Linux Base - Capitolo n. 5

Edizioni ByteMan (14-11-2005)

revisione: 25/01/2008

Configurazione (parte 1)

Windows tiene tutte le configurazioni in un unico file binario, chiamato **registro**, che per essere modificato necessita di un apposito strumento software: **regedit**.
Linux mantiene i file di configurazione globali nella directory **/etc/** sotto forma di normalissimi file di testo modificabili con un comune **editor**.
Volendo fare una analogia con vecchi ricordi legati al DOS si tratta di una serie di file simili ai vari **autoexec.bat** e **config.sys**.
Ma occorre tenere presente che a causa della multiutenza di Linux è molto spesso possibile per ciascun utente scegliere i settaggi che ritiene più appropriati per un determinato programma senza alterare le impostazioni globali, mettendo un opportuno file di configurazione nella propria home directory. In genere questi file sono invisibili (iniziano cioè con un punto **"."**) ed hanno lo stesso nome del loro analogo di /etc/ valido invece per tutto il sistema (globale).

In questa sezione passeremo in rassegna i principali file di configurazione. È da tenere presente che per molti di essi è disponibile una man page che ne spiega il formato, accessibile con:

```
man nomefile
```

o nel caso di omonimia con un comando o una funzione, con:

```
man 5 nomefile
```

Qui daremo soltanto una descrizione sommaria di base, per questo vale sempre la pena controllare la suddetta documentazione che contiene tutti i dettagli.
Esistono delle convenzioni generali, abbastanza seguite, valide per ciascuno di questi file, come ignorare le righe vuote o considerare il carattere # come l'inizio di un commento.

/etc/fstab

Questo file, il cui nome sta per ***f**ile **s**ystem **tab**le* contiene le indicazioni dei filesystem disponibili per il sistema. È il file che il programma di avvio **init** va a leggere quando deve montare i dischi ed è pertanto di importanza fondamentale.

Può essere modificato con i diritti di amministratore, ma può essere letto con il comando:

```
cat /etc/fstab
```

Il formato del file è molto semplice, ogni linea definisce un filesystem (vedi seminario 2) che sta su un device, linee vuote o che iniziano per **#** vengono ignorate, i campi di ogni linea sono separati da spazi o tabulatori. La man page **man fstab** ne spiega i dettagli.

Vediamone un esempio:

```
# /etc/fstab: static file system information.
#
# file system   mount point     type      options                       dump  pass
/dev/hdb5       /               ext2      defaults,errors=remount-ro     0     1
/dev/hdb6       none            swap      sw                             0     0
proc            /proc           proc      defaults                       0     0
/dev/fd0        /floppy         auto      defaults,user,noauto           0     0
/dev/cdrom      /cdrom          iso9660   defaults,ro,user,noauto        0     0
/dev/sr0        /mnt/cdrom      iso9660   defaults,ro,user,noauto        0     0
/dev/hdb1       /boot           ext2      rw                             0     2
```

```
/dev/hda1       /mnt/win        vfat    defaults,user,noauto            0       0
/dev/hdc4       /mnt/zip        auto    defaults,user,noauto            0       0
```

La tabella è stata incolonnata per comodità didattica, nella realtà i vari campi possono essere separati al minimo con uno *spazio* o un carattere *tab*. Talvolta l'area delle opzioni, sempre per comodità di impaginazione, risulterà spezzata su più righe, questo nella realtà non avviene, ogni riga è un elemento completo della tabella.

Come si può notare ogni linea contiene 6 campi:

1. **file system**: il device su cui si trovano i file
2. **mount point**: la directory in cui il device deve essere montato
3. **type**: il tipo di filesystem
4. **options**: le opzioni di montaggio
5. **dump**: se effettuare un dump del filesystem
6. **pass**: l'ordine in cui eseguire il check del filesystem all'avvio

Analizziamo i vari campi con qualche dettaglio:

- Il primo campo descrive il ***device*** su cui sta il file system da montare, nel caso in questione si hanno due hard disk (**/dev/hda** e **/dev/hdb** con varie partizioni), un floppy (**/dev/fd0**), uno zip (**/dev/hdc4**), un CDROM ed un masterizzatore scsi (**/dev/cdrom** e **/dev/sr0** (nel caso del CDROM si è usato un link simbolico).
  Quando si vorrà aggiungere un altro device e montarlo occorrerà allora determinare qual'è il suo corrispondente file di dispositivo e inserirlo in questo campo.
  Si noti ancora che, avendo abilitato il supporto nel kernel, è posibile montare anche il filesystem **/proc** che contiene una serie di file virtuali generati al volo dal kernel per accedere ad alcune delle informazioni interne.
  Se ci fossero stati dei file montati via NFS (cioè file condivisi sulla rete) si sarebbero avuti anche campi del tipo:

  ```
  192.168.1.20:/home    /mnt/nfs    nfs    defaults,user,noauto      0    0
  ```

- Il secondo campo indica il ***mount point*** cioè la directory dove i file del nuovo dispositivo saranno resi disponibili.
  Se il filesystem non deve essere montato, come nel caso della partizione di swap, si usa la parola chiave **none**.

- Il terzo campo, ***type***, indica il tipo di filesystem che sta sul device che si vuole montare. I tipi più comuni sono **ext2** (il filesystem standard di Linux), **vfat** (il filesystem di Windows), **iso9660** (il filesystem dei CDROM). Ma ne esistono altri, non usati nell'esempio, come **nfs**, o **hfs** che è il filesystem del MacOS; oppure i nuovi filesystem per Linux: **ext3** e **reiserfs**.
  Si noti come in un caso sia presente invece la parola chiave **swap** ad indicare che in quel caso il dispositivo non contiene un filesystem, ma va usato per la swap.
  È inoltre possibile usare la parola chiave **auto** per dire al sistema di cercare di determinare automaticamente il tipo di filesystem.

- Il quarto campo, ***options***, indica le opzioni con cui si può montare il filesystem, esse devono essere indicate con una lista di valori separati da virgole, i valori possibili sono indicati nella man page del comando **mount** che riassumiamo brevemente di seguito:

  - ***user*** consente anche agli utenti normali di montare il filesystem
  - ***nouser*** non consente agli utenti normali di montare il filesystem
  - ***dev*** consente l'uso di file di dispositivo sul filesystem
  - ***nodev*** non consente l'uso di file di dispositivo sul filesystem
  - ***auto*** tutti i filesystem contrassegnati con questa opzione vengono montati

dal comando ***mount -a*** che viene eseguito all'avvio del sistema.

- ***noauto*** il filesystem deve essere montato esplicitamente
- ***exec*** consente l'esecuzione di programmi sul filesystem
- ***noexec*** non consente l'esecuzione di programmi sul filesystem
- ***suid*** consente che i bit suid e sgid abbiano effetto
- ***nosuid*** non consente che i bit suid e sgid abbiano effetto
- ***sync*** tutto l'I/O sul filesystem deve essere sincrono
- ***async*** tutto l'I/O sul filesystem deve essere asincrono
- ***ro*** monta il filesystem in read-only
- ***rw*** monta il filesystem in read-write
- ***default*** usa le opzioni di default: ***rw***, ***suid***, ***dev***, ***exec***, ***auto***, ***nouser***, e ***async***. *Nel caso in cui venga usata questa opzione, successive specificazioni di altre opzioni sovrascrivono i corrispondenti valori di default*.

- Il quinto campo, ***dump***, relativo alla manutenzione del filesystem, indica se il filesystem deve venire preso in esame dal comando dump per il backup.
- Il sesto campo, ***pass***, riguardante la manutenzione, indica la sequenza con cui il comando **fsck** deve eseguire i controlli sull'integrità dei filesystem. Il numero **1** deve essere assegnato al filesystem di boot, mentre i restanti avranno il numero **2**; il valore **0** permette di evitare ogni tipo di controllo automatico all'avvio.

Dal punto di vista dell'amministrazione base si ha a che fare con **/etc/fstab** tutte le volte che si aggiunge un disco, o un nuovo dispositivo, o si cambiano le partizioni. In questo caso occorre identificare qual'è il file di dispositivo da usare e scegliere nel filesystem una directory su cui montarlo. Deve poi essere specificato il filesystem da usare (o **auto** se si vuole tentare il riconoscimento automatico).

Nell'esempio si noti come per zip e floppy si sia consentito agli utenti di montare il filesystem, ma si sia disabilitato il montaggio all'avvio, e pure il controllo dello stato del filesystem, dato che non è detto che il floppy o lo zip siano sempre inseriti nel driver.

Lo stesso vale per il CDROM e il masterizzatore, per i quali si è pure aggiunto l'opzione di montaggio in read-only. Si noti inoltre l'opzione speciale per il filesystem di root, per il quale si è indicato di rimontare il filesystem in read only nel caso di errori.

Nel caso di disco fisso andrà poi scelto se montarlo all'avvio o meno, e in questo caso usare il sesto campo per indicare in quale ordine rispetto agli altri dovrà essere effettuato il controllo del filesystem (il primo deve essere il filesystem usato come radice).

/etc/mtab

Questo file contiene la "**m**ounted filesystem **tab**le", cioè informazioni, aggiornate dinamicamente in tempo reale, sui file system montati su un sistema Unix in genere.
Viene usato da alcuni programmi che hanno bisogno di queste informazioni, ma viene generato automaticamente e **NON deve** essere alterato. Può essere letto con il comando:

```
cat /etc/mtab
```

Il suo formato è simile a quello di **/etc/fstab** (che però ha una natura statica), ma alcuni campi hanno un significato diverso, nell'ordine:

1. **file system**: il device su cui si trovano i file

2. **mount point**: la directory in cui il device è montato
3. **type**: il tipo di filesystem utilizzato
4. **options**: le opzioni di montaggio
5. **dump**: frequenza di dump fatti (in giorni)
6. **pass**: numero di passaggi in fsck paralleli

Nell'esempio riportato, di seguito, possiamo notare, tra le altre cose, la presenza del filesystem **reiserfs**, l'assenza della partizione di **swap** che viene usata senza essere montata, la presenza di **usbs** e del dispositivo di **automount** che consente il riconoscimento automatico non appena viene inserita la chiavetta usb.

```
# file system   mount point     type      options                          dump  pass

/dev/hda7       /               reiserfs  rw                                0     0
proc            /proc           proc      rw,nodiratime                     0     0
sysfs           /sys            sysfs     rw                                0     0
tmpfs           /dev/shm        tmpfs     rw                                0     0
/dev/hda1       /mnt/hda1       reiserfs  rw,nosuid,nodev                   0     0
/dev/hda2       /mnt/hda2       reiserfs  rw,nosuid,nodev                   0     0
/dev/hda3       /mnt/hda3       reiserfs  rw,nosuid,nodev                   0     0
/dev/hda6       /mnt/hda6       reiserfs  rw,nosuid,nodev                   0     0
/dev/hda8       /mnt/hda8       reiserfs  rw,nosuid,nodev                   0     0
/dev/hdc1       /mnt/hdc1       reiserfs  rw,nosuid,nodev                   0     0
/dev/hdc2       /mnt/hdc2       reiserfs  rw,nosuid,nodev                   0     0
/dev/hdc3       /mnt/hdc3       ext3      rw,nosuid,nodev                   0     0
devpts          /dev/pts        devpts    rw,gid=5,mode=620                 0     0
usbfs           /proc/bus/usb   usbfs     rw,devmode=0666                   0     0
/dev/sdc        /mnt/sdc        vfat      rw,noexec,nosuid,nodev,sync,
                                          uid=1000,gid=1000,
                                          shortname=mixed,quiet,
                                          umask=0002,showexec               0     0
automount(pid3531) /mnt/auto    autofs    rw,fd=4,pgrp=3531,
                                          minproto=2,maxproto=4             0     0
```

Può risultare istruttiva la comparazione con il corrispondente **fstab** letta nello stesso computer:

```
# /etc/fstab: filesystem table.
#
# file system   mount point     type      options                          dump  pass
/dev/hda7       /               reiserfs  defaults                          0     1
proc            /proc           proc      defaults                          0     0
/dev/fd0        /floppy         vfat      defaults,user,noauto,
                                          showexec,umask=022                0     0
usbfs           /proc/bus/usb   usbfs     devmode=0666                      0     0
sysfs           /sys            sysfs     defaults                          0     0
tmpfs           /dev/shm        tmpfs     defaults                          0     0
/dev/cdrom      /cdrom          iso9660   defaults,ro,users,noexec,noauto   0     0
/dev/cdrom1     /cdrom1         iso9660   defaults,ro,users,noexec,noauto   0     0
/dev/dvd        /dvd            iso9660   defaults,ro,users,noexec,noauto   0     0
# Added by KNOPPIX
/dev/hda1       /mnt/hda1       reiserfs  auto,users,exec                   0     0
# Added by KNOPPIX
/dev/hda2       /mnt/hda2       auto      auto,users,exec                   0     0
# Added by KNOPPIX
/dev/hda3       /mnt/hda3       reiserfs  auto,users,exec                   0     0
# Added by KNOPPIX
/dev/hda5       none            swap      defaults                          0     0
# Added by KNOPPIX
/dev/hda6       /mnt/hda6       reiserfs  auto,users,exec                   0     0
# Added by KNOPPIX
/dev/hda8       /mnt/hda8       reiserfs  auto,users,exec                   0     0
# Added by KNOPPIX
/dev/hdc1       /mnt/hdc1       reiserfs  auto,users,exec                   0     0
```

```
# Added by KNOPPIX
/dev/hdc2       /mnt/hdc2       reiserfs  auto,users,exec                0      0
# Added by KNOPPIX
/dev/hdc3       /mnt/hdc3       ext3      auto,users,exec                0      0
# Added by KNOPPIX
/dev/sdc        /mnt/sdc        vfat      noauto,users,exec,umask=000    0      0
```

Gli editor

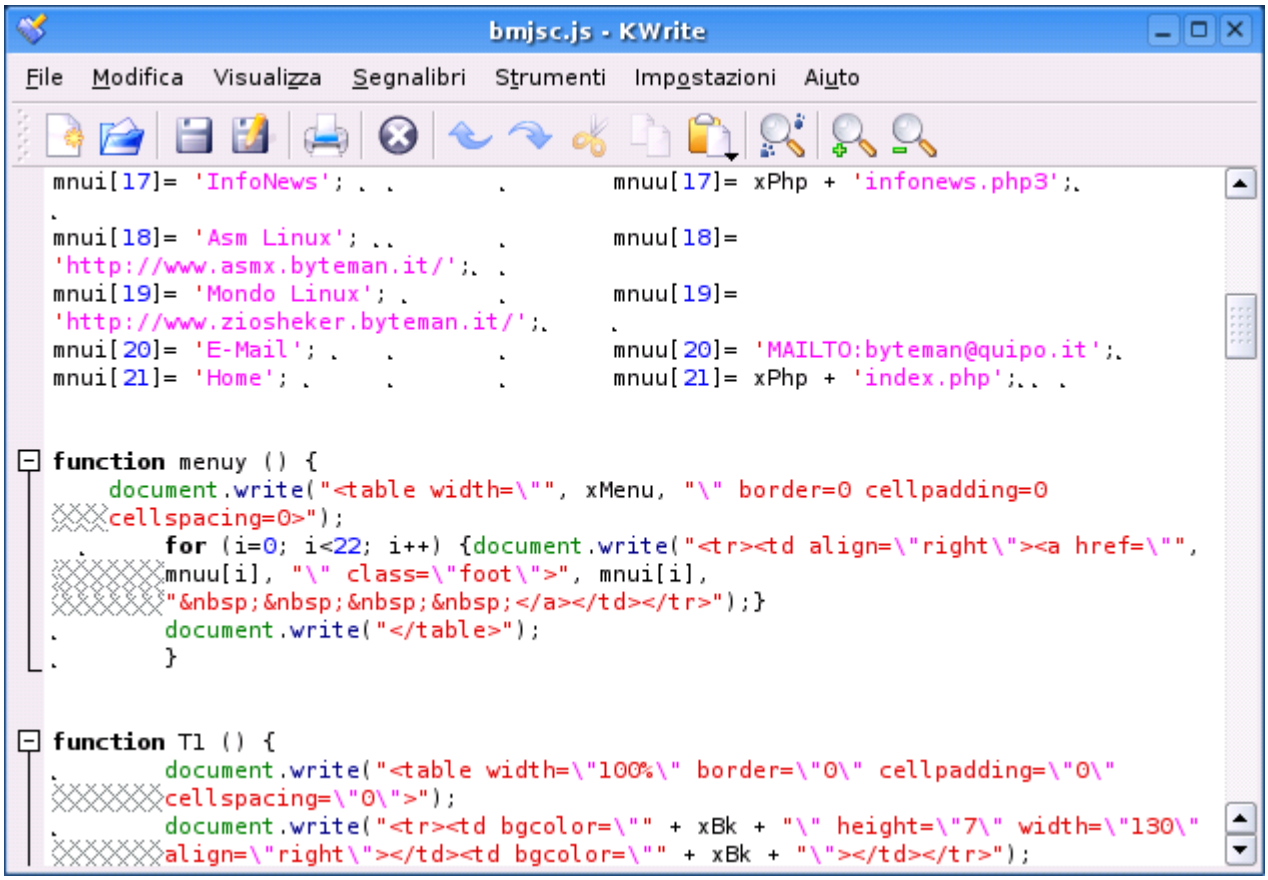
Strumento essenziale nel mondo dell'informatica, gli editor non possono mancare in Linux, anzi probabilmente sono tantissimi e rischiano di far perdere l'orientamento a chi inizia.
Ne esistono sia con interfaccia grafica, sia a linea di comando. Ma per motivi di essenzialità ne indicheremo solo due: **kwrite** (grafico) e **vim** (a linea di comando).

kwrite

Tipico dell'interfaccia **kde**, l'editor **kwrite** è facilissimo da usare soprattutto per chi è abituato ai vari *notepad* o *wordpad*. I paragoni, in questi casi, sono molto relativi, ma **kwrite** fornisce tutto quello che ci si potrebbe aspettare da un editor di testi evoluto:

- copia/incolla
- drag&drop
- manuale nella propria lingua
- possibilità di attivazione a linea di comando
- sintassi colorata in quanto è in grado di riconoscere in modo preciso le primitive di diversi linguaggi: ADA, Perl, C/C++, Pascal e HTML
- riconfigurabilità delle combinazioni di tasti
- ricerca e sostituzione
- opzioni di lavoro personalizzabili
- ... e molto altro ancora

Ecco una classica schermata dell'editor in funzione con un file javascript:



vim

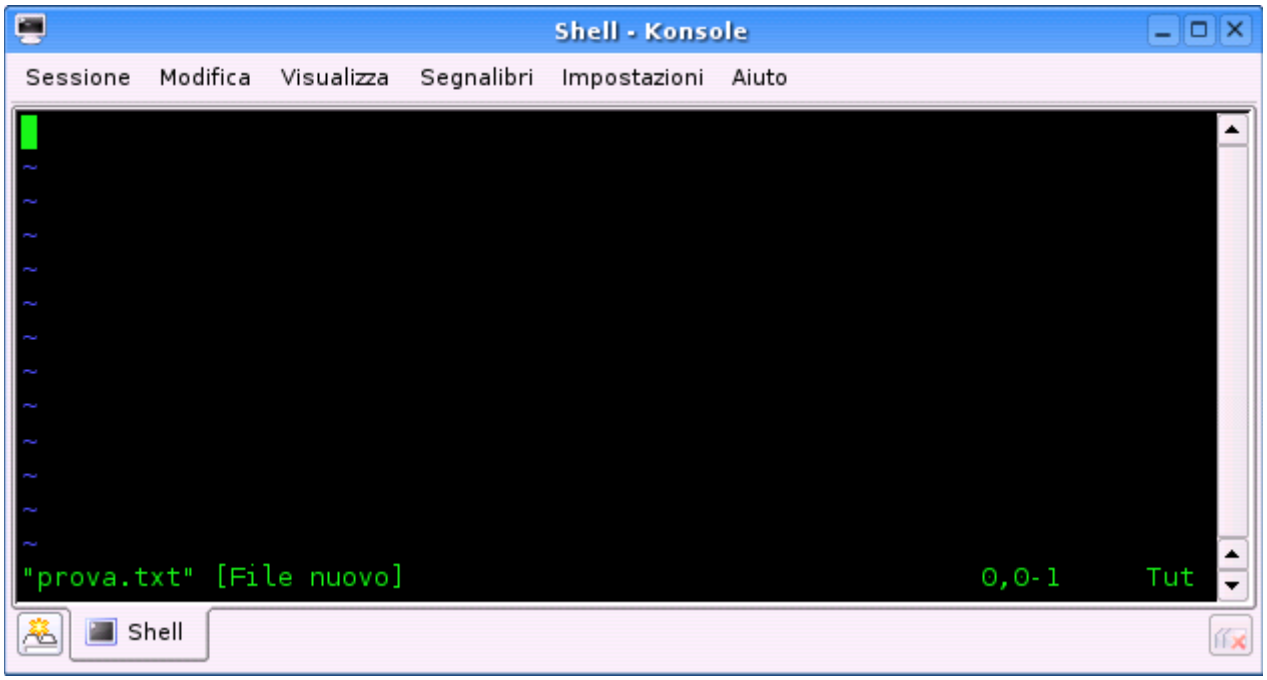
Come si è già capito dagli incontri precedenti Linux nasce come tutti gli Unix con l'interfaccia a linea di comando, e tutte le funzionalità del sistema sono accessibili fin da quel livello. Questo è particolarmente vero per quanto riguarda l'amministrazione di sistema per tutta una serie di motivazioni:

- I vari tool grafici mettono a disposizione solo un limitato insieme di opzioni, e non danno mai il livello di controllo ed affidabilità che si può raggiungere operando direttamente sui file di configurazione con speciali editor creati appositamente.
- Quando un eventuale problema di qualche tipo su disco avrà danneggiato qualche file essenziale o bloccato il sistema all'avvio si potrà sempre usare una distribuzione su floppy per rimettere le cose a posto, ma in questo malaugurato caso non saranno disponibili editor grafici.

Abbiamo visto che una delle caratteristiche fondamentali di Linux è quella di tenere tutte le configurazioni in file di testo, accessibili e modificabili con un qualunque editor, ed in genere raccolti nella directory /etc/. Per questo lo strumento principale di ogni amministratore di sistema è l'editor di testi, possibilmente a linea di comando.

Noto inizialmente con il nome **vi** è stato uno dei primi editor evoluti presenti in un sistema Unix.
Deriva dagli editor di linea e ne eredita alcune caratteristiche, in particolare il fatto di essere un **editor modale**, in cui cioè i comandi e il loro effetto dipendono dalla corrente modalità di operazioni (stato) in cui si trova il programma.
Questa caratteristica lo rende senz'altro il meno intuitivo e uno dei più difficili da usare per il novizio, succede spesso infatti che al primo impatto non si riesca neanche ad uscire dall'editor.
Il principale vantaggio è che essendo molto leggero, lo si trova installato praticamente su qualunque sistema, spesso anche nella versione più moderna nota con il nome di **vim**.
Anche se ne faremmo volentieri a meno, occorre tuttavia conoscerlo se non altro per affrontare situazioni di emergenza.

Ecco una classica schermata dell'editor in funzione dopo che è stato avviato con: **vim prova.txt**:



Come già detto **vim** è un editor modale, il comando **vim nomefile**, apre il file in ***modalità comando*** in una finestra principale, dove compare il testo (in caso di nuovo file le righe vuote sono sostituite da **~**) ed in cui ci si muove con le frecce, tralasciando libera l'ultima linea, usata per dare i comandi o ricevere le informazioni.

Tutti i comandi di **vim** sono eseguiti con pressioni di singoli tasti, ma la possibilità di dare il comando dipende dalla modalità in cui si trova l'editor al momento. I modi sono sostanzialmente due: ***comando*** ed ***inserimento***. Quando ci si trova in modalità comando occorre scrivere qualcosa e questo viene mostrato nella riga finale.

Una delle cose da capire subito con **vim** è che una volta che si è entrati in modalità inserimento non è possibile dare più alcun comando (cosa spesso molto irritante perchè rende impossibile ai neofiti uscire dall'editor), occorrerà prima tornare in modalità comando (premendo il tasto **ESC**).

Ricordiamo subito che **ESC** commuta in ***modalità comando*** mentre **i** commuta in ***modalità inserimento***. Molti comandi iniziano con il carattere **:** (due punti). Quella che segue è una lista dei comandi principali:

`:ex nomefile`	*apertura file*
`:w`	*salva*
`:w nomefile`	*salva con nome*
`:q`	*uscita se non ci sono modifiche*
`:qw`	*uscita dopo aver salvato*
`:q!`	*uscita senza salvare*
`u`	*undo dell'ultima modifica*
`yy`	*copia una riga*
`p`	*incolla la riga*
`dd`	*taglia una riga*
`d`	*cancella un carattere*
`/testo`	*ricerca testo*
`:h`	*help*
`i`	*entra in modo inserimento*
`ESC`	*entra in modo comandi*

Per maggiori approfondimenti è meglio consultare sia le **manpages** relative a **vim** sia delle apposite pubblicazioni, come quella citata in **Schede e Guide**, e... *good luck, o veramente!!!*

Help - Aiuto - HowTo

Certamente lo studio e la conoscenza di Linux richiedono un impegno non indifferente e tra le *difficoltà*, paradossalmente, c'è quella dell'eccesso di informazioni tra cui cercare aiuto e chiarimenti durante il proprio percorso di apprendimento. E' importante quindi avere dei punti di riferimento sicuri, all'interno del proprio sistema, il luogo più comodo dove ricercare documentazione, ma forse proprio per questo meno ovvio e quindi spesso ignorato.

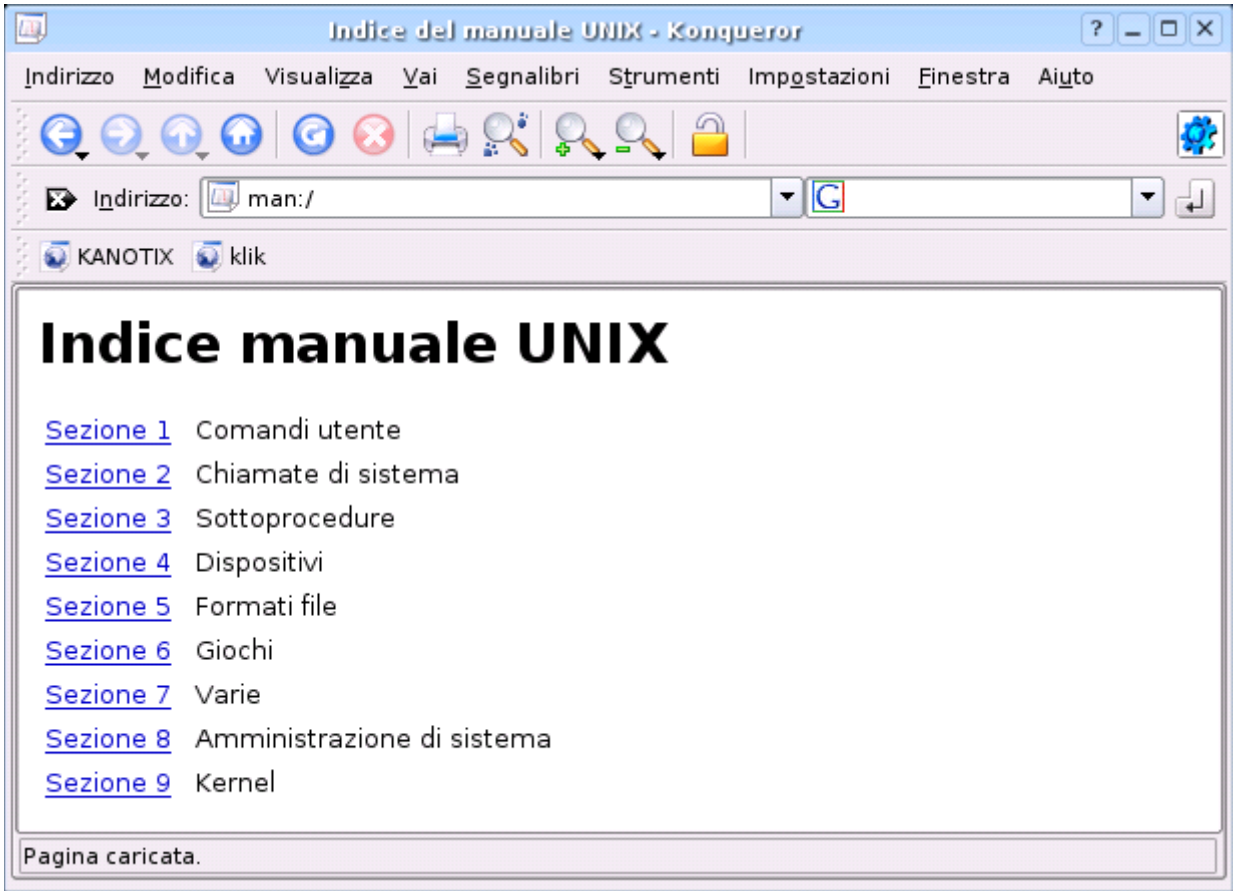
man

man, abbreviazione di manual, e' la forma tradizionale della documentazione in linea per i sistemi operativi Unix e Linux. Le *man pages* sono file formattati in modo speciale, ne esistono per la maggior parte dei programmi e vengono distribuite assieme al software. Sono scritte originariamente in inglese, ma se la distribuzione in uso ha la localizzazione si possono consultare nella propria lingua. Per l'italiano, in distribuzioni basate su Debian (Knoppix, Kanotix, Mepis, Ubuntu,...) è sufficiente installare il paccheto **manpages-it**
Come già visto nel seminario 1, digitando **man nomecomando** verra' visualizzata la pagina del manuale relativa a **nomecomando**.

Poiche' sono veramente molte, le pagine del manuale vengono raggruppate in sezioni numerate. Questo sistema e' utilizzato da cosi tanto tempo che spesso si incontreranno dei riferimenti con l'indicazione **man**(1) a significare che il comando è documentato nella sezione 1. Specificare la sezione nella quale man ricerca la pagina da visualizzare e' utile nel caso di argomenti multipli con lo stesso nome.

Sezione 1	*comandi utente (soltanto l'introduzione)*
Sezione 2	*chiamate di sistema*
Sezione 3	*chiamate delle librerie C*
Sezione 4	*dispositivi (es.: hd, sd,)*
Sezione 5	*formati dei file e protocolli (es.: wtmp, /etc/passwd, nfs)*
Sezione 6	*giochi (introduzione)*
Sezione 7	*convenzioni, pacchetti macro, etc. (es.: nroff, ascii)*
Sezione 8	*amministrazione di sistema (soltanto l'introduzione)*
Sezione 9	*kernel*

Oltre che dalla linea di comando, se si utilizza come browser **Konqueror**, è possibile visualizzare la pagina man di programma scrivendo nella barra degli indirizzi **#programma** oppure **man:/programma**. Scrivendo invece solamente **#** oppure **man:/** verrà visualizzato il menu generale delle pagine man, nel quale si potrà navigare, molto più comodamente, seguendo i link tra le varie sottosezioni.



info

Un'altra fonte di informazioni, prevalentemente in lingua inglese, è accessibile dal proprio sistema tramite il comando **info nomecomando**, dove nomecomando è il nome del programma di cui si vuole visualizzare la pagina info.
Se si utilizza come browser **Konqueror**, è possibile visualizzare la pagina info di un programma scrivendo nella barra degli indirizzi **info:/programma**. Scrivendo invece solamente **info:/** verrà visualizzato l'elenco principale delle pagine info.

doc

Altra documentazione generica dei programmi installati, solitamente disponibile solo in inglese, la si può trovare in differenti directory a seconda della distribuzione. Generalmente si trova comunque nella directory **/usr/share/doc/** oppure in **/usr/doc/**.

-h --help

Innumerevoli comandi hanno la possibilità di richiamare un piccolo help integrato nel comando stesso, semplicemente scrivendo il comando seguito da --help o -h.
Ricordiamo che in Linux le opzioni di ciascun comando vengono contrassegnate con un segno **-** (meno) seguito da una lettera. Con il passare del tempo questo sistema, seppure molto veloce, ha reso piuttosto criptiche alcune righe di comando a causa dell'eccesso di opzioni. Per motivi di chiarezza, non di velocità naturalmente, sono state introdotte le opzioni lunghe contrassegnate da **--** (meno meno) seguito da una parola.
Per le applicazioni in ambiente grafico è solitamente previsto un menu chiamato Aiuto o Help che contiene informazioni sul programma in uso. Si noti che se è presente solo la voce Help invece della voce tradotta (Aiuto) è probabile che il programma non sia stato tradotto o che manchi un aiuto in linea in italiano o che non sia presente una corretta localizzazione italiana del sistema.

HOWTO

Si tratta di guide monografiche riguardanti vari aspetti del sistema GNU/Linux, del Software Libero e dell'informatica in generale. A seconda della distribuzione utilizzata è possibile trovarli in posizioni diverse del proprio filesystem, in inglese, ma spesso anche in italiano se è stato installato il pacchetto opportuno e se il documento è stato tradotto. Per Debian e distribuzioni basate su Debian esistono i pacchetti **doc-linux-it**, in formato HTML tradotti in italiano, e **doc-linux-it-text**, in formato testo. Una volta installati tali pacchetti, le traduzioni italiane degli HOWTO saranno disponibili rispettivamente nelle directory **/usr/share/doc/HOWTO/it-html/** e **/usr/share/doc/HOWTO /it-txt/**; se invece si installa la versione inglese degli HOWTO, in HTML attraverso il pacchetto doc-linux-html o in testo semplice attraverso il pacchetto doc-linux-text, questi saranno disponibili rispettivamente nelle directory /usr/share/doc/HOWTO/en-html/ e /usr/share/doc/HOWTO/en-txt/.

whatis

Cerca la parola chiave (keyword) specificata all'interno del database whatis. Produce una descrizione molto breve dei comandi di sistema, una sorta di guida di riferimento tascabile per i comandi. Equivalente a **man -f**

```
whatis nasm
nasm (1)              - the Netwide Assembler - portable 80x86 assembler

man -f nasm
nasm (1)              - the Netwide Assembler - portable 80x86 assembler
```

apropos

Analogo a whatis, ma cerca stringhe e non parole complete, di conseguenza può dare risultati più articolati.
Equivalente a **man -k**

```
apropos asm
nasm (1)              - the Netwide Assembler - portable 80x86 assembler
ndisasm (1)           - the Netwide Disassembler - 80x86 binary file disassembler

man -k asm
nasm (1)              - the Netwide Assembler - portable 80x86 assembler
ndisasm (1)           - the Netwide Disassembler - 80x86 binary file disassembler
```

help

Brevi informazioni sui comandi interni, in lingua inglese, possono essere ottenute tramite il comando **help**. Digitato senza parametri fornisce una lista dei comandi interni, aggiungendovi il nome del comando desiderato ne illustra brevemente la funzione.

```
help              Lista dei comandi interni
help alias        Informazioni sul comando interno alias
```

Il secondo comando produce in output quanto segue:

```
alias: alias [-p] [name[=value] ... ]
    `alias' with no arguments or with the -p option prints the list
    of aliases in the form alias NAME=VALUE on standard output.
    Otherwise, an alias is defined for each NAME whose VALUE is given.
    A trailing space in VALUE causes the next word to be checked for
    alias substitution when the alias is expanded.  Alias returns
    true unless a NAME is given for which no alias has been defined.
```

Comandi Base (parte 4)

Operazioni con i file

Tra le operazioni che è possibile compiere sui file ci sono quelle relative ai cosiddetti collegamenti o link.
Linux ha due tipi di collegamenti:

- **link fisico** (hard link) che crea un collegamento fisico a un file esistente. Questo non è altro che un nuovo nome (*un alias*) del file che resta unico; ogni modifica fatta al file è quindi valida per tutti i nomi. Se si cancella uno dei nomi il file continua ad esistere fino a quando esiste almeno un nome; per cancellare il file occorre cancellare tutti i suoi nomi. Il limite è che può funzionare solo nell'ambito dello stesso file-system. Inoltre non funziona per le cartelle.
- **link simbolico** (soft link) che è costituito sostanzialmente da un piccolo file puntatore che contiene il nome del file collegato. Questo tipo di link funziona anche tra file system diversi.

Come utenti Windows si ha probabilmente dimestichezza con i collegamenti (link simbolici), che sono quelle icone con la freccina ed il cui nome inizia con 'Collegamento a'. Si tratta di un sistema molto comodo per radunare in un unico posto le applicazioni e i documenti usati più frequentemente, che però materialmente restano dove sono. Il collegamento è un semplice puntatore.

Vediamo, in Linux, con un esempio la differenza tra un file con un hard link e uno con un soft link, eseguiamo allora i seguenti comandi:

```
touch prova              creazione del file prova
ln prova prova-h         creazione link fisico
ln -s prova prova-s      creazione link simbolico
ls -il prova*            listato (con inode) dei vari file prova generati
```

Otterremo come output qualcosa del genere:

```
286483 -rw-r--r--  2 alunno itis 0 2005-11-12 10:47 prova
286483 -rw-r--r--  2 alunno itis 0 2005-11-12 10:47 prova-h
286500 lrwxrwxrwx  1 alunno itis 5 2005-11-12 10:47 prova-s -> prova
```

Come si vede, **prova** e **prova-h** hanno lo stesso inode (286483), mentre **prova-s** ne ha uno diverso 8286500.
I link simbolici si notano perchè hanno la lettera **l** all'inizio del blocco permessi, ed alla fine, inoltre, viene indicato dove punta il collegamento: **-> prova**.

Per quanto riguarda il termine **inode** per il momento è sufficiente fare questa riflessione: le persone richiamano i file e le directory usando il loro nome, i sistemi Linux invece li richiamano per mezzo di un numeretto detto **inode**.
I collegamenti simbolici utilizzano un inode extra, invece i collegamenti fisici utilizzano lo stesso. In altre parole con il collegamento simbolico si crea un altro file (anche se particolare), con il collegamento fisico si punta allo stesso file o se si preferisce SONO lo stesso file chiamato con nomi diversi.

Questa spiegazione non è ovviamente esaustiva, ma è sufficiente per iniziare. Per ulteriori informazioni usate il comando: **man ln**.

Se in **/dev** date un'occhiata ai vari devices, troverete probabilmente anche il device **/dev/cdrom** che non è affatto un device, ma un collegamento simbolico al vero device **/dev/hdb**

```
ls -l /dev/cd*
lrwxrwxrwx  1 root root      8 2005-11-12 06:54 /dev/cdburner -> /dev/hdb
lrwxrwxrwx  1 root root      8 2005-11-12 06:54 /dev/cdrom -> /dev/hdb
lrwxrwxrwx  1 root root      8 2005-11-12 06:54 /dev/cdrom1 -> /dev/hdd
```

I collegamenti simbolici in Konqueror sono mostrati in *corsivo* e mettendoci sopra il mouse appare nella parte inferiore la destinazione cui puntano..

Riassumendo, per la creazione dei collegamenti si utilizza il seguente comando:

```
ln [opz]   TARGET LINK_NAME
```

- Senza nessuna opzione, tenta di creare collegamenti fisici verso target (hard link); possibile solo verso file della stessa partizione.
- con opzione **-s** crea collegamenti simbolici (più usato).
- E' possibile specificare una cartella come ultimo parametro: in questo caso, in essa viene creato un link per ogni file specificato.

Informazioni spazio su disco

Per avere una dettagliata statistica sull'utilizzo dello spazio del disco fisso in tutte le partizioni, directory e sottodirectory di Linux utilizzeremo i comandi **du** e **df**.

Il primo, **du** (disk usage), restituisce informazioni su dimensioni di file e directory. Usato senza argomenti, analizza lo spazio su disco per la directory corrente mostrando ricorsivamente le dimensioni dei file contenuti anche nelle eventuali sottodirectory. Si suggerisce di provarlo anche con le opzioni base in combinazione tra di loro.

```
du [opz] [NOME_FILE|NOME_DIRECTORY)...]
```

- **-h** (human-readable) aggiunge a ciascuna dimensione un suffisso, come M per Megabyte, in modo da rendere più comprensibile l'interpretazione dell'output.
- **-c** aggiunge il totale di tutte le dimensioni dei singoli file, dopo che sono stati processati tutti. Si può usare per scoprire l'ingombro totale sul disco di un insieme dato di file o directory.
- **-s** (summarize) visualizza solo lo spazio occupato dagli argomenti dati, e non dalle loro sottodirectory.
- Consultare il manuale per altre opzioni.

Il secondo comando **df** (disk filling) visualizza una tabella con indicazioni sull'occupazione di spazio su ogni partizione del sistema. Il comando:

```
df -h
```

può visualizzare una tabella riassuntiva come la seguente:

```
Filesystem          Dimens. Usati Disp. Uso% Montato su
/dev/hda7             38G   33G  5,6G  86% /
tmpfs                506M     0  506M   0% /dev/shm
/dev/hda1             12G  2,2G  9,1G  20% /mnt/hda1
/dev/hda2             12G  2,1G  9,2G  19% /mnt/hda2
/dev/hda3             12G  1,4G  9,8G  13% /mnt/hda3
/dev/hda6             38G   18G   21G  47% /mnt/hda6
/dev/hda8             38G  1,8G   36G   5% /mnt/hda8
/dev/hdc1             12G  7,5G  4,2G  65% /mnt/hdc1
/dev/hdc2             12G   33M   12G   1% /mnt/hdc2
/dev/hdc3             62G   23G   36G  40% /mnt/hdc3
/dev/sdc             490M   56M  434M  12% /mnt/sdc
/dev/sdd1            124M   14M  110M  12% /mnt/sdd1
```

Montaggio/Smontaggio dei dischi

La gestione del montaggio e dello smontaggio dei filesytem, tipicamente a cura dell'amministratore del sistema, si effettua con i comandi **mount** e **umount** e per il tramite del file di configurazione **/etc/fstab**.

Il comando **mount**, usato da solo senza alcuna opzione, fornisce una tabella come la seguente:

```
/dev/hda7           on /              type reiserfs  (rw)
proc                on /proc          type proc      (rw,nodiratime)
sysfs               on /sys           type sysfs     (rw)
tmpfs               on /dev/shm       type tmpfs     (rw)
/dev/hda1           on /mnt/hda1      type reiserfs  (rw,nosuid,nodev)
/dev/hda2           on /mnt/hda2      type reiserfs  (rw,nosuid,nodev)
/dev/hda3           on /mnt/hda3      type reiserfs  (rw,nosuid,nodev)
/dev/hda6           on /mnt/hda6      type reiserfs  (rw,nosuid,nodev)
/dev/hda8           on /mnt/hda8      type reiserfs  (rw,nosuid,nodev)
/dev/hdc1           on /mnt/hdc1      type reiserfs  (rw,nosuid,nodev)
/dev/hdc2           on /mnt/hdc2      type reiserfs  (rw,nosuid,nodev)
/dev/hdc3           on /mnt/hdc3      type ext3      (rw,nosuid,nodev)
devpts              on /dev/pts       type devpts    (rw,gid=5,mode=620)
usbfs               on /proc/bus/usb  type usbfs     (rw,devmode=0666)
/dev/sdc            on /mnt/sdc       type vfat      (rw,noexec,nosuid,nodev,sync,
                                                      uid=1000,gid=1000,shortname=mixed,
                                                      quiet,umask=0002,showexec)
automount(pid3527)  on /mnt/auto      type autofs    (rw,fd=4,pgrp=3527,minproto=2,maxproto=4)
/dev/sdd1           on /mnt/sdd1      type vfat      (rw,noexec,nosuid,nodev,sync,
                                                      uid=1000,gid=1000,shortname=mixed,
                                                      quiet,umask=0002,showexec)
```

con tutte le informazioni relative ai filesystem montati in un determinato momento (la tabella è stata qui formattata per esigenze di chiarezza, ma i vari campi spesso sono separati soltanto da spazi).

Ma la forma più frequente del comando **mount** è la seguente:

```
mount [-o opzioni] [-t tipo] device puntodimontaggio

mount /dev/hdc1 /mnt/hdc1
esempio generico
mount -t vfat /dev/fd0 /mnt/floppy
monta il primo floppy disk, formattato vfat, nel punto di montaggio /mnt/flopyy
mount [-t auto] /dev/hda10 /mnt/extra
monta la partizione /dev/hda10 in /mnt/extra, effettuando l'autoriconoscimento del filesystem
mount -o ro, loop -t iso9660 nomeiso.iso /mnt/iso
monta un'immagine iso (nomeiso.iso) nel punto di montaggio /mnt/iso
```

- Se *device* o *puntodimontaggio* sono specificati in **fstab**, uno dei due può essere omesso
- È possibile specificare un'immagine di filesystem (se il kernel supporta il loopback), tramite l'opzione loop
- *tipo* indica il tipo del filesystem. Spesso identificato automaticamente (tranne, ad es, vfat vs fat16)
- **Attenzione**: un utente ordinario non può specificare parametri personalizzati per mount. Può però montare i filesystem specificati in fstab contrassegnati con **user**.

Il comando **umount**, invece, provvede a smontare un dispositivo montato. La sua forma generale è:

```
umount puntodimontaggio

umount /mnt/hdc1
esempio generico
```