
             chiede che scheda è (es. rl0 per realtek),se si vuole usare
             IPV6 selezionare yes, DHCP no. Poi si presenta la schermata
             di settaggio:
      "host": nome.my.domain
      "Domain": my.domain
      "ipv4 gateway": GW oppure ip della macchina(es. 192.168.0.1)
      "name server" : server DNS del proprio provider
      "IPV4 address" : ad es.  192.168.0.1
      "netmask" : 255.255.255.0
      "extra options to ifconfig":
```

Tornare al menu di FIG. A2, andare su "*XFree86*" (*Configure XFree86 Server*), entrare e settare il server X (per il momento quello di default è XFree86 v.3.3.6, ma va ottimamente) con "*XF86Setup*" (tool grafico molto intuitivo e utilizzato anche su Linux). In alternativa c'è il tool testuale "*xf86config*".

Una volta configurato e testato XFree86, andare sul menu "*Desktop*" (*Configure XFree86 Desktop*), entrarci e selezionare il windows manager preferito. Se si possiede lo spazio si può tranquillamente installare "*KDE2*" o "*GNOME+Sawfish*". Se si possiede invece poco spazio o una macchina lenta si consiglia di optare per il veloce "*Windowmaker*" (oppure i più obsoleti "*fvwm*" e "*Afterstep*"). In un secondo tempo volendo si possono installare più windows manager contemporaneamente. Una volta selezionato il Windows Manager preferito andare su "*Exit*". A questo punto si dovrebbe percepire che il tool di installazione fa partire il CD-ROM per installare il desktop scelto.

Entrare ora sempre dal menu di FIG. A2 in "*Startup*" (*Configure system startup options*) e selezionare oltre alle cose già selezionate, le opzioni:

```
[X]  lpd            This host has a printer and wants to run lpd.
[X]  linux          This host wants to be able to run linux binaries.
```

Andare su "*Exit*". Il sistema ora sta installando i binari di compatibilità cioè una sorta di emulatore di Linux.

Ora la macchina è dotata di una FreeBSD4.x (4.4 o 4.5 se ci si è dotati di dischi recenti o si è avuta la possibilità di scaricare dal web una immagine ISO). Andare su "*Exit*" dal menu di FIG.A2, poi andare quindi nel menu di FIG. A1, selezionare con tasto TAB "*Exit Sysinstall*" oppure con il tasto "*X*" per riavviare la macchina.

D.2.4 Aggiustamenti dopo l’installazione

Se si è installato il BSD-Bootloader, FreeBSD si avvierà subito, altrimenti se si possiede già una distribuzione Linux installata bisogna avviarla come al solito con LILO che è già installato, loggarsi come root, aprire con un editor di testo il file:

`/etc/lilo.conf`

ed aggiungere in fondo al file (ammesso ad esempio che si sia installato FreeBSD su `/dev/ad0s2` e Linux su `/dev/hda2`) le righe:

```
other=/dev/hda2
      label=FreeBSD
      table=/dev/hda
```

Salvare il file `/etc/lilo.conf`, e digitare con i permessi di root il comando:

```
#lilo -v
```

che serve per aggiornare l’MBR del disco fisso.
Se l'output di "**`lilo -v`**" non ha dato errori, vuol dire che tutto va bene. A questo punto si può riavviare la macchina e far partire FreeBSD dal prompt di LILO selezionando la voce "*FreeBSD*".

Appendice E

LILO

LILO (*LInux LOader*) è una procedura molto comune per il caricamento di Linux negli elaboratori con architettura i386. Permette di avviare anche altri sistemi operativi eventualmente residenti nello stesso elaboratore in cui si usa Linux. In questa sezione si vedono solo alcuni aspetti del suo funzionamento, quelli che dovrebbero bastare nella maggior parte delle situazioni.

E.1 Organizzazione essenziale

La procedura LILO è 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 Linux, sia per altri sistemi operativi eventuali. Può contenere anche il file del kernel, o più file di kernel differenti, quando per questo non si usa semplicemente la directory radice. Più precisamente, contiene almeno i file seguenti:

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

Nella tabella seguente sono elencati i codici esadecimali corrispondenti ad alcuni dispositivi per le unità di memorizzazione:

Dispositivo	**Codice**	**Dispositivo**	**Codice**
/dev/fd0	200	/dev/hdb2	342
/dev/fd1	201	/dev/sda	800
/dev/hda	300	/dev/sda1	801
/dev/hda1	301	/dev/sda2	802
/dev/hda2	302	/dev/sdb	800
/dev/hdb	340	/dev/sdb1	801
/dev/hdb1	341	/dev/sdb2	802

1024 cilindri
Quando si utilizza l'architettura i386, il firmware, cioè il BIOS, solitamente non è in grado di accedere a settori oltre il 1024-esimo cilindro (cioè oltre il cilindro numero 1023). Di conseguenza, il programma che si occupa di caricare il kernel di qualunque sistema operativo, si deve avvalere delle funzioni del BIOS (perché inizialmente non c'è ancora un sistema operativo funzionante) e quindi non può raggiungere file oltre quel limite dei 1024 cilindri.
La directory `/boot/` (con tutto il suo contenuto) e il kernel, devono trovarsi fisicamente entro il 1024-esimo cilindro. Non basta che la partizione inizi entro il limite per garantire che questi file si trovino effettivamente in quella zona. In caso di necessità, si può utilizzare una partizione apposita per questi file, nella parte sicura del disco. È poi sufficiente montare questa partizione nel file system generale, eventualmente riproducendo la directory `/boot/` attraverso un semplice collegamento simbolico.

E.2 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 ciò viene richiesto; nel terzo caso si utilizza un dischetto in modo da non alterare il sistema di avvio già 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, è necessario ripetere l'installazione attraverso l'eseguibile `lilo`.

E.2.1 /etc/lilo.conf

`/etc/lilo.conf` è 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 è riservata al DOS e la seconda a Linux. L'esempio permette di avviare 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'unità 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 sì 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 alcunché, 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 è sempre così. Se si vuole preparare un sistema di avvio per un sistema Linux residente in un'altra partizione (magari un dischetto), si vuole forse fare riferimento a un kernel che si trova lì. La cosa potrebbe non essere tanto intuitiva e si consiglia di documentarsi in rete.

- `label=linux`
  Definisce il nome utilizzato per fare riferimento a questo kernel. Poteva essere qualunque cosa, in questo caso il nome `linux` è 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 sì 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 è 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` è utile per ricordare che si tratta dell'avvio di quel sistema operativo.

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

E.2.2 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`.

Opzioni principali:

`-C` *file_di_configurazione*
Permette di indicare un file di configurazione differente rispetto al solito `/etc/lilo.conf`.

`-v`
Rende il comando lilo *verbose* (cioè in grado di far vedere all'utente cosa sta facendo il comando stesso).

Esempi:

`#` **`lilo -C ./mio.conf`**
Installa il sistema di avvio utilizzando la configurazione del file `mio.conf` contenuto nella directory corrente.

E.3 Configurazione di LILO più in dettaglio

Riassumendo, LILO è uno dei sistemi di avvio di kernel Linux e di altri sistemi operativi, specifico per gli elaboratori di architettura i386. Il suo file di configurazione è `/etc/lilo.conf`.
Solitamente, il file di configurazione viene creato in modo predefinito già in fase di installazione, utilizzando opzioni generiche.

Nella sostanza le direttive di configurazione hanno la forma di assegnamenti a variabili, intese come opzioni che hanno un ruolo nella fase di avvio del sistema. A parte il caso delle righe bianche e di quelle vuote, che vengono ignorate, oltre alla possibilità di indicare dei commenti preceduti dal simbolo `#`, si usa la sintassi seguente:

```
nome= valore_assegnato
nome="valore_assegnato"
opzione_booleana
```

In particolare:

- ogni direttiva deve essere disposta su una riga propria;
- a seconda del contesto, i valori assegnati possono essere sensibili alla differenza tra maiuscole e minuscole;
- è ammissibile l'uso di uno spazio, prima e dopo il simbolo = che rappresenta l'assegnamento, ma in generale si preferisce ometterlo;
- se si deve assegnare una stringa contenente uno o più spazi, occorre racchiuderla tra virgolette;
- alcune direttive rappresentano un'opzione booleana, per cui è sufficiente annotarne il nome senza alcun assegnamento, per indicare implicitamente l'abilitazione dell'opzione relativa;
- gli assegnamenti che non si possono ricondurre a direttive di configurazione, vengono intesi come assegnamenti a variabili di ambiente che poi sono passate al processo iniziale, tali e quali, rispettando anche l'uso delle lettere maiuscole o minuscole.

Le direttive di configurazione sono organizzate in sezioni: quelle della parte iniziale rappresentano la configurazione generale, mentre le sezioni specificano le particolarità delle voci che si possono selezionare nel momento dell'avvio del sistema operativo.

E.3.1 Direttive di configurazione globale

Le direttive che appaiono all'inizio del file di configurazione, prima della dichiarazione delle sezioni specifiche, riguardano tutte le sezioni sottostanti. Implicitamente appartengono alla sezione globale che non viene dichiarata espressamente.

Nel seguito vengono descritte alcune di queste.

- `backup=`*file*
  `force-backup=`*file*
  La prima delle due direttive, fa sì che nel momento in cui si installa il nuovo settore di avvio, venga fatta una copia di quello vecchio nel file specificato, a meno che il file in questione ci sia già, nel qual caso la copia non viene rifatta. In alternativa, la seconda direttiva non tiene conto dell'esistenza o meno del file, che eventualmente viene sovrascritto.

- `boot=`*file_di_dispositivo*
  Indica il nome del file di dispositivo nel quale installare il settore di avvio. In generale si tratta del file di dispositivo corrispondente a tutto il primo disco, `/dev/hda`, altrimenti, specie se si tratta di una partizione, significa che deve essere poi un altro sistema di avvio a prendersi carico dell'avvio di questo settore particolare.

- `compact`
  Cerca di riunire le richieste di lettura relative a settori adiacenti in un'unica operazione, allo scopo di ridurre il tempo necessario a caricare il sistema operativo. L'uso dei questa direttiva è particolarmente utile nella realizzazione di dischetti di avvio.

- `default=`*riferimento_alla_sezione_predefinita*
  Permette di definire quale voce selezionare in modo predefinito, tra quelle disponibili, in mancanza di una scelta precisa da parte dell'utente. Il nome che viene assegnato si riferisce a quanto dichiarato all'interno delle sezioni con la direttiva `IMAGE=`*nome*.

- `delay=`*decimi_di_secondo*
  Permette di specificare un ritardo, espresso in decimi di secondo, prima di avviare il sistema. Potrebbe essere necessario in alcune situazioni particolari, per dare il tempo a qualche componente fisica dell'elaboratore di inizializzarsi.

- `fix-table`
  Questa opzione booleana, consente la correzione automatica della tabelle delle partizioni, all'inizio delle partizioni stesse, nel caso queste non corrispondano allo standard normale. Questa opzione può creare dei disguidi se nel disco sono installati altri sistemi operativi con le loro convenzioni particolari.

- `ignore-table`
  Con questa opzione booleana si fa in modo che vengano ignorate eventuali anomalie nella tabelle delle partizioni.

- `install=`*file*
  Con questa direttiva si specifica esplicitamente il nome del file contenente il settore di avvio da installare. Se non si indica questa direttiva, viene usato in modo predefinito il file `/boot/boot.b`.

- `keytable=`*file*
  Questa direttiva stabilisce una rimappatura della tastiera secondo la codifica riportata nel file indicato. Il file in questione deve essere generato appositamente, tenendo conto della mappa di partenza (quella del BIOS) e di quella di destinazione.
  Per ottenere questo file, si utilizza un programma che fa parte del pacchetto che compone LILO: può trattarsi di `keytab-lilo.pl` o di `keytab-lilo`. Questo utilizza le mappe di definizione della tastiera di un sistema Linux normale, per generare ciò che serve.

- `map=`*file*
  Specifica la posizione e il nome del file contenente la mappa necessaria per raggiungere il kernel e altre informazioni indispensabili all'avvio. Se non si indica esplicitamente tale direttiva, viene creato e usato il file `/boot/map` in modo predefinito.

- `message=`*file*
  Indica un file di testo contenente un messaggio che deve essere visualizzato all'avvio, prima dell'invito di LILO. La lunghezza massima del testo è di 65535 byte; in particolare, il carattere <*FF*> (che si ottiene normalmente con la combinazione [`Ctrl+l`]), genera una ripulitura dello schermo. È importante sottolineare che lo spostamento o la modifica di questo file richiede la ricostruzione del file di mappa, ovvero `/boot/map`.

- `nowarn`
  Disabilita l'emissione di messaggi di avvertimento.

- `prompt`
  Richiede la comparsa dell'invito. Di solito si usa questa direttiva assieme a `timeout`, per fissare un tempo massimo oltre il quale viene selezionata automaticamente la voce predefinita.

- `timeout=`*decimi_di_secondo*
  Questa direttiva stabilisce un tempo di attesa, espresso in decimi di secondo, per la selezione di una voce di avvio attraverso la tastiera, trascorso il quale viene scelta automaticamente quella predefinita (che può essere la prima, oppure quella dichiarata con la direttiva `default`). Lo zero indica di non attendere alcunché, mentre il valore -1 stabilisce un tempo indefinito. Se non si stabilisce questa direttiva, il tempo predefinito per la pausa è di cinque secondi, pari al valore 50.

- `verbose=`*n*
  Permette di stabilire il livello di dettaglio desiderato per le informazioni emesse dall'eseguibile `lilo`. Si usano valori numerici interi, generalmente da zero a cinque, dove il valore più alto fornisce informazioni maggiori.

E.3.2 Direttive globali e specifiche

Un gruppo di direttive particolari, può essere usato sia in modo particolare, all'interno di sezioni che riguardano le varie voci di avvio, oppure anche in modo globale, prima della dichiarazione di tali sezioni, dove rappresentano l'impostazione predefinita nel caso non siano utilizzate nuovamente nelle sezioni:

- `append=`*parametri_di_avvio_del_kernel*
  Aggiunge la stringa indicata tra i parametri del kernel.

- `read-only`
- `read-write`
  La prima direttiva specifica che in fase di avvio il file system deve essere montato in sola lettura. Ciò è necessario per la verifica e l'eventuale riparazione del file system, quando successivamente il sistema provvede automaticamente a rimontarlo in lettura e scrittura. La seconda direttiva specifica che il file system deve essere montato sia in lettura che in scrittura

- `root=`*file*
  Indica il file di dispositivo che deve essere montato come file system principale. Se non si utilizza questa direttiva, si intende implicitamente che si tratti della partizione o del disco in cui si trova già il file del kernel.

- `vga=`**{**`normal`|`extended`|`ask`|*n***}**
  Specifica la modalità video VGA che deve essere impostata all'avvio. La parola chiave `normal` richiede espressamente la modalità testo normale, pari a 80x25; `extended` richiede la modalità testo 80x50; `ask` fa in modo che venga richiesto all'utente in fase di avvio; infine, un valore numerico corrisponde a una scelta equivalente dal menù che si otterrebbe con l'opzione `ask`.

- `lock`
  Questa direttiva abilita la registrazione della riga di comando utilizzata all'avvio, relativa alla propria voce di avvio, allo scopo di riutilizzarla in modo predefinito negli avvii successivi.

- `password=`*parola_d'ordine*
  Fa in modo che venga richiesta la parola d'ordine indicata per poter procedere. Naturalmente, occorre tenere presente che il file di configurazione contenente tale informazione, dovrebbe essere protetto in qualche modo, almeno dagli accessi di utenti diversi dall'amministratore.

- `restricted`
  Questa direttiva può essere usata solo assieme a `password` e serve a stabilire che la richiesta di tale parola d'ordine avviene solo nel caso di inserimento di parametri di avvio per il kernel.

- `single-key`
  La direttiva `single-key` consente di avviare un'immagine con la pressione di un solo tasto, senza l'aggiunta di un [`Invio`] finale. Per ottenere questo risultato, si può fare in modo che le varie direttive `label` definiscano dei nomi composti da un solo carattere, oppure si aggiunge alla direttiva `label` la direttiva `alias`, dove però si deve specificare un carattere differente dall'iniziale usata nel nome abbinato a `label`. In questo senso, è comune utilizzare delle direttive `alias` contenenti solo un numero.
  L'avvio attraverso la pressione di un tasto singolo, impedisce l'inserimento di parametri per il kernel. Di conseguenza, per poter selezionare l'avvio, sia con un tasto singolo che con un nome, si usano sia le direttive `label` che le direttive `alias`, con l'accorgimento di non ripetere le iniziali.

E.3.3 Sezioni delle voci di avvio

Le voci selezionabili all'avvio, sono descritte all'interno di sezioni, che hanno la stesso aspetto delle direttive normali. Si tratta precisamente di queste due direttive:

`image=`*file_immagine_del_kernel_da_avviare*

`other=`*file_di_dispositivo*

Nel primo caso si fa riferimento a una sezione relativa a una voce di avvio per un kernel Linux; nel secondo si tratta dell'avvio di un altro settore di avvio, presumibilmente di un sistema operativo diverso da Linux.
Tutte le direttive successive a una di queste due, fino alla dichiarazione di una sezione successiva eventuale, rappresentano impostazioni particolari. In questo ambito, si possono indicare le direttive già descritte in precedenza, tranne quelle di competenza esclusivamente globale, oltre a quelle che vengono descritte qui in particolare.

- `label=`*nome*
  Indica il nome attribuito a questa voce di avvio, che potrà essere selezionato al momento dell'invito.

- `alias=`*nome*
  Specifica un nome alternativo per la voce a cui si riferisce.

- `loader=`*file*
  Si usa nell'ambito di una sezione `other`, per indicare il file contenente il codice necessario per il caricamento di un settore di avvio successivo. In condizioni normali si tratta del file `/boot/chain.b`, che viene utilizzato in modo predefinito quando non si specifica questa direttiva.

- `table=`*file_di_dispositivo*
  Specifica, attraverso il file di dispositivo corrispondente, la tabella di partizione relativa al sistema operativo che si intende avviare. Si usa di solito nelle sezioni `other`, quando non si tratta dell'avvio di Linux.

E.4 Esempio

L'esempio seguente può avviare un sistema Linux in due modi differenti, attraverso il file `/boot/vmlinuz` e `/boot/vmlinuz.1`, oppure un altro sistema operativo (in questo caso si tratta di Windows).
La presenza di direttive `alias`, fa sì che si possano selezionare le voci per nome, potendo così aggiungere anche dei parametri per il kernel, oppure attraverso una sola cifra numerica.
Si può osservare che la voce `linux`, ovvero `1`, richiede l'inserimento di una parola d'ordine nel caso si vogliano inserire dei parametri di avvio; inoltre, nel caso della voce `prova`, ovvero `2`, è impedito l'inserimento di parametri di avvio, attraverso la direttiva `lock`.

```
boot=/dev/hda
vga=normal
read-only
prompt
timeout=-1
single-key
message=/boot/message

image=/boot/vmlinuz
  label=linux
  alias=1
  root=/dev/hda4
  initrd=/boot/initrd
  password=segreto
  restricted
image=/boot/vmlinuz.1
  label=prova
  alias=2
  root=/dev/hda4
  initrd=/boot/initrd.1
  lock
other=/dev/hda1
  label=windows
  alias=3
  table=/dev/hda
```

Nell’utilizzo di un utente comune, difficilmente si trovano file di configurazione di questo genere. Di solito, infatti, per il normale utilizzo di Linux sul proprio computer domestico, la

complessità del file di configurazione non supera quella vista al paragrafo riguardante `/etc/lilo.conf`.

E.5 Rimuovere LILO

Se si è installato LILO su un driver contenente un sistema DOS o Windows, si può sempre ripristinare il boot sector con il comando:

```
fdisk /MBR
```

dove MBR sta per Master Boot Record. Tale operazione può essere fatta utilizzando ad esempio un disco di ripristino di Windows in cui sia presente l'eseguibile `fdisk`.