Comandi per gli Utenti Linux

In questo capitolo imparerete i comandi di Linux e come utilizzarli.

Obiettivi : In questo capitolo, i futuri amministratori Linux impareranno come:

✔ **Spostarsi** nell'albero di sistema.
✔ **Creare** un file di testo, **visualizzare** il suo contenuto e **modificarlo**.
✔ **Utilizzare** i comandi Linux più utili.

🏁 **comandi utente**, **linux**

Conoscenza: ⭐
Complessità: ⭐

Tempo di lettura: 40 minuti

Generalità

I sistemi Linux attuali hanno utilità grafiche dedicate al lavoro di un amministratore. Tuttavia, è importante essere in grado di utilizzare l'interfaccia in modalità riga di comando per diversi motivi:

- La maggior parte dei comandi di sistema sono comuni a tutte le distribuzioni Linux, questo non è il caso degli strumenti grafici.
- Può accadere che il sistema non si avvii correttamente ma che un interprete di comando di backup rimanga accessibile.
- L'amministrazione remota viene eseguita dalla riga di comando con un terminale SSH.
- Per preservare le risorse del server, l'interfaccia grafica è installata o lanciata su richiesta.
- L'amministrazione è eseguita da scripts.

L'apprendimento di questi comandi consente all'amministratore di connettersi a un terminale Linux, di gestirne le risorse e i file, di identificare la stazione, il terminale e gli utenti collegati, ecc.

Gli utenti

L'utente di un sistema Linux è definito nel file `/etc/passwd`, da:

- Un **nome di login**, o più comunemente chiamato "login", che non può contenere spazi.
- Un identificatore numerico: **UID** (User Identifier).
- Un identificatore di gruppo: **GID** (Group Identifier).
- Un **interprete di comandi**, ad esempio una shell, che può essere diversa da un utente all'altro.
- Una **directory di connessione**, ad esempio la **directory home**.

In altri file l'utente sarà definito da:

- Una **password**, che verrà crittografata prima di essere memorizzata (`/etc/shadow`).
- Un **prompt dei comandi**, o **prompt** login, che sarà simboleggiato da un `#` per gli amministratori e da un `$` per gli altri utenti (`/etc/profile`).

A seconda della politica di sicurezza implementata sul sistema, la password dovrà contenere un certo numero di caratteri e soddisfare determinati requisiti di complessità.

Tra gli interpreti di comando esistenti, la **Bourne-Again Shell** (`/bin/bash`) è quella utilizzata più frequentemente. È assegnata per impostazione predefinita ai nuovi utenti. Per vari motivi, gli utenti avanzati di Linux possono scegliere shell alternative tra la Korn Shell (`ksh`), la C Shell (`csh`), etc.

La directory di accesso dell'utente è per convenzione memorizzata nella directory `/home` della workstation. Conterrà i dati personali dell'utente e i file di configurazione delle sue applicazioni. Per impostazione predefinita, al login, la directory di accesso è selezionata come directory corrente.

Un'installazione di tipo workstation (con interfaccia grafica) avvia questa interfaccia sul terminale 1. Essendo Linux multiutente, è possibile connettere più utenti più volte, su diversi **terminali fisici** (TTY) o **terminali virtuali** (PTS). I terminali virtuali sono disponibili all'interno di un ambiente grafico. Un utente passa da un terminale fisico ad un altro usando `Alt` + `Fx` dalla riga di comando o utilizzando `CTRL` + `Alt` + `Fx` in modalità grafica.

La shell

Una volta che l'utente è collegato a una console, la shell visualizza il **prompt dei comandi**. Si comporta quindi come un ciclo infinito, ripetendo lo stesso schema a ogni istruzione inserita:

- Visualizza il prompt dei comandi.
- Lettura del comando.
- Analisi della sintassi.
- Sostituzione di caratteri speciali.

- Esecuzione del comando.
- Visualizza il prompt dei comandi.
- etc.

La sequenza chiave `CTRL` + `C` è usata per interrompere un comando in esecuzione.

L'uso di un comando segue generalmente questa sequenza:

```
comando [opzione(i)] [argomento(i)]
```

Il nome del comando è spesso in **minuscolo**.

Uno spazio separa ogni oggetto.

Le **opzioni abbreviate** iniziano con un trattino (`-l`), mentre le **opzioni lunghe** iniziano con due trattini (`--list`). Un doppio trattino (`--`) indica la fine dell'elenco delle opzioni.

È possibile raggruppare alcune opzioni brevi insieme:

```
$ ls -l -i -a
```

è equivalente a:

```
$ ls -lia
```

Dopo un'opzione possono esserci più argomenti:

```
$ ls -lia /etc /home /var
```

In letteratura, il termine "opzione" è equivalente al termine "parametro," che è più comunemente usato nella programmazione. Il lato opzionale di un'opzione o argomento è simboleggiata dall'inclusione in parentesi quadre `[` e `]`. Quando è possibile più di un'opzione, Una barra verticale chiamata "pipe" li separa `[a|e|i]`.

Comandi generali

comandi `apropos` , `whatis` e `man`

È impossibile per un amministratore a qualsiasi livello conoscere tutti i comandi e le opzioni in dettaglio. Solitamente è disponibile un manuale per tutti i comandi installati.

comando `apropos`

Il comando `apropos` ti consente di cercare per parola chiave all'interno di queste pagine dei manuali:

Opzioni	Descrizione
`-s`, `--sections list` o `--section list`	Limitato alle sezioni dei manuali.
`-a` o `--and`	Visualizza solo la voce corrispondente a tutte le parole chiave fornite.

Esempio:

```
$ apropos clear
clear (1)            - clear the terminal screen
clear_console (1)    - clear the console
clearenv (3)         - clear the environment
clearerr (3)         - check and reset stream status
clearerr_unlocked (3) - nonlocking stdio functions
feclearexcept (3)    - floating-point rounding and exception handling
fwup_clear_status (3) - library to support management of system firmware
updates
klogctl (3)          - read and/or clear kernel message ring buffer; set
console_loglevel
sgt-samegame (6)     - block-clearing puzzle
syslog (2)           - read and/or clear kernel message ring buffer; set
console_loglevel
timerclear (3)       - timeval operations
XClearArea (3)       - clear area or window
XClearWindow (3)     - clear area or window
XSelectionClearEvent (3) - SelectionClear event structure
```

Per trovare il comando che consentirà di cambiare la password di un account:

```
$ apropos --exact password  -a change
chage (1)            - change user password expiry information
passwd (1)           - change user password
```

comando `whatis`

Il comando `whatis` visualizza la descrizione del comando passata come argomento:

```
whatis clear
```

Esempio:

```
$ whatis clear
clear (1)            - clear the terminal screen
```

comando `man`

Una volta trovato con `apropos` o `whatis`, il manuale è letto da `man` ("Man è tuo amico"). Questo set di manuali è diviso in 8 sezioni, raggruppando le informazioni per argomento, la sezione predefinita è la 1:

1. Programmi o comandi eseguibili.
2. Chiamate di sistema (funzioni date dal kernel).
3. Chiamate di libreria (funzioni date dalla libreria).
4. File speciali (di solito si trovano in /dev).
5. Formati di file e convenzioni (file di configurazione come etc/passwd).
6. Giochi (come le applicazioni basate sui personaggi).
7. Varie (es. man (7)).
8. Comandi di amministrazione del sistema (di solito solo per root).
9. Routine del Kernel (non standard).

È possibile accedere alle informazioni su ciascuna sezione digitando `man x intro`, dove `x` è il numero della sezione.

Il comando:

```
man passwd
```

dirà all'amministratore le opzioni, etc, del comando passwd. Mentre:

```
$ man 5 passwd
```

lo informerà sui file relativi al comando.

Navigare nel manuale con le frecce `↑` e `↓`. Uscire dal manuale premendo il tasto `q`.

comando `shutdown`

Il comando `shutdown` consente di **spegnere elettronicamente** un server Linux, immediatamente o dopo un certo periodo di tempo.

```
shutdown [-h] [-r] time [message]
```

Specificare l'ora di spegnimento nel formato `hh:mm` per un'ora precisa, o `+mm` per un ritardo in minuti.

Per forzare un arresto immediato, usa la parola `now` al posto del tempo. In questo caso, il messaggio opzionale non viene inviato agli altri utenti del sistema.

Esempi:

```
[root]# shutdown -h 0:30 "Server shutdown at 0:30"
[root]# shutdown -r +5
```

Opzioni:

Opzioni	Osservazioni
`-h`	Arresta il sistema elettronicamente.
`-r`	Riavvia il sistema.

comando `history`

Il comando `history` visualizza la cronologia dei comandi inseriti dall'utente.

I comandi sono memorizzati nel file `.bash_history` nella directory di accesso dell'utente.

Esempio di un comando history

```
$ history
147 man ls
148 man history
```

Opzioni	Commenti
`-w`	Scrive la cronologia corrente nel file della cronologia
`-c`	Cancella la cronologia della sessione corrente (ma non il contenuto del file `.bash_history`).

- Manipolazione della cronologia:

Per manipolare la history, i seguenti comandi immessi dal prompt dei comandi permetteranno di:

Chiavi	Funzione
!!	Richiama l'ultimo comando eseguito.
!n	Richiama il comando per il suo numero nell'elenco.

Chiavi	Funzione
!string	Richiama il comando più recente che inizia con la stringa.
↑	Naviga nella cronologia andando indietro nel tempo a partire dal comando più recente.
↓	Naviga nella cronologia andando avanti nel tempo.

Autocompletamento

Il completamento automatico è di grande aiuto.

- Completa i comandi, i percorsi inseriti o i nomi dei file.
- Una pressione del tasto `TAB` completa la voce nel caso di una soluzione singola.
- Nel caso di più soluzioni, premere `TAB` una seconda volta per visualizzare le opzioni.

Se premendo due volte il tasto `TAB` non vengono presentate opzioni, non c'è soluzione al completamento attuale.

Visualizzazione e Identificazione

commando `clear`

Il comando `clear` cancella il contenuto della schermata del terminale. Più precisamente, sposta la visualizzazione in modo che il prompt dei comandi si trovi in cima allo schermo, sulla prima riga.

Su un terminale fisico, il display sarà permanentemente nascosto, mentre in un'interfaccia grafica, una barra di scorrimento permetterà di tornare indietro nella cronologia del terminale virtuale.

> **Suggerimento**
>
> `CTRL` + `L` avrà lo stesso effetto del comando `clear`

comando `echo`

Il comando `echo` è usato per visualizzare una stringa di caratteri.

Questo comando è più comunemente usato negli script amministrativi per informare l'utente durante l'esecuzione.

L'opzione `-n` indica nessuna stringa di output newline (di default, stringa di output newline).

```
shell > echo -n "123";echo "456"
123456

shell > echo "123";echo "456"
123
456
```

Per vari motivi, allo sviluppatore dello script potrebbe essere necessario utilizzare sequenze speciali (a partire da un carattere `\`). In questo caso, sara usata l'opzione `-e`, che consentirà l'interpretazione della sequenza.

Tra le sequenze usate frequentemente, possiamo menzionare:

Sequenza	Risultato
`\a`	Invia un bip sonoro
`\b`	Indietro
`\n`	Aggiunge una interruzione di linea
`\t`	Aggiunge un tab orizzontale
`\v`	Aggiunge un tab verticale

comando `date`

Il comando `date` visualizza la data e l'ora. Il comando ha la seguente sintassi:

```
date [-d yyyyMMdd] [format]
```

Esempi:

```
$ date
Mon May 24 16:46:53 CEST 2021
$ date -d 20210517 +%j
137
```

In quest'ultimo esempio, l'opzione `-d` visualizza una determinata data. L'opzione `+%j` formatta

questa data per mostrare solo il giorno dell'anno.

> **Attenzione**
>
> Il formato di una data può cambiare in base al valore della lingua definito nella variabile ambientale '$LANG'.

La visualizzazione della data può seguire i seguenti formati:

Opzione	Formato
`+%A`	Nome completo del giorno della settimana della località (ad esempio, Domenica)
`+%B`	Nome completo del mese della località (ad esempio, Gennaio)
`+%c`	Data e ora di Locale (ad esempio, Gio Mar 3 23:05:25 2005)
`+%d`	Giorno del mese (ad es. 01)
`+%F`	Data nel formato `YYYY-MM-DD`
`+%G`	Anno
`+%H`	Ora (00..23)
`+%j`	Giorno dell'anno (001..366)
`+%m`	Numero del mese (01..12)
`+%M`	Minuto (00..59)
`+%R`	Tempo nel formato `hh:mm`
`+%s`	Secondi dal 1° gennaio 1970
`+%S`	Secondo (00..60)
`+%T`	Tempo nel formato `hh:mm:ss`

Opzione	Formato
`+%u`	Giorno della settimana (`1` per Lunedì)
`+%V`	Numero della settimana (`+%V`)
`+%x`	Data nel formato `GG/MM/AAAA`

Il comando `date` consente anche di modificare la data e l'ora del sistema. In questo caso, verrà utilizzata l'opzione `-s` .

```
[root]# date -s "2021-05-24 10:19"
```

Il formato da utilizzare usando l'opzione `-s` è il seguente:

```
date -s "yyyy-MM-dd hh:mm[:ss]"
```

comando `id` , `who` e `whoami`

Il comando `id` viene utilizzato per visualizzare informazioni su utenti e gruppi. Per impostazione predefinita, non viene aggiunto alcun parametro utente e vengono visualizzate le informazioni dell'utente e del gruppo attualmente connessi。

```
$ id rockstar
uid=1000(rockstar) gid=1000(rockstar) groups=1000(rockstar),10(wheel)
```

Le opzioni `-g` , `-G` , `-n` e `-u` visualizzano il gruppo principale GID, sottogruppo GIDs, nomi al posto di identificatori numerici, e l'UID dell'utente.

Il comando `whoami` visualizza il login dell'utente corrente.

Il solo comando `who` visualizza i nomi degli utenti connessi:

```
$ who
rockstar tty1   2021-05-24 10:30
root     pts/0  2021-05-24 10:31
```

Dal momento che Linux è multi-utente, è possibile che più sessioni siano aperte sulla stessa stazione, sia fisicamente che attraverso la rete. È interessante sapere quali utenti sono connessi, se non solo per comunicare con loro inviando messaggi.

- tty: rappresenta un terminale.
- pts/: rappresenta una console virtuale in un ambiente grafico con il numero dopo aver

rappresentato l'istanza della console virtuale (0, 1, 2...)

L'opzione `-r` visualizza anche il runlevel (vedi capitolo "startup").

Albero Dei File

In Linux, l'albero dei file è un albero invertito, chiamato **albero gerarchico singolo**, la cui radice è la directory `/`.

La **directory corrente** è la directory in cui si trova l'utente.

La **directory di connessione** è la directory di lavoro associata all'utente. Le directory di accesso sono, per impostazione predefinita, memorizzate nella directory `/home`.

Quando l'utente accede, la directory corrente è la directory di accesso.

Un **percorso assoluto** fa riferimento ad un file dalla radice attraversando l'intero albero fino al livello del file:

- `/home/groupA/alice/file`

Un **percorso relativo** fa riferimento allo stesso file attraversando l'intero albero dalla directory corrente:

- `../alice/file`

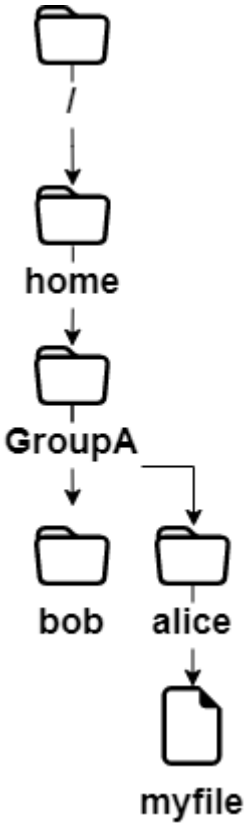
Nell'esempio sopra, il "`..`" si riferisce alla directory principale della directory corrente.

Una directory, anche se è vuota, conterrà necessariamente almeno **due riferimenti**:

- `.`: riferimento a se stessa.
- `..`: riferimento alla directory principale della directory corrente.

Un percorso relativo può quindi iniziare con `./` o `../`. Quando il percorso relativo si riferisce a una sottodirectory o ad un file nella directory corrente, il `./` è spesso omesso. L'inserimento del riferimento `./` sarà veramente richiesto solo per l'esecuzione di un file eseguibile.

Gli errori nei percorsi possono causare molti problemi: dalla creazione di cartelle o file nei luoghi sbagliati, alle eliminazioni involontarie, ecc. È quindi fortemente raccomandato di utilizzare il completamento automatico quando si immettono i percorsi.



Nell'esempio precedente, si vuole fornire la posizione del file `myfile` dalla directory di bob.

- In un **percorso assoluto**, la directory corrente non ha importanza. Iniziamo dalla radice e scendiamo fino alle directory `home`, `groupA`, `alice` e infine il file `myfile`: `/home/groupA/alice/myfile`.
- In un **percorso relativo**, il nostro punto di partenza è la directory corrente `bob`, saliamo di un livello con `..` (i.e., nella directpry `groupA`), poi giù nella directory di alice, e infine il file `myfile`: `../alice/myfile`.

comando `pwd`

Il comando `pwd` (Print Working Directory) visualizza il percorso assoluto della directory corrente.

```
$ pwd
/home/rockstar
```

Per utilizzare un percorso relativo per fare riferimento a un file o a una directory, o per usare il comando `cd` per spostarsi in un'altra directory, è necessario conoscere la sua posizione nell'albero dei file.

A seconda del tipo di shell e dei diversi parametri del suo file di configurazione, il prompt del terminale (noto anche come prompt dei comandi) visualizzerà il percorso assoluto o relativo della directory corrente.

comando `cd`

Il comando `cd` (Cambia Directory) ti permette di cambiare la directory corrente -- in altre parole, di spostarti attraverso l'albero.

```
$ cd /tmp
$ pwd
/tmp
$ cd ../
$ pwd
/
$ cd
$ pwd
/home/rockstar
```

Come puoi vedere nell'ultimo esempio sopra, il comando `cd` senza argomenti sposta la directory corrente alla `home directory`.

comando `ls`

Il comando `ls` visualizza il contenuto di una directory.

```
ls [-a] [-i] [-l] [directory1] [directory2] […]
```

Esempio:

```
$ ls /home
.    ..    rockstar
```

Le opzioni principali del comando `ls` sono:

Opzione	Informazione
`-a`	Visualizza tutti i file, anche quelli nascosti. I file nascosti in Linux sono quelli che iniziano con un `.`.
`-i`	Visualizza i numeri di inode.
`-l`	Utilizza un formato di elenco lungo, cioè ogni riga visualizza informazioni di formato lungo per un file o una directory.

Il comando `ls`, tuttavia, ha molte opzioni (vedi `man`):

Opzione	Informazione

Opzione	Informazione
`-d`	Visualizza le informazioni di una directory invece di elencare i suoi contenuti.
`-g`	Come l'opzione -l, ma non elenca il proprietario.
`-h`	Visualizza le dimensioni dei file nel formato più appropriato (byte, kilobyte, megabyte, gigabyte, ...). `h` stà per Human Readable. Deve essere utilizzato con l'opzione -l.
`-s`	Visualizza la dimensione assegnata di ogni file, in blocchi. Nel sistema operativo GNU/Linux, "block" è l'unità di archiviazione più piccola nel file system, un blocco è uguale a 4096Byte.
`-A`	Visualizza tutti i file nella directory tranne `.` e `..`
`-R`	Visualizza il contenuto delle sottodirectory in modo ricorsivo.
`-F`	Visualizza il tipo di file. Stampa un `/` per una directory, `*` per gli eseguibili, `@` per un collegamento simbolico, e niente per un file di testo.
`-X`	Ordina i file in base alle loro estensioni.

- Descrizione delle colonne generate dall'esecuzione del comando `ls -lia`:

```
$ ls -lia /home
78489 drwx------ 4 rockstar rockstar 4096 25 oct. 08:10 rockstar
```

Valore	Informazione
`78489`	Numero di inode.
`drwx------`	Tipo di file (`d`) e permessi (`rwx------`).
`4`	Numero di sottodirectory. (`.` e `..` incluse). Per un file, rappresenta il numero di collegamenti diretti e 1 rappresenta se stesso.
`rockstar`	Proprietà dell'utente.
`rockstar`	Proprietà del gruppo.

Valore	Informazione
`4096`	Per i file, mostra la dimensione del file. Per le directory, mostra il valore fisso di 4096 byte occupati dal nome del file. Per calcolare la dimensione totale di una directory, usa `du -sh rockstar/`
`25 oct. 08:10`	Ultima data di modifica.
`rockstar`	Il nome del file (o directory).

Nota

Gli **Alias** sono spesso già posizionati nelle distribuzioni comuni.

Questo è il caso dell'alias `ll`:

```
alias ll='ls -l --color=auto'
```

Il comando `ls` ha molte opzioni. Ecco alcuni esempi avanzati di utilizzo:

- Elenca i file in `/etc` in base all'ultima modifica:

```
$ ls -ltr /etc
total 1332
-rw-r--r--.  1 root root    662 29 may   2021 logrotate.conf
-rw-r--r--.  1 root root    272 17 may.   2021 mailcap
-rw-------.  1 root root    122 12 may.  2021 securetty
...
-rw-r--r--.  2 root root     85 18 may.  17:04 resolv.conf
-rw-r--r--.  1 root root     44 18 may.  17:04 adjtime
-rw-r--r--.  1 root root    283 18 may.  17:05 mtab
```

- Elenca i file `/var` di dimensioni superiori a 1 megabyte ma inferiori a 1 gigabyte. L'esempio qui riportato utilizza i comandi avanzati `grep` con le espressioni regolari. I novizi non devono lottare troppo, ci sarà un tutorial speciale per introdurre queste espressioni regolari in futuro.

```
$ ls -lhR /var/ | grep ^\- | grep -E "[1-9]*\.[0-9]*M"
...
-rw-r--r--. 1 apache apache 1.2M 10 may.  13:02 XB RiyazBdIt.ttf
-rw-r--r--. 1 apache apache 1.2M 10 may.  13:02 XB RiyazBd.ttf
```

```
-rw-r--r--. 1 apache apache 1.1M 10 may.  13:02 XB RiyazIt.ttf
...
```

Naturalmente, si consiglia vivamente di utilizzare il comando `find`.

```
$ find /var -size +1M -a -size -1024M -a -type f -exec ls -lh {} \;
```

- Mostra i permessi di una cartella:

Per conoscere i permessi di una cartella, ad esempio `/etc`, il seguente comando **non** sarebbe appropriato:

```
$ ls -l /etc
total 1332
-rw-r--r--.  1 root root     44 18 nov.  17:04 adjtime
-rw-r--r--.  1 root root   1512 12 janv.  2010 aliases
-rw-r--r--.  1 root root  12288 17 nov.  17:41 aliases.db
drwxr-xr-x.  2 root root   4096 17 nov.  17:48 alternatives
...
```

Il comando precedente visualizzerà il contenuto della cartella (all'interno) per impostazione predefinita. Per la cartella stessa, è possibile utilizzare l'opzione `-d`.

```
$ ls -ld /etc
drwxr-xr-x. 69 root root 4096 18 nov.  17:05 /etc
```

- Ordina per dimensione del file, prima il più grande:

```
$ ls -lhS
```

- formato ora/data con `-l`:

```
$ ls -l --time-style="+%Y-%m-%d %m-%d %H:%M" /
total 12378
dr-xr-xr-x. 2 root root 4096 2014-11-23 11-23 03:13 bin
dr-xr-xr-x. 5 root root 1024 2014-11-23 11-23 05:29 boot
```

- Aggiungere la barra ***trailing slash*** alla fine delle cartelle:

Per impostazione predefinita, il comando `ls` non visualizza l'ultima slash di una cartella. In alcuni casi, come per gli script, ad esempio, è utile visualizzarla:

```
$ ls -dF /etc
/etc/
```

- Nascondi alcune estensioni:

```
$ ls /etc --hide=*.conf
```

comando `mkdir`

Il comando `mkdir` crea una directory o un albero di directory.

```
mkdir [-p] directory [directory] [...]
```

Esempio:

```
$ mkdir /home/rockstar/work
```

La directory "rockstar" deve essere presente per creare la directory "work".

Altrimenti, si deve usare l'opzione `-p`. L'opzione `-p` crea le directory genitore se queste non esistono.

> **Pericolo**
>
> Non è consigliabile usare i nomi dei comandi Linux come directory o nomi di file.

comando `touch`

Il comando `touch` modifica il timestamp di un file o crea un file vuoto se il file non esiste.

```
touch [-t date] file
```

Esempio:

```
$ touch /home/rockstar/myfile
```

Opzione	Informazione
`-t date`	Cambia la data di ultima modifica del file con la data specificata.

Formato data: `[AAAA]MMJJhhmm[ss]`

Suggerimento

Il comando `touch` è usato principalmente per creare un file vuoto, ma può essere utile, ad esempio, per i backup incrementali o differenziali. Infatti, l'unico effetto dell'esecuzione di un `touch` su un file sarà quello di forzarne il salvataggio durante il backup successivo.

comando `rmdir`

Il comando `rmdir` elimina una directory vuota.

Esempio:

```
$ rmdir /home/rockstar/work
```

Opzione	Informazione
`-p`	Rimuove la cartella genitore o le cartelle fornite, se sono vuote.

Suggerimento

Per eliminare una cartella non vuota e il suo contenuto, utilizzare il comando `rm`.

comando `rm`

Il comando `rm` elimina un file o una directory.

```
rm [-f] [-r] file [file] [...]
```

Pericolo

Qualsiasi eliminazione di un file o directory è definitiva.

Opzioni	Informazione

Opzioni	Informazione
`-f`	Non chiedere se eliminare.
`-i`	Chiedi se cancellare.
`-r`	Elimina una cartella e cancella ricorsivamente le sue sottocartelle.

> **Nota**
>
> Il comando `rm` non chiede conferma quando si eliminano i file. Tuttavia, con una distribuzione Red Hat/Rocky, `rm` chiede la conferma della cancellazione perché il comando `rm` è un `alias` del comando `rm -i`. Non sorprenderti se su un'altra distribuzione, come Debian, ad esempio, non ottieni una richiesta di conferma.

L'eliminazione di una cartella con il comando `rm`, sia che la cartella sia vuota o meno, richiede l'aggiunta dell'opzione `-r`.

La fine delle opzioni è segnalata alla shell da un doppio trattino `--`.

Nell'esempio:

```
$ >-hard-hard # To create an empty file called -hard-hard
hard-hard
[CTRL+C] To interrupt the creation of the file
$ rm -f -- -hard-hard
```

Il nome del file hard-hard inizia con un `-`. Senza l'uso del `--` la shell avrebbe interpretato il `-d` in `-hard-hard` come un'opzione.

command `mv`

Il comando `mv` muove e rinomina un file.

```
mv file [file ...] destination
```

Esempi:

```
$ mv /home/rockstar/file1 /home/rockstar/file2
$ mv /home/rockstar/file1 /home/rockstar/file2 /tmp
```

Opzioni	Informazione
`-f`	Non chiedere conferma per la sovrascrittura del file di destinazione.
`-i`	Richiedere conferma per la sovrascrittura del file di destinazione (default).

Alcuni casi concreti ti aiuteranno a capire le difficoltà che possono sorgere:

```
$ mv /home/rockstar/file1 /home/rockstar/file2
```

Rinomina `file1` in `file2`. Se `file2` esiste già, sostituisci il contenuto del file con `file1`.

```
$ mv /home/rockstar/file1 /home/rockstar/file2 /tmp
```

Sposta `file1` e `file2` nella cartella `/tmp`.

```
$ mv file1 /repexist/file2
```

Sposta `file1` in `repexist` e lo rinomina `file2`.

```
$ mv file1 file2
```

`file1` è rinominato con `file2`.

```
$ mv file1 /repexist
```

Se esiste la cartella di destinazione, `file1` viene spostato in `/repexist`.

```
$ mv file1 /wrongrep
```

Se la directory di destinazione non esiste, `file1` viene rinominato in `wrongrep` nella directory principale.

comando `cp`

Il comando `cp` copia un file.

```
cp file [file ...] destination
```

Esempio:

```
$ cp -r /home/rockstar /tmp
```

Opzioni	Informazione
`-i`	Richiesta di conferma per la sovrascrittura (default).
`-f`	Non chiedere conferma per la sovrascrittura del file di destinazione.
`-p`	Mantiene il proprietario, le autorizzazioni e il timestamp del file copiato.
`-r`	Copia una directory con i suoi file e sottodirectory.
`-s`	Crea un collegamento simbolico invece di copiare.

```
cp file1 /repexist/file2
```

`file1` viene copiato in `/repexist` con il nome `file2`.

```
$ cp file1 file2
```

`file1` viene copiato come `file2` in questa cartella.

```
$ cp file1 /repexist
```

Se esiste la directory di destinazione, `file1` viene copiato in `/repexist`.

```
$ cp file1 /wrongrep
```

Se la directory di destinazione non esiste, `file1` è copiato sotto il nome `wrongrep` nella directory principale.

Visualizzazione

comando `file`

Il comando `file` visualizza il tipo di un file.

```
file file1 [files]
```

Esempio:

```
$ file /etc/passwd /etc
/etc/passwd:    ASCII text
/etc:        directory
```

comando `more`

Il comando `more` visualizza il contenuto di uno o più file schermata per schermata.

```
more file1 [files]
```

Esempio:

```
$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
```

Usando il tasto `ENTER`, lo spostamento è linea per linea. Usando il tasto `SPACE`, lo spostamento è pagina per pagina. `/text` Ti consente di cercare la corrispondenza nel file.

comando `less`

Il comando `less` visualizza il contenuto di uno o più file. Il comando `less` è interattivo e ha i propri comandi per l'uso.

```
less file1 [files]
```

I comandi specifici per `less` sono:

Comando	**Azione**
`h`	Aiuto.
`↑` `↓` `→` `←`	Sposta su, giù di una linea, o a destra e sinistra.
`Invio`	Sposta giù di una riga.
`Spazio`	Sposta giù di una pagina.
`PgUp` e `PgDn`	Sposta su o giù di una pagina.
`gg` e `G`	Passa alla prima e all'ultima pagina

Comando	Azione
`/text`	Cerca il testo.
`q`	Chiude il comando `less`.

comando `cat`

Il comando `cat` concatena il contenuto di più file e visualizza il risultato sull'output standard.

```
cat file1 [files]
```

Esempio 1 - Visualizzazione del contenuto di un file in output standard:

```
$ cat /etc/passwd
```

Esempio 2 - Visualizzazione del contenuto di più file in output standard:

```
$ cat /etc/passwd /etc/group
```

Esempio 3 - Combinare il contenuto di più file in un unico file utilizzando il reindirizzamento dell'output:

```
$ cat /etc/passwd /etc/group > usersAndGroups.txt
```

Esempio 4 - Visualizzazione della numerazione di linea:

```
$ cat -n /etc/profile
     1    # /etc/profile: system-wide .profile file for the Bourne shell
(sh(1))
     2    # and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
     3
     4    if [ "`id -u`" -eq 0 ]; then
     5      PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin"
     6    else
…
```

Example 5 - Mostra la numerazione delle righe non vuote:

```
$ cat -b /etc/profile
     1    # /etc/profile: system-wide .profile file for the Bourne shell
(sh(1))
     2    # and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

     3    if [ "`id -u`" -eq 0 ]; then
```

```
     4      PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin"
     5    else
…
```

comando `tac`

Il comando `tac` fa quasi il contrario del comando `cat`. Visualizza il contenuto di un file a partire dalla fine (che è particolarmente interessante per la lettura dei log!).

Esempio: Visualizza un file di log visualizzando prima l'ultima riga:

```
[root]# tac /var/log/messages | less
```

comando `head`

Il comando `head` visualizza l'inizio di un file.

```
head [-n x] file
```

Opzione	Descrizione
`-n x`	Visualizzare le prime `x` righe del file

Per impostazione predefinita (senza l'opzione `-n`), il comando `head` visualizzerà le prime 10 righe del file.

comando `tail`

Il comando `tail` visualizza la fine di un file.

```
tail [-f] [-n x] file
```

Opzione	Descrizione
`-n x`	Visualizza le ultime `x` righe del file
`-f`	Visualizza le modifiche al file in tempo reale

Esempio:

```
tail -n 3 /etc/passwd
sshd:x:74:74:Privilege-separeted sshd:/var/empty /sshd:/sbin/nologin
tcpdump::x:72:72::/:/sbin/nologin
user1:x:500:500:grp1:/home/user1:/bin/bash
```

Con l'opzione `-f`, le informazioni di modifica del file verranno sempre emesse a meno che l'utente non esca dallo stato di monitoraggio con `CTRL` + `C`. Questa opzione è molto utilizzata per tracciare i file di log (i registri) in tempo reale.

Senza l'opzione `-n`, il comando `tail` mostra le ultime 10 righe del file.

comando `sort`

Il comando `sort` ordina le linee di un file.

Permette di ordinare il risultato di un comando o il contenuto di un file in un determinato ordine, numerico, alfabetico, per dimensione (KB, MB, GB) o in ordine inverso.

```
sort [-k] [-n] [-u] [-o file] [-t] file
```

Esempio:

```
$ sort -k 3,4 -t ":" -n /etc/passwd
root:x:0:0:root:/root:/bin/bash
adm:x:3:4:adm:/var/adm/:/sbin/nologin
```

Opzione	**Descrizione**
`-k`	Specifica le colonne da separare. È possibile specificare più colonne.
`-n`	Richiede un ordinamento numerico.
`-o file`	Salva l'ordinamento nel file specificato.
`-t`	Specificare un delimitatore, che richiede che i contenuti del file corrispondente siano contenuti di colonne regolarmente delimitate, altrimenti non possono essere ordinati correttamente.
`-r`	Inverte l'ordine del risultato. Usato insieme all'opzione `-n` per ordinare dal più grande al più piccolo.
`-u`	Rimuovi i duplicati dopo l'ordinamento. Equivalente a `sort file | uniq`.

Il comando `sort` ordina il file solo sullo schermo. Il file non è modificato dall'ordinamento. Per salvare l'ordinamento, utilizzare l'opzione `-o` o un reindirizzamento dell'output `>`.

Per impostazione predefinita, i numeri sono ordinati in base al loro carattere. Pertanto, "110" sarà prima di "20", che a sua volta sarà prima di "3". L'opzione `-n` deve essere specificata in modo che i blocchi di caratteri numerici siano ordinati in base al loro valore.

Il comando `sort` inverte l'ordine dei risultati, con l'opzione `-r`:

```
$ sort -k 3 -t ":" -n -r /etc/passwd
nobody:x:65534:65534:Kernel Overflow User:/:/sbin/nologin
systemd-coredump:x:999:997:systemd Core Dumper:/:/sbin/nologin
polkitd:x:998:996:User for polkitd:/:/sbin/nologin
```

In questo esempio, il comando `sort` ordinerà il contenuto del file `/etc/passwd` dal più grande uid (identificatore utente) al più piccolo.

Alcuni esempi avanzati di utilizzazione del comando `sort`:

- Mischiando i valori

Il comando `sort` permette anche di mescolare i valori con l'opzione `-R`:

```
$ sort -R /etc/passwd
```

- Ordinamento degli indirizzi IP

Un amministratore di sistema si trova ben presto di fronte all'elaborazione di indirizzi IP dai log dei suoi servizi, come SMTP, VSFTP o Apache. Questi indirizzi vengono tipicamente estratti con il comando `cut`.

Ecco un esempio con il file `dns-client.txt`:

```
192.168.1.10
192.168.1.200
5.1.150.146
208.128.150.98
208.128.150.99
```

```
$ sort -nr dns-client.txt
208.128.150.99
208.128.150.98
192.168.1.200
192.168.1.10
5.1.150.146
```

- Ordinamento dei file mediante la rimozione dei duplicati

Il comando `sort` sa come rimuovere i duplicati dall'output del file utilizzando l'opzione `-u`.

Ecco un esempio con il file `colours.txt`:

```
Red
Green
Blue
Red
Pink
```

```
$ sort -u colours.txt
Blue
Green
Pink
Red
```

- Ordinamento dei file in base alle dimensioni

Il comando `sort` sa come riconoscere le dimensioni dei file, da comandi come `ls` con l'opzione `-h`.

Ecco un esempio con il file `size.txt`:

```
1.7G
18M
69K
2.4M
1.2M
4.2G
6M
124M
12.4M
4G
```

```
$ sort -hr size.txt
4.2G
4G
1.7G
124M
18M
12.4M
6M
2.4M
1.2M
69K
```

comando `wc`

Il comando `wc` conteggia il numero di righe, parole e/o byte di un file.

```
wc [-l] [-m] [-w] file [files]
```

Opzione	Descrizione
`-c`	Conta il numero di byte.
`-m`	Conta il numero di caratteri.
`-l`	Conta il numero di linee.
`-w`	Conta il numero di parole.

Ricerca

comando `find`

Il comando `find` cerca la posizione di file o cartelle.

```
find directory [-name name] [-type type] [-user login] [-date date]
```

Poiché le opzioni del comando `find` sono numerose, è meglio fare riferimento al comando `man`.

Se la cartella di ricerca non è specificata, il comando `find` effettuerà la ricerca dalla cartella corrente.

Opzione	Descrizione
`-perm permissions`	Cerca i file in base ai loro permessi.
`-size size`	Ricerca dei file in base alle dimensioni.

opzione `-exec` del comando `find`

È possibile utilizzare l'opzione `-exec` del comando `find` per eseguire un comando su ogni riga del risultato:

```
$ find /tmp -name *.txt -exec rm -f {} \;
```

Il comando precedente cerca tutti i file nella cartella `/tmp` denominati `*.txt` e li elimina.

Comprendere l'opzione `-exec`

Nell'esempio precedente, il comando `find` costruirà una stringa che rappresenta il comando da eseguire.

Se il comando `find` trova tre file chiamati `log1.txt`, `log2.txt` e `log3.txt`, allora il comando `find` costruirà la stringa sostituendo le parentesi nella stringa `rm -f {} \;` con uno dei risultati della ricerca, e lo farà tante volte quanti sono i risultati.

Questo ci darà:

```
rm -f /tmp/log1.txt ; rm -f /tmp/log2.txt ; rm -f /tmp/log3.txt
;
```

Il carattere `;` è un carattere speciale della shell che deve essere protetto da `\` per evitare che venga interpretato troppo presto dal comando `find` (e non nel `-exec`).

Suggerimento

`$ find /tmp -name *.txt -delete` fa la stessa cosa.

comando `whereis`

Il comando `whereis` ricerca i file relativi a un comando.

```
whereis [-b] [-m] [-s] command
```

Esempio:

```
$ whereis -b ls
ls: /bin/ls
```

Opzione	Descrizione
`-b`	Esegue la ricerca solo del file binario.

Opzione	Descrizione
`-m`	Ricerca solo per le pagine man.
`-s`	Ricerca solo per file sorgente.

command `grep`

Il comando `grep` cerca una stringa in un file.

```
grep [-w] [-i] [-v] "string" file
```

Esempio:

```
$ grep -w "root:" /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Opzione	Descrizione
`-i`	Ignora le maiuscole della stringa cercata.
`-v`	Esclude le linee contenenti la stringa.
`-w`	Cerca la parola esatta.

Il comando `grep` restituisce la riga completa contenente la stringa cercata. * Il carattere speciale `^` viene utilizzato per cercare una stringa all'inizio di una riga. * Il carattere speciale `$` viene utilizzato per cercare una stringa alla fine di una riga.

```
$ grep -w "^root" /etc/passwd
```

Nota

Questo comando è molto potente e si raccomanda vivamente di consultare il suo manuale. Ha numerosi derivati.

È possibile cercare una stringa in un albero di file con l'opzione `-R`.

```
grep -R "Virtual" /etc/httpd
```

Meta-caratteri (wildcards)

I Meta-caratteri sostituiscono uno o più caratteri (o anche un'assenza di caratteri) durante una ricerca. Questi meta-caratteri sono anche noti come wildcards.

Possono essere combinati.

Il carattere `*` sostituisce una stringa composta da qualsiasi carattere. Il carattere `*` può anche rappresentare un'assenza di caratteri.

```
$ find /home -name "test*"
/home/rockstar/test
/home/rockstar/test1
/home/rockstar/test11
/home/rockstar/tests
/home/rockstar/test362
```

I Meta-caratteri consentono ricerche più complesse sostituendo tutto o parte di una parola. Sostituiscono semplicemente le incognite con questi caratteri speciali.

Il carattere `?` sostituisce un singolo carattere, qualunque esso sia.

```
$ find /home -name "test?"
/home/rockstar/test1
/home/rockstar/tests
```

Le parentesi quadre `[` e `]` sono usate per specificare i valori che un singolo carattere può assumere.

```
$ find /home -name "test[123]*"
/home/rockstar/test1
/home/rockstar/test11
/home/rockstar/test362
```

Nota

Circondare sempre le parole contenenti metacaratteri con `"` per evitare che vengano sostituite dai nomi dei file che soddisfano i criteri.

> **Attenzione**
>
> Non confondete i meta-caratteri della shell con i meta-caratteri delle espressioni regolari. Il comando `grep` utilizza meta-caratteri di espressione regolare.

Reindirizzamenti e pipes

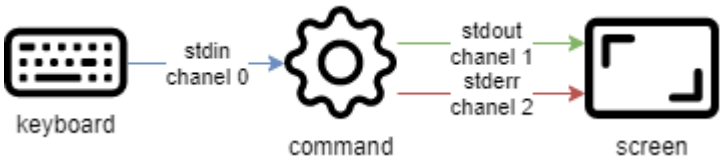
Standard input e output

Sui sistemi UNIX e Linux, ci sono tre flussi standard. Consentono ai programmi, attraverso la libreria `stdio.h`, di inviare e ricevere informazioni.

Questi flussi sono chiamati canale X o descrittore di file X.

Per impostazione predefinita:

- la tastiera è il dispositivo di input per il canale 0, chiamato **stdin** ;
- lo schermo è il dispositivo di uscita per i canali 1 e 2, chiamati **stdout** e **stderr**.



stderr riceve i flussi di errore restituiti da un comando. Gli altri flussi sono diretti a **stdout**.

Questi flussi puntano ai file delle periferiche, ma poiché tutto è un file in UNIX/Linux, i flussi di I/O possono essere facilmente indirizzati ad altri file. Questo principio è la forza della shell.

Redirezione Input

È possibile reindirizzare il flusso di input da un altro file con il carattere `<` o `<<`. Il comando leggerà il file invece della tastiera:

```
$ ftp -in serverftp << ftp-commands.txt
```

Nota

Solo i comandi che richiedono l'input da tastiera saranno in grado di gestire il reindirizzamento dell'input.

Il reindirizzamento dell'ingresso può anche essere utilizzato per simulare l'interattività dell'utente. Il comando leggerà il flusso di ingresso finché non incontrerà la parola chiave definita dopo il reindirizzamento dell'ingresso.

Questa funzione è utilizzata per i comandi interattivi negli script:

```
$ ftp -in serverftp << END
user alice password
put file
bye
END
```

La parola chiave `END` può essere sostituita da qualsiasi parola.

```
$ ftp -in serverftp << STOP
user alice password
put file
bye
STOP
```

La shell esce dal comando `ftp` quando riceve una linea contenente solo la parola chiave.

Attenzione

La parola chiave finale, qui `END` o `STOP`, deve essere l'unica parola sulla linea e deve essere all'inizio della linea.

Il reindirizzamento dello standard input è usato raramente, perché la maggior parte dei comandi accetta un nome di file come argomento.

Il comando `wc` potrebbe essere usato in questo modo:

```
$ wc -l .bash_profile
27 .bash_profile # the number of lines is followed by the file name
$ wc -l < .bash_profile
27 # returns only the number of lines
```

Redirezione Output

L'output standard può essere reindirizzato ad altri file usando il carattere `>` o `>>`.

Il semplice reindirizzamento `>` sovrascrive il contenuto del file di output:

```
$ date +%F > date_file
```

Quando viene utilizzato il carattere `>>`, indica che il risultato del comando viene aggiunto al contenuto del file.

```
$ date +%F >> date_file
```

In entrambi i casi, il file viene creato automaticamente quando non esiste.

L'output di errore standard può anche essere reindirizzato ad un altro file. Questa volta sarà necessario specificare il numero del canale (che può essere omesso per i canali 0 e 1):

```
$ ls -R / 2> errors_file
$ ls -R / 2>> errors_file
```

Esempi di reindirizzamento

Reindirizzamento di 2 output verso 2 file:

```
$ ls -R / >> ok_file 2>> nok_file
```

Reindirizzamento di 2 output a un singolo file:

```
$ ls -R / >> log_file 2>&1
```

Reindirizzamento del ***stderr*** a un "pozzo senza fondo" (`/dev/null`):

```
$ ls -R / 2>> /dev/null
```

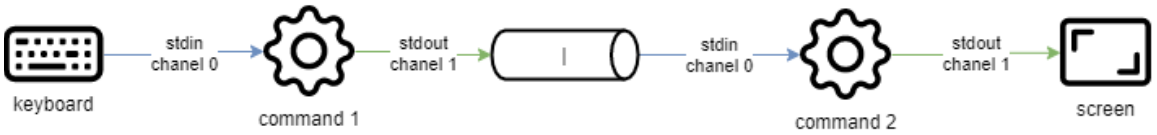
Quando entrambi i flussi di uscita vengono reindirizzati, nessuna informazione viene visualizzata sullo schermo. Per utilizzare sia il reindirizzamento dell'uscita che il mantenimento della visualizzazione, è necessario utilizzare il comando `tee`.

Pipes

Una **pipe** è un meccanismo che consente di collegare l'output standard di un primo comando all'ingresso standard di un secondo comando.

Questa comunicazione è unidirezionale ed è fatta con il simbolo `|`. Il simbolo della pipe `|` è

ottenuto premendo il tasto `SHIFT` + `|` contemporaneamente.



Tutti i dati inviati dal controllo a sinistra della pipe attraverso il canale di uscita standard vengono inviati al canale di ingresso standard del controllo a destra.

I comandi particolarmente utilizzati dopo una pipe sono i filtri.

- Esempi:

Mostra solo l'inizio:

```
$ ls -lia / | head
```

Mostra solo la fine:

```
$ ls -lia / | tail
```

Ordina il risultato:

```
$ ls -lia / | sort
```

Conta il numero di parole / caratteri:

```
$ ls -lia / | wc
```

Cerca una stringa nel risultato:

```
$ ls -lia / | grep fichier
```

Punti Speciali

comando `tee`

Il comando `tee` viene utilizzato per reindirizzare l'output standard di un comando a un file mantenendo la visualizzazione sullo schermo.

Viene combinato con la pipe `|` per ricevere come input l'output del comando da reindirizzare:

```
$ ls -lia / | tee fic
$ cat fic
```

L'opzione `-a` aggiunge al file invece di sovrascriverlo.

comandi `alias` e `unalias`

L'uso di **alias** è un modo per chiedere alla shell di ricordare un particolare comando con le sue opzioni e di dargli un nome.

Per esempio:

```
$ ll
```

sostituirà il comando:

```
$ ls -l
```

Il comando `alias` elenca gli alias per la sessione corrente. Gli alias sono stabiliti per impostazione predefinita sulle distribuzioni Linux. Qui, gli alias per un server Rocky Linux:

```
$ alias
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias vi='vim'
alias which='alias | /usr/bin/which --tty-only --read-alias --show-dot --show-tilde'
```

Gli alias sono definiti solo temporaneamente, per il tempo della sessione utente.

Per un uso permanente, devono essere creati nel:

- file `.bashrc` nella directory di accesso dell'utente;
- file `/etc/bashrc` per tutti gli utenti.

Attenzione

Particolare attenzione deve essere prestata quando si utilizzano alias che possono essere potenzialmente pericolosi! Ad esempio, un alias creato senza una conoscenza di base di amministratore:

```
alias cd='rm -Rf'
```

Il comando `unalias` ti consente di eliminare gli alias.

Per eliminare un singolo alias:

```
$ unalias ll
```

Per eliminare tutti gli alias:

```
$ unalias -a
```

Per disabilitare temporaneamente un alias, la combinazione è `\<nome alias>`.

Ad esempio se digitiamo:

```
$ type ls
```

potrebbe restituire quanto segue:

```
ls is an alias to « ls -rt »
```

Ora che questo è noto, possiamo vedere i risultati dell'utilizzo dell'alias o disabilitarlo in una volta con il carattere `\` eseguendo il seguente:

```
$ ls file*   # order by time
file3.txt  file2.txt  file1.txt
$ \ls file*  # order by name
file1.txt  file2.txt  file3.txt
```

Alias e Funzioni Utili

- alias di `grep`.

Colora il risultato del comando `grep`: `alias grep='grep --color=auto'`

- funzione `mcd`

È comune creare una cartella e poi spostarsi al suo interno: `mcd() { mkdir -p "$1"; cd "$1"; }`

- funzione `cls`

Spostarsi in una cartella ed elencarne il contenuto: `cls() { cd "$1"; ls; }`

- funzione `backup`

Crea una copia di backup di un file: `backup() { cp "$1"{,.bak}; }`

- funzione `extract`

Estrae qualsiasi tipo di archivio:

```
extract () {
  if [ -f $1 ] ; then
    case $1 in
      *.tar.bz2) tar xjf $1 ;;
      *.tar.gz) tar xzf $1 ;;
      *.bz2) bunzip2 $1 ;;
      *.rar) unrar e $1 ;;
      *.gz) gunzip $1 ;;
      *.tar) tar xf $1 ;;
      *.tbz2) tar xjf $1 ;;
      *.tgz) tar xzf $1 ;;
      *.zip) unzip $1 ;;
      *.Z) uncompress $1 ;;
      *.7z) 7z x $1 ;;
      *)
        echo "'$1' cannot be extracted via extract()" ;;
    esac
  else
    echo "'$1' is not a valid file"
  fi
}
```

- Se `l'alias cmount` restituisce quanto segue: `alias cmount="mount | column -t"`

Si può quindi usare `cmount` per mostrare tutti i mount del sistema in colonne come la seguente: `[root]# cmount`

che restituisce il nostro filesystem montato nel seguente formato:

```
/dev/simfs   on   /                                    type   simfs
(rw,relatime,usrquota,grpquota)
proc         on   /proc                                type   proc
(rw,relatime)
sysfs        on   /sys                                 type   sysfs
(rw,relatime)
none         on   /dev                                 type   devtmpfs
(rw,relatime,mode=755)
none         on   /dev/pts                             type   devpts
(rw,relatime,mode=600,ptmxmode=000)
none         on   /dev/shm                             type   tmpfs
(rw,relatime)
none         on   /proc/sys/fs/binfmt_misc             type   binfmt_misc
(rw,relatime)
```

Il carattere `;`

Il carattere `;` concatena i comandi.

Una volta che l'utente ha premuto
`INVIO, tutti i comandi verranno eseguiti in sequenza nell'ordine di immissione.`

```
$ ls /; cd /home; ls -lia; cd /
```

Verificare le proprie Conoscenze

:heavy_check_mark: Cosa definisce un utente sotto Linux? (7 risposte)

:heavy_check_mark: Cosa caratterizza un'opzione lunga per un comando?

:heavy_check_mark: Quali comandi ti permettono di cercare aiuto su un comando?

- [ ] `google`
- [ ] `chuck --norris`
- [ ] `info`
- [ ] `apropos`
- [ ] `whatis`

:heavy_check_mark: Quale comando ti permette di vedere la cronologia di un utente?

:heavy_check_mark: Quale comando ti permette di cercare del testo in un file?

- [ ] `find`
- [ ] `grep`

:heavy_check_mark: Quale comando ti permette di cercare un file?

- [ ] `find`
- [ ] `grep`

:heavy_check_mark: Quale comando reindirizza il flusso di errore di un comando a un nuovo file `errors.log`?

- [ ] `ls -R / 2> errors.log`
- [ ] `ls -R / 2>> errors.log`
- [ ] `ls -R / 2> errors.log 2>&1`

Ultimo aggiornamento: 22 maggio 2023

Author: Antoine Le Morvan

Contributors: Steven Spencer, Aditya Putta, Franco Colussi, Grammaresque