Bash

È una shell: interprete di comandi

CLI: Command Line Interface

al prompt si digitano comandi

```
$ echo ciao
ciao
```

`echo` è il comando che *ripete* il parametro

```
$ date
Wed Sep 13 05:00:59 PM CEST 2023
```

così si ottiene l'ora esatta

sintassi tipica dei comandi

comando [*parametri opzionali*] [*target del comando*]

in altre parole

verbo [*complementi vari*] [*complemento oggetto*]

esempio1: lista i file (sottinteso della directory corrente)

```
$ ls
ls
Desktop    Downloads  Pictures  Templates  lecture_examples
Documents  Music      Public    Videos     unibot
```

Le opzioni solitamente sono indicate con un `-` e un carattere, es `-l`. In alternativa le opzioni possono essere indicate con `--` e una stringa, es `--all`. Le opzioni possono avere parametri.

esempio2: lista i file in formato *lungo* i.e. dettagliato (sottinteso della directory corrente)

```
$ ls -l
total 40
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Desktop
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Documents
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Downloads
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Music
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Pictures
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Public
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Templates
drwxr-xr-x 2 so2324 so2324 4096 Aug 26 17:33 Videos
drwxr-xr-x 2 so2324 so2324 4096 Sep  2 12:11 lecture_examples
drwxr-xr-x 3 so2324 so2324 4096 Oct 26  2022 unibot
```

esempio3: lista la directory '/' (la root directory)

```
$ ls /
bin   etc    initrd.img      lib64       mnt   root  srv  usr      vmlinuz.old
boot  extra  initrd.img.old  lost+found  opt   run   sys  var
dev   home   lib             media       proc  sbin  tmp  vmlinuz
```

esempio4: lista in formato lungo la root directory:

```
$ ls -l /
total 72
lrwxrwxrwx   1 root root     7 Aug 26 17:22 bin -> usr/bin
drwxr-xr-x   3 root root  4096 Aug 26 17:28 boot
drwxr-xr-x  17 root root  3340 Aug 28 13:59 dev
drwxr-xr-x 136 root root 12288 Aug 26 17:31 etc
drwxr-xr-x   8 root root  4096 Aug 26 12:07 extra
drwxr-xr-x   8 root root  4096 Aug 26 17:30 home
lrwxrwxrwx   1 root root    30 Aug 26 17:26 initrd.img -> boot/initrd.img-6.1.0-10-amd64
lrwxrwxrwx   1 root root    31 Aug 26 17:26 initrd.img.old -> boot/initrd.img-5.10.0-22-amd6
lrwxrwxrwx   1 root root     7 Aug 26 17:22 lib -> usr/lib
lrwxrwxrwx   1 root root     9 Aug 26 17:22 lib64 -> usr/lib64
drwx------   2 root root 16384 Sep 20  2018 lost+found
drwxr-xr-x   2 root root  4096 Sep 20  2018 media
drwxr-xr-x   2 root root  4096 Nov  6  2020 mnt
drwxr-xr-x   2 root root  4096 Sep 20  2018 opt
dr-xr-xr-x 229 root root     0 Aug 26 17:29 proc
drwx------  17 root root  4096 Sep 13 17:12 root
drwxr-xr-x  25 root root   760 Sep 13 17:08 run
lrwxrwxrwx   1 root root     8 Aug 26 17:22 sbin -> usr/sbin
drwxr-xr-x   2 root root  4096 Sep 20  2018 srv
dr-xr-xr-x  13 root root     0 Aug 26 17:29 sys
drwxrwxrwt  13 root root  4096 Sep 13 12:06 tmp
drwxr-xr-x  15 root root  4096 Aug 26 17:22 usr
drwxr-xr-x  11 root root  4096 Sep 20  2018 var
lrwxrwxrwx   1 root root    27 Aug 26 17:26 vmlinuz -> boot/vmlinuz-6.1.0-10-amd64
lrwxrwxrwx   1 root root    28 Aug 26 17:26 vmlinuz.old -> boot/vmlinuz-5.10.0-22-amd64
```

il comando più importante `man`

Il comando `man` consente di consultare il manuale di ogni comando:

```
$ man ls
LS(1)                            User Commands                                LS(1)

NAME
       ls - list directory contents

SYNOPSIS
```

```
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory by default).  Sort
       entries alphabetically if none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

.................
       -l     use a long listing format
.................
```

Ovviamente anche il comando `man` ha la sua pagina di manuale

```
$ man man
MAN(1)                          Manual pager utils                          MAN(1)

NAME
       man - an interface to the system reference manuals

SYNOPSIS
       man [man options] [[section] page ...] ...
       man -k [apropos options] regexp ...
       man -K [man options] [section] term ...
       man -f [whatis options] page ...
       man -l [man options] file ...
       man -w|-W [man options] page ...

DESCRIPTION
       man is the system's manual pager.  Each page argument given to man is nor-
       mally the name of a program, utility or function.  The manual page associ-
       ated with each of these arguments is then found and displayed.  A section,
       if provided, will direct man to look only in that section of  the  manual.
       The default action is to search in all of the available sections following
       a pre-defined order (see DEFAULTS), and to show only the first page found,
       even if page exists in several sections.

       The  table  below  shows the section numbers of the manual followed by the
       types of pages they contain.

       1   Executable programs or shell commands
       2   System calls (functions provided by the kernel)
       3   Library calls (functions within program libraries)
       4   Special files (usually found in /dev)
```

```
       5   File formats and conventions, e.g. /etc/passwd
       6   Games
       7   Miscellaneous (including macro packages and conventions), e.g. man(7),
           groff(7), man-pages(7)
       8   System administration commands (usually only for root)
       9   Kernel routines [Non standard]

       A manual page consists of several sections.
...........
```

Come indicato `man` non fornisce solo le pagine dei manuali per i comandi ma anche per funzioni di libreria, system call, strumenti di amministrazione. ogni *sezione* è identificata da un numero.

Nella prima riga c'è scritto il tipo di manuale (e.g. `MAN(1)` indica che si tratta della sezione 1: Executable programs or shell commands.).

Se si vuole limitare la ricerca alla sezione 1:

```
$ man 1 ls
LS(1).....
```

L'opzione `-k` (keyword) consente di cercare una pagina di manuale partendo da una parola chiave:

```
$ man -k -s 1 manual
aclocal (1)          - manual page for aclocal 1.16.1
aclocal-1.16 (1)     - manual page for aclocal 1.16.5
apropos (1)          - search the manual page names and descriptions
automake (1)         - manual page for automake 1.16.1
automake-1.16 (1)    - manual page for automake 1.16.5
avr-man (1)          - a man(1) replacement to access the avr-libc manual pages
man (1)              - an interface to the system reference manuals
man-recode (1)       - convert manual pages to another encoding
manconv (1)          - convert manual page from one encoding to another
manpath (1)          - determine search path for manual pages
whatis (1)           - display one-line manual page descriptions
whereis (1)          - locate the binary, source, and manual page files for a com...
xman (1)             - Manual page display program for the X Window System
```

l'albero del File System

In UNIX/Linux c'è un solo albero del FIle System, quello che contiene directory, sottodirectory ecc.

La root directory alla radice dell'albero (che si indica con `/`) ha alcune directory standard:

- `/bin` Comandi base per utenti "comuni"
- `/sbin` Comandi per la gestione del sistema, non destinati ad utenti comuni

- `/dev` Device file per accedere a periferiche o sistemi di memorizzazione
- `/etc` File di configurazione del sistema
- `/home` "Home directory" degli utenti
- `/lib` Librerie condivise dai programmi e utili per il loro funzionamento
- `/tmp` Directory dei file temporanei
- `/var` Dati variabili, code di stampa, log del sistema
- `/usr` Contiene gran parte dei programmi esistenti nel sistema

Per motivi storici i programmi e le librerie non necessari al boot o per ripristinare il sistema stanno nella directory `/usr`.

- `/usr/bin` Comandi per utenti "comuni"
- `/usr/include` File header per programmi C o C++
- `/usr/lib` Librerie condivise dai programmi e utili per il loro funzionamento
- `/usr/local` Programmi e librerie locali al sistema (non gestiti dalle distribuzioni
- `/usr/sbin` Comandi per la gestione del sistema, non destinati ad utenti comuni

Le home directory degli utenti sono nella directory `home`. La directory personale dell'utente maria è `/home/maria`

Nota: nel sistema distribuito del Dipartimento gli utenti sono normalmente identificati con il nome nel Directory Service di Ateneo (DSA). Normalmente è *nome.cognome* con talvolta un suffisso numerico per gestire le omonimie. Le home directory sono divise per categorie di utenti. Lo studente Alan Turing userebbe come username `alan.turing` e avrebbe come home directory `/home/students/alan.turing`.

La shell (in realtà come tutti i processi) ha una directory corrente Quando parte una nuova shell (su un terminale si fa il login con username e paswword o quando si apre una finestra con un terminale virtuale) normalmente la directory corrente è la home directory.

Per indicare un file si possono usare percorsi (pathname)

- assoluti: indicando il percorso dalla radice: `/home/renzo/sistemi_operativi/hello_world.c` i pathname assoluti iniziano con il carattere `/`
- relativi: indicando il percorso dalla dir corrente. Es se ha dir corrente è `/home/renzo` si indica il file precedente anche come `sistemi_operativi/hello_world.c`
- relativi indiretti. Sono percorsi relativi dove compare `..` per *salire* alla directory precedente: es `../lucia/prog2.c`.

In realtà nei sistemi UNIX i pathname non servono solo per indicare i file ma sono uno strumento per individuare dispositivi (device) o canali di comunicazione (socket UNIX o fifo per esempio). Ad ognuna di queste entità viene associato un file speciale (un finto file). Il pathname del file consente dare un nome utilizzabile nei comandi o nella scrittura dei programmi.

wildcard Nella scrittura dei pathname la shell interpreta alcuni simboli speciali:

- `*` (asterisco) sostituisce qualsiasi stringa
- `?` sostituisce un singolo carattere
- `[..]` sostituisce un carattere di quelli identificati nel range

esempi:

- `*.pdf` tutti i file con suffisso pdf
- `2023*` tutti i file che iniziano con 2023
- `mio???` tutti i file che hano un nome di 6 lettere e iniziano con mio
- 'prova[1-6].c i file prova1.c, prova2.c ... fino a prova6.c (se esistono).

i file standard, ridirezione e pipe

Alla loro attivazione da parte della shell i comandi hanno tre file standard aperti:

- `stdin` o standard input
- `stdout` o standard output
- `stderr` o standard error

Se non diversamente specificato i tre file fanno riferimenti al terminale dal quale è stato lanciato il programma.
Per esempio il comando `cat` senza parametri semplicemente copia `stdin` in `stdout`.

```
$ cat
```

a questo punto ogni riga scritta viene ripetuta dal sistema. (premere *ctrl-d* per terminare).

È possibile *ridirezionare* i file standard usando i caratteri `<` per lo `stdin` e `>` per lo `stdout`.

es:

```
$ cat > prova
```

ogni riga scritta viene scritta nel file `prova`. (premere sempre *ctrl-d* per terminare). Lo `stdout` è stato *ridirezionato* al file prova.

se ora si digita:

```
cat < prova
```

questo comando mostrerà le righe fornite al comando precedente. `cat` copia sempre lo `stdin` in `stdout` ma ora lo `stdin` è stato ridirezionato.

quindi il comando:

```
cat < originale > copia
```

copia il file `originale` nel file `copia` (esempio didattico, per copiare un file si usa normalmente il comando `cp` che è più versatile ed efficiente.

Infine è possibile concatenare comandi in modo che lo `stdout` di uno diventi lo `stdin` di un secondo tramite il simbolo `|` (barra verticale). Viene chiamata operazione di *pipe*.

es:

```
$ ls | sort -r
```

fa in modo che lo `stdout` di `ls` venga rielaborato dal comando `sort`. Il risultato di queso comando composito é di ottenere la lista dei file della directory corrente in ordine alfabetico decrescente (`-r` è l'opzione *reverse sort*).

. . . e lo standard error? viene usato dai programmi/comandi per segnalare errori. E' distinto dallo `stdout`. Se gli errori fossero segnalati su `stdout` quando questo è ridirezionato su di un file o una pipe questi non comparirebbero sul terminale e verrebbero salvati o rielaborati come se fossero dati.

i comandi più comuni

cd: cambia la directory corrente es: `cd /etc`. Se chiamato senza parametro `cd` torna alla home directory

ls: lista i file alcuni esempi sono stati visti sopra. Senza parametri *lista* i file della dir corrente.

```
$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2502 Aug 26 17:31 /etc/passwd
```

Il file /etc/passwd è lungo 2502 caratteri, è stato creato il 26 agosto alle 17:31, appartiene all'utente root e al gruppo root (root è il "superutente" del sistema che ha pieno accesso a tutte le informazioni). I primi caratteri indicano:

- `-`: che è un file (se fosse stata una directory ci sarebbe `d`, se device `c` o `b`. . . .
- `rw-` indica che il proprietario (root) può leggere e scrivere questo file
- `r--` gli altri componenti del gruppo root possono solo leggere
- `r--` anche tutti gli altri utenti possono leggere questo file.

Prendiamo un altro esempio:

```
$ ls -l /bin/bash
-rwxr-xr-x 1 root root 1265648 Apr 23 23:23 /bin/bash
```

Il file `/bin/bash` è sempre proprietà di root/gruppo root ma le x in `-rwxr-xr-x` indicano che è un file eseguibile. Sia l'utente root (prima x), sia gli altri elementi del grupo root (senconda x) sia tutti gli altri possono lanciare l'esecuzione del programma.

```
$ ls -d -l /home/so2324/
drwx------ 19 so2324 so2324 4096 Sep 13 17:24 /home/so2324/
```

l'opzione -d indica che anche se è una directory vogliamo le informazioni sulla dir stessa e non l'elenco dei file contenuti. Questa directory può essere letta e scritta solo dall'utente so2324. Il bit 'r' indica la possibilità di vederne il contenuto, il permesso 'x' nelle directory indica che si può *attraversare* cioè accedere file e directory (al buio se il bit 'r' manca).

whoami Quando un utente ha crisi di identità es:

```
$ whoami
so2324
```

groups Indica i gruppi ai quali appartiene l'utente

```
$ groups
so2324 sudo users kvm
```

file Indica il tipo del file e.g.:

```
$ file Documents/
Documents/: directory
$ file /bin/firefox
/bin/firefox: POSIX shell script, ASCII text executable
$ file /tmp/b.c
/tmp/b.c: C source, ASCII text
```

cp: copia file Ha due possibili sintassi: con due parametri il primo è il pathname del file sorgente e il secondo parametro il pathname della destinazione. Con due o più parametri, se l'ultimo è il pathname di una directory, tutti i file vengono copiati nella directory, mantenendo il proprio nome.

```
$ cp /etc/passwd /tmp/elenco_utenti
$ cp /etc/group /etc/hostname /tmp
```

mv: sposta file Ha la stessa sintassi di `cp` ma sposta i file invece che copiarli. `mv` serve anche per rinominare un file (spostandolo all'interno della stessa directory).

ln serve per creare *link* cioè per poter chiamare un file con più nomi. Ci sono due tipi di link: link *fisico* (detto anche *hard link*) e link *simbolico*.

Esempio per fare in modo che il file `prova` si chiami anche `test` si usa il comando:

```
$ ln prova test
```

Attenzione: non viene creata una copia, si ottiene in quasto modo un secondo nome che identifica lo stesso file.

```
$ echo ciao > prova
$ ln prova test
$ ls -l prova test
-rw-r--r-- 2 so2324 so2324 5 Sep 25 08:59 prova
-rw-r--r-- 2 so2324 so2324 5 Sep 25 08:59 test
```

Per creare un link simbolico si usa sempre il comando `ln` ma con il parametro `-s`:

```
$ ln -s prova symtest
$ ls -l prova symtest
-rw-r--r-- 2 so2324 so2324 5 Sep 25 08:59 prova
lrwxrwxrwx 1 so2324 so2324 5 Sep 25 09:01 symtest -> prova
```

in questo caso `symtest` è sono un rimando al file prova.

rm cancella un file, o meglio *il nome* di un file. Il file viene veeramente cancellato quando non esiste più alcun nome per riferirsi ad esso.

```
$ rm prova
```

Attenzione: `rm` non ammette ripensamenti. Non ci sono *cestini* o comandi miracolosi, quando un file è cancellato, è perso.

mkdir crea una directory

```
$ mkdir /tmp/prova
```

rmdir cancella una directory

```
$ rmdir /tmp/prova
```

la directory deve essere vuota.

La history di bash

Bash memorizza la storia dei comandi precedentemente digitati, è una funzionalità comoda per ripetere, correggere comandi già eseguiti o per eseguirne di simili.

Di seguito si riportano alcune direttive utili:

- digitando `!!` si ripette l'ultimo comando,
- con i tasti freccia in alto e in basso si *naviga* nei comandi precedenti con la possibilità di modificarli prima di inviarli,
- premendo *ctrl-r* si cerca nella storia i comandi precedenti che contengono la sequenza di caratteri inseriti dopo il *ctrl-r*.