

UNIX-like for dummies

di Vittorio Romolini
versione 2008-03-05 n° 4

Indice

Indice	3
Introduzione ed installazione	5
Cos'è UNIX, cosa sono Linux e BSD, cosa sono le distribuzioni	5
L'utente ed il sistema di riferimento	5
Sistemi con/senza interfaccia grafica	5
Come ottenere il sistema operativo ed installarlo	6
Il partizionamento dei dischi rigidi	6
Installare altri programmi	7
Iniziamo a sporcarci le mani	8
Il boot-loader	8
Login e logout	8
Console e terminali	8
Accreditarsi come amministratori	9
Il manuale	9
Opzioni dei comandi	10
Terminare l'OS	10
Creare alias	10
Applicazioni e comandi vari	10
Files e redirezione	13
L'organizzazione dei files	13
Operazioni di base su files e directories	13
Cercare files	15
Ridirigere gli input/output/error	16
Ancora redirezione: la stampa su carta	16
Introduzione ai permessi in UNIX	17
Altri due permessi: s, t	17
Impostare il proprietario ed i permessi di un file	17
I flag dei files	18
Utenti, VI, script ed archiviazione	19
Gestione degli utenti	19
L'editor di testi VI	19
Script per la shell	21
Archiviazione: compressione e masterizzazione	22
Mounting e gestione di partizioni	23
Cos'è ed a cosa serve il mount	23
Le 4 fette, e terminologia nei diversi sistemi	23
I nomi delle partizioni di disco in UNIX	23
I file-systems montati automaticamente al boot	24
Creare, montare e smontare file-systems	25
Controllare e partizionare dischi	25
Mount di file-system remoti	26
Processi	28
Visualizzazione dei processi e delle loro informazioni	28
Invio di segnali	28
Processi in background e foreground	28
Gestione della priorità	29
Lanciare programmi all'avvio	29
Networking	30
Conoscere e configurare le interfacce	30
Wireless	31
Reti wireless protette	31
DSL	32
Nomi di host e DNS	32
Ping, traceroute, tabella di routing e tabella di ARP	33
Statistiche di rete	33
Sniffing	33
Firewall	34
Applicazioni varie di rete	35
Bibliografia e fonti varie	36
Registro delle modifiche (changelog)	37

Introduzione ed installazione

Cos'è UNIX, cosa sono Linux e BSD, cosa sono le distribuzioni

UNIX è un sistema operativo (cioè un programma che gestisce le risorse hardware e software del computer).

Linux e BSD sono anch'essi sistemi operativi, e sono considerati UNIX-like poiché derivati da UNIX.

Più in dettaglio, Linux è il kernel (nucleo) di sistema operativo, cioè è un programma non direttamente utilizzabile da un utente umano ma bensì da altri software che costituiscono appunto l'interfaccia tra il kernel e l'utenza.

Le distribuzioni Linux sono sistemi operativi con kernel Linux al quale sono affiancati numerosi altri programmi che ne consentono l'utilizzo. Ne esistono di numerose: Mandriva, Suse, Ubuntu, e tante altre.

Anche i sistemi BSD sono numerosi; tra essi, i più noti sono FreeBSD e OpenBSD.

I sistemi operativi (che spesso saranno brevemente chiamati OS, cioè Operating Systems) derivati da UNIX sono quindi numerosi e diversificati tra loro, ma la loro comune origine attribuisce ad essi un elevato numero di caratteristiche condivise. I comandi descritti in questo testo sono dunque da intendersi generalmente utilizzabili nei sistemi derivati da UNIX, salvo poi alcune sporadiche differenze tra le numerose distribuzioni.

L'utente ed il sistema di riferimento

Questa guida è pensata e realizzata innanzitutto per l'utente principiante che si trova per a voler/dover usare un sistema operativo UNIX-like ma non sa minimamente dove mettere le mani.

Durante la realizzazione di questo testo ho adoperato Linux Mandrake 10.1, FreeBSD 4.8 e soprattutto Linux Ubuntu (versioni 6 e 7) che consiglio come banco di prova per via della sua semplicità d'utilizzo.

Sistemi con/senza interfaccia grafica

Una interfaccia grafica per l'utente (GUI, Graphic User Interface) è l'insieme di pulsanti, menù, icone, immagini ed altri elementi grafici che consentono all'utente di interagire molto semplicemente con il sistema operativo.

In un computer dove gira un OS senza interfaccia grafica, è possibile impartire comandi solo in forma testuale (avrà probabilmente visto schermi di PC con sfondo nero dove sono visualizzate solo scritte classicamente bianche o verdi). In questo primo momento si può chiamare l'ambiente testuale come "console": in seguito l'argomento sarà affrontato più in dettaglio.

In un'ambiente con interfaccia grafica, invece, è possibile "far fare qualcosa" al computer – ad esempio – facendo doppio click su un'icona, trascinare un oggetto rappresentato sullo schermo (sia esso una finestra, l'icona di un file, un oggetto all'interno di un documento di testo verso un software di disegno, ...), eccetera.

Gli ambienti con interfaccia grafica sono stati sviluppati successivamente a quelli senza interfaccia grafica proprio per semplificare, attraverso il software, l'utilizzo dell'hardware del computer.

Gli ambienti senza interfaccia grafica sono ancora diffusi:

- sui server, che non essendo principalmente e direttamente utilizzati da persone ma bensì da altri computer che si collegano ad essi attraverso l'infrastruttura di rete, non necessitano dell'interfaccia grafica (che tralaltro richiede un maggiore impiego di risorse hardware)
- laddove sono rimasti installati (o vengono installati ex novo) vecchi sistemi operativi per necessità economiche, formative o altro.

In realtà la divisione non è così netta tra sistemi con o senza GUI:

- ogni sistema con GUI è provvisto di un emulatore della console (pensa al "Prompt dei comandi MS-DOS")
- su alcuni sistemi il programma che gestisce la GUI può essere installato, disinstallato, abilitato e disabilitato a piacere dell'utente. FreeBSD, ad esempio, all'avvio opera in modalità testuale ma, su lancio di appositi comandi (e se sono stati installati appositi files nel sistema) è possibile usare l'interfaccia grafica. Ubuntu, invece, per default è in modalità grafica, ma è sempre possibile utilizzare anche l'interfaccia testuale

Può sembrare curioso che in un sistema come FreeBSD, per default a comando testuale, sia possibile avviare l'interfaccia grafica ed all'interno di questa utilizzare un emulatore di console.



NOTA: DESKTOP MANAGER

L'elaborazione delle interfacce grafiche è gestita da appositi software chiamati genericamente "desktop manager".

I più diffusi desktop manager sono Gnome e KDE.

Occorre notare che, mentre i comandi testuali sono sostanzialmente (ma non totalmente) uguali tra tutti i sistemi UNIX-like, le interfacce grafiche sono in alcuni casi molto differenti tra loro; tuttavia due versioni dello stesso desktop manager per OS differenti saranno identiche in termini di utilizzo, per cui una volta acquisito un minimo di dimestichezza con i principali desktop-manager non dovresti avere grandi difficoltà. Questi due livelli di similitudine permettono quindi l'immediata utilizzabilità di un sistema da parte di un utente abitudinario a sfruttare le potenzialità di un altro sistema che ha però un'interfaccia analoga (sia essa testuale o grafica).

Questa guida si pone come obiettivo l'introduzione al mondo di UNIX e derivati, perciò non posso che tenere il più possibile come riferimento il minimo comun denominatore tra tutti questi OS, ed illustrare quindi i comandi lanciabili dalla console e – dove l'ho ritenuto opportuno – citare i programmi con interfaccia grafica più noti.

Nell'utilizzo quotidiano di un sistema operativo recente, utilizzerai quasi unicamente i software con GUI, ma i comandi illustrati in

questa guida ti saranno comunque utili:

- per fare ogni cosa che puoi fare tramite GUI, ma anche molto altro: infatti mentre attraverso la console puoi fare “tutto”, non è garantita la stessa potenza all'interfaccia grafica
- per fare le stesse cose anche su ambienti senza GUI, o con una GUI che non conosci, o con una GUI che comunque non ti offre quelle specifiche funzionalità che ti servono
- spesso, anche per fare queste cose in maniera più veloce ed efficiente

Come ottenere il sistema operativo ed installarlo

Potenzialmente, ogni sistema operativo può essere ottenuto ed installato in maniera differente.

Alcuni OS sono reperibili a pagamento, tuttavia:

- spesso è possibile scaricare gratuitamente dal web l'immagine del disco d'installazione, da masterizzare su un disco reale (CD o DVD a seconda dei casi)
- in alcuni casi (come ad esempio per Ubuntu attraverso il servizio on-line ShipIt) ci si può far spedire a casa in maniera totalmente gratuita il disco d'installazione (il tempo necessario alla consegna arriva fino a 6 settimane)

Se invece si dispone già di un sistema operativo recente derivato da UNIX, è probabile che offra la possibilità di autoaggiornamento.

Come per ogni sistema operativo, prima di procedere all'installazione è raccomandabile un backup dei propri dati: in alcune fasi della procedura si potrebbero cancellare erroneamente anche grandi moli di dati sensibili (ciò accade quando l'utente è distratto).

L'installazione di un sistema operativo con licenza libera è composta sostanzialmente da una serie di fasi visibili all'utente:

- la configurazione di base (gli account-utente, la data e l'ora di sistema, il tipo di tastiera, la lingua, ...)
- il partizionamento dei dischi rigidi
- la copia dei files del sistema

Se da un lato la prima parte è solitamente molto semplice ed autoesplicativa, e l'ultima è realizzata dall'installer senza necessità dell'intervento umano... la seconda fase oltre ad essere molto importante va completata con estrema attenzione leggendo scrupolosamente il prossimo paragrafo, pur tenendo presente che la prima volta che si installa un sistema operativo è meglio avere a fianco a sé qualcuno con esperienza in materia.

Nel complesso, l'installazione è comunque semplice e guidata passo-passo. Ad ogni modo in alcuni casi (come in Ubuntu) è possibile anche utilizzare il sistema operativo stesso che viene eseguito direttamente dal disco d'installazione, durante la sua copia e configurazione sul disco rigido: diventa così possibile accedere ad internet e navigare sul web per cercare informazioni su come completare l'installazione.

Sempre in merito ad Ubuntu, ti consiglio di scaricare l'ottima guida ufficiale all'utilizzo pratico della sua versione Desktop (è un file PDF linkato in <http://help.ubuntu.com>).

Il partizionamento dei dischi rigidi

Prima di tutto... alcune premesse:



NOTA: COSA SONO LE PARTIZIONI ED I FILE-SYSTEM

Un hard disk (anche noto come disco rigido, o disco fisso) è un dispositivo hardware dentro il computer, ed è finalizzato alla memorizzazione di files codificati in lunghe sequenze di 1 e di 0.

Una partizione è un'astrazione logica che semplicemente rappresenta una parte della memoria di un “supporto di memorizzazione di massa” (o “dispositivo di storage” qual'è appunto un hard disk, ma anche una penna USB, ed altri hardware ancora).

L'accesso al contenuto di una partizione avviene tramite il file-system, ossia un “modo” di memorizzare i files dentro la partizione. Esistono numerosi file-system, tra questi: gli ambienti Microsoft Windows usano soprattutto FAT32 e NTFS, mentre i sistemi Linux più recenti salvano generalmente i dati su file-system EXT3.

Attenzione (1): su ciascuna partizione c'è uno ed un solo file-system, e su ciascuna partizione/file-system si può installare al massimo un solo sistema operativo.

Attenzione (2): un sistema operativo può accedere a tutte le partizioni.. tra quelle di cui conosce il file-system! Ad esempio Windows di per sé non riesce ad accedere a partizioni EXT3 (si può supplire installando appositi driver), mentre le distribuzioni Linux accedono oltre che alle EXT3 anche alle FAT e numerosi altri file-system.

Durante l'installazione del sistema la fase di partizionamento permette di specificare quali e quante partizioni creare (e/o mantenere) sui dischi rigidi del computer. Sarà dunque possibile scegliere la soluzione più adatta alle proprie esigenze, che solitamente sono una tra le seguenti:

- installazione del solo sistema UNIX-like sul disco
- installazione del sistema UNIX-like su un disco dove si trovano già installati (e si vogliono mantenere) altri sistemi operativi

Poiché questa guida è rivolta a coloro che affrontano i primi passi in un mondo finora sconosciuto, non posso che consigliare di seguire una strategia prudente e dunque lasciare installato il proprio OS abituale (presumibilmente Microsoft Windows) su una partizione da rimpicciolire così da creare spazio per nuove partizioni dove sarà possibile installare l'OS UNIX-like.

Per illustrare la logica dei passi da fare durante il partizionamento, ecco un esempio:

- hai un disco da 100 GB (gigabytes) con due partizioni FAT32: una (chiamiamola “win_c”) contiene MS Windows, mentre l'altra (chiamiamola “win_hidden”, di circa 2 GB) ti è solitamente invisibile ma è stata creata dall'installazione di Windows stesso

- stai installando Linux Ubuntu (dunque il software di partizionamento usato è il buon Gparted) e lo vuoi copiare su una partizione con filesystem adatto (EXT3)
- vuoi creare anche una partizione (chiamiamola “win_d”) in cui tenere i tuoi files personali, accessibile sia da Windows che da Linux Ubuntu

Una importante riflessione da fare è: “Quanto spazio riservare sul disco rigido al sistema operativo, ai dati, ... ?”.

In generale devi considerare che almeno i primi tempi la necessità di spazio sul sistema UNIX-like sarà molto basso:

- il sistema operativo più i pur numerosi software aggiuntivi che porta con sé occupano, nel caso di Ubuntu 6 (uscito nel giugno 2006 e che quindi può essere assunto come limite superiore dello spazio richiesto dagli OS antecedenti a tale anno), circa 2.7 gigabytes.
- sono necessari almeno 500 MB per uno swapping ottimale.



NOTA: LO SWAP

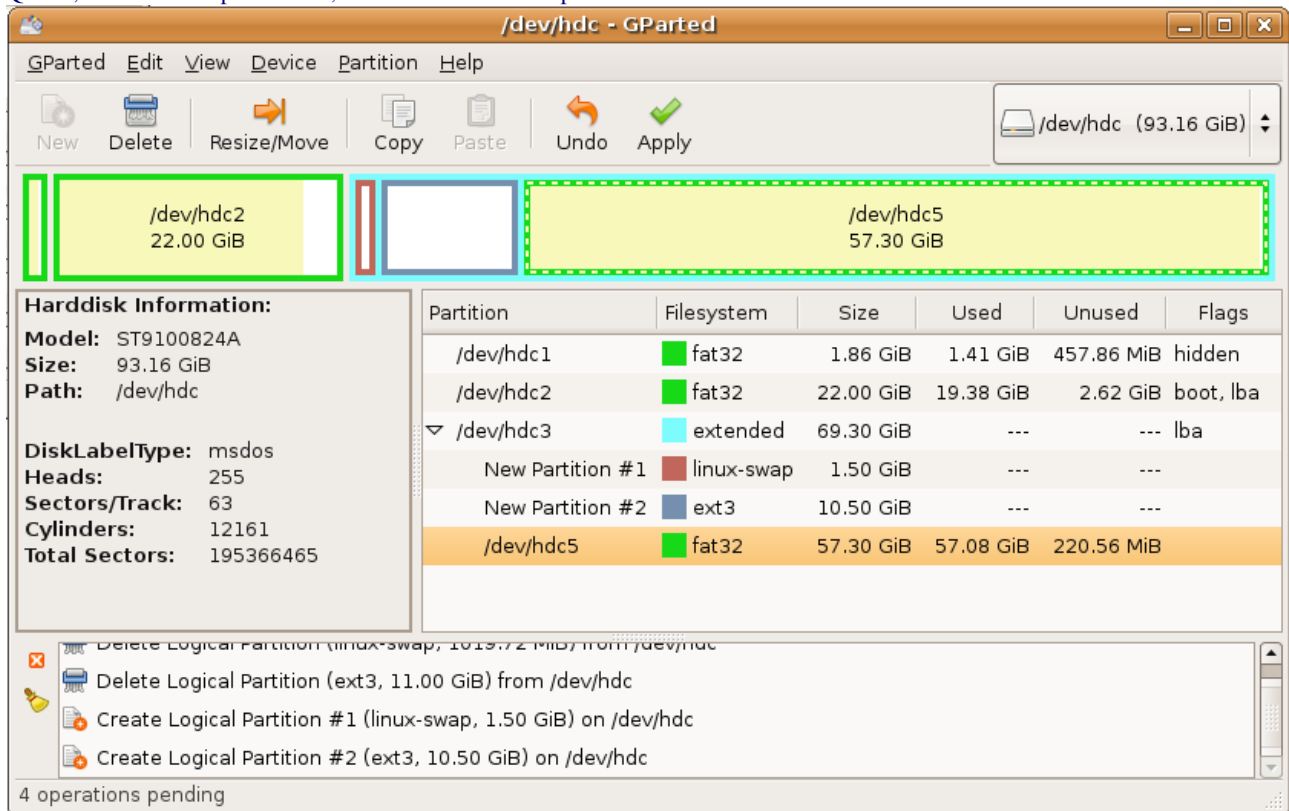
Quando un computer attraverso il processore esegue i compiti richiesti, elabora dei dati che tiene temporaneamente nella memoria centrale (la RAM), ben più veloce di quella usata nei dischi rigidi.

Poiché tale memoria risulta spesso insufficiente, il sistema operativo provvede a tenere nella memoria centrale solo i dati ai quali il processore accede più frequentemente, spostando (operazione di swap-out) quelli meno usati in un “parcheggio temporaneo” sul disco rigido.

Quando il processore riavrà bisogno dei dati swappati, l'OS ricopierà tali dati (swap-in) in memoria centrale, a discapito di altri dati che subiranno lo swap-out.

Ma il sistema operativo dove mette questi dati sul disco rigido? Ogni OS ha un suo metodo: Windows crea un file di swap (anche detto di paging) dentro una partizione, Linux invece usa una partizione apposita da creare in fase di installazione.

Quindi, arrotondando per eccesso, l'installazione minima per Ubuntu 6 richiedeva almeno 3.5 GB.



Nel nostro caso, disponendo di 100 GB, possiamo anche abbondare creando:

- una partizione da 10.5 GB con filesystem EXT3 dove saranno copiati i files di Linux Ubuntu
- una partizione da 1.5 GB per lo swap (il file-system da usare in questo caso è proprio “linux-swap”)

Lo spazio rimanente (100 – 12 = 88 GB) può essere così distribuito:

- lasciando inalterata “win_hidden” (circa 2 GB)
- ridimensionando “win_c” a 22 GB
- assegnando il resto (64 GB) alla nuova partizione “win_d” appena creata con file-system FAT32 in modo che sia facilmente accessibile da Windows

Subito dopo il ripartizionamento del disco, l'installer di Ubuntu ti chiederà di specificare i mount-point, ossia i percorsi ai quali saranno agganciati i file-system (delle varie partizioni presenti nel sistema).

L'argomento mounting sarà ripreso in dettaglio in seguito della guida: per ora è sufficiente usare i seguenti mount-point d'esempio:

- /media/win_c per la partizione “win_c” identificata fisicamente – ad esempio – da /dev/hdc2
- /media/win_d per la partizione “win_d” identificata fisicamente – ad esempio – da /dev/hdc5
- /media/win_hidden per la partizione “win_hidden” identificata fisicamente – ad esempio – da /dev/hdc1
- / per la partizione dove si installa Ubuntu, indicata in figura (uno screenshot di gparted) come “New Partition #2”

Installare altri programmi

L'installer del sistema operativo provvede ad installare anche applicazioni addizionali che probabilmente però non esauriscono le tue esigenze. Per aggiungere altri programmi tra quelli disponibili, esistono numerosi metodi:

- la copia dei files del programma
- la compilazione dei codici sorgente del programma
- l'utilizzo di installer appositi, spesso differenti a seconda del sistema che usi (files RPM , DEB , ...)

Molti OS forniscono programmi per la gestione dei software installati, il recupero degli aggiornamenti, la disinstallazione, la ricerca degli applicativi di cui si ha bisogno, e tanto altro ancora. Ad esempio, in Ubuntu sono disponibili `apt-get` da console e `synaptic` (Synaptic Package Manager) con interfaccia grafica Gnome.

Iniziamo a sporcarci le mani

Il boot-loader

All'avvio del computer, se sono installati più sistemi operativi – a meno di aver indicato diversamente in fase di installazione – sarà eseguito il “boot loader”, un programma che ti consente di selezionare il sistema operativo da far partire, potendo scegliere ovviamente tra quelli installati: Windows, Linux, FreeBSD,

Qualora, inoltre, avessi installato più versioni dello stesso OS, potrai scegliere quale specifica versione avviare.

I boot loader esistenti sono svariati e sono installati assieme all'OS UNIX-like: al variare del sistema in generale varia anche il boot loader installato. Ad esempio Ubuntu 6 e 7 installano GRUB, mentre altri installano LILO, ...

I più diffusi boot loader sono configurabili (puoi ad esempio impostare una password, l'immagine di sfondo, i colori del testo, l'OS da avviare per default, ... e tanti altri dettagli), ma questi aspetti puoi approfondirli sul web.

Login e logout

Appena terminata la “fase di boot” in cui sistema operativo carica in memoria tutto ciò di cui ha bisogno si prospettano due diversi scenari:

- viene avviata l'interfaccia grafica
- si possono impartire comandi tramite la console

In entrambi i casi, inizialmente si dovrebbe poter solo effettuare il login per “autenticarsi”, cioè dire chi sei e darne una prova (ad esempio digitando un username ed una password).

Non tutti gli utenti possono fare le stesse cose: operazioni critiche devono ad esempio essere permesse solo all'amministratore o a persone di sua fiducia. Perciò per utilizzare il sistema operativo occorre prima autenticarsi.

Notare che per evidenti motivi di sicurezza, quasi sempre nella console la digitazione della password non comporta l'echo (cioè la ristampa su schermo) dei caratteri premuti, né di caratteri speciali quali ad esempio l'asterisco usato invece in alcuni sistemi con interfaccia grafica.

Durante l'avvio dell'OS, il sistema scrive in un registro (un “log”) le operazioni compiute ed eventuali messaggi di errore. Nel caso di sistemi senza interfaccia grafica, inoltre, questo registro viene anche stampato su schermo. E' comunque possibile rileggere il contenuto del registro lanciando il comando `dmesg`.

Quando si termina di lavorare, oppure quando ci si deve allontanare dalla postazione di lavoro in presenza di estranei nei paraggi che potrebbero leggere o modificare dati sensibili, è opportuno chiudere la sessione, ossia effettuare il “logout”. Questa operazione è intuitiva e semplice attraverso l'interfaccia grafica, mentre tramite console basta lanciare un opportuno comando come `Logout`.

Console e terminali

Come già accennato, attraverso la console è possibile lanciare comandi espressi sotto forma testuali, che un programma apposito (la “shell”) interpreta ed esegue.

Ogni utente ha una shell default che viene utilizzata quando apre una console. Le shell sono numerose (`sh`, `bash`, `tcsh`, ...), ma su ogni sistema derivato da UNIX di può sempre trovare almeno la `sh`.

All'inizio di ogni riga della console viene stampata una serie di informazioni. Tra queste solitamente ci sono:

- il nome dell'utente corrente
- il nome della directory corrente (altrimenti detta “directory di lavoro”, working directory)

Ad esempio

```
vittorio@21dicembre:/bin$
```

può apparire all'inizio di ciascuna riga sulla console di una shell in esecuzione sul computer 21dicembre da parte dell'utente

```
vittorio
```

, con `/bin` come directory corrente di lavoro.


NOTA: LE DIRECTORIES

Una directory è un contenitore di files: in Microsoft Windows viene chiamata “Cartella” (Folder)

Le directories permettono dunque l'organizzazione gerarchica di grandi moli di files.

In ogni caso sarà stampato anche un carattere particolare immediatamente prima del prompt (ossia dell'attesa di inserimento da parte dell'utente). Questo carattere, nella shell `sh` come anche in tante altre, è:

- il cancelletto `#` se si è loggati come amministratore (“root”)
- altrimenti è il dollaro `$`, come mostrato nell'esempio precedente.



NOTA: LA MULTIPROGRAMMAZIONE

La multiprogrammazione di un sistema operativo è la possibilità di mantenere in esecuzione “contemporaneamente” più programmi come ad esempio un internet browser, un player audio, un editor di testi, ed altro ancora. Un sistema operativo si dice invece monoprogrammato se permette l'esecuzione di un programma alla volta.

In un ambiente senza interfaccia grafica, la multiprogrammazione del sistema sarebbe sprecata se l'utente potesse interagire solamente con un programma per volta per via della disponibilità di una sola console: la soluzione è la presenza di più “terminali”, cioè una specie di “schermi virtuali” a disposizione dell'utente. Ti sono forniti dunque un certo numero di terminali testuali: su ognuno di essi puoi così tenere in esecuzione un'applicazione a piacimento: ti basterà cambiare terminale per cambiare l'applicazione

in uso senza per questo dover terminare gli altri programmi, che restano in uso sugli altri terminali.

Se il sistema supporta questa funzionalità, inoltre, avrai un terminale dove gira l'interfaccia grafica.

Riformulando quanto dichiarato precedentemente, dunque, subito dopo il boot in tutti i terminali viene presentata la richiesta di login: in terminali differenti possono loggarsi account-utenti differenti, che possono anche essere utilizzati dalla stessa persona fisica (ma questo non interessa al computer) oppure da persone distinte. Alcuni programmi, inoltre, permettono la creazione di "terminali remoti", che consentono ad utenti collegati in rete di fare login ed usare le risorse del computer come se vi si trovasse di fronte.

Per passare da un terminale all'altro basta premere una combinazione di tasti.

Ad esempio in FreeBSD per selezionare il terminale numero 2 si deve premere:



In Linux Ubuntu invece la combinazione è:



Ogni terminale è considerato come un qualsiasi altro dispositivo hardware. Il nome di ogni terminale inizia con `tty`, seguito dal suo numero d'ordine: ad esempio, considerando che la numerazione dei dispositivi comincia da zero, il terzo terminale si chiama `tty2`.

Attraverso il comando `exit` si termina la shell. Notare che se si opera senza interfaccia grafica, questo comando comporta implicitamente il logout.

Dalla console puoi avviare l'ambiente grafico con il comando `startx` purché esso sia installato.

In FreeBSD:

- per tornare alla console senza terminare l'ambiente grafico, premere
- per tornare all'ambiente grafico (se precedentemente avviato), premere

In Linux, invece, la combinazione di tasti provoca la terminazione forzata dell'ambiente grafico (utile nel caso lo si voglia riavviare in modo da beneficiare da subito di eventuali modifiche alla configurazione che richiederebbero altrimenti il riavvio del computer).

Accreditarsi come amministratori

Talvolta capita di dover eseguire operazioni critiche che potrebbero arrecare danni al sistema, e che quindi sono permesse solo all'amministratore: gestione degli utenti, dei dispositivi hardware, della configurazione del sistema operativo, ...



NOTA: I NOMI DELL'AMMINISTRATORE

L'amministratore in questa guida è riferito in vari modi: "root", "super-user", o semplicemente "amministratore"... ma è sempre lui. Il nome esatto dell'account utente dell'amministratore è `root`.

Fai attenzione che il termine "root" (in inglese: radice) ha anche altri utilizzi: la root directory (cioè `/`); la home directory dell'utente "root" (cioè `/root`).

Siccome normalmente non si opera come super-user, occorre accreditarsi come tali per compiere queste operazioni.

Per farlo, si potrebbe uscire dal proprio account e fare login come `root`, oppure lanciare il comando `su` (cioè "super-user") che – dopo aver ricevuto correttamente tramite la tastiera la password di amministrazione – apre una nuova shell e permette di operare come `root`. Per tornare alla shell precedente, basta chiudere la nuova shell col solito comando `exit`.

Infine, in alcuni OS esistono istruzioni (come `sudo`, che sta per "Super-User DO") con le quali è possibile impartire direttamente un singolo comando come amministratore, senza dover aprire una shell apposta. Ad esempio, l'istruzione che segue apre come `root` il file di configurazione `/etc/fstab` con l'editor di testi `nano`.

```
sudo nano /etc/fstab
```

Negli ambienti con interfaccia grafica è possibile avviare applicazioni visuali con comandi come `gksudo` (con desktop manager Gnome) o `kdesudo` (con KDE).

Il manuale

Se sei un utente di Microsoft Windows, credo tu ne abbia raramente utilizzato l'help.

Al contrario, il manuale nei sistemi UNIX è molto utilizzato, e ti sarà di fondamentale sostegno durante i tuoi primi passi in questo nuovo ambiente. Voglio inoltre farti notare come questa guida, oltre al fatto di non essere orientata verso uno specifico sistema operativo, vuole essere solo un indirizzamento, una prima spinta verso l'esplorazione di questi sistemi, e proprio per questo non può essere esaustiva, né tentare di esserlo. Di conseguenza, sarà il Manuale il tuo secondo Virgilio in questo inferno di bit, e sarà esso a supplire alle numerose ed inevitabili mancanze di queste pagine.

La guida in linea è consultabile sia tramite interfaccia grafica, che da console. Per quanto riguarda il primo modo, non credo ci siano difficoltà d'uso. La consultazione del manuale mediante la console invece è un attimo più complessa. Prima di sapere come avviarlo ed impiegarlo, è opportuno sapere i fondamentali su come utilizzarlo:



per scorrere tra le pagine



per uscire dal manuale

Per saperne di più, è possibile chiedere al manuale stesso

```
man man
```

Più in generale... per sapere come usare un certo comando basta lanciare

```
man comando
```

Per ricercare tutti i comandi che contengono nel loro nome o nella loro breve descrizione, ad esempio, la parola chiave “edi”

```
man -k edi
```

Il lancio del comando precedente produce come output un elenco di voci: sarà successivamente possibile chiamare `man` su ogni specifico elemento per trovare quello che serve. Ogni voce dell’elenco è il nome della pagina di manuale, seguito – tra parentesi – dal numero della sezione del manuale in cui tale pagina si trova. Le sezioni sono: 1) comandi; 2) chiamate di sistema;

Equivalente di `man -k cercare` :

```
apropos cercare
```

Introduzione della sezione 2 :

```
man 2 intro
```

Infine, oltre a questa piccola guida, al manuale ed al resto dell’help fornito dal sistema operativo, ricorda di consultare il web (con i suoi forum, i motori di ricerca, ...), che è una fonte inesauribile di spunti di approfondimento e di soluzioni ai problemi pratici in cui potresti incorrere.

Opzioni dei comandi

I comandi, per convenzione, ricevono in ingresso:

- le “opzioni”. Come puoi aver notato precedentemente, le opzioni sono singole lettere che si passano subito dopo il carattere meno -, e se occorre indicare più opzioni, è sufficiente indicare il meno una volta e fargli seguire l’elenco delle lettere delle diverse opzioni (nulla impedisce, tuttavia, di indicarle separatamente o a gruppi). A volte in questa forma sono fornite solo le opzioni abbreviate, ossia forme meno chiare da leggere ma più veloci da digitare per esprimere opzioni solitamente introdotte da una coppia di caratteri “meno”.
- i “predicati” o “sottocomandi”, sono parole introdotte da un solo “meno” oppure da una coppia di segni “meno”. Spesso le “opzioni” sono solo delle forme brevi (ma meno leggibili) dei corrispondenti “predicati”.
- i parametri, altrimenti detti “argomenti in ingresso”. Non sono introdotti da alcun carattere.

Nota che alcuni predicati (e le opzioni corrispondenti) a volte devono essere seguiti da un separatore (uno spazio o un carattere “uguale”) ed un valore. Ciò accade nel caso di comandi particolarmente complessi, come ad esempi `find` (vedi oltre in questa guida).

Ovviamente, però, tutte queste sono solo convenzioni: nulla impedisce ad un programmatore di sviluppare applicazioni che attendono in ingresso delle opzioni indicate in maniera differente (ad esempio con lo slash alla maniera di Microsoft Windows).

Infatti, ad esempio, esistono comandi come `who am i`, dove in realtà il comando è `who` mentre `am i` è un predicato che specifica quali sono le informazioni particolari di cui si ha bisogno.

Terminare l'OS

Per terminare correttamente il sistema operativo (ossia provocarne lo “shutdown”) puoi utilizzare il comando `shutdown`, il cui comportamento può essere modificato attraverso il passaggio di alcune opzioni.

```
shutdown opzioni-varie quando messaggio
```

Tra le opzioni-varie si può specificare cosa deve accadere dopo lo shutdown del sistema operativo:

- effettuare un riavvio della macchina (reboot): `r`
- spegnere la macchina oppure – se ciò non è possibile via software – bloccarla (halt): `h`

Il parametro quando (l’unico obbligatorio) permette di istruire il comando sul momento in cui lo shutdown deve avvenire. Tale momento può essere indicato in vari modi:

- esplicitamente, ad esempio nella forma `hh:mm` (all’ora `hh` e minuto `mm`)
- relativamente all’istante in cui si lancia il comando, ad esempio nella forma `+m` (tra `m` minuti)
- utilizzando l’alias `now` (equivalente a `+0`), che sta per “adesso”

Ad esempio, per riavviare tra 1 minuto:

```
shutdown -r +1
```

Uno shutdown programmato può essere annullato:

```
shutdown -c
```

Infine, specificando l’opzione facoltativa messaggio si fa sì che al momento dell’inizio della procedura di `shutdown` venga mostrato un messaggio sulla console.

Creare alias

Un alias è “un altro modo di chiamare” qualcosa.

Un alias `X` di un comando `Y` è un comando che, se lanciato, provoca il lancio di `Y`. Gli alias sono pratici per velocizzare la chiamata di comandi lunghi da digitare sulla console.

```
alias nome_nuovo_comando='comando_ed_eventuali_parametri'
```

Ad esempio, dopo aver creato l’alias `spegni_pc` così come segue, chiamandolo si invocherà lo spegnimento del computer:

```
alias spegni_pc='shutdown -h now'
```

Applicazioni e comandi vari

L'installazione di un sistema, ed in particolare di una distribuzione Linux, porta con sé numerosi programmi d'utilità, cioè software non strettamente necessario al funzionamento del sistema operativo ma che implementa funzionalità sfruttabili dall'utente.

Come già visto, è inoltre possibili installare software aggiuntivo, ed i programmi disponibili sono tantissimi... dai più inutili (come il pur simpatico `xeyes` presente in tutti gli ambienti con interfaccia grafica) ai più funzionali.

Nota che le applicazioni per la rete (browser, client di posta, ...), alcune per la masterizzazioni e qualche altro programma sono elencati in altri paragrafi.

Editor di testi

Esistono numerosi editor di testi negli ambienti UNIX, sia con che senza interfaccia grafica. Uno sempre presente è `vi`, utilizzabile tramite la console ed illustrato in seguito in questa guida, ma la cui complessità è abbastanza alta. A seconda del sistema utilizzato sono però disponibili altri editor più semplici da usare, come `edit` in FreeBSD, oppure il semplice `nano`.

Nei sistemi con interfaccia grafica esistono sempre degli editor ben più user-friendly ed al contempo più funzionali sia di `vi` che di `edit`, come ad esempio `kwrite` (con desktop manager KDE) o `gedit` (con Gnome), o anche intere suite come OpenOffice (con la quale ho redatto questa guida) complete di editor di testi, foglio elettronico, editor di presentazioni, eccetera. Un altro software ricco di funzionalità con interfaccia grafica è `scribus` per l'impaginazione e la pubblicazione.

Grafica

Per la grafica è possibile utilizzare l'avanzatissimo programma di disegno `gimp` che, proprio per via della sue numerose funzionalità, inizialmente è un po' complesso da imparare ad usare.

`inkscape` è invece adatto per la grafica vettoriale, mentre `blender` lo è per il 3D modeling.

Puoi usare `hugin`, infine, per comporre assieme più foto in maniera da ottenere fotografie panoramiche.

Programmazione

I sistemi operativi derivati da UNIX sono molto diffusi tra gli sviluppatori, e ciò fa sì che l'installazione di tali sistemi operativi copi sul disco anche molti compilatori, interpreti e documentazione: da `as` per lo GNU Assembler alle guide per Python e Perl, dal compilatore `gcc` al linker `ld`, da `mcs` del Mono Project all'interprete Java...


Vari

Per conoscere nome e versione del sistema operativo

```
uname
```

Opzioni di `uname`

- tipo di macchina `m`
- nome dell'host `n`
- release dell'OS `r`
- nome dell'OS `s`
- versione dell'OS `v`
- tutte le informazioni `a`

`yes` produce continuamente in output una stringa `S` finché non viene terminato (premendo ). Tale stringa `S` è quella che ha ricevuto in ingresso oppure – per default – il carattere “y”. Questo comando è apparentemente inutile, ma in realtà ha una praticità che sarà più chiara in seguito.

`bc` è un calcolatore aritmetico. Dopo averlo avviato, è possibile eseguire calcoli ed assegnarne il risultato a delle variabili:

```
a=sqrt(9)
```

```
b=a*2
```

```
b
```

Restituisce il percorso dove si trova il binario di un certo comando :

```
type comando
```

Pulisce lo schermo:

```
clear
```

Elenco dei comandi recenti:

```
history
```

Pulisce (“clear”) la storia dei comandi recenti:

```
history -c
```

Data ed ora corrente:

```
date
```

Ora corrente nel formato Ore:Minuti :

```
date '+%H:%M'
```

Calendario del mese corrente:

```
cal
```

Calendario del 1984:

```
cal 1984
```

Calendario del gennaio 1984:

```
cal 1 1984
```

Mostra gli utenti loggati nel sistema:

```
who
```

Mostra informazioni sull'utente corrente:

```
who am i
```

Chi sta usando un certo file? Qual'è l'utente e qual'è il programma in esecuzione che lo usano (ID e nome)?

```
fuser -v percorso-file
```

Termina i programmi in esecuzione che stanno usando files nella directory di esempio /usr/games

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

Mostra il tipo di un file

```
file percorso-file
```

Visualizza stringhe di testo all'interno di un file binario

```
strings percorso-file
```

Spesso scaricando files in internet si legge che il file stesso ha un certo "hash MD5". Una volta che il file è stato scaricato, è possibile controllarne l'integrità con il seguente comando e confrontandone l'output con l'hash:

```
md5sum percorso-file
```

Il comando `split` permette invece di dividere un file in "pezzetti" (utile ad esempio per poter copiare un grosso file audio su alcuni dischi floppy).

Ordinamento alfabetico delle righe di un testo, ad esempio contenuto in un file :

```
sort percorso-file
```

Ricerca di una stringa in un testo dentro un file

```
grep "stringa" percorso-file
```

Opzioni di `grep`:

- ricerca case-insensitive (cioè insensibile alla differenza tra maiuscole e e minuscole): `i`
- ritorna anche il numero della linea (a partire dall'inizio dell'input) nella quale la stringa è stata trovata: `n`
- ritorna anche il byte-offset (a partire dall'inizio dell'input) nella quale la stringa da cercare è stata trovata: `b`
- stampa anche un numero `N` di righe dopo ("after") l'occorrenza della stringa cercata: `A N`
- ritorno solo il nome del file dove la stringa da cercare è stata trovata `v`

Stampa tutto (ad esempio il contenuto di un file)

```
cat percorso-file
```

Opzioni di `cat` :

- numera le righe `n`
- salta le righe vuote `b`

L'utilizzo di `cat` produce spesso un output eccessivamente lungo, che non entra in un'unica schermata.

Stampa tutto (ad esempio il contenuto di un file) in ordine inverso

```
tac percorso-file
```

Stampa tutto (ad esempio il contenuto di un file) numerando le righe

```
nl percorso-file
```

Stampa una riga per volta (ad esempio il contenuto di un file)

```
less percorso-file
```

Stampa una schermata per volta (ad esempio il contenuto di un file)

```
more percorso-file
```

Visualizza la prima parte (ad esempio del contenuto di un file)

```
head percorso-file
```

Per visualizzare solo le prime `N` righe (ad esempio del contenuto di un file)

```
head -n N percorso-file
```

Visualizza l'ultima parte (ad esempio del contenuto di un file)

```
tail percorso-file
```

Questi comandi non operano solo su files ma anche su flussi di dati provenienti da altri programmi: questo argomento sarà trattato dettagliatamente nel paragrafo sulla redirectione.

Files e redirezione

L'organizzazione dei files

Negli ambienti derivati da UNIX ogni file, directory e dispositivo hardware ha un proprio indirizzo a partire dalla directory radice.

Il percorso della directory radice (in inglese "root directory") è / .

Per riferire un elemento nel filesystem è possibile utilizzare:

- il suo percorso assoluto, ossia l'indirizzo del file a partire dalla root directory
- il suo percorso relativo, ossia l'indirizzo del file a partire dalla directory corrente di lavoro

Mostra il percorso assoluto della directory corrente ("print working directory"):

```
pwd
```

Cambia la directory corrente:

```
cd directory-da-aprire
```

All'interno della root directory si trova una gerarchia di sotto-directories simile alla seguente, ma variabile a seconda dello specifico OS che si utilizza:

- /bin Sta per "binary". I più importanti comandi UNIX, come le shell, sono realizzati mediante programmi contenuti in questa directory.
- /boot Contiene i files del boot-loader
- /dev Sta per "devices". Qua sono situati i dispositivi (hard disks, lettori cd, modem, terminali, ...). UNIX vede i dispositivi come file speciali: questo comporta una notevole semplificazione in molti aspetti
- /etc Files di configurazione del sistema (degli utenti: `passwd` ; dei messaggi del giorno: `motd` ; e moltissimi altri ancora...). In `/etc/defaults` si trovano i files di configurazione di default, mentre in `/etc/opt` vengono posizionati i file di configurazione generale dei programmi non di sistema ("opzionali"). Questi files sono accessibili in scrittura solo all'amministratore, mentre la lettura a volte è permessa anche ad altri utenti: i permessi saranno esaminati in seguito in questa guida.
- /home Qua si trovano le "home directories" degli utenti (eccetto dell'amministratore, chiamato `root` , la cui home directory è `/root` , da non confondere con la root directory).
Ogni utente può accedere alla propria home directory riferendola, invece che con il suo percorso, attraverso la tilde ~ (carattere ASCII numero 126). Inoltre, è possibile riferire la home directory di un certo utente attraverso ~utente .
Notare che alcuni sistemi operativi posizionano le home directories, invece, dentro la directory `/usr` .
- /lib Librerie essenziali e moduli del sistema operativo.
- /mnt Sta per "mounts". In questa directory vengono solitamente "montati" i dispositivi di storage come le partizioni di hard disk, i floppy disk, i cd (vedi successivamente). Alcuni sistemi usano per questa finalità `/media`
- /opt Qua si trovano i programmi aggiuntivi
- /sbin Contiene programmi di sistema, in genere utilizzabili solo dall'amministratore
- /tmp Contiene files temporanei: ogni utente può scrivervi al suo interno (ma solo sui propri files).
- /usr Raggruppa moltissimi files e directories utilizzati dall'utente o da programmi non di sistema
 - /usr/bin con i programmi non di sistema ed utilizzabili in generale da tutti gli utenti
 - /usr/include con gli headers da includere nei codici sorgente C / C++
 - /usr/lib contenente le librerie dei programmi in `/usr`
 - /usr/sbin contiene programmi non indispensabili al sistema, ma il cui utilizzo dovrebbe essere di competenza del solo utente `root`
 - /usr/share con files condivisi dai programmi come icone, documentazione, ...
 - /usr/src al cui interno sono posizionati codici sorgente di programmi
- /var Contiene dati variabili, log, code di stampa, ...

Operazioni di base su files e directories

Creare un file vuoto (o se esistente ne aggiorna le date di ultimo accesso e modifica) :

```
touch nome-file
```



NOTA: GLI I-NODE

I files e le directories sono implementi sul disco attraverso gli I-NODE, che consistono in una struttura di dati descrittiva dell'elemento stesso.

In altre parole, ogni partizione ha una tabella di tutti gli elementi (files, directories, ...) che contiene, e per ciascun elemento sono memorizzate alcune informazioni in un blocco di dati chiamato appunto I-NODE. Queste informazioni sono ad esempio: riferimenti a dove si trova il contenuto vero e proprio del file all'interno del disco, le date di creazione ed ultima modifica del file, i permessi, ...

Creare una directory:

```
mkdir nome-directory
```

I links

All'interno di una directory puoi creare quanti link (a files, directories, dispositivi, ...) vuoi.

**NOTA: I LINK**

Mentre il contenuto di un file audio è la codifica di suoni, il contenuto di un file testuale è la codifica di un testo, eccetera, ... il contenuto di una directory è un elenco di collegamenti (“links”, o più precisamente “hard links”) agli I-NODE degli elementi che la directory appunto contiene.

Riassumendo, un I-NODE è la struttura dati descrittiva di un elemento sulla partizione (un file, una directory, ...), mentre un link è un riferimento ad un I-NODE.

Nota che ogni directory contiene sempre almeno due links: quello a sé stessa (.) e quello alla directory padre (chiamato ..): attraverso questi link è possibile riferire un altro file nel file system tramite un percorso relativo. Ad esempio, posto che nella directory superiore sia presente un file T, il percorso relativo di T è ../T.

Attraverso appositi comandi, si può comandare la creazione in una directory X di un link ad un elemento F che sta in un'altra directory Y. L'I-NODE del file F sarà così accessibile tramite due percorsi: sia dentro X, sia dentro Y.

Creare un link B allo stesso I-NODE puntato da A (un link ad un file preesistente):

```
ln A B
```

Creare un link simbolico C ad A:

```
ln -s A C
```

**NOTA: I LINK SIMBOLICI**

Un link simbolico ad un elemento F è in realtà un link ad un nuovo file il cui contenuto è il percorso di F. Che differenza c'è dunque tra un link ed un link simbolico? Posti gli ultimi due comandi di esempio, se si rimuove il link A:

- il link B continua a funzionare in quanto si riferisce all'I-NODE (non coinvolto dalla rimozione del link A) che era puntato da A
- l'accesso al link C genera un errore (in quanto punta ad A che non esiste più, benché l'I-NODE che esso puntava continui ad esistere).

Rimozione

Rimuovere un link:

```
rm nome-file
```

**NOTA: IL CONTATORE DEI LINKS**

Ogni I-NODE ha un contatore dei links (hard, non simbolici) che lo puntano. La rimozione di un link comporta la cancellazione anche dell'I-NODE solo se il suo contatore dei links diventa zero.

Nel comando di rimozione – come anche la copia, lo spostamento e tantissimi altri – possono essere utilizzati i caratteri jolly * e ? :

- l'asterisco è un segnaposto per un numero qualsiasi (anche zero) di qualunque carattere
- il punto interrogativo lo è per uno ed un solo carattere.

Rimuovere nome-file-1, nome-file-2, ... nome-file-9, e tutti i link con estensione tmp :

```
rm nome-file-? *.tmp
```

Rimuovere un link ad una directory vuota :

```
rmdir nome_dir
```

Rimuovere un link ad una directory ed il suo contenuto ricorsivamente (utile nel caso si voglia rimuovere una directory che non è vuota):

```
rm -R nome_dir
```

Copia

Il comando di copia cp crea copie degli I-NODE puntati dai links ricevuti in ingresso. Ad esempio, l'istruzione seguente accede all'I-NODE riferito dal link A, crea un I-NODE con lo stesso contenuto, ed infine crea un link B al nuovo I-NODE.

```
cp A B
```

Vi sono due possibili modi di lanciare cp :

- se l'ultimo parametro è una directory esistente, gli I-NODE puntati dagli altri links passati al comando vengono copiati all'interno di tale directory
- altrimenti i parametri ammessi in ingresso sono solo due: il primo è il nome del link dell'I-NODE da copiare, il secondo è il percorso e nome dove posizionare il link alla copia dell'I-NODE. Questo è il caso dell'esempio di poche righe fa.

Esempio 1: copia i primi 3 files dentro la directory indicata come 4° parametro

```
cp file1 file2 file3 dest_dir
```

Esempio 2: copia il file sorgente dentro la directory dest_dir con il nome dest_file

```
cp sorgente dest_dir/dest_file
```

Copiare una directory ricorsivamente (cioè anche il contenuto delle sue subdirectories):

```
cp -R sorgente destinazione
```

Copiare una directory con l'opzione d comporta la copia degli eventuali links simbolici come tali, non la copia dei files a cui i collegamenti si riferiscono:

```
cp -d sorgente destinazione
```

Inoltre l'opzione p fa sì che vengano preservate le proprietà ed i permessi sul file (ha effetto nel caso si stiano copiando files di altri utenti, ma si vuole tralaltro lasciare l'utente proprietario originario sui nuovi files):

```
cp -p sorgente destinazione
```

Quindi, riassumendo, per copiare ricorsivamente files di altri utenti, mantenendo intatte proprietà e permessi e lasciando i links simbolici come tali il comando da usare è:

```
cp -dpR sorgente destinazione
```

Spostare: “taglia & incolla”

Spostare (e rinominare) links può avvenire in due modi come per la copia: da una sorgente ad una destinazione, oppure un elenco di sorgenti verso una directory di destinazione.

```
mv sorgente destinazione
mv sorgente_1 sorgente_2 sorgente_N destinazione
```

Differenze

Controllare se due files coincidono o sono differenti:

```
diff file1 file2
```

diff mostra, se le trova, le differenze (aggiunte, modifiche, eliminazioni) tra i due files, indicando per ciascuna il numero di riga ed il contesto (ossia il contenuto delle righe attigue). Se non viene trovata alcuna differenza, diff non produce alcun output.

Opzioni del comando diff :

- non sensibile a maiuscole e minuscole: **i**
- non sensibile a differenti spaziature orizzontali: **b**
- non sensibile a linee vuote aggiunte o mancanti: **B**
- effettua un confronto binario (per default invece realizza un confronto del testo): **--binary**

Elenco e proprietà di files, e note sui files nascosti

Stampare su schermo il contenuto di una directory:

```
ls
```

Opzioni del comando ls :

- mostra anche i files nascosti: **a**
- stampa anche informazioni aggiuntive (diritti, creatore,...): **l**
- tra le informazioni aggiuntive, mostra le dimensioni in una forma più comprensibile (cioè non sempre in bytes, ma in caso di files grossi in KB, MB, GB, ...): **h**
- stampa anche il numero identificativo dell'I-NODE: **i**
- stampa informazioni sulla specifica directory invece che di tutto il suo contenuto: **d**
- ricorsivo (continua sulle sottodirectories): **R**

ls accetta come parametro anche il nome/percorso di specifici files e/o directories (possono essere elencati, si possono usare i caratteri speciali). In tale caso, ls visualizzerà solamente le informazioni richieste relative ai files/directories indicati.

Esempio: stampa tutte le informazioni di tutti i files (anche quelli nascosti) della directory dir-esempio e del suo sottoalbero, mostrando le dimensioni dei files in maniera umana

```
ls -Rlh dir-esempio
```

Esempio: stampa le informazioni aggiuntive (comprensive del numero di I-NODE) della directory corrente

```
ls -ldi
```

Esempio: come il precedente, ma anche sulla directory superiore

```
ls -ldi . . .
```

In alcuni ambienti è possibile indicare a ls che l'output deve essere colorato (in modo da renderlo leggibile più velocemente) attraverso il predicato **--color**.

Come accennato, attraverso l'opzione **a** è possibile visualizzare i files nascosti. Più in dettaglio, i files sono considerati come "nascosti" quando il loro nome comincia con il carattere punto, perciò per nascondere un file chiamato esempio basta rinominarlo .esempio. Il comando **ls -a** mostra quindi anche i files (e, al solito, directories, dispositivi, ...) i cui nomi cominciano col punto. Osserva che il comando **rm *** non rimuove i files nascosti, benché il carattere jolly "asterisco" significhi "nessuno, uno o più caratteri qualsiasi".

Le dimensioni

Per conoscere la dimensione – "disk used" – della directory corrente e di tutto il suo contenuto, riassumendo l'output con l'opzione **s** e mostrandolo in formato comprensibile con **h** :

```
du -sh
```

Per conoscere la dimensione di una directory esempio e di tutto il suo contenuto:

```
du -sh esempio
```

Il comando **du** è applicabile anche sui files.

Nel capitolo sul mounting imparerai ad usare anche **df** ("disk free") che restituisce l'ammontare dello spazio libero nelle diverse partizioni montate.

Cercare files

Stampa l'elenco dei files (che si trovano all'interno della directory corrente) il cui nome comincia con **ciao** ed è seguito da un solo carattere:

```
ls ciao?
```

Stampa l'elenco dei files (che si trovano all'interno della directory corrente) nel cui nome compare **ciao** :

```
ls *ciao*
```

La stessa ricerca precedente, ma all'interno di ciascuna subdirectory della directory corrente (ma non dentro i loro sottoalberi):

```
ls */*ciao*
```

La stessa ricerca precedente, ma all'interno di tutto il sottoalbero a partire dalla directory corrente (.) :

```
find . -name '*ciao*' -print
```

Il comando `find` prevede diverse possibilità sia di selezione dei file che di esecuzione di azioni su di essi. In generale `find` attende come argomenti:

- prima l'elenco delle directories su cui agire
- poi un elenco di opzioni e predicati

Per ogni file presente nelle directories richieste, vengono valutati i predicati di selezione (ricevuti in ingresso) e, se questi sono soddisfatti, i predicati di azione (ricevuti in ingresso).

Ciascun predicato di selezione può assumere in generale valore vero o falso. La valutazione dei predicati avviene per ogni file “in AND logico tra loro con cortocircuito”, cioè in sequenza da sinistra a destra pur tenendo presente che la valutazione stessa viene terminata al primo predicato falso. È inoltre possibile utilizzare espressioni logiche più complesse, attraverso l'impiego degli operatori or (-o) e negazione (!), e possibilità di parentesizzazione (\ (e \)).

`find` mette a disposizione diversi predicati di selezione:

- `-name` selezione del nome
- `-perm` selezione sui permessi
- `-type` selezione sul tipo di file: ordinario, directory, ecc.
- `-user` selezione sull'utente
- `-group` selezione sul gruppo
- `-size` selezione sulla dimensione: esatta (=) o con limite inferiore (<) o superiore (>) rispetto ad un numero n di unità di misura (b per bytes, k per kilobytes)
- `-atime` selezione della data di ultimo accesso
- `-mtime` selezione della data di ultima modifica

Nel primo esempio di `find` puoi anche notare il predicato d'azione `-print`, che produce la stampa su schermo del nome del file (in tale esempio un ipotetico file chiamato “arrivederci” non sarebbe stato stampato in quanto la valutazione dei predicati si sarebbe fermata a `-name` (risultato in quel caso falso), non arrivando a `-print`).

Un altro predicato di azione è `-exec` che esegue un'operazione sul file, o anche `-ok` che differisce da `-exec` per via della richiesta di conferma prima dell'esecuzione del comando. Per quanto riguarda questi due predicati occorre considerare che:

- per riferire il file corrente nel comando si usa la notazione `{}`, che fa da segnaposto al percorso completo del file stesso
- alla fine del comando di `-exec` e `-ok` occorre inserire il carattere di terminazione “punto e virgola”

Esempio: rimozione (`-exec rm {} ;`) in tutto il filesystem (dalla root directory / in tutte le sue sottodirectories) dei files che sono stati acceduti l'ultima volta da più di 7 giorni (`-atime +7`), e che: o si chiamano `tmp` oppure hanno come estensione `tmp`

```
find / \( -name 'tmp' -o -name '*.tmp' \) -atime +7 -exec rm {} ;
```

Ridirigere gli input/output/error

Numerosi comandi (tra i quali `sort`, `grep`, `less`, `more`, `cat`, `head` e `tail`) elaborano i dati ricevuti in due possibili vie:

- come parametro d'ingresso oppure, in sua assenza, ...
- ... attraverso il flusso in input

Per comprendere cosa sia la redirectione di flussi di dati, pensa all'output di un programma da console: in genere viene stampato sulla console, ossia il flusso in output viene rediretto verso la console. La redirectione consiste appunto nello specificare da dove e verso dove vanno indirizzati i flussi di informazioni.

Si può dunque, ad esempio, comandare a `sort` di ordinare dei dati provenienti da uno specifico input ed inviare la lista ordinata verso un'altro specifico output. È inoltre possibile indicare la destinazione dei messaggi di errore.

Solitamente, tuttavia, queste tre sorgenti e destinazioni di flussi non sono indicate, per cui quando sono omesse vengono in realtà utilizzati dei flussi standard:

- lo “standard input” cioè la tastiera come sorgente dell'input
- lo “standard output” cioè la console come destinazione dell'output
- lo “standard error” cioè ancora la console come destinazione dei messaggi di errore

Per ridirigere un flusso di un comando occorrono due elementi:

- l'operatore di reindirizzamento
- dove e come reindirizzare il flusso

Se l'output viene rediretto su un file, l'operatore da utilizzare è `>` oppure se si vuole scrivere in append (aggiunta alla fine) si usa l'operatore `>>`. Per ridirigere lo standard error invece occorre utilizzare `2>` (`2>>` in append), ed infine per l'input è `<`.

Se invece il flusso di output viene rediretto verso un altro programma (per il quale il flusso è da considerarsi quindi di input), l'operatore è il pipe |.

Esempio 1: appende le informazioni di una certa directory in un certo file

```
ls -ld directory >> file
```

Esempio 2: recupera il contenuto della directory corrente, lo ordina alfabeticamente, e ne stampa le prime righe

```
ls | sort | head
```

Esempio 3: ad un programma che pone una lunga sequenza di richieste di conferma (ad esempio una certa operazione su ogni file di una lunga lista) si può rispondere sempre affermativamente utilizzando il comando `yes` (visto precedentemente) senza opzioni

```
yes | programma_verboso
```

Esempio 4: stampa di un messaggio di saluto sul terminale numero 2

```
echo "CIAO!" > /dev/tty2
```

Ancora redirezione: la stampa su carta

Tantissime stampanti sono utilizzabili negli ambienti UNIX grazie al servizio CUPS (Common Unix Printing System). Nei sistemi operativi più moderni, basta collegare la stampante al computer tramite USB affinché l'OS la riconosca e la renda immediatamente utilizzabile.

Mentre ancora una volta in un ambiente con interfaccia grafica il modo per stampare su carta è intuitivo e semplice da individuare (e spesso consiste nel semplice click sui menù File->Print...), da console le operazioni da compiere sono un po' più articolate. Per stampare il contenuto di un file occorre prima passare il file stesso al comando `pr` ("prepare file for printing") che ne elabora il contenuto considerando ad esempio i salti di pagina, e quindi passare il suo output in ingresso all'apposita utilità di stampa su stampante (`lp` oppure `lpr` a seconda dell'OS):

```
pr file | lp
```

A parte la preparazione dei contenuti da stampare, per specificare la stampante di destinazione occorre utilizzare uno dei seguenti comandi (ancora una volta dipende dell'OS che stai utilizzando):

```
lp -dnome-stampante
```

```
lpr -Pnome-stampante
```

Per conoscere la coda di stampa

```
lpstat
```

Anche per rimuovere una richiesta di stampa il comando varia a seconda dell'OS: può essere `cancel` o `lprm`.

Introduzione ai permessi in UNIX

Come visto precedentemente, eseguendo il comando `ls -l` è possibile leggere informazioni aggiuntive su ciascun I-NODE linkato da files nella directory corrente. Il primo campo è una stringa di dieci caratteri, di cui:

- il primo identifica il tipo di elemento (`-` per i files ordinari, `d` per le directories, `l` per i link simbolici, `s` per i socket, `c` per i dispositivi a caratteri, `b` per i dispositivi a blocchi, `p` per le code fifo)
- i successivi 9 sono i permessi sull'I-NODE (chiamati collettivamente con il termine "mode"), divisi in 3 gruppi da 3 caratteri ciascuno:
 - il primo gruppo di tre caratteri specifica i diritti dell'utente proprietario (User)
 - i secondi tre caratteri rappresentano i diritti degli utenti che sono dello stesso gruppo-utenti del proprietario (Group)
 - infine gli ultimi tre caratteri identificano i diritti di tutti gli altri utenti (Others) sul tale file

I permessi accordati vengono intuitivamente indicati con una lettera significativa mentre quelli negati con un trattino.

Sostanzialmente i permessi (e le lettere che li identificano) sono: lettura (`r`), scrittura (`w`) ed esecuzione (`x`).

I permessi vanno interpretati in modo differente a seconda che il file sia ordinario oppure sia una directory. Se il file è ordinario i permessi significano:

- `r` = lettura del file (esaminazione e copia)
- `w` = apertura in scrittura del file
- `x` = esecuzione del file

Se invece il file è una directory i permessi significano:

- `r` = lettura della directory (elencazione senza informazioni aggiuntive dei links che contiene)
- `w` = rinominazione della directory o modifica del suo contenuto (aggiunta, rinominazione e cancellazione di links)
- `x` = esecuzione (elencazione con informazioni aggiuntive del suo contenuto; possibilità di impostarla come directory corrente di lavoro)

Osserva che il comando `rm`, poiché rimuove il link all'I-NODE tra quelli contenuti nella directory che lo contiene, considera il diritto di scrittura sulla directory nella quale si trova il link che deve rimuovere, non il permesso di scrittura del link.

Oltre al formato testuale, si può utilizzare quello numerico, che è più veloce da scrivere ma meno intuitivo per un neofita. Per ciascun gruppo di permessi, il diritto di lettura è rappresentato dalla cifra 4, quello di scrittura 2 e quello di esecuzione 1. Perciò se si vuole consentire lettura ed esecuzione occorre sommare $4+1=5$.

Quindi ad esempio `777` sta per "tutti i permessi a tutti", mentre `750` sta per "accesso completo per il proprietario, lettura ed esecuzione per il gruppo del proprietario, nessun permesso per gli altri".

Altri due permessi: `s`, `t`

A differenza di ciò che hai letto sinora, in generale il "mode" di un file (cioè l'insieme dei permessi) è composto da 4 blocchi invece di 3: il primo è solitamente 0 in forma numerica, ma può rappresentare i permessi `t` ("sticky bit") e `s` ("set UID or GUID").

Sticky bit

Storicamente, lo sticky-bit veniva assegnato agli eseguibili, in modo che una volta avviati il loro codice eseguibile rimanesse residente in memoria per eventuali altre esecuzioni, così da migliorare le performances del computer.

Oggi lo sticky-bit viene usato invece con le directories. Come visto nel paragrafo precedente, nei casi di modifica ed eliminazione di un link viene considerato il bit di scrittura della directory, a prescindere dal proprietario dei file. Applicando alla directory lo sticky-bit, invece, un utente può modificare/eliminare un file esistente in una directory se e solo se è proprietario del file stesso o della directory (oppure ovviamente se è l'amministratore).

Lo sticky-bit `t` è solitamente assegnato alla directory `/tmp`. Viene mostrato nel gruppo dei permessi di “others” al posto del permesso di esecuzione; inoltre, il suo valore numerico è 1, cioè se assegnato assieme a “tutti i diritti a tutti”, si esprime come 1777 (`rwXrwxrwt`).

Esempio: l'utente Alessio crea una directory dove ogni altro utente può mettere dei propri files per condividerli con gli altri, con permessi `rwXrwxrwt`. Successivamente Barbara vi crea un file `F` che in un secondo tempo Carlo proverà, senza riuscirci, ad eliminare non essendo né il proprietario del file (che è Barbara), né della directory (che è Alessio), né l'amministratore (che è `root`). Se invece la directory fosse stata creata senza lo sticky-bit ma con tutti gli altri permessi compreso quello di scrittura, Carlo sarebbe riuscito ad eliminare il file `F` di Barbara.

Set UID, Set GUID

Il bit `s` (con valore numerico 6 dato dalla somma di `SUID=4` e `SGID=2`) viene invece applicato ai programmi eseguibili assieme al bit `x` in modo che possano essere eseguiti con i privilegi dell'utente o gruppo proprietario dei files stessi.

Ad esempio, il programma `passwd` per la modifica della password degli utenti deve poter accedere a importanti files di configurazione del sistema e per questo è di proprietà dell'amministratore. Tuttavia esso deve essere utilizzabile da tutti gli utenti, e per questo ha il bit `s` settato.

Quando, ad esempio, si imposta il bit `s` ad un file con i permessi `r-xr-xr-x` (0555), si ottengono i nuovi permessi `r-sr-sr-x` (6555). Notare che se il bit `s` viene assegnato ad un file sui quali l'utente ed il gruppo proprietari non hanno diritti (ad esempio `r--r--`), la lettera visualizzata è una `S` maiuscola: `r-Sr-Sr--`.

Impostare il proprietario ed i permessi di un file

Il proprietario di un file è il suo creatore, ma può essere modificato.

Per cambiare il proprietario di un file `X`, impostandolo su `Y`

```
chown Y X
```

Per impostare il gruppo proprietario di un file `X`, impostandolo su `Z`

```
chgrp Z X
```

Per impostare di un file `X` sia il proprietario su `Y` che il gruppo su `Z`

```
chown Y:Z X
```

Per impostare il proprietario di una directory ma anche di tutto ciò che si trova nel suo sotto-albero, occorre utilizzare l'opzione `-R` (notare la maiuscola).

Invece per impostare i permessi su files e directories (vale anche qui l'opzione `-R` per la ricorsività) si usa `chmod`, che attende in ingresso i diritti da applicare ed il file da modificare.

```
chmod permessi elenco-files
```

I diritti si esprimono attraverso la forma numerica oppure con una serie di caratteri. Tale sequenza è logicamente divisa in tre parti: “chi”, “operatore” e “diritti”.

- “chi” è una combinazione di: `u` (utente proprietario), `g` (gruppo del proprietario), `o` (altri). Per indicare tutti quanti, basta `a` che è equivalente a `ugo`
- “operatore” può essere `+` in caso di assegnazione, `-` in caso di revoca, oppure `=` in caso di impostazione
- “diritti” sono sostanzialmente i soliti `r`, `w`, `x`, `s` e `t`.

Si possono separare più modifiche dei permessi su uno stesso file utilizzando la virgola.

Esempio 1: aggiunge il permesso di eseguibilità a tutti gli utenti

```
chmod a+x file
```

Esempio 2: vengono tolti i permessi di lettura e scrittura per gruppo e altri, e viene aggiunto il permesso di eseguibilità per il proprietario

```
chmod go-wr,u+x file
```

Esempio 3: agli altri viene assegnato il solo permesso di lettura (se non c'era permesso di lettura viene accordato e se c'erano altri permessi vengono negati)

```
chmod o=r file
```

L'operatore di impostazione può anche non essere seguito da alcun permesso: in tale caso si intende revocare ogni eventuale diritto.

I permessi assegnati automaticamente ad un file al momento della sua creazione sono impostabili attraverso l'utility `umask` alla quale occorre indicare in ingresso in forma numerica ottale i permessi da disabilitare (non quelli accordati).

Se ad esempio si vuole che i permessi standard siano pari a `755` (accesso completo per il proprietario; lettura ed esecuzione per il gruppo del proprietario e per gli altri), si deve disabilitare la scrittura a Group e Others, e quindi lanciare il comando `umask 22`

Se invece si vogliono specificare i permessi in forma simbolico-testuale, occorre indicare i permessi da consentire (non quelli da proibire). Perciò il comando che segue è uguale al precedente: `umask u=rwx,go=rX`

I flag dei files

Esistono infine ulteriori permessi configurabili, ma essi ed i comandi da usare variano a seconda della partizione di disco nella quale si trova il file ed al sistema operativo utilizzato.

Ad esempio, in FreeBSD ogni file ha dei “flags” che possono essere modificati con l'utility `chflags`.

Esempio 1: attribuisce il flag `sunlink` (impossibilità di rimozione)

```
chflags sunlink file
```

Esempio 2: toglie il flag `sunlink`

chflags nosunlink file

Utenti, VI, script ed archiviazione

Gestione degli utenti

Se si è amministratori del sistema, si possono apportare modifiche all'insieme degli utenti e dei gruppi-utente.

Per creare un utente, impostandone tutti i parametri: nome, password, percorso della home directory, shell default...:

```
adduser -silent
```

Similmente, `addgroup` serve a creare gruppi di utenti.

Per eliminare un utente:

```
rmuser nome-utente
```

Ogni utente può cambiare la propria password utilizzando il comando `passwd`.

L'editor di testi VI

VI è un editor di testi da console totalmente user-unfriendly (poco amichevole nei confronti dell'utente), ma in compenso è presente in tutti i sistemi UNIX-like, è ricco di funzionalità, ed ha numerose scorciatoie da tastiera per le più disparate azioni di modifica e spostamento nel testo (anche per questo ha una curva di apprendimento molto lenta).

Se quindi utilizzi un sistema con interfaccia grafica o comunque hai altri (e più semplici) editor testuali da console come ad esempio `edit` o `nano`, puoi anche non imparare ad usare VI.

VI può operare in due diverse modalità: "input mode" o "command mode". L'utente in input-mode può inserire testo nel documento, mentre in command-mode impartisce dei comandi all'editor (sempre attraverso la tastiera).











Dall'input-mode si passa al command-mode premendo .

Note:

- tutti i comandi seguenti si intendono impartiti in command-mode
- i segnaposti chiamati **N** sono da intendersi numerici










Inserimento












Dal command-mode si passa all'input-mode impartendo diversi comandi:

-  si inserisce testo sovrascrivendolo al preesistente
-  si inserisce un solo carattere sovrascrivendolo al preesistente
-  sostituzione: si cancella il testo a partire dalla posizione corrente e quindi vi si inserisce del nuovo testo (eventualmente nessuno: in tal caso l'operazione realizzata è la sola cancellazione)
-  sostituzione dall'inizio della riga corrente anziché dalla posizione corrente
-  si appende testo alla fine della riga corrente
-  si inserisce testo dopo la posizione corrente del cursore
-  si inserisce testo all'inizio della riga corrente
-  si inserisce testo a partire dalla posizione corrente del cursore, facendo scorrere avanti il testo preesistente
-  si inserisce una nuova riga sopra quella corrente
-  si inserisce una nuova riga sotto quella corrente

Spostarsi







Ci sono diverse maniere per spostarsi all'interno del documento:

-  per spostarsi alla **N**ª riga del documento
-  per spostarsi alla **N**ª riga della schermata
-  (pipe) per spostarsi alla **N**ª colonna della riga corrente
-  oppure  per spostarsi al carattere immediatamente a sinistra
-  oppure  per spostarsi alla riga inferiore
-  oppure  per spostarsi alla riga superiore

- **I** oppure  per spostarsi al carattere immediatamente a destra
-  alla pagina precedente
-  alla pagina successiva
- **N**  per spostarsi **N** pagine indietro (backward)
nota: se **N** non viene specificato, viene assunto uguale ad 1
- **N**  per spostarsi **N** pagine avanti (forward)
nota: se **N** non viene specificato, viene assunto uguale ad 1
- **N**  per spostarsi **N** righe indietro
nota: se **N** non viene specificato, viene assunto uguale all'ultimo specificato
nota 2:  invece di  se si vuole rimanere sulla stessa riga e colonna nello schermo
- **N**  per spostarsi **N** righe avanti
nota: se **N** non viene specificato, viene assunto uguale all'ultimo specificato
nota 2:  invece di  se si vuole rimanere sulla stessa riga e colonna nello schermo
- **N**  per spostarsi **N** paragrafi indietro
nota: se **N** non viene specificato, viene assunto uguale ad 1
- **N**  per spostarsi **N** paragrafi avanti
nota: se **N** non viene specificato, viene assunto uguale ad 1
- **N**  per spostarsi **N** parole indietro
nota: se **N** non viene specificato, viene assunto uguale ad 1
- **N**  per spostarsi **N** parole avanti
nota: se **N** non viene specificato, viene assunto uguale ad 1

















Ricerche

Ti puoi spostare all'interno di un documento anche per effetto di una ricerca. Le ricerche consistono, come facilmente intuibile, nel chiedere a VI di cercare l'occorrenza di qualcosa all'interno del documento. Questo "qualcosa" può essere una stringa di caratteri, ma anche un'espressione regolare (cerca in rete se non sai di cosa si tratta, e per scoprire come esprimerla in VI).

-  qualcosa  per spostarsi alla prima occorrenza (successiva alla posizione corrente) di qualcosa
-  qualcosa  per spostarsi alla prima occorrenza (precedente alla posizione corrente) di qualcosa
-  ripete l'ultimo comando di ricerca
-  ripete l'ultimo comando di ricerca ma nella direzione opposta

Eliminazioni, spostamenti e copie

Operazioni di elimina, taglia, copia e incolla

-  per cancellare il carattere corrente
- **N**  per cancellare **N** caratteri a partire dal corrente
-  per cancellare il capo-a-rigo alla fine della riga corrente, così da spostare la riga seguente alla fine della corrente
-  ,  ,  e  (come già visto) per cancellare e sovrascrivere
-   per tagliare tutta la riga corrente (cioè cancellarla dal documento e posizionarla nella clipboard)
- per tagliare **N** righe a partire dall'inizio della corrente  **N** 
- per copiare tutta la riga corrente  
- per copiare **N** righe a partire dall'inizio della corrente  **N** 
-  per incollare il contenuto della clipboard prima della posizione del cursore

- **p** per incollare il contenuto della clipboard dopo la posizione del cursore

Puoi anche utilizzare più clipboard: oltre a quella “senza nome”, infatti, ne puoi creare altre “con nome”. Per tagliare, copiare ed incollare da/in una specifica clipboard con nome, i comandi sono simili ai precedenti con in più, sostanzialmente, la sola aggiunta del nome della clipboard da usare. Puoi trovare, su questo specifico argomento come sugli altri, spiegazioni più dettagliate in rete.

Apri, salva e chiudi

Operazioni di apertura, chiusura e salvataggio:

- **:w** **Enter** scrive il documento (salva il file)
- **:w** `percorso-file` **Enter** scrive il documento ad un certo percorso
- **:wq** **Enter** scrive il documento e chiude VI
- **:q** **Enter** chiude VI se il documento non è stato modificato dall'ultimo salvataggio
- **:q!** **Enter** chiude VI senza salvare eventuali modifiche
- **:e** `percorso-file` **Enter** carica il file indicato
- **:e!** **Enter** ricarica il file annullando le modifiche
- **:r** `percorso-file` **Enter** legge il file indicato e ne inserisce il contenuto dopo la riga attiva

Altro

Operazioni di annullamento:

- **U** delle operazioni sulla riga corrente da quando il cursore si è posta su di essa
- **u** dell'ultima operazione al documento

Infine:

- **Ctrl+G** per mostrare informazioni sul file
- **Ctrl+R** per ridisegnare la schermata corrente (refresh)
- **:set number** **Enter** per mostrare i numeri di riga
- **:set nonumber** **Enter** per non mostrare i numeri di riga
- **:set ignorecase** **Enter** per considerare uguali nelle ricerche le forme minuscole e maiuscole della stessa lettera
- **:set noignorecase** **Enter** per considerare differenti nelle ricerche le lettere minuscole e maiuscole della stessa lettera

Per l'elenco completo dei comandi, di cui io ho riportato solo una parte, consulta il manuale.

Script per la shell

Uno script per la shell è un file col permesso di esecuzione settato, che contiene una serie di comandi – eventualmente condizionati – che la shell deve interpretare ed eseguire. Lanciare uno script è come lanciare i singoli comandi uno dopo l'altro, solo che chiaramente nel primo caso è la shell a fare tutto da sola, con conseguente vantaggio per l'utente.

Gli script possono dunque essere utilizzati per automatizzare operazioni ripetitive e noiose.

La prima riga di uno script deve iniziare con la sequenza speciale **#!** seguita dal nome della shell (**sh**, **bash**, ...) da utilizzare per interpretare lo script stesso. Nelle righe successive invece si trova il corpo vero e proprio dello script con tutte le istruzioni ed eventuali commenti (introdotti da **#**).

Esempio: definisco una variabile, stampo dei messaggi di testo sullo schermo (tramite il comando **echo** già visto) e l'output di un comando, concateno una variabile d'ambiente ad una stringa, stampo il valore di una mia variabile, ed infine elimino interattivamente (**-i**) un certo file.

```
#!/bin/sh
MIA_VARIABILE=20184
echo "Ciao!"
Sono andato capo a rigo..."
echo "Data: `date`"
echo "Questo è il percorso del mio desktop:"
echo ${HOME}/Desktop
echo "Un numero: "
```

```
echo $MIA_VARIABILE
rm -i file_eliminare
```

Come si vince dal listato precedente:

- per memorizzare il contenuto di una variabile basta usare il solo nome della variabile stessa e l'operatore uguale (=)
- per leggere il contenuto di una variabile si antepone il simbolo del dollaro al nome della variabile; qualora però questa notazione sia ambigua (ad esempio perché immersa tra altri caratteri), è necessario racchiudere il nome della variabile tra parentesi graffe
- l'output di un comando può essere convertito in stringa racchiudendo il comando stesso tra apici ` (codice ASCII 96).

Alcune variabili d'ambiente:

- \$# numero degli argomenti passati allo script
- \$1, \$2, ... gli argomenti passati allo script
- \$* tutta la stringa degli argomenti
- \$- opzioni dell'esecuzione dello script
- \$\$ il PID, cioè il numero identificativo del processo (la gestione dei processi è descritta più avanti in questa guida)
- \$HOME la home directory dell'utente corrente
- \$PATH i percorsi, separati caratteri di due punti, dove vengono cercati i comandi riferiti per nome senza percorso
- \$PWD la directory di lavoro corrente

Chiamando il comando `read`, che accetta in ingresso una variabile di destinazione, è possibile far digitare dei valori all'utente. Il flusso sequenziale di esecuzione può essere alterato tramite alcuni costrutti come `if`, `case`, `in`, `for`, `while` ed altri, ed istruzioni del tipo di `break`, `continue` e `exit`.

Alcuni operatori condizionali:

- `-a` and logico
- `-o` or logico
- `!` negazione
- `-d` è una directory?
- `-f` non è una directory?
- `-r` lettura consentita?
- `-w` scrittura consentita?
- `-s` l'operando non è vuoto?
- `-lt` è minore?
- `-gt` è maggiore?
- `-eq` gli operandi sono due numeri uguali
- `==` gli operandi sono due stringhe uguali

Ecco infine un semplice script d'esempio che stampa un conto alla rovescia a partire da un numero inserito dall'utente:

```
#!/bin/sh
echo "Inserisci un numero intero nell'intervallo [1;10]"
read numero
if [ $numero -lt 1 -o $numero -gt 10 ]
then
    echo "Valore esterno all'intervallo consentito."
    exit 1
fi
echo "Conto alla rovescia..."
while [ $numero -gt 0 ]
do
    echo $numero
    numero=`expr $numero - 1`
done
echo "Zero!!!"
```

Ovviamente queste poche righe sono solo introduttive nel campo degli script per le shell, e perciò ancora una volta rimando ad altre guide l'approfondimento su questo argomento e le sue potenzialità più avanzate, come ad esempio la possibilità di intercettare segnali tramite `trap`.

Archiviazione: compressione e masterizzazione

Creare un archivio compresso:

```
tar -cf archivio elenco-elementi-archiviare
```

Ogni elemento dell'elenco dei file e delle directories da archiviare va separato dagli altri tramite uno spazio.

L'opzione `n` fa sì che l'archiviazione di una directory non comprenda il contenuto delle sue sottodirectories.

Elenco del contenuto di un archivio:

```
tar -tf archivio
```

Per avere informazioni aggiuntive su ciascun elemento, utilizzare l'opzione "verbose" `v`.

Estrazione (nella directory corrente di lavoro):

```
tar -xf archivio
```

`gzip` è un programma di compressione attraverso il quale viene creato un file compresso per ogni file indicato negli argomenti. Esso è in grado di comprimere solo file regolari e soltanto singolarmente: per ogni file ne viene generato un altro con l'estensione `.gz`, che

può essere decompresso tramite `gunzip` .

Sono talvolta disponibili, inoltre, i programmi `zip` e `unzip` per comprimere e decomprimere files zip.

Anche nei sistemi derivati da UNIX è possibile masterizzare cd e dvd, essendo disponibili numerosi programmi come ad esempio:

- `k3b` (con interfaccia grafica KDE)
- `cdrecord` (da console)
- `gnomebaker` (con interfaccia grafica Gnome)

Sempre per quanto riguarda i CD, si possono utilizzare `cdda2wav` e `cdparanoia` (entrambi da console) per estrarre tracce da un CD audio e convertirle in files WAV (ripping), oppure con lo stesso fine `grip` e `sound-juicer` con interfaccia grafica Gnome.

Mounting e gestione di partizioni

Cos'è ed a cosa serve il mount



NOTA: PARTIZIONI E FILE-SYSTEMS

Se non ricordi bene cosa sono ed a quali finalità rispondono, rileggi l'apposita nota nel capitolo introduttivo.

L'operazione di mount consiste nell' "agganciare" un file-system presente su un certo dispositivo di storage (un hard disk, un cd, un floppy, ...) al file-system principale (quello dove è installato l'OS ed il cui albero inizia da / ed è anche chiamato "root file-system") per poterlo utilizzare: leggere un file al suo interno, navigare tra le directories, rimuovere un link, eccetera.

Se ad esempio disponi di varie partizioni, di cui una prima dove hai installato il sistema operativo ed una seconda dove si trovano alcuni files, non potrai accedere a questi ultimi files se non hai collegato la seconda partizione alla prima: solo collegandola ("montandola") avrai la possibilità di esprimere un percorso che ti porti al suo interno per accedere ai files desiderati.

La directory radice del root file-system, come già visto, è / : è appunto all'interno di una o più sue sottodirectories che vengono montati i file-systems delle altre eventuali partizioni (per convenzione, i file-systems vengono montati all'interno di directories dentro /mnt oppure /media).

La directory dove viene montato un file-system è detta "punto di mount" per tale file-system. Ad esempio si potrebbe montare un floppy in /mnt/floppy o una partizione di un certo hard disk esterno in /mnt/disco-esterno .

Per riferire un ipotetico file lista-spesa.txt nella directory radice del floppy, ti basterà perciò formulare il percorso /mnt/floppy/lista-spesa.txt .

Quando si monta un file-system in una directory dir che fa da punto di mount, il sottoalbero del root-filesystem discendente da dir viene temporaneamente "nascosto", e sarà nuovamente accessibile al momento dello smontaggio. Ciò significa che se in una directory dir che contiene il file foto-bella.png si monta una partizione, accedendo al percorso del punto di mount non si troverà più foto-bella.png , ma solo il contenuto della partizione montata (finché tale partizione resterà montata).

Lo smontaggio implica inoltre il flush.



NOTA: IL FLUSH

Quando si comanda il sistema operativo di scrivere sul disco, in realtà l'OS non "obbedisce" subito per ragioni di efficienza (e dunque per migliorare le prestazioni del sistema), rimandando le scritture in momenti più convenienti come ad esempio i periodi di inattività.


Il flush consiste appunto nella effettiva realizzazione delle scritture su disco comandate ma non ancora avvenute .

Staccare fisicamente un dispositivo di memorizzazione (quale ad esempio una drive-pen USB) senza averne correttamente smontato i file-systems, espone dunque al rischio che il flush non sia avvenuto completamente: è anzi possibile che in tale maniera si sia danneggiato il file-system ed i suoi contenuti non siano quindi recuperabili.

Le 4 fette, e terminologia nei diversi sistemi

Ogni disco può essere suddiviso al massimo in 4 "fette", ma ognuna di queste fette può contenere svariate partizioni.

I nomi usati per indicare di "fette" e partizioni sono differenti in ambienti differenti:

	UNIX	MS Windows
Fetta	Slice	Partizione 
Partizione	Partizione	Unità

Dunque una partizione per Windows è una slice per i sistemi UNIX-like, mentre una partizione per gli OS derivati da UNIX è – come avevi già visto – una unità in Windows (solitamente A: per i floppy, C: per il primo "hard disk", eccetera ...).

I file-systems perciò sono relativi alle partizioni in UNIX ed alle unità in MS Windows.

Per chiarezza, e salvo quando diversamente indicato, quando in questa guida ho scritto "partizioni" ho inteso fare riferimento alle partizioni nel senso dei sistemi operativi UNIX che, in effetti, sono l'argomento della guida.

I nomi delle partizioni di disco in UNIX

Nel precedente paragrafo hai appreso che ogni disco contiene (al massimo) 4 slices all'interno delle quali si trovano le partizioni. Per identificare le differenti slice in un disco e le differenti partizioni in una slice si usano (a seconda dell'OS) valori progressivi numerici o letterali.

In FreeBSD per identificare in maniera assoluta una partizione nel sistema basta concatenare:

- il nome del dispositivo (vedi oltre)
- la lettera s , che sta per "slice"
- il numero della slice (a partire da 1)
- la lettera della partizione

In Linux invece per riferire una partizione si omette l'indicazione della slice:

- il nome del dispositivo (vedi oltre)
- il numero della partizione (a partire da 1)

Il nome di un dispositivo (di storage, ma non solo) è costituito da un prefisso indicante:

- il tipo del dispositivo stesso (vedi oltre)
- un valore progressivo (a seconda dell'OS, un numero a partire da 0 oppure una lettera a partire da a); se ad esempio nel computer sono installati tre lettori cd, il nome in UNIX di ciascuno di questi dispositivi potrebbe essere costituito dal prefisso comune a tutti i lettori cd seguito dalle cifre 0 (per il primo), 1 (per il secondo) e 2 (per il terzo lettore cd).

In FreeBSD (nel quale la distinzione dei dispositivi simili avviene tramite un numero a partire da 0) i prefissi dei più comuni dispositivi di memorizzazione di massa sono:

- ad hard disk ATAPI IDE
- da hard disk SCSI
- acd lettori cd-rom ATAPI IDE
- cd lettori cd-rom SCSI
- fd lettore di floppy disk

In Linux (che usa invece una lettera a cominciare dalla a per distinguere i dispositivi simili) invece i nomi sono:

- hd supporti di memorizzazione ATAPI IDE (hard disk, lettori cd, ...)
- sd supporti di memorizzazione SCSI (hard disk, lettori cd, ...)
- fd lettore di floppy disk



NOTA: SCSI O IDE?

Se non sai qual'è il tipo d'interfaccia ed il protocollo di trasferimento dei dati utilizzato da un tuo dispositivo, prova a cercare in rete le informazioni sul modello del disco.

Ad ogni modo, probabilmente è SCSI se il tuo è un computer portatile, IDE su un desktop pc.

Linux inoltre crea dei link simbolici chiamati ad esempio `cdrom` e `cdrw` per riferire i relativi dispositivi (indipendentemente se IDE o SCSI) che fanno da lettori cd e masterizzatori cd.

Per esempio, quindi:

- in FreeBSD `ad1s3e` riferisce la quinta (e) partizione della terza (3) slice del secondo (1) IDE hard disk (ad)
- in Linux `hdb5` riferisce la quinta (5) partizione del secondo (b) IDE hard disk (hd)

Nel paragrafo sull'organizzazione del root-filesystem hai già appreso che tutti i dispositivi sono rappresentati come files ed elencati nella directory `/dev` : in essa puoi trovarvi dunque anche le partizioni dei dispositivi di storage.

I file-systems montati automaticamente al boot

Al boot il sistema operativo monta (in generale) tutti i file-systems indicati nel file di configurazione `/etc/fstab` .

Tale file contiene una lista di righe nel seguente formato:

partizione punto-di-mount tipo-file-system opzioni dump-freq ordine

Il dispositivo, e dove montarlo

Il primo campo è il nome della partizione da montare (ad esempio `ad1s3e` secondo la forma ammessa da FreeBSD), mentre il secondo è il percorso del punto di mount (ad esempio `/mnt/mia_partizione`) che deve essere una directory già esistente.

Il tipo del file-system

Il terzo campo contiene il tipo del file-system da montare. I tipi di file-systems riconosciuti e quindi montabili variano a seconda del sistema operativo utilizzato. I più comuni sono:

- i file-system tipici in UNIX `ufs` , `ext2` e `ext3` (`ext3` è come `ext2` ma in più è `journalized`, cioè in sostanza più sicuro nei casi di crash del sistema o del suo spegnimento improvviso per mancanza d'alimentazione elettrica)
- la FAT32 (tipica dei sistemi MS Windows), chiamata `msdos` oppure `vfat` per il supporto ai nomi lunghi
- i dischi ottici `iso9660` oppure `cd9660`
- i file-systems remoti `nfs`
- l'area di paging usata dal sistema operativo chiamata `swap` .

Le opzioni

Attraverso le opzioni (quarta colonna) si può indicare con più precisione il modo col quale si vuole montare le partizioni. Ne esistono molte (se ne possono usare anche più d'una, separate da spazi), sia di specifiche del tipo di partizione sia generiche, ed al solito sono consultabili nel manuale. Le opzioni generiche più utilizzate sono:

- `ro` indica che la partizione è in sola lettura
- `rw` indica che la partizione può anche essere scritta
- `noauto` specifica che la partizione, benché sia elencata nel file, non deve essere montata automaticamente al boot del sistema operativo
- `sync` vuol dire che il flush delle scritture (descritto precedentemente) avviene immediatamente appena le scritture sono comandate
- `user` consente anche agli utenti non root di poter montare la partizione
- `umask=XXX` serve per specificare la maschera `XXX` dei permessi da negare ai files nella partizione

Un esempio di opzione specifica di un tipo di partizione è `utf8` sui file-systems `vfat` , che consente caratteri particolari (come ad esempio gli accentati) nei nomi di files.

Dumping & checking

Il quinto campo è la frequenza di dumping: il dump di un file-system è il backup dei files che contiene.

Il sesto campo, infine, serve a specificare l'ordine di scansione alla ricerca di errori (una sorta di scandisk periodico all'avvio del computer, effettuato dall'utility `fsck`):

- 0 se la partizione non deve essere mai controllata all'avvio
- 1 se la è la partizione del root-filesystem
- maggiore di 1 negli altri casi

Un valore minore (eccetto 0) produce come effetto che la partizione sarà controllata prima di altre.

Si ipotizzi che una certa partizione venga montata a `/mnt/x`, e che un'altra lo sia a `/mnt/x/y`. In tale caso il numero d'ordine si rende significativo: posto che per entrambe le partizioni debba ancora essere maggiore di uno, quello della partizione montata in `/mnt/x` ovviamente deve essere minore di quello della seconda partizione, altrimenti la ricerca degli errori non avverrebbe su tutto il filesystem.

Esempio

Nella seguente riga esemplificativa di `/etc/fstab` si specifica che la partizione `/dev/sda1` con filesystem `vfat` deve poter essere montata successivamente al boot (`noauto`) da tutti (`user`) in `/media/esterno`. Questa riga specifica inoltre che la codifica dei nomi dei files sia `utf8` (quindi col supporto alle lettere accentate), che tutti gli utenti dispongono di tutti i permessi sui files (`umask=000`) della partizione e che la partizione non sia oggetto né di backup (il primo 0) né di scansioni (il secondo 0) all'avvio:

```
/dev/sda1 /media/esterno vfat utf8,user,umask=000,noauto 0 0
```

Gli effetti di eventuali modifiche al file di configurazione `/etc/fstab` (che viene letto dal sistema al suo avvio) non sono però immediati: saranno visibili solo dal prossimo riavvio della macchina. Per aggiornare immediatamente l'insieme delle partizioni disponibili, si può ricorrere ad alcuni metodi alternativi tra loro:

- se l'opzione è riconosciuta, si lancia il comando `mount -a`
- in alcuni sistemi è necessario segnalare ad uno specifico programma in esecuzione – chiamato `mountd` – della nuova configurazione. Il metodo per inviare segnali ai processi è descritto in seguito

Creare, montare e smontare file-systems

Conoscere l'elenco dei file-systems correntemente montati:

```
mount
```

Per conoscere lo spazio libero su ogni file-system:

```
df
```

Per conoscere lo spazio libero su ogni file-system in maniera più comprensibile utilizzare l'opzione `h` (iniziale di "human"):

```
df -h
```

Montare un file-system:

```
mount opzioni partizione punto-di-mount
```

L'opzione `t` introduce il tipo di file-system da montare. Ad esempio, per montare in `/mnt/floppy` un classico floppy formattato per MS-DOS e fisicamente inserito nell'apposito lettore:

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

Al comando `mount` si possono passare anche altre opzioni di mounting, tra cui quelle viste durante l'analisi di `/etc/fstab`, tramite l'opzione `o` (seguito dall'effettiva opzione: `rdonly`, `sync`, ...).

Per montare un file-system indicato in `/etc/fstab` con l'opzione `noauto`, invece, basta passare al comando il solo punto-di-mount oppure il solo nome della partizione. Ad esempio:

```
mount /mnt/floppy
```

Per montare tutti i file systems presenti in `/etc/fstab` (come visto nel paragrafo precedente):

```
mount -a
```

Per smontare un filesystem:

```
umount punto-di-mount
```

Per creare il file immagine del floppy disk e del CD

```
cp /dev/fd0 file-immagine-floppy
```

```
cp /dev/cdrom file-immagine-cd
```



NOTA: IL FILE IMMAGINE DI UN DISCO

Il file immagine di un disco non è un'immagine grafica, una foto, un disegno. È tutt'altro: è un file che contiene i dati e la struttura tipica di un dispositivo di memorizzazione dati, come ad esempio un floppy, un CD o un DVD.

L'immagine di un disco è l'esatta copia digitale del disco, tramite la quale tutti i dati del disco stesso vengono salvati in un file che al suo interno ricalca esattamente la struttura del file-system e le informazioni di integrità dei dati presenti sul supporto.

Copiando ad esempio l'immagine di un CD in un file, dunque, sarà possibile rimontare in seguito il file-system nel file-immagine anche senza avere più a disposizione il vero e proprio CD.

Per montare i file immagine del floppy e del CD

```
mount -t vfat -o loop file-immagine-floppy punto-di-mount-floppy
```

```
mount -t iso9660 -o loop file-immagine-cd punto-di-mount-cd
```

Formattare un floppy disk nel lettore `/dev/fd0`

```
fdformat /dev/fd0
```

Creare partizioni (ad esempio una FAT32 con supporto ai nomi lunghi sul floppy disk precedentemente formattato)

```
mkfs.vfat /dev/fd0
```

Infine, attraverso il comando `mkisofs` è possibile creare immagini ISO di cd sulla base di files e directories su partizioni montate. Tale immagine può tralaltro definire un “bootable cd” (ossia un disco d'avvio per il computer).

Controllare e partizionare dischi

`fsck` e le sue versioni specializzate per i diversi tipi di partizione (`e2fsck`, `dosfsck`, ...) eseguono un controllo del dispositivo indicato come parametro in ingresso, alla ricerca (ed eventualmente correzione) dei blocchi danneggiati.

```
e2fsck /dev/hda3
```

Per modificare la tabella delle partizioni di un certo disco, cioè per “ripartizionarlo”:

```
fdisk disco
```

Sempre per il partizionamento, ma attraverso una comoda interfaccia grafica per Gnome, puoi usare `gparted` di cui ho anche inserito uno screenshot all'inizio della guida, nelle pagine sull'installazione dell'OS.

Mount di file-system remoti

Ipotizza di avere a tua disposizione due computer, chiamiamoli A e B, in una rete LAN già configurata (per il networking consulta l'apposito capitolo): puoi montare su A un file-system che si trova in una partizione di un dispositivo di storage fisicamente alloggiato nel computer B.



NOTA: CLIENT E SERVER

Nell'esempio appena fatto, B è detto “server” in quanto è l'entità che offre dei servizi all'esterno, mentre A è il “client” che accede ai servizi offerti dal “server”.

Esistono vari modi di montare file-system remoti, ognuno con un proprio protocollo di networking: in seguito verrà utilizzato NFS (Network File System).



NOTA: I PROTOCOLLI DI NETWORKING

Un protocollo di networking è sostanzialmente costituito dalle procedure e dalle regole con le quali dei computers dialogano tra loro in una rete.

Innanzitutto, B deve “esportare” (in linguaggio Microsoft Windows : “condividere”) tale file-system: le singole directories e gli interi file-systems da esportare devono essere indicati in `/etc/exports`.

Come nel caso di `/etc/fstab`, eventuali modifiche a `/etc/exports` non saranno considerate finché non viene opportunamente segnalato ad un componente del sistema di “rileggere” la configurazione (tale componente è `nfsd` in Linux, ed il già citato `mountd` in FreeBSD). Per l'invio di segnali ai processi, consulta il capitolo apposito.

Importare

Non appena B inizia ad esportare, ad esempio, la sua directory `/progetti`, allora un utente sul computer A può importarla montandola al percorso `/progetti_di_b`:

- lanciando il comando `mount`,
- indicando come tipo di file-system `nfs`,
- specificando come partizione il nome o l'indirizzo del server (cioè B, il cui indirizzo per esempio è 192.168.1.50),
- ed indicando infine il nome della directory esportata ed il mount-point:

```
mount -t nfs 192.168.1.50:/progetti /progetti_di_b
```



NOTA: GLI INDIRIZZI IP

Un computer in una rete è identificabile attraverso un indirizzo univoco. Generalmente il “protocollo di rete” utilizzato è l'IP – Internet Protocol – per il quale la forma degli indirizzi è quella degli “indirizzi IP”.

Un indirizzo IP (versione 4) è costituito da una quadrupla di numeri da 0 a 255 separati da punti.

Come modificare `/etc/exports`

Ora che hai chiaro il procedimento, occorre fare un passo indietro, ossia a “come” modificare il file delle esportazioni. Bisogna notare innanzitutto che la struttura di `/etc/exports` varia a seconda del sistema operativo.

- In Linux essa è orientata agli elementi da esportare: ogni riga contiene...
 - un mount point...
 - ... ed una lista di macchine (in inglese “host”) o gruppi di rete (introdotti dal simbolo @) autorizzati a montare il file system, riferibili per nome o indirizzo ed eventualmente con i caratteri jolly * e ? in modo da poter indicare più host con poco sforzo;
 - eventualmente, ogni macchina/gruppo può essere seguito da una lista di opzioni di mount (riportate tra parentesi, non introdotte dal meno e separate tra loro da virgole).

Esempio: esportazione della directory `/progetti` sia in lettura che in scrittura verso un primo host, ed in sola lettura verso un'altra seconda macchina; esportazione ancora verso il primo host anche della directory `/doc`.

```
/progetti    192.168.1.1 192.168.1.2(ro)
/doc         192.168.1.1
```

Per default la scrittura è consentita, ma può anche essere esplicitata utilizzando l'opzione `rw`. Per impedirla, come visto, usa l'opzione `ro` (“read-only”).

- In FreeBSD, invece, la struttura del file è orientata ai singoli host, dei quali non possono esserci doppioni su righe differenti: su ogni riga si specificano innanzitutto una serie di directories, quindi delle opzioni di esportazione, ed infine l'host o gli host a cui tale directories vanno esportate.

Dunque, ad esempio, le stesse esportazioni viste per Linux si esprimono in FreeBSD come segue (al solito l'indentazione non è significativa):

```
/progetti      /doc          192.168.1.1
/progetti      - ro         192.168.1.2
```

Altre opzioni generiche di esportazione in FreeBSD sono:

- `nohide` per esportare anche gli eventuali file-systems montati in sottodirectories
- `alldirs` per consentire all'host che importa di montare a partire da una sottodirectory dell'elemento esportato (cioè, se ad esempio si esporta `/x`, consente al client di montare a partire da `/x/y`)

Autenticazione

Prima di concludere è inoltre opportuno approfondire l'argomento delle credenziali dell'host client presso l'host che esporta, e come regolarle in `/etc/exports`.

Il comportamento normale che un utente si aspetterebbe è di poter accedere ai file nel server proprio come farebbe in un normale file system. Questo, però, richiede che siano usati gli stessi user-id (`uid`) e group-id (`gid`) sia sul cliente che sul server per gli stessi utenti e gruppi: questo chiaramente non è sempre vero.

Inoltre, non è desiderabile che l'utente `root` (`uid=0`) del client sia trattato come `root` anche quando accede ai file nel server NFS: per questo il `root` del client viene normalmente mappato come `nobody` (normalmente: `uid=-2` e `gid=-2`) sul server. Sempre salvo diversa indicazione in `/etc/exports`, invece, gli utenti non `root` vengono accreditati sul server con gli stessi `uid` e `gid` che hanno sul client.

Per modificare/eplicitare il comportamento standard in Linux si usano le opzioni:

- `root_squash` o `no_root_squash` per impostare esplicitamente o disabilitare lo “squashing” (schiacciamento) del `root` su `nobody`
- `squash_uids` e `squash_gids` per specificare un elenco di `uid` e `gid` che dovrebbero essere trasformati in `nobody`, come ad esempio:
`squash_uids=0-15,20,25-50`
- `all_squash` o `no_all_squash` per impostare o disabilitare esplicitamente lo “squashing” di tutti gli utenti
- `anonuid` e `anongids` per impostare `uid` e `gid` da applicare sul server agli utenti “schiacciati”
- `map_static=file-di-map` abilita la correlazione statica degli `uid` e dei `gid` come specificato nel file.

Il formato del file ha questo aspetto (al solito i cancelletti introducono commenti):

```
#      remoto      locale
uid    0-99        -           # schiaccia questi
uid    100-500    1000       # trasforma 100-500 in 1000-1500
gid    0-49      -           # schiaccia questi
gid    50-100    700       # trasforma 50-100 in 700-750
```

In FreeBSD invece:

- `maproot=uid` mappa il `root` del client sullo specifico `uid` del server (il `gid` è quello dell'utente identificato da `uid`)
- `mapall=uid` mappa tutti gli utenti del client, compreso `root`, sullo specifico `uid` del server (il `gid` è nuovamente quello dell'utente identificato da `uid`)

Processi

Visualizzazione dei processi e delle loro informazioni



NOTA: I PROCESSI

Un processo è un programma in esecuzione.

Per conoscere varie informazioni periodicamente aggiornate sui processi, si usa il seguente comando che tralaltro fornisce anche un prompt di comandi avanzati (premendo h si ottiene il menù delle azioni possibili):

```
top
```

Lista dei processi in esecuzione:

```
ps
```

Opzioni di ps :

- per stampare anche i processi in esecuzione degli altri utenti a
- per stampare differenti informazioni aggiuntive u oppure l
- per stampare anche la priorità dei processi (vedi oltre) 0 nice
- per elencare anche i demoni x



NOTA: I PROCESSI DEMONI

Un processo demone è un particolare processo sempre in esecuzione che offre dei particolari servizi (e per questo in Windows un demone viene chiamato “Servizio”) ad altri processi.

Proprio per via dal fatto che i demoni non sono direttamente utilizzati dall'utente, essi non hanno un'interfaccia-utente né grafica né da console, e sono controllabili solo attraverso i loro “front-end” cioè i processi che si frappongono tra essi e l'utenza.

Talvolta è necessario riferirsi a dei processi: ciò è possibile attraverso il loro numero identificativo PID (process-id).

Per scoprire il PID di un processo si può sfruttare ps in combinazione con grep passando a quest'ultimo il comando (o parte di esso) utilizzato per lanciare tale processo in esecuzione:

```
ps -aux | grep processo
```

Un altro modo per conoscere il PID di un processo di cui si conosce il nome è il ricorso al comando pidof . Il seguente esempio richiede il PID di ogni istanza della shell bash in esecuzione. Se non ve ne sono, non verrà restituito alcunché. Se ve ne sono, pidof le elenca separate da uno spazio.

```
pidof bash
```

Per conoscere l'utilizzo della memoria:

```
free
```

Per conoscere l'elenco dei processi in esecuzione strutturato a forma d'albero (in modo da poter vedere le relazioni tra processi padri e figli):

```
pstree
```



NOTA: PARENTELE TRA PROCESSI

Immagina che un processo A lanci in esecuzione un altro processo B: allora A sarà il “processo padre” di B e coerentemente B sarà detto “processo figlio” di A.

Risalendo l'albero di parentela di tutti i processi si trova il processo init (che ha PID 1).

Invio di segnali

In alcuni casi è necessario inviare segnali ai processi, come ad esempio in FreeBSD per segnalare a mountd che /etc/fstab è stato modificato. Per inviare un segnale ad un processo occorre innanzitutto recuperare il PID del processo attraverso uno dei metodi disponibili.

Una volta scoperto il PID puoi inviare un segnale in questa maniera:

```
kill -segnale pid
```

Riprendiamo l'esempio della notifica a mountd . Poniamo che tale processo abbia PID=178 : dobbiamo dunque lanciare tramite il comando kill il segnale di aggiornamento (HUP in FreeBSD , SIGHUP in Linux) al processo numero 178 .

```
kill -HUP 178          # in FreeBSD
```

```
kill -SIGHUP 178      # in Linux
```

Per conoscere l'elenco dei segnali:

```
kill -l
```

Per scoprire il significato dei segnali:

```
man signal
```

Esistono due categorie di segnali:

- gli “interpretabili” (ad esempio HUP / SIGHUP) al cui arrivo il processo destinatario decide se ignorarli o come agire di conseguenza
- i “non interpretabili” (ad esempio KILL / SIGKILL) che hanno un significato ben preciso da rispettare (KILL / SIGKILL provocano l'uccisione del processo, cioè la sua terminazione forzata)

Processi in background e foreground

Elenca i processi in background (in gergo “jobs”) o stoppati:

jobs

Premendo **Ctrl+C** si uccide il processo in esecuzione.

Premendo **Ctrl+Z** si sospende il processo in esecuzione, e si ottiene un identificatore che serve per riferirlo qualora lo si rinvoglia mandare in esecuzione.

Per rimandare in esecuzione in foreground un processo sospeso usa `fg`, mentre per rimandarlo in esecuzione in background usa `bg`:

`fg id-processo-sospeso`

Per lanciare un comando sin dall'inizio in background

comando &

Per eseguire un processo ad un'ora predeterminata e gestire queste operazioni pianificate (anche dette “scheduled”) sono disponibili svariati comandi, tra i quali `at`, `cron` e `crontab`: consulta il manuale per saperne di più in materia.

Quando un utente effettua il logout, i programmi che avvia vengono implicitamente terminati. Per lanciare un comando e far sì che la sua esecuzione prosegua anche dopo un eventuale logout, occorre passarlo a `nohup` e richiederne l'esecuzione in background

`nohup comando &`

Gestione della priorità



NOTA: LA PRIORITÀ DI UN PROCESSO

Un sistema multiprogrammato consente l'esecuzione contemporanea (o per essere più preciso, apparentemente tale) di più processi: ognuno di essi viene fatto eseguire al processore, il quale passa all'esecuzione del successivo processo al verificarsi di particolari condizioni oppure alla fine della “quota di tempo continuato di esecuzione sul processore” concessa al processo corrente.

Questo intervallo di tempo e la frequenza con la quale un processo vengono selezionati per essere eseguiti dipendono dalla priorità del processo: se esso ha una maggiore priorità (tipica dei programmi necessari al buon funzionamento del sistema operativo) sarà eseguito più spesso di altri, velocizzandone perciò l'esecuzione complessiva (ovviamente a scapito dei processi con priorità minore).

Nei sistemi UNIX si può impostare la priorità di un processo, sommandole un valore numerico chiamato “nice” (in Linux va da -20 a +19): il risultato va inteso in senso inverso, ossia al suo crescere diminuisce la priorità del processo (e viceversa).

Si può lanciare un comando assegnandogli un certo “nice” da sommare alla priorità iniziale di default: se il valore è negativo si ottiene la velocizzazione dell'esecuzione del processo, se è nullo non si ottiene alcuna differenza rispetto al comportamento normale, altrimenti si produce un rallentamento nell'esecuzione.

`nice -n priorità comando-da-eseguire`

Attraverso il comando `renice` è invece possibile modificare la priorità di uno o più processi già in esecuzione, in base al PID, all'utente proprietario oppure al gruppo proprietario.

Per modificare la priorità del processo con un certo PID

`renice nice -p pid`

Si può indicare anche più di un processo

`renice nice -p pid-1 pid-2`

Per modificare la priorità dei processi di un certo utente

`renice nice -u utente`

Si possono anche realizzare istruzioni più complesse che modificano la priorità in base a più criteri di selezione: PID, utente e gruppo. Ad esempio il comando seguente imposta il nice ad 1 (quindi rallentandoli) per i processi con pid 987 e 32, e per i processi degli utenti “giulia” e “diego”

`renice +1 -p 987 -u giulia diego -p 32`

Occorre osservare che solo l'utente `root` può sia aumentare che diminuire la priorità di tutti i processi, mentre gli altri utenti possono solo diminuire la priorità dei propri processi.

Lanciare programmi all'avvio

Per specificare che il sistema deve compiere dei determinati compiti ad ogni suo avvio, li si può inserire – ad esempio – nello script di configurazione `/etc/rc.local` prima della riga:

`exit 0`

Occorre però verificare che lo script abbia il permesso di esecuzione da parte dell'account proprietario (cioè dell'amministratore, che è anche l'unico che può modificare quel file).

Questo script potrà tornarti utile per molti comandi di networking, i cui effetti sono validi solamente nella sessione di lavoro corrente e cioè fino allo spegnimento del calcolatore.

Nota, infine, che il momento del login di un utente (magari in ambiente grafico) è differente dall'avvio del sistema operativo, dunque non ha senso voler lanciare l'internet browser Firefox da `/etc/rc.local`. I vari desktop manager dispongono di applicazioni dedicate alla specifica dei programmi che si desidera vengano eseguiti all'avvio del desktop manager stesso.

Networking

Conoscere e configurare le interfacce



NOTA: LE INTERFACCE DI RETE

Un'interfaccia di rete è un dispositivo che permette al computer di collegarsi in una rete di computers. Esempi di interfacce sono le classiche schede di rete cablate con porta RJ-45, le interfacce per reti wireless, eccetera.

Ogni interfaccia può essere connessa ad una rete (a prescindere se locale o pubblica) ed ha dunque un proprio indirizzo di rete (generalmente IP come descritto in precedenza in questa guida).

Inoltre, ogni interfaccia ha anche un indirizzo di livello fisico chiamato MAC Address: il motivo per cui siano necessari questi (ed altri) indirizzi esula dall'argomento di questa guida.

Il sistema operativo offre inoltre un'interfaccia aggiuntiva virtuale (con indirizzo IP fisso 127.0.0.1), detta di loopback, che permette la connessione a sé stesso: può sembrare inutile, ma non lo è affatto.

Ogni interfaccia di rete è contenuta come ogni altro dispositivo (reale o virtuale) dentro `/dev`, ed nomi di tali dispositivi è composto da:

- una prima parte fissa rappresentativa del tipo di interfaccia. Ad esempio:
 - `eth` interfacce che usano il protocollo fisico "Ethernet", sia cablate che wireless: le più diffuse. Attenzione: in alcuni sistemi le interfacce wireless hanno un altro prefisso
 - `ppp` interfacce che usano il protocollo fisico "Point-to-Point Protocol" per comunicare. In seguito vedrai dei comandi che creano interfacce `ppp` virtuali sopra una vera interfaccia Ethernet (PPPoE: PPP-over-Ethernet)
 - `lo` interfacce di loopback
- una seconda parte nella quale è riportato un numero progressivo (a partire da 0) che serve a distinguere un'interfaccia tra le altre dello stesso tipo

Quindi ad esempio `eth0` è la prima interfaccia Ethernet.

Per conoscere le informazioni di base sulle proprie interfacce di rete (come ad esempio l'indirizzo IP) :

`ifconfig`

Da un punto di vista più "hardware", invece, si possono visualizzare i dispositivi di rete presenti nel calcolatore:

`lshw -C network`

Se non riceve alcun argomento, `ifconfig` mostra solamente lo stato delle interfacce attualmente definite.

Se invece gli viene passato in ingresso un nome di interfaccia mostra lo stato solo dell'interfaccia specificata:

`ifconfig interfaccia-di-rete`

Disponendo di privilegi di amministrazione ed aggiungendo altre opzioni al comando precedente, si può configurare l'interfaccia (le modifiche hanno effetto solo nella sessione di lavoro corrente). Si può ad esempio renderla promiscua (cioè fa rsì che essa propaghi all'OS anche i pacchetti che sono destinati ad altre interfacce magari di altri computer), attivarla/disattivarla e molto altro ancora.

```
ifconfig interfaccia-di-rete promisc      # modalità promiscua (necessaria per lo "sniffing")
ifconfig interfaccia-di-rete -promisc    # modalità non promiscua
ifconfig interfaccia-di-rete up          # attivazione
ifconfig interfaccia-di-rete down        # disattivazione
```

Si può anche impostare che un'interfaccia operi col protocollo IPv6 al posto del classico IP (versione 4), e quindi configurarla di conseguenza.

Per impostare staticamente l'indirizzo IP di una specifica interfaccia di rete:

`ifconfig interfaccia-di-rete nuovo-indirizzo-ip`

Si possono altresì indicare la `netmask` (cioè la maschera utilizzata per capire se un dato indirizzo appartiene alla rete locale o non ne fa parte) e l'indirizzo di broadcast (l'indirizzo usato per inviare dei messaggi in un solo "colpo" a tutte le interfacce in ascolto nella rete locale).

Esempio: assegnazione dell'indirizzo IP 192.168.1.2, creazione della maschera di rete 255.255.255.128 (in questo modo la classe C è divisa in due sottoreti da 126 indirizzi disponibili ciascuna), assegnazione di 192.168.0.127 come indirizzo di broadcast, ed attivazione dell'interfaccia:

```
ifconfig eth0 192.168.1.2 netmask 255.255.255.128 broadcast 192.168.0.127 up
```

La configurazione delle interfacce di rete attraverso `ifconfig`, come hai già letto, è applicata dall'OS solamente fino alla chiusura della sessione di lavoro durante la quale tale configurazione è stata impartita.

Per configurare in maniera definitiva le interfacce è possibile percorrere tre strade alternative:

- usare software con interfaccia grafica, che solitamente è sviluppato in maniera tale da memorizzare le configurazioni anche per i riavvii successivi del computer. Questo metodo è però molto variabile tra i diversi sistemi (ed in particolare tra i differenti desktop manager: Gnome, KDE, ...), in compenso il suo uso è abbastanza intuitivo
- modificare il file di configurazione delle interfacce: pure in questo caso continuano ad esserci differenze (anche notevoli), ma al variare di OS e non più di desktop-manager.

Tale file si chiama:

- `/etc/rc.conf` in FreeBSD
- `/etc/network/interfaces` in Linux

Qualunque sia il file, esso è modificabile senza troppe difficoltà tenendo presente i parametri di `ifconfig` e le altre righe già presenti nel file di configurazione.

- usare lo script `/etc/rc.local` già descritto (ma poiché esiste un altro file per configurare le interfacce, è meglio non

ricorrere a questo metodo)

Per ottenere dinamicamente da un server apposito (server DHCP) l'indirizzo IP ed altre informazioni come la netmask ed assegnare tutto ciò ad una interfaccia :

```
dhclient interfaccia
```

Mentre invece per cancellare le impostazioni ottenute dinamicamente via DHCP:

```
dhclient -r interfaccia
```

Wireless



NOTA: LE RETI WIRELESS

Le interfacce di rete wireless permettono la connessione a reti di computer senza la necessità di stendere cavi.

Esistono due tipi di reti wireless: con infrastruttura oppure ad-hoc. Il primo tipo prevede la presenza di un veloce nodo centrale della rete (chiamato Access-Point, in breve AP) che smista le comunicazioni e fa da ponte verso un'eventuale rete cablata. Una rete wireless del secondo tipo può invece essere realizzata da pochi host tra di loro.

Di qualunque tipo sia, una rete wireless ha:

- un nome identificativo chiamato **ESSID**
- un canale e la corrispondente frequenza di trasmissione nell'etere
- il bitrate, cioè la velocità di trasmissione
- dei parametri (eventualmente disabilitati) di autenticazione e di cifratura dei dati trasmessi

Una rete wireless con infrastruttura è contraddistinta inoltre dall'indirizzo fisico **BSSID** dell'Access Point.

Per scoprire quali interfacce di rete offrono connettività wireless e per recuperare alcune informazioni sulla loro attuale configurazione:

```
iwconfig
```

Per cercare le reti wireless disponibili in zona:

```
iwlist interfaccia scan
```

Nel seguito di questo paragrafo si ipotizza che l'unica interfaccia wireless disponibile sia `eth1`, e dunque nel caso d'esempio:

```
iwlist eth1 scan
```

Attivazione e configurazione d'esempio dell'interfaccia:

```
ifconfig altra-interfaccia down      # Disattivo le altre interfacce (eth0 ...)
dhclient -r eth1                       # Rilascio la configurazione DHCP pre-esistente
ifconfig eth1 up                       # Attivo l'interfaccia
iwconfig eth1 essid 'prova'            # Imposto l'ESSID sul valore specificato, racchiuso tra apici
iwconfig eth1 rate 11M                 # Imposto il bit-rate su 11 Mbit/secondo
iwconfig eth1 key off                   # Nessuna cifratura dei dati
iwconfig eth1 mode Managed              # In modalità normale
dhclient eth1                           # Ottengo tramite DHCP un ind. IP per l'interfaccia
```

Si può anche specificare a quale Access Point ci si vuole collegare, specificandone il **BSSID**:

```
iwconfig eth1 ap bssid
```

Per default, il valore del parametro `ap` è `any`.

Anche in questo caso le configurazioni durano il tempo della sessione nella quale sono state assegnate, svanendo allo spegnimento del computer. Per renderle persistenti, modifica (ad esempio in Linux) `/etc/network/interfaces`.

Reti wireless protette

Oltre alla conoscenza dei comandi illustrati sinora per l'associazione ad una rete wireless non protetta (cioè senza meccanismi di autenticazione o cifratura dei dati), è oggi di notevole importanza la capacità di collegarsi a reti senza fili protette.

I protocolli di autenticazione e cifratura sono svariati, ma più frequentemente si ha a che fare con i casi seguenti.

WEP

Richiede l'indicazione della password, che puoi immettere sia in formato esadecimale sia come stringa di testo:

```
iwconfig interfaccia key chiave-in-esadecimale
```

```
iwconfig interfaccia key s:testo-chiave
```

Alcune schede di rete richiedono inoltre che venga lanciato il comando

```
iwconfig interfaccia key restricted
```

WPA

Per poter comunicare in una rete protetta con WPA dal proprio host è innanzitutto necessario aver installato `wpa_supplicant` ed aver creato il file `/etc/wpa_supplicant.conf`.

Questo file va dunque aperto e modificato al fine di configurare l'associazione dell'interfaccia alla rete wireless.

Dopo una serie di istruzioni generali, puoi definire una serie di blocchi `network`, una per ogni rete wireless protetta alla quale ti vuoi collegare. L'ordine di questi blocchi è significativo: se durante la lettura del file da parte del sistema due o più blocchi possono essere usati (poiché la rete individuata dall'ESSID è presente) allora l'interfaccia sarà associata solo alla prima rete descritta da questi blocchi.

Il protocollo WPA è complesso e con diverse funzionalità le cui descrizioni esulano da questa guida, perciò per semplificare ipotizzo che la protezione della rete alla quale desideri collegarti sia basata su una semplice password (chiamata PSK, pre-shared key). La

PSK deve essere lunga almeno 8 caratteri e non più di 63.

Poiché la cifratura della password richiede un certo impegno delle risorse del calcolatore, si può risparmiare questo tempo perso cifrando direttamente la PSK utilizzando il comando `wpa_passphrase`, che restituisce un codice di 64 cifre esadecimali che andrà inserito nel file di configurazione.

Alternativamente, in tale file si può inserire la password tra doppi apici.

Le procedure da seguire differiscono lievemente a seconda della versione del protocollo WPA usato nella rete:

- **WPA 1**

```
ap_scan=1
ctrl_interface=/var/run/wpa_supplicant
network={
    ssid=essid-tra-doppi-apici
    scan_ssid=0 # mettere 1 se la rete non è pubblicizzata dall'AP
    proto=WPA
    key_mgmt=WPA-PSK
    psk=psk
    pairwise=TKIP
    group=TKIP
}
```
- **WPA 2**

```
ctrl_interface=/var/run/wpa_supplicant
network={
    ssid=essid-tra-doppi-apici
    psk=psk
    key_mgmt=WPA-PSK
    proto=RSN
    pairwise=CCMP
}
```

Una volta configurato `/etc/wpa_supplicant.conf` basterà lanciare i comandi (nota l'assenza di spazi tra le opzioni ed i rispettivi valori):

```
wpa_supplicant -w -Ddriver -iinterfaccia -c/etc/wpa_supplicant.conf -ddhclient interfaccia
```

Dove `driver` è il nome del driver dell'interfaccia, che puoi scoprire eseguendo `lshw -C network` e cercando il valore dell'attributo `driver` nella campo `configuration` della interfaccia wireless in questione.

DSL



NOTA: HAI BISOGNO DI LEGGERE QUESTO PARAGRAFO?

Hai un modem DSL? Se la risposta è no, puoi saltare al prossimo paragrafo: questo tratta comandi che non ti servono.

Hai un router tra il modem DSL e la rete locale (ad esempio il tuo computer)? Se la risposta è sì, puoi anche in questo caso saltare al prossimo paragrafo: sarà il router, automaticamente, a compiere le operazioni descritte in questo paragrafo.

Se invece hai un modem DSL ma non disponi del router, continua a leggere!

I modem DSL sono abbastanza diffusi sul mercato e forse ne usi uno anche tu: potrebbe risultarti utile sapere come configurare la connessione ad internet tramite la scheda di rete.

Occorre innanzitutto lanciare (come al solito da super-user) il comando `pppoeconf`. Questo programma richiede l'inserimento di alcune informazioni: username, password, DNS, ... ed infine ti chiede se vuoi che la connessione sia aperta automaticamente all'avvio del sistema (boot).

Per avviare manualmente la connessione DSL dopo averla configurata:

```
pon dsl-provider
```

Nota: con il comando precedente viene tralaltro creata una nuova interfaccia `ppp` virtuale gestita dal demone `pppd`.

Per chiudere la connessione DSL, invece:

```
poff dsl-provider
```

Per leggere il log delle connessioni DSL:

```
plog
```

Nomi di host e DNS

Attraverso il comando `host` è possibile conoscere l'indirizzo IP di una macchina a partire dal suo nome simbolico (e viceversa).

Esempio:

```
host abcd.qualche_sito.it
host 123.45.67.89
```

La traduzione dal nome simbolico all'indirizzo IP avviene in più fasi:

- se le impostazioni di default in `/etc/host.conf` sono state lasciate intatte, in una prima fase l'OS verifica se il nome simbolico non sia già tradotto staticamente in `/etc/hosts`: se ciò fosse vero il sistema riesce a tradurre immediatamente il nome senza dover accedere alla rete, con un conseguente guadagno in velocità. Ciò comporta che se accedi spesso ad un host remoto, puoi indicarne in `/etc/hosts` la traduzione del nome simbolico in indirizzo IP. La sintassi di questo file è molto semplice da comprendere e dunque non dovresti aver problemi a modificare il file nel caso tu ne avessi bisogno.

- in un secondo tempo viene tentata la traduzione dinamica attraverso l'uso del protocollo DNS, la cui configurazione (come ad esempio gli indirizzi IP dei server DNS da usare) si trova in `/etc/resolv.conf`. In questo file, oltre alle righe vuote ed ai commenti introdotti dal cancelletto `#`, si trovano delle righe così formate:

comando parametro

Esistono sostanzialmente due tipi di comando:

- `nameserver`, che attende come parametro l'indirizzo IP di un server DNS
- `domain`, che attende come parametro il dominio di default che viene aggiunto ai nomi simbolici di host che ne sono sprovvisti. Ad esempio, se la tua macchina si chiamasse `a.miodominio.it`, puoi riferire `b.miodominio.it` semplicemente chiamandolo `b` se hai definito il comando `domain miodominio.it`

Per conoscere il nome dell'host su cui si è loggati:

```
hostname
```

Per impostare il nome dell'host

```
hostname nuovo-hostname
```

Ping, traceroute, tabella di routing e tabella di ARP

Per verificare che una scheda Ethernet sia configurata correttamente, devi fare due prove:

- prima, effettua un ping verso l'interfaccia stessa (utilizzando l'indirizzo dell'interfaccia, recuperabile da `ifconfig`)
`ping indirizzo-interfaccia`
se non riesce, vuol dire che l'interfaccia è configurata erroneamente
- poi un ping verso un'altra macchina sulla rete locale
`ping host`
se non riesce, vuol dire che l'interfaccia è di per sé configurata bene, mentre i problemi di comunicazione possono dipendere da altri fattori, tra i quali: il mezzo trasmissivo potrebbe non funzionare, forse l'host specificato effettivamente non esiste, la configurazione dell'interfaccia potrebbe essere differente rispetto a quella della rete locale, ...

Il comando `ping` continua il suo test (ossia l'invio di pacchetti ICMP ECHO_REQUEST e l'attesa delle risposte) finché non viene terminato forzatamente, oppure fino a che non ha inviato un numero `N` di pacchetti indicato in ingresso al comando:

```
ping -c N host
```

Attraverso il comando `traceroute` è invece possibile scoprire uno dei possibili percorsi compiuti dei pacchetti per arrivare ad una certa destinazione, dove per "percorso" ovviamente si intende la sequenza di host intermedi attraversati. Per ciascuno di essi vengono effettuate tre prove, di cui vengono restituiti i tempi necessari per raggiungerli. Occorre notare che alcuni host intermedi possono non "dichiararsi" per motivi di sicurezza o prestazioni.

```
traceroute abcd.qualche_sito.it
```



NOTA: LA TABELLA DI ROUTING

Il routing è il processo di selezione di un percorso per arrivare ad una destinazione.

La tabella di routing serve a memorizzare, per ciascun host di destinazione conosciuto, qual'è l'interfaccia migliore sulla quale inviare i messaggi ad esso mandati affinché arrivino il più prima possibile.

Per leggere la tabella di routing:

```
netstat -r
```

Per velocizzare la stampa della tabella di routing, si può indicare di non convertire gli indirizzi in nomi simbolici specificando l'opzione `n`.

Attraverso il comando `route` si possono aggiungere delle entrate della tabella di routing:

- instrada i pacchetti diretti ad un certo host attraverso l'interfaccia `eth1`
`route add -host 192.128.2.1 dev eth1`
- instrada i pacchetti diretti ad una certa rete (con una certa netmask) attraverso l'interfaccia `eth0`
`route add -net 192.56.76.0 netmask 255.255.255.0 dev eth0`
- aggiunge il default gateway, ovvero la route da usare in tutti gli altri casi non previsti da route specifiche
`route add default gw 192.168.2.1`

Notare che è necessario inserire anche la route all'indirizzo specificato come default gateway, dichiarando l'interfaccia che va utilizzata.

Al posto dell'indirizzo numerico, si può inserire il nome simbolico del default gateway, purchè sia poi "traducibile" dal sistema grazie ad un'apposita entrata nel file `/etc/hosts`

Nota: anche il comando `route` permette di agire sulla tabella di routing corrente, ma le modifiche non sono persistenti.



NOTA: ARP (ADDRESS RESOLUTION PROTOCOL)

ARP è un protocollo che permette all'OS di scoprire l'indirizzo fisico (MAC Address) corrispondente ad un indirizzo IP dato.

Per leggere la tabella di ARP dove si trovano le traduzioni correnti tra indirizzi IP e MAC:

```
arp -a
```

Per leggere nella tabella di ARP l'entrata relativa ad uno specifico host:

```
arp -a host
```

Attraverso il comando `arp` puoi anche aggiungere (`-s indirizzo-IP indirizzo-fisico`) e rimuovere (`-d indirizzo-IP`) entrate della tabella di ARP.

Statistiche di rete

Stato di tutte le interfacce di rete

```
netstat -i
```

Stato di una specifica interfaccia di rete

```
netstat -I interfaccia
```

Traffico dei pacchetti su tutte le interfacce di rete, aggiornato ogni n secondi (indica anche il numero di pacchetti scartati)

```
netstat -w n -d
```

Traffico dei pacchetti su una specifica interfaccia di rete, aggiornato ogni n secondi

```
netstat -w n -I interfaccia
```

Utilizzando l'opzione s puoi recuperare le statistiche sull'utilizzo di specifici protocolli di rete o famiglie di protocolli.

Esempio 1 (FreeBSD): mostra le statistiche sull'utilizzo del protocollo TCP, e quindi le azzera

```
netstat -szp tcp
```

Esempio 2 (FreeBSD): mostra le statistiche sull'utilizzo della famiglia di protocolli INET

```
netstat -sf inet
```

Un'interessante software con interfaccia grafica è `etherape`, che mostra graficamente l'attività sulla rete.

Sniffing

`tcpdump` è uno sniffer da console che permette di monitorare il traffico di rete con filtri flessibili per limitare l'output a video secondo varie regole di matching di pacchetti.

Mostra solo gli header dei pacchetti:

```
tcpdump
```

Opzioni varie:

- mostra anche il contenuto dei pacchetti `-x`
- stampa solo i primi N bytes del contenuto dei pacchetti `-x -s N`
- sniffing su una specifica interfaccia `-i interfaccia`
- per terminare dopo lo sniffing di n pacchetti `-c n`
- salvare il dump dello sniffing in un file per analizzarlo successivamente `-w percorso-file-dump`
- ricaricare dump per analizzarlo `-r percorso-file-dump`

Per recuperare solo i pacchetti che soddisfano determinate condizioni puoi utilizzare i filtri di selezione, che si passano a `tcpdump` sotto la forma di una stringa.

Esempio 1: sniffing dei pacchetti TCP inviati da un certo host da una certa porta

```
tcpdump "tcp src 10.20.30.40 src port 50"
```

Esempio 2: sniffing dei pacchetti TCP inviati da un certo host ad un certo altro su una data porta

```
tcpdump "tcp src 10.20.30.40 dst 10.20.30.1 dst port 700"
```

Esempio 3: sniffing dei pacchetti TCP provenienti da uno di due specifici host

```
tcpdump "tcp and (src 10.20.30.40 or src 10.20.30.1)"
```

Uno sniffer con interfaccia grafica molto diffuso è, invece, `ethereal`.

Firewall

Il firewall `ipfw` (non presente in tutti i sistemi derivati da UNIX, ma comunque utile per comprendere la logica di gestione di un firewall), utilizzabile da console, scarta i pacchetti in base alle loro proprietà, ossia senza considerarne i contenuti, siano essi innocui oppure dannosi per la sicurezza del sistema.

Le proprietà considerate sono solitamente:

- l'indirizzo e/o porta di provenienza,
- l'indirizzo e/o porta di destinazione,
- il protocollo utilizzato,
- il tipo di pacchetto.

Il firewall opera in base ad un elenco di regole, il cui ordine è significativo: ogni regola ha un numero che la identifica, ed in caso di regole contrastanti vale quella con ID minore.

`ipfw` è un firewall "esclusivo", in quanto l'ultima regola (la numero 65535) prevede l'accettazione di tutti i pacchetti, ed è quindi compito dell'amministratore indicare le regole di esclusione dando loro dei numeri identificativi minori, cioè specificare i casi nei quali il firewall deve scartare i pacchetti.

Stampa l'elenco delle regole attive del firewall:

```
ipfw show
```

Su ogni riga dell'elenco delle regole sono presenti:

- l'ID della regola
- il numero dei pacchetti che hanno matchato la regola
- la dimensione in kilobytes dei pacchetti che hanno matchato la regola
- la regola

Ogni regola è composta da due elementi:

- l' azione, che può essere, tra le altre:

- allow (o equivalenti: accept , pass , permit) : accetta il pacchetto
- deny (o l'equivalente drop) : scarta il pacchetto
- unreach : come deny , ma in più segnala all'host sorgente che la destinazione è irraggiungibile
- la descrizione dei pacchetti interessati dalla regola

Esempio 1: scarta i pacchetti IP dall'host 10.20.30.40 all'host 120.3.4.5 (non indicando una porta, si intende tutte le porte)
deny ip from 10.20.30.40 to 120.3.4.5

Esempio 2: scarta i pacchetti TCP dalla porta 80 dell'host 10.20.30.40 alla macchina sulla quale gira il firewall

deny tcp from 10.20.30.40 80 to me

Esempio 3: scarta i pacchetti provenienti da qualunque host che avviano connessioni TCP con “questo” host

deny tcp from any to me setup

Esempio 4: scarta tutti i pacchetti TCP da qualunque host a “questo” host che appartengono ad una connessione TCP già stabilita

deny tcp from any to me established

Esempio 5: scarta tutti i pacchetti ICMP di ECHO REPLY da qualunque host a “questo” host (consulta il manuale di ipfw per conoscere tutti gli icmptypes)

deny icmp from any to me icmptypes 0

Per aggiungere una regola si usa il predicato add seguito dall'ID, dall'azione e dalla descrizione e dalla regola:

ipfw add id azione descrizione

L'ID della regola in realtà è facoltativo, e se omissso sarà assunto uguale a quello dell'ultima regola inserita, incrementato di 100.

Esempio: consente solo ICMP ECHO REQUEST (e la corrispondente risposta) a questo host

ipfw add 50 allow icmp from any to me icmptypes 8

ipfw add allow icmp from me to any icmptypes 0

ipfw add deny all from any to any

Elimina la regola corrispondente ad un certo ID

ipfw delete id

Eliminare tutte le regole (eccetto l'ultima, incancellabile):

ipfw flush

Un firewall con interfaccia grafica è firestarter .

Applicazioni varie di rete



NOTA: LA DOCUMENTAZIONE

Per l'utilizzo di tutte le applicazioni seguenti, ed in particolare per la configurazione dei demoni, rimando alla lettura di documentazioni specifiche ed alle fonti citate nella bibliografia poiché insegnarti ad usarli non è lo scopo di questo testo.

Instant messaging e chat

Per la ricezione dei messaggi sulla console:

- mesg y consente la ricezione di messaggi
- mesg n impedisce la ricezione di messaggi
- mesg stampa se la ricezione dei messaggi sulla console è consentita

Se sono stati installati, talk ed il corrispondente demone talkd permettono di dialogare da console con altre persone. Gli interlocutori possono:

- sia trovarsi sullo stesso sistema
talk nome-utente
- sia operare su un altro host di cui occorre fornire il nome simbolico o l'indirizzo IP
talk nome-utente@altro-computer

Esistono tantissimi programmi con interfaccia grafica per chat e instant messaging, molto più user-friendly di talk e tra questi vi sono:

- gaim, kopete, amsn e tanti altri: accedono a servizi di instant messaging molto diffusi
- planimo è un instant messenger per reti locali

irc è un client IRC in esecuzione su console, mentre loqui ha una più usabile interfaccia grafica.

Posta elettronica

Sono numerosissimi anche i client di posta elettronica. Se disponi di un sistema con interfaccia grafica, non hai motivo di imparare ad usare i client a riga di comando (come mail , mailx e pine), dato che puoi utilizzare, ad esempio, il ben più intuitivo mozilla-thunderbird .

HTTP: Web client & server

wget è una utility da console per scaricare documenti dal web, che segue i collegamenti nelle pagine HTML e crea una versione locale del sito web remoto. Questo programma ricrea completamente la struttura di files e directories del sito originale, in modo da permettere una lettura offline delle pagine web. Ne esiste anche una versione con interfaccia grafica: si chiama gwget .

Ma a parte wget , i client HTTP più usati sono i web browser... e ne esistono parecchi, sia per ambienti ad interfaccia grafica (mozilla-firefox , opera , konqueror , ...) che per quelli testuali (come lynx).

Il più noto e diffuso programma che consente connessioni HTTP al proprio sistema (“web server”) è Apache: il suo demone si chiama httpd o apache2 a seconda delle versioni.

Può essere molto utile installare un motore di interpretazione di script server-side da affiancare al web-server, come ad esempio php5 : in questa maniera potrai usare anche pagine dinamiche nei siti web ospitati sul server.

Shell remote

Per avviare una shell remota (già descritta) occorre usare `telnet` , mentre il corrispondente demone `telnetd` consente connessioni remote al proprio host. Attraverso programmi come `telnet` è possibile tra le altre cose connettersi a specifiche porte sugli host remoti e quindi ai servizi su esse forniti.

Al fine di avviare una shell remota che sia criptata, invece, bisogna utilizzare `ssh` (al solito il demone è `sshd`):

`ssh nome-utente@server-a-cui-connettersi`

FTP client & server

Per stabilire una connessione ad una directory FTP per caricarvi o scaricarvi files va usato `ftp` , mentre per consentire connessioni FTP al proprio computer puoi installare il demone `vsftpd` .

`ftp server-a-cui-connettersi`

Bibliografia e fonti varie

Per la scrittura di questa guida ho tratto spunto da:

- lezioni del corso “Organizzazione di Sistemi Operativi e Reti” di Ing. Informatica all'Università di Pisa
- manuale di FreeBSD 6.0
- manuale di Mandrake Linux 10.1 (ora Mandriva)
- manuale di Ubuntu Linux 6 e 7
- <http://www.univ.trieste.it/~nircdc/doc/baseunix/>
- <http://www.univ.trieste.it/~nircdc/doc/oldunix/>
- <http://openskills.info/topic.php?ID=156>
- <http://www.to.infn.it/groups/group4/mirror/linux/AppuntiLinux/AL-indgen.html>
- http://ubuntuguide.org/wiki/Ubuntu_dapper_it
- <http://help.ubuntu.com/>
- WikiPedia, sia [EN] sia [IT]
- <http://ubuntuforums.org/showthread.php?t=571188>

Registro delle modifiche (changelog)

I cambiamenti apportati a questa guida sono stati:

- versione 4 del 2008-03-05
reti wireless protette; lshw; aggiunta di route nella tabella di routing; dhclient; lanciare programmi all'avvio del sistema e del desktop manager; rilettura totale con correzione di errori vari e rielaborazione delle frasi; eliminazione del make
- versione 3.1 del 2007-11-05
riformulazione degli spunti sui boot loader ; wireless: rate, ap any, key; nota sull'utilità dei comandi per i modem DSL; rielaborazione delle note sugli OS di riferimento e su quello consigliato
- versione 3 del 2007-07-04
ristrutturazione della guida (capitoli, paragrafi ed i sottoparagrafi per i paragrafi lunghi), cambio del modo in cui sono rappresentati i tasti (da un font apposito a immagini), ridimensionamento del testo, introduzione delle note informatiche, spiegazione UNIX Linux e distribuzioni, dettagli sul partizionamento durante l'installazione dell'OS, riscrittura del paragrafo sulle opzioni, correzioni in merito alle redirezioni, correzioni tra i comandi di VI, spiegazione delle slices sui dischi e della diversa terminologia in materia tra UNIX e Windows, eliminazione m-tools, reti wireless, correzione di errori vari (sia di digitazione, sia semantici)
- versione 2.2 del 2006-11-01
du su files, approfondimento sul comando shutdown, cenno a nano, cenno ai tools per lo sviluppo, approfondimento di yes, cenno a addgroup, split, controllo e partizionamento di dischi, pidof, at e crontab, irc
- versione 2.1 del 2006-09-28
approfondimento dei comandi di VI
- versione 2 del 2006-08-09
generale rielaborazione sintattica, sistemi con/senza GUI, ottenere ed installare il sistema operativo, introduzione alla gestione delle installazioni, boot-loader, rielaborazione console e terminali, opzioni dei comandi, approfondimento del root file system, approfondimento della redirezione di stdin-stdout-stderr, rielaborazione dei permessi e dei flags dei files, rielaborazione della gestione degli utenti, visualizzare e nascondere i numeri di riga in VI, approfondimento dell'archiviazione e della masterizzazione, rielaborazione dei nomi delle partizioni sul disco, approfondimento del mounting, creare e montare immagini di dischi, formattare partizioni, approfondimento di nice e renice, approfondimento della traduzione dei nomi simbolici di host in rete, ampliamento delle applicazioni varie di rete, aggiunti in più punti della guida altri comandi ed in alcuni casi anche le loro opzioni principali, e numerosi piccole altre aggiunte
- versione 1.2 del 2006-06-11
free, exit, halt, reboot, configurazione ed uso di connessioni DSL, accenno ai browser, type, opzioni di cat, arch, ricorsività di chmod, du, fuser, ls -color, accreditarsi come amministratori, opzioni di uname, montare il file immagine di un cd, pstree, head, tail, file, strings, mkfs
- versione 1.1 del 2006-05-06
uname, yes, permessi in forma numerica, sticky-bit e bit suid-sgid, umask, hostname, traceroute, ipfw flush
- versione 1 del 2006-01-12