

✓ LINUX in Compresse v1.01

by Alfio "Il Profeta" Pafumi

(volevo chiamarlo LINUX in Pillole, ma esiste già qualcosa con questo nome e quindi...)

codename: L4F

(email: musashi.m@gmail.com)

Note dell'autore: questa dispensa è stata scritta utilizzando il libro "UNIX: manuale per l'amministratore di sistema", gli appunti presi a lezione di Laboratorio di Sistemi Operativi del professor G. Fiumara e quanto appreso smanettando. La distribuzione utilizzata è Mandrake Linux 9.2

Comunque, se qualcosa è scritto male, se ho frainteso o omesso qualcosa... IO NON VOGLIO SAPERE NULLA! (chi vi ha detto di usare questa dispensa? Studiate dai libri e dai *vostri* appunti!)

Seramente, spero possa essere utile e che non ci siano errori o inesattezze (o almeno non troppi)! Buono studio! ;o)))

P.S. Avrei potuto condire il tutto con foto dell'autore e dei colleghi dello stesso, ma avendo un po' di buon gusto ho preferito evitare.....

:: INDICE ::

- ✓ COMANDI DI BASE
- ✓ LILO E AVVIO DEL SISTEMA
- ✓ RUNLEVEL (LIVELLI DI ESECUZIONE) DEL SISTEMA
- ✓ SCRIPT DI AVVIO
- ✓ SPEGNIMENTO E RIAVVIO DEL SISTEMA
- ✓ I POTERI DI ROOT
- ✓ SUDO E /ETC/SUDOERS
- ✓ IL FILESYSTEM
- ✓ TIPI DI FILE
- ✓ MONTARE DISCHI
- ✓ DEVICE E TERMINALI
- ✓ AGGIUNGERE E RIMUOVERE UN UTENTE
- ✓ I SEGNALI
- ✓ MONITORAGGIO DEI PROCESSI
- ✓ IL BACKUP
- ✓ LE OPERAZIONI PERIODICHE: CRON
- ✓ SYSLOG
- ✓ OPERATORI PER REINDIRIZZARE I FLUSSI
- ✓ APPENDICE A: l'editor VI
- ✓ APPENDICE B: la shell BASH
- ✓ APPENDICE C: I CARATTERI JOLLY: cercare files, ma non solo
- ✓ APPENDICE D: l'utility TAR

Convenzioni:

comandi
files
directory

:: COMANDI DI BASE ::

<code>man comando/file</code>	Questo comando visualizza l'aiuto di Linux relativamente al comando o file indicato. Man rappresenta la documentazione di UNIX/Linux. E' forse la più importante risorsa di Linux, per qualsiasi dubbio, usate il man (se non basta c'è Internet).
<code>cd nome_dir</code>	Cambia directory
<code>rm nome_file</code>	Cancella i file indicati
<code>mkdir nome_dir</code>	Crea una directory
<code>rmdir nome_dir</code>	Cancella la directory indicata, se vuota
<code>rm -r nome_dir</code>	Cancella la directory indicata, anche se non vuota.
<code>touch nome_file</code>	Crea un file vuoto
<code>pwd</code>	Mostra il path corrente
<code>ls</code>	Mostra i nomi dei files contenuti nella directory
<code>ls -l</code>	Mostra i files contenuti nella directory visualizzando più informazioni
<code>cp nome_file destinazione</code>	Copia il file o i files indicati nella destinazione specificata
<code>mv nome_file destinazione</code>	Sposta il file o i files indicati nella destinazione specificata
<code>whereis nome_file</code>	Cerca il file indicato
<code>find nome_dir -name nome_file</code>	Cerca il file indicato, a partire dalla directory specificata. Ha molte più opzioni di whereis. Consultate il manuale, usando man find
<code>uname -a</code>	Mostra informazioni sul sistema, fra cui la versione del kernel.
<code>hostname</code>	Mostra il nome del computer
<code>df</code>	Mostra lo spazio libero
<code>cat nome_file</code>	Visualizza il contenuto di un file di testo specificato, ma può essere usato per concatenare più files in uno solo (vedi man cat per ulteriori informazioni). Non usare con file binari!
<code>more nome_file</code>	Visualizza il contenuto di un file di testo specificato, permettendo di scorrere in avanti il testo. Non usare con file binari!
<code>less nome_file</code>	Visualizza il contenuto di un file di testo specificato, permettendo di scorrere il testo in avanti e indietro. Non usare con file binari!
<code>tail nome_file</code>	Di default visualizza le ultime 10 righe di <i>nome_file</i> (utile per consultare i files di log, visualizzando così solo gli ultimi messaggi di log)
<code>ifconfig</code>	Visualizza informazioni sulla rete
<code>clear</code>	Pulisce lo schermo

:: LILO E AVVIO DEL SISTEMA ::

LILO (LInux LOader) serve a gestire più sistemi operativi installati sulla stessa macchina.

Si configura editando il file [/etc/lilo.conf](#)

Le principali impostazioni sono:

default	Indica qual è la scelta da avviare se l'utente non interviene
timeout	Indica il tempo di attesa prima di scegliere l'opzione di default. E' espresso in decimi di secondi, quindi time = 100 indica un attesa di 10 secondi

❖ **IMPORTANTE:** per effettuare le modifiche al sistema, **NON** basta editare il file [/etc/lilo.conf](#) e salvarlo. Dopo E' **NECESSARIO** lanciare il comando:

lilo -t

per testare le modifiche effettuate (-t sta per test e finora non avete in alcun modo modificato il boot del sistema!), se l'output del precedente comando non segnala errori, allora lanciate il comando:

lilo

per effettuare realmente le modifiche. Al prossimo riavvio vedrete il risultato delle vostre modifiche.

Se non si avvia, allora... avete fatto danno.....

:: RUNLEVEL (LIVELLI DI ESECUZIONE) DEL SISTEMA ::

Non sono altro che differenti modalità in cui può essere eseguito Linux. Non pensate a nulla di trascendentale, vuol semplicemente dire che a runlevel differenti, di default vengono avviati, servizi differenti (non tutti differenti, solo che ad alcuni runlevel alcuni servizi non vengono avviati, in altri runlevel vengono avviati servizi in più o servizi diversi).

In poche parole, i programmi caricati in memoria, e quindi le cose che potrete fare, saranno differenti (ma nulla vieta di avviare il sistema in un runlevel, che non avvia uno specifico servizio e poi avviare noi, manualmente, quel servizio)

Esempio:

Se voglio usare l'interfaccia grafica X11 (è il server grafico di Linux, che poi permette di usare i vari KDE, Gnome, ecc.) avvio il sistema a runlevel 5 (quello a cui corrisponde, appunto, l'interfaccia grafica sulla maggior parte dei sistemi Linux), ma nulla mi vieta di avviare il sistema a runlevel 3 (che praticamente è uguale al 5 ma senza l'interfaccia grafica) e poi avviare l'interfaccia grafica quando lo ritengo più opportuno, con il comando **startx**.

● **NOTA:** Poco sopra è stato usato il termine server, in gergo informatico non indica solo un computer connesso in rete, al quale altri computer (client) possono collegarsi. Server è qualsiasi "cosa" (computer, processo) sta in attesa di richieste e, quando arriva una di tali richiesta, si comporta di conseguenza.

Il server grafico serve al sistema per utilizzare l'interfaccia grafica, così come altri servono ad utilizzare altre periferiche. Esempi: il server sonoro (ALSA) serve per avere il sonoro, il server di stampa (CUPS) serve a stampare, ecc.

Runlevel:

(sono 7, vanno da 0 a 6, possono variare da distribuzione a distribuzione e possono essere

personalizzati dall'utente; qui elencheremo il caso più comune)

0	spegnimento *
1	modalità monoutente
2	modalità multiutente SENZA rete
3	modalità multiutente CON rete
4	solitamente inutilizzato (e quindi utilizzabile in futuro o personalizzabile dall'utente)
5	modalità multiutente CON rete e INTERFACCIA GRAFICA
6	riavvio *

* (non impostare mai come runlevel predefinito all'avvio!!!)

→ Per cambiare il runlevel corrente, il comando da usare è **telinit X**, dove invece di **X** dovreste scrivere il numero che indica il runlevel a cui volete passare.

Il runlevel predefinito si imposta editando il file `/etc/inittab`. La riga da modificare è:

id:X:initdefault:

invece della **X** troverete un numero da 1 a 5, che indica il runlevel che il sistema utilizza di default.

Ad ogni runlevel corrisponde una directory:

`/etc/rc0.d/` runlevel 0
`/etc/rc1.d/` runlevel 1
...e così via fino a...
`/etc/rc6.d/` runlevel 6

all'interno di ogni directory si trovano dei files, o esattamente dei link ai files relativi agli script da eseguire, quando si passa a quel runlevel.

Come detto, le directory `rc(numero).d` contengono in realtà solo dei link ai files originali, che sono invece contenuti in `/etc/rc.d`

(questo perché se lo stesso servizio viene avviato da più runlevel non ha senso avere più copie dello stesso file, anzi è un inutile spreco di spazio. Il sistema invece ha un solo file vero e proprio e poi tutti i link necessari ai vari runlevel)

:: SCRIPT DI AVVIO ::

La maggior parte dei sistemi Linux usa degli script d'avvio in stile SYSTEM V.

Il sistema quando si avvia, procede eseguendo gli script a partire dal livello 0 fino a quello di esecuzione predefinito.

Gli script sono caratterizzati da un nome che inizia sempre per S oppure K poi un numero e un nome che indica il servizio. Ad esempio `S22nome`.

- quando si passa da un livello inferiore ad uno superiore, il sistema tramite il processo `init`, lancia con l'argomento **start** tutti gli script che iniziano per S, in ordine ascendente in base al numero.
- quando si passa da un livello superiore ad uno inferiore, invece `init` esegue con l'argomento **stop** tutti gli script che iniziano per K, in ordine discendente.

Quindi se vogliamo eseguire un servizio all'avvio di un determinato runlevel, dobbiamo creare all'interno della directory `rcnumero.d` corrispondente un link simbolico al servizio stesso.

Esempi:

```
ln -s /etc/init.d/sshd /etc/rc2.d/S99sshd
```

il comando precedente crea un link simbolico nella directory `/etc/rc2.d/` allo script di avvio di `sshd` contenuto nella directory `/etc/init.d` con il numero 99. Questo indica che quando il sistema passerà a runlevel 2 eseguirà l'avvio (notare l'iniziale S) di questo servizio fra gli ultimi (questo è indicato dal numero 99, abbastanza alto)

```
ln -s /etc/init.d/sshd /etc/rc0.d/S25sshd
```

il comando precedente crea un link simbolico nella directory `/etc/rc0.d/` allo script di avvio di `sshd` contenuto nella directory `/etc/init.d` con il numero 25. Questo indica che quando il sistema passerà a runlevel 0 eseguirà lo stop (notare l'iniziale K) di questo servizio.

:: SPEGNIMENTO E RIAVVIO DEL SISTEMA ::

Come detto prima, per spegnere il sistema si deve passare al runlevel 0 e per riavviare si deve passare al runlevel 6.

E' possibile compiere queste due operazioni in molti modi, ma alla fine ciò che fa il sistema ogni volta che lo spegnete, non è altro che passare a runlevel 0, così come passa a runlevel 6 ogni volta riavviate (esclusi ovviamente lo spegnimento e il riavvio fatti brutalmente premendo i tasti sul case, o in caso di arresto di corrente)

Per spegnere il sistema si può usare il comando **shutdown** a cui dobbiamo però passare i parametri adeguati, cioè cosa fare e quando farlo.

Esempio 1:

```
shutdown -h now
```

Spegne il sistema (notare il **-h** che sta per halt)

Esempio 2:

```
shutdown -r now
```

Riavvia il sistema (notare il **-r** che sta per reboot)

In entrambi i casi abbiamo usato **now** che indica al sistema di effettuare lo shutdown adesso (ovviamente spegnimento e riavvio non avvengono all'istante, il sistema inizia da subito a terminare i vari processi, ma ci vuole un po' perché questo avvenga). Invece di usare **now** potremmo indicare un tempo espresso in secondi.

Se vogliamo semplicemente spegnere o riavviare il sistema, senza particolari esigenze o parametri da passare, esistono due comandi più brevi:

reboot	riavvia il sistema
halt	arresta il sistema

♦ **IMPORTANTE:** solitamente, a meno che il sistema non sia configurato diversamente, solo l'utente root (cioè l'amministratore di sistema) può spegnere o riavviare il pc.

Se pensate al vostro pc, può sembrare una limitazione doversi *loggar*e come root, solo ad esempio, per spegnerlo (ma basta dare all'utente con cui usate il pc il "potere" di usare **halt** o **reboot**, usando l'utility **sudo**) ma se pensate ad un sistema dove vari utenti possono collegarsi e lavorare contemporaneamente, appare evidente che un normale utente NON può e NON deve avere la possibilità di spegnere o riavviare il pc quando gli pare e piace.

:: I POTERI DI ROOT ::

L'utente root, nei sistemi Unix/Linux è l'amministratore di sistema. Se si è effettuato il login come un altro utente si può diventare root con il comando **su**

● **NOTA:** Root ha potere virtualmente illimitato. E' altamente sconsigliato usare l'interfaccia grafica come root.

:: SUDO E /ETC/SUDOERS ::

Sudo è un'utility che permette di gestire meglio i poteri di root. In pratica, se vogliamo dare ad un utente la possibilità di eseguire alcuni compiti da amministratore (concretamente, lanciare comandi che solo root può usare) invece di dargli la password di root (scelta decisamente troppo pericolosa e fiduciosa), è meglio usare sudo, che sta Super User DO.

Le impostazioni di sudo si creano e modificano editando il file [/etc/sudoers](#).

Per editare tale file sé buona norma usare l'apposito comando **visudo**.

Una volta editato correttamente il file [/etc/sudoers](#) l'utente o gli utenti indicati nel file [/etc/sudoers](#) potranno usare il comando o i comandi assegnati loro.

Sintassi:

nomeutente nomehost=[opzione password]:comando1[,comando2,comando3]

nomeutente	l'utente che potrà usare sudo con il comando specificato
nomehost	il nome di host su cui è valida questa regola di sudo. Questo parametro serve perché sudo è pensato per essere multi host. Cioè un solo file di sudo può essere condiviso fra più computer collegati fra loro.
opzione password	se non scriviamo nulla il sistema chiederà la password all'utente quando farà uso del comando sudo associato ai comandi <i>comando1</i> , <i>comando2</i> , ecc; se vogliamo che l'utente non debba inserire la password dobbiamo scrivere <i>NOPASSWD</i>
comando1,comando2	comando1,comando2 = il comando o i comandi con cui l'utente può usare sudo

Esempio:

pippo localhost=NOPASSWD:/sbin/halt,/sbin/reboot

L'utente pippo potrà usare i comandi /sbin/halt ed /sbin/reboot sul computer localhost senza inserire password.

pippo potrà usare il comando **halt** digitando:

sudo /sbin/halt

● **NOTA:** Per ogni parametro, cioè al posto di nomeutente, host o dei comandi, si potrebbero usare degli alias, che vanno definiti prima di definire la regola di sudo. Cioè gli alias vanno scritti prima di quando scriviamo i comandi per cui vogliamo usare sudo.

◆ **IMPORTANTE:** finora, quando si diceva di usare il comando **halt**, è stato scritto soltanto il comando senza path. In realtà halt si trova nella directory **/sbin/** semplicemente tale directory fa parte dei PATH predefiniti in cui il sistema cerca i comandi dell'utente root e quindi non è necessario specificarla (se è root ad eseguire il comando). Se invece come nell'ultimo esempio, è un utente comune a lanciarli è necessario specificare il percorso completo, affinché il sistema li trovi (in teoria si potrebbe aggiungere la directory in questione nel PATH dell'utente, ma in questo caso è insolito che un utente comune abbia nel proprio path **/sbin/** perché tale directory, come il nome suggerisce*, contiene gli eseguibili di sistema, ed un utente non dovrebbe avere la necessità di usare molti comandi di sistema)

* **sbin** = **s**-system **bin**-ary

Perché ho scritto “importante” prima del paragrafetto precedente? Perché a volte un comando che avete usato in precedenza senza problemi, potrebbe sembrare non funzionare perché magari il sistema (o meglio voi) non ha aggiornato il path i cui cerca i comandi e quindi per eseguire il comando “incriminato” è necessario usare il percorso completo!

:: IL FILESYSTEM ::

directory importanti:

/	è la directory root, in Linux tutto parte da qui
/bin/	contiene gli eseguibili
/boot/	contiene i files necessari per il boot del sistema (l'immagine del kernel, ecc.)
/etc/	contiene gran parte dei files di configurazione
/home/	contiene le homedirectory degli utente
/proc/	contiene le immagini dei processi in esecuzione
/root/	è la homedirectory dell'utente root
/sbin/	contiene gli eseguibili di sistema
/usr/	contiene una miniatura del sistema (al suo interno ci sono fra altre, le directory /bin/, /sbin/)

:: TIPI DI FILES ::

quando lanciate il comando `ls -l` ogni rigo indica un file. Il primo carattere di ogni rigo indica il tipo di file.

Ci sono 7 tipi di file.

-	file normale
d	directory
b	file di dispositivo a blocchi
c	file di dispositivo a caratteri
s	socket di dominio UNIX, sono file socket locali (da distinguere dai socket usati per comunicare in rete)
p	PIPE denominato
l	link simbolico

:: MONTARE DISCHI ::

Windows gestisce il sistema effettuando una suddivisione delle risorse in base alla loro “natura”. Insomma distingue hard disk, floppy, lettori ottici, come risorse differenti a cui permette di accedere attraverso l'esplorazione delle cosiddette “Risorse del computer”.

Linux, invece, vede qualsiasi componente (floppy, unità cd/dvd, ma anche stampanti, scanner) come parte del filesystem. La radice del filesystem è la directory / e qualsiasi dispositivo di memorizzazione (hard disk, partizioni*, unità cd/dvd, floppy)

● * NOTA: le partizioni sono hard disk logici, cioè vengono viste da Linux come hard disk. Non c'è un comando per accertarsi con assoluta certezza di quanti siano gli hard disk fisici installati nel computer, senza aprirlo (anche se secondo me ciò non è del tutto esatto, almeno in alcune distribuzioni di Linux.....).

Con Linux, prima di usare un disco bisogna montarlo (questo vale per i floppy, i cd, gli hard disk, pendrive... in pratica per qualsiasi tipo di supporto di memorizzazione)

Montare un disco vuol dire semplicemente associare un dispositivo ad una directory.

Inseguito, si potrà accedere a quel disco e quindi al suo contenuto, semplicemente andando nella directory in cui l'abbiamo montato.

Il comando da usare è **mount**.

Sintassi:

`mount file_di_dispositivo punto_di_mount`

● NOTA: il punto di mount è una directory

Esempio:

`mount /dev/hdb1 /mnt/win_c`

Se vogliamo rimuovere un dispositivo montato sul pc, ad esempio togliere il cd, è buona norma prima smontarlo dal sistema.

Una volta montato un dispositivo, esso può essere smontato con il comando **umount** seguito dal nome del dispositivo o del punto di mount. (NOTA: non c'entra nulla con il “montare” o “smontare” fisicamente l'hardware).

Sintassi:

`umount file_di_dispositivo`

oppure

`umount punto_di_mount`

Esempi:

`umount /dev/hdb1`

`umount /mnt/win_c`

❖ **ATTENZIONE:** un dispositivo non può essere smontato se è occupato, cioè se ci sono file aperti o programmi che stanno usando file su quel dispositivo, ecc.

Se provando a smontare un dispositivo il sistema dà errore dicendo che non può compiere l'operazione, perché il dispositivo è occupato, dobbiamo chiudere tutti i programmi che usano file di quel dispositivo. Se anche così risulta occupato, bisogna usare il comando:

`fuser file_di_dispositivo`

esso darà come output, i processi che usano il dispositivo. Potremo quindi *killare* i processi e poi smontare il dispositivo usando **umount** come spiegato sopra.

Linux, all'avvio, deve necessariamente montare alcune partizioni, ad esempio la partizione di root (quella che contiene il sistema operativo stesso) e può anche montare le partizioni che l'amministratore desidera. Per fare in modo che Linux faccia tutto ciò bisogna editare il file [/etc/fstab](#)

Struttura del file [/etc/fstab](#)

ogni rigo si riferisce ad un dispositivo ed è formato da 6 campi, separati da spazi

Esempio:

dev/hda1 / ext2 defaults 1 1

(primo) (secondo) (terzo) (quarto) (quinto) (sesto)

(primo)	file di dispositivo
(secondo)	punto di mount
(terzo)	tipo di filesystem
(quarto)	opzioni di mount (esempio lettura/scrittura, cioè rw oppure solo lettura, cioè ro)
(quinto)	usato per dire al programma dump se deve effettuare il backup

(primo)	file di dispositivo
(sesto)	serve per il programma fsck per sapere in che ordine deve controllare il filesystem (il valore 0 indica di non effettuare il controllo, il filesystem di root / dovrebbe avere valore 1)

Il file [/etc/mstab](#) presenta invece tutti i filesystem correntemente montati (quindi se volete sapere se un filesystem è montato o no, potete visualizzare il contenuto di questo file). La sua struttura è uguale a quella di [/etc/fstab](#)

:: DEVICE E TERMINALI ::

- I devices sono i dispositivi, cioè le periferiche. I file di dispositivi (device files) sono files che servono al sistema per comunicare con le periferiche. Esistono file di dispositivo a **blocchi** (utilizzati dal sistema per comunicare con le periferiche a blocchi, ad esempio: **dischi rigidi, lettori cd/dvd**) o a **caratteri** (utilizzati dal sistema per comunicare con le periferiche a caratteri, ad esempio: **il terminale, le porte seriali**).

→ I files di dispositivo sono contenuti nella directory [/dev/](#) e nelle sue sottodirectory.

- Il terminale, molto molto brevemente, è quello che vi permette di interagire con il sistema, digitare comandi, ecc.

Di default su Linux esistono più terminali, indicati da `TTYnumero`, dove al posto di *numero* c'è un numero da 1 a 6 (di solito).

Potete passare da un terminale all'altro usando la combinazione di tasti `<ALT> + numero`

- ♦ **IMPORTANTE:** Se in seguito a qualche problema, programma sballato o operazione poco ortodossa o errata* il terminale si comporta in maniera non appropriata, cioè alla pressione dei tasti vengono visualizzate lettere errate, simboli incomprensibili, ecc. lanciate il comando **reset** e date `<INVIO>` anche alla cieca (cioè anche se non compare nulla a video o compaiono simboli strani).

Questo dovrebbe ripristinare il corretto funzionamento del terminale.

E' consigliabile subito dopo usare il comando **clear** per ripulire lo schermo.

- * per esempio cercare di visualizzare, con il comando **less** o **more**, il contenuto di un file binario, cosa che fa sballare il terminale per la presenza nei file binari di speciali caratteri che modificano il comportamento del terminale stesso.

:: AGGIUNGERE E RIMUOVERE UN UTENTE ::

Per aggiungere un utente si possono usare i comandi **useradd** o **adduser**, ma dovete sapere come farlo editando i files appositi.

La procedura è la seguente:

- editare i files:
 - [/etc/passwd](#) contiene le informazioni sugli account degli utenti
 - [/etc/shadow](#) contiene le informazioni sulle password

[/etc/group](#) contiene le informazioni sui gruppi

- impostare la password dell'utente con il comando

passwd nomeutente

- NOTA: attenzione! La password va digitata per due volte. Se una delle due volte sbagliamo, il sistema ci chiede la password per la TERZA volta, ma siccome poi non sa se confrontare questa terza password con la prima o la seconda, continuerà a darci errore. La cosa da fare è interrompere il comando con <CTRL> + C e ripetere l'operazione di impostazione della password, lanciando nuovamente **passwd nomeutente**

Struttura del file [/etc/passwd](#) (7 campi separati da :)

username:x:UID:GID:campo GECOS:home directory:shell di login

(primo)	nome di login
(secondo)	campo password (si mette una x per non lasciarlo vuoto, ma la password in realtà sarà salvata nel file /etc/shadow)
(terzo)	identificativo dell'utente, il numero con cui il sistema identifica l'utente, di solito va da 501 in poi
(quarto)	identificativo del gruppo
(quinto)	informazioni varie, di solito nome per esteso, numero di telefono, indirizzo, ecc.
(sesto)	directory home
(settimo)	shell predefinita per l'utente di solito /bin/bash

Struttura del file [/etc/shadow](#) (9 campi separati da :)

(primo)	username
(secondo)	password crittografata, si mette un asterisco *, perché la password deve essere crittografata ed ovviamente non può essere scritta a mano
(terzo)	data dell'ultima modifica della password
(quarto)	min numero di giorni tra due modifiche alla password
(quinto)	max numero di giorni tra due modifiche alla password
(sesto)	numero di giorni d'anticipo con cui avvisare che la password sta scadendo
(settimo)	numero di giorni di attesa prima che l'account sia disabilitato
(ottavo)	data di scadenza dell'account
(nono)	al momento inutilizzato in Linux, riservato a scopi futuri

- NOTA 1: la password è codificata con l'algoritmo di crittografia DES, la cui particolarità è che due password in chiaro, uguali, codificate con tale algoritmo in momenti diversi, daranno come risultato due password codificate diverse. Questo è importante perché vuol dire che non posso scoprire se due utenti hanno la stessa password solo guardando le password codificate.
- NOTA 2: la date sono espresse in giorni a partire dall'1 Gennaio 1970

Struttura del file [/etc/group](#) (4 campi separati da :)

nomegruppo*:GID:username1,username2

(primo)	nome del gruppo
(secondo)	password crittografata. Viene usata raramente per i gruppi. Di solito si inserisce un asterisco.
(terzo)	GID, Group IDentifier, identificativo del gruppo. Un numero con cui il sistema identifica il gruppo.
(quarto)	Elenco dei membri del gruppo separati da virgole.

✓ **e siccome siamo figli...**

quando si lavora sul serio in un grosso centro dove più persone potrebbero avere i poteri di root, per modificare i files [/etc/passwd](#) e [/etc/shadow](#) è preferibile usare il comando **vipw** senza parametri (che apre prima il file [/etc/passwd](#) e dopo chiede se aprire o meno il file [/etc/shadow](#)).

Tale comando avvia una sessione particolare di VI, che blocca i file [/etc/passwd](#) e [/etc/shadow](#) in modo che finché non abbiamo finito nessun altro potrà cercare di modificarli, questo per evitare che qualcun altro possa aprirli e/o salvarli mentre ci stiamo lavorando noi (con il pericolo di creare una situazione di inconsistenza nei suddetti files)

Poi creiamo la home directory dell'utente (il comando per creare una directory è **mkdir**)

Possiamo creare la homedirectory dove vogliamo, ma per convenzione si crea una cartella con lo stesso nome scelto come username all'interno di [/home](#)

Poi dobbiamo cambiare proprietario e gruppo della directory appena creata.

● **NOTA:** Linux è un S.O. multiutente, quindi ogni file, directory ha un suo proprietario ed un gruppo di appartenenza.

Quando compiamo l'operazione di aggiunta di un utente e creiamo la sua directory home, siamo *loggati* come root, quindi anche la directory creata apparterrà a root. Ma in realtà deve appartenere all'utente!!!!!! è per lui che stiamo facendo questa fatica, no? (No! È per passare l'esame ;-)
pensiero di F.)

Quindi dopo aver creato la homedirectory ne modifichiamo proprietario e gruppo usando il comando **chown** con la seguente sintassi

chown *nuovoproprietario:nuovogruppo nomedirectory*

→ **Cancellazione di un utente**

Per cancellare un utente potete usare il comando **userdel**, oppure editare i files precedente, rimuovendo le impostazioni dell'utente da cancellare.

NOTA: `userdel`, di default, non cancella la homedirectory ed i files dell'utente eliminato.

:: I SEGNALI ::

I segnali sono delle interruzioni (interrupt) che vengono lanciate ad un processo.

- possono essere inviati da un processo all'altro, per comunicare
- possono essere inviati dal terminale, con una combinazione di tasti, ad un processo per terminarlo, sospenderlo, ecc. (esempio `<CTRL> + C` oppure `<CTRL> + Z`)
- possono essere inviati dall'amministratore ad un processo con l'uso del comando `kill`, con risultati variabili, dipendentemente dalle opzioni passate a `kill`
- possono essere inviati dal kernel ad un processo in caso di errori gravi (esempio: divisione per 0)

Per lanciare segnali ad un processo si usa il comando `kill` seguito dal tipo di *segnale* che vogliamo inviare (che viene espresso con un numero o con il nome, vedi Tabella sotto, preceduto da un trattino -) e dal *pid* (process identifier, identificativo del processo) del processo a cui vogliamo inviarlo.

Sintassi:

`kill -segnale pid`

I segnali UNIX che dovrebbero essere noti a tutti gli amministratori

#	Nome	Descrizione	Azione di default	Può essere catturato?	Può essere bloccato?	Effettua il core dump?
1	HUP	Hangup	Terminazione	SI	SI	No
2	INT	Interrupt	Terminazione	SI	SI	No
3	QUIT	Quit	Terminazione	SI	SI	SI
9	KILL	Kill	Terminazione	No	No	No
*	BUS	Errore di bus	Terminazione	SI	SI	SI
*	SEGV	Segmentation fault	Terminazione	SI	SI	SI
15	TERM	Terminazione software	Terminazione	SI	SI	No
*	STOP	Stop	Stop	No	No	No
*	TSTP	Stop da tastiera	Stop	SI	SI	No
*	CONT	Continua dopo lo stop	Viene ignorato	SI	No	No
*	WINCH	La finestra è stata cambiata	Viene ignorato	SI	SI	No
*	USR1	Definito dall'utente	Terminazione	SI	SI	No
*	USR2	Definito dall'utente	Terminazione	SI	SI	No

Varia da sistema a sistema. Si veda il file [/usr/include/signal.h](#) o [man signal](#) per informazioni più specifiche.

:: MONITORAGGIO DEI PROCESSI ::

ps	istantanea dei processi in esecuzione
pstree	istantanea dei processi in esecuzione rappresentati in forma di albero
top	visualizzazione dei processi che si aggiorna di tanto in tanto (di solito ogni 5 sec o meno nei sistemi moderni)

ps auxw	<p>a visualizza tutti i processi, anche quelli di sistemi, di altri utenti, ecc</p> <p>u visualizza informazioni in formato utente, cioè con più informazioni, con l'intestazione, ecc.</p> <p>x visualizza anche i processi senza terminale. Quindi non soltanto i processi lanciati dal terminale, ma anche gli altri.</p> <p>w mette a disposizione un rigo in più per visualizzare le informazioni. Se le informazioni su di un processo non entrano in una sola riga, invece di essere troncate (come accade di default), vengono visualizzate a capo. Si possono mettere quante w si vuole. Per ogni w sarà disponibile, se necessario un rigo in più. Solitamente una sola w basta.</p>
ps elf	<p>e mostra l'ambiente del processo</p> <p>l mostra informazioni in formato lungo</p> <p>f le righe di comando sono mostrate in un albero</p>

:: IL BACKUP ::

Il backup è gestito usando i comandi **dump** e **restore**.

dump serve per effettuare il backup dei dati.

caratteristiche principali:

- mantiene i permessi dei file ed anche il nome del proprietario e del gruppo
- permette backup incrementali (cioè possiamo definire dei livelli di backup, da **0** a **9**, di solito). Esempio: se effettuiamo un backup di livello 3, il sistema farà la copia solo dei file aggiunti o modificati dall'ultimo backup di livello 2 ad oggi e non di tutti i files, permettendoci così di risparmiare tempo per l'esecuzione del backup stesso e spazio sui supporti usati per salvare i dati.

Opzioni importanti di dump:

u	fa sì che dump aggiorni il file /etc/dumpdates al termine della sua esecuzione. Se non si specifica mai u tutti i dump sono di livello 0 perché non si tiene mai traccia dei dump precedenti
----------	---

restore serve per ripristinare i dati precedentemente memorizzati con dump.

- permette di navigare fra i file salvati usando i comandi ls, cd, pwd

Ci sono altri modi per salvare i dati, vedi APPENDICE D: l'utility TAR

:: LE OPERAZIONI PERIODICHE: CRON ::

cron è il demone delle operazioni periodiche, il nome deriva da quello del dio del tempo, Cronos.

Si gestisce tramite l'editazione delle crontable (o, brevemente, **crontab** che è anche l'omonimo comando da usare per gestire le crontable e quindi cron)

Ogni utente ha una propria crontable.

Uso del comando **crontab**:

crontab -e	edita la crontable dell'utente
crontab -l	visualizza la crontable dell'utente
crontab -r	rimuove la crontable dell'utente

Struttura di una crontable (6 campi separati da spazi, ma nell'ultimo campo gli spazi sono semplici caratteri)

minuti_del_giorno ora_del_giorno giorno_del_mese mese_dell_anno giorno_della_settimana comando

i primi 5 campi indicano il momento o i momenti in cui dovrà essere eseguito il comando, il 6° campo contiene il comando da eseguire.

- NOTA 1: giorno_della_settimana va indicato con un numero da 0 a 6, dove 0 indica domenica
- NOTA 2: se specifichiamo, ad esempio, giorno_del_mese 10 e giorno_della_settimana 1, cioè lunedì, il comando NON verrà eseguito ogni lunedì 10, MA tutti i lunedì della settimana e ogni 10 del mese. Insomma *basta che una sola delle tue condizioni si verifichi*.

Esempio 1:

```
00 9 * * * rm- r /tmp/*.*
```

il comando **rm-r /tmp/*.*** verrà eseguito alle 9.00 tutti i giorni (l'asterisco indica che qualsiasi valore va bene affinché CRON esegua il comando).

Esempio 2:

```
00 9 10 * 1 rm- r /tmp/*.*
```

il comando **rm-r /tmp/*.*** verrà eseguito alle 9.00 tutti i giorni 10 del mese e tutti i lunedì della settimana.

Esempio 3:

```
00 9,12,15,18,21 * * * rm- r /tmp/*.*
```

il comando **rm-r /tmp/*.*** verrà eseguito alle 9.00, alle 12.00, alle 15.00, alle 18.00, alle 21.00 tutti i giorni.

- NOTA: una versione di cron più avanzata, VIXIE CRON, permette di scrivere l'impostazione precedente in maniera molto più compatta ed elegante:

Esempio 4:

- **NOTA 2:** anche i valori usati nel campo livello vanno scelti fra valori predefiniti, che vanno da debug (il meno grave) fino ad arrivare ad emerg (emergency, il più grave) passando per altri quali warn, alert (giusto per citarne alcuni)

Per effettuare la rotazione dei files di log, affinché non occupino troppo spazio su disco e per evitare semplicemente di cancellare periodicamente i files di log (soluzione semplicistica e poco prudente, che ci lascia senza il loro prezioso ausilio, nel caso in cui dovessimo avere dei problemi proprio il giorno dopo averli cancellati del tutto, ad esempio) è bene effettuare una rotazione dei files di log, per fare ciò in passato andavano scritti degli script adatti alle nostre esigenze, oggi esiste il programma **logrotate**, pensato proprio per questo scopo.

IMPORTANTE: ci sono files di log che è bene non toccare (quindi neanche copiare, rinominare, ecc.) nemmeno se siete gli amministratori del sistema. Due buone ragioni:

- hanno una struttura particolare ed alcuni programmi non funzioneranno se questi files vengono inavvertitamente danneggiati
- sono files binari e non devono essere visualizzati usando more, less o simili, altrimenti manderanno in tilt il terminale (vedi DEVICE E TERMINALI)

Essi sono:

/var/log/wtmp	Contiene informazioni sui login e logout degli utenti. Per visualizzarne il contenuto usate il comando last
/var/run/utmp	Contiene informazioni su chi sta usando correntemente il sistema
/var/log/lastlog	Contiene informazioni sugli ultimi login degli utenti. Per visualizzarne il contenuto usate il comando lastlog

:: OPERATORI PER REINDIRIZZARE I FLUSSI ::

(“MAI INCROCIARE I FLUSSI! SAREBBE MALE” cit.: da Ghostbusters, questa frase non ha alcun senso riferita a Linux!)

Ad ogni programma sono solitamente associati 3 flussi (in inglese *stream*):

- standard input, di solito la tastiera
- standard output, di solito il monitor
- standard error, solitamente è lo stesso dello standard output (ma non è obbligatorio che sia così)

Questi flussi possono essere reindirizzati, usando operatori appositi:

> ridirige lo standard output

esempio: **ls > elenco.txt**

salva l'output del comando **ls** in un file chiamato elenco.txt

Se il file non esiste, lo crea; se esiste lo sovrascrive.

< ridirige lo standard input

2> ridirige lo standard error

→ altri operatori:

>> ridirige il flusso in modalità “append”
esempio: **ls >> elenco.txt**
salva l'output del comando **ls** in un file chiamato **elenco.txt**
Se il file non esiste, lo crea; se esiste aggiunge il nuovo testo alla fine del file.

| fa in modo che l'output di un programma diventi l'input del successivo

esempio 1: **ls | more**

l'output di **ls**, cioè la lista di file e directory, diventa l'input di **more** e quindi potremmo scorrere l'elenco in avanti

esempio 2: **ls | grep prova**

l'output di **ls**, cioè la lista di file e directory, diventa l'input di **grep**, a cui passiamo anche l'argomento “prova”. (Per capire meglio l'esempio leggete la documentazione su **grep** con **man grep**)

L'utente vedrà quindi come risultato solo i files nel cui nome compare la stringa “prova”.

Quindi, se nella directory corrente ci sono 4 files: *file1 file2 fileprova1 fileprova2*, lanciando il comando **ls** visualizzeremo i 4 files. Se invece lanciamo **ls | grep prova** il risultato sarà solo *fileprova1 fileprova2*

; permette di concatenare più comandi, eseguendoli in sequenza

esempio:

comando1 ; comando2

&& permette di concatenare più comandi, MA il secondo viene eseguito solo se il primo non ha dato errore.

esempio 1:

comando1 && comando2

Di solito si usa per eseguire i passi necessari ad installare un programma di cui si hanno i sorgenti:

esempio 2:

./configure && make && makeinstall

● NOTA: per eseguire **makeinstall** dovete essere root.

:: APPENDICE A: l'editor VI ::

Storico editor testuale dei sistemi UNIX/Linux. Si avvia con:

vi nomefile

L'uso di VI è caratterizzato da due modalità: modalità comandi e modalità editazione.

Quando si avvia VI ci si trova in modalità comandi.

I comandi sono molti, fra i più importanti ci sono:

:q	esci
:q!	forza l'uscita
:e	ripristina l'ultima versione del documento salvata (insomma, elimina tutte le modifiche fatte dall'ultimo salvataggio)
:e!	
:w	salva
:w!	forza salvataggio (ad esempio se è un file a sola lettura, ma comunque chi sta editando il file deve avere il permesso di scrittura su quel file)
:wq	salva ed esce

- NOTA: se proviamo ad usare **:qw** otterremo un errore perché, dato l'ordine degli argomenti, è come se cercassimo di uscire dal programma e poi salvare..... ovviamente non va bene.

In modalità comandi è possibile compiere quindi operazioni sul file (anche cancellare caratteri, in alcune versioni di VI) MA NON scrivere.

Per fare questo dobbiamo passare alla modalità editazione, premendo il tasto <a> oppure <i>.

Per passare nuovamente alla modalità comandi (per salvare il file o per uscire, ecc.) basta premere <ESC>

:: APPENDICE B: la shell BASH ::

Le shell correntemente installate su una distribuzione Linux possono essere viste visualizzando il contenuto del file [/etc/shells](#). E' possibile cambiare shell usando il comando **chsh** seguito dal nome della shell che volete usare.

La bash (Bourne Again Shell) è la shell più utilizzata dalla maggior parte di utenti Linux/UNIX, tanto che molte distribuzioni la usano come shell di default.

Combinazioni di tasti:

frecce <SU> e <GIU'>	permettono di scorrere i comandi precedentemente utilizzati (quindi non dobbiamo riscriverli ogni volta)
<CTRL> + <C>	interrompe il comando in esecuzione
<CTRL> + <D>	
<CTRL> + <Z>	mette in pausa il comando in esecuzione
<TAB>	completamento automatico

files di configurazione della BASH

- file generale:

[/etc/bashrc](#)

le sue impostazioni sono valide per tutti gli utenti (a meno che non vengano *bypassate* dalle impostazioni specifiche dell'utente)

- file specifici dell'utente:

.bashrc e **.bash_profile** posizionati nella homedirectory dell'utente. Notare il punto come lettera iniziale del nome. Indica che è un file nascosto, per visualizzarlo occorre usare **ls -a**

:: APPENDICE C: I CARATTERI JOLLY: cercare files, ma non solo ::

Linux possiede vari strumenti per cercare files (vedi i comandi **whereis**, **find**, **locate**) ma una cosa molto importante è saper usare i caratteri jolly (wildchars). Sotto Linux i più importanti sono:

?	Punto interrogativo, al suo posto può esserci un qualsiasi carattere
*	Asterisco, al suo posto possono esserci un qualsiasi numero di caratteri o anche non esserci caratteri
[0-9]	Questa sintassi indica che al suo posto può esserci qualsiasi numero da 0 a 9
[a-z]	Questa sintassi indica che al suo posto può esserci qualsiasi lettera da a a z minuscole
[a-zA-Z]	Questa sintassi indica che al suo posto può esserci qualsiasi lettera da a a z minuscole e da A a Z maiuscole

Esempio:

Se vogliamo cercare tutti i file all'interno della directory corrente che contengono la stringa **"uno"** in qualche parte del loro nome, possiamo usare il comando

whereis *uno*

Potete combinare più caratteri jolly, anche differenti, ad esempio:

whereis ?uno*

oppure

whereis [a-z]uno*

● NOTA: ovviamente i caratteri jolly possono essere usati non solo nella ricerca, ma anche quando vogliamo cercare/rinominare/copiare files/directory.

● NOTA 2: Se usati bene i caratteri jolly possono aiutarvi a compiere velocemente operazioni altrimenti lunghe e ripetitive. Usati male potete però far danno (immaginate di cancellare files usando i caratteri jolly, se sbagliate potreste cancellare anche files che non volevate eliminare).

:: APPENDICE D: l'utility TAR ::

E' un'utility di archiviazione che serve ad immagazzinare directory e files in un unico file archivio. La sua particolarità è quella di preservare i permessi dei file, il proprietario ed il gruppo di appartenenza.

Opzionalmente può anche comprimere i files per ottenere un file archivio di dimensioni minori.

Opzioni:

c	c = create, serve per creare un archivio
x	x = extract, serve per estrarre il contenuto di un archivio
t	t = test, serve per testare un archivio

v	v = verbose, visualizza i messaggi sulle operazioni che sta compiendo
f	è seguita dal nome del file archivio
z	z = serve per dire che il file archivio è compresso, se usato con c crea un file compresso; se usato con t o x testa o estrae un file compresso

Sintassi:

- Per la creazione di un archivio:
tar cvf nomefiletar directory_o_file_da_archiviare
- Per l'estrazione di un archivio:
tar xvf nomefiletar directory_di_destinazione
- Per il test di un archivio:
tar cvf nomefiletar

Per gli archivi compressi basta aggiungere l'opzione z quindi la sintassi sarà:

- Per la creazione di un archivio compresso:
tar czvf nomefiletar directory_o_file_da_archiviare
- Per l'estrazione di un archivio compresso:
tar xzvf nomefiletar directory_di_destinazione
- Per il test di un archivio compresso:
tar czvf nomefiletar