

Operating Systems

- Scripting Linux -

Ing. Marco Morana
marco.morana@unipa.it

Materiale utile

- Scott Granneman
 - *Linux: Codice e comandi essenziali*
- man pages!!!

```
man(1)                                     man(1)

NAME
    man - format and display the on-line manual pages

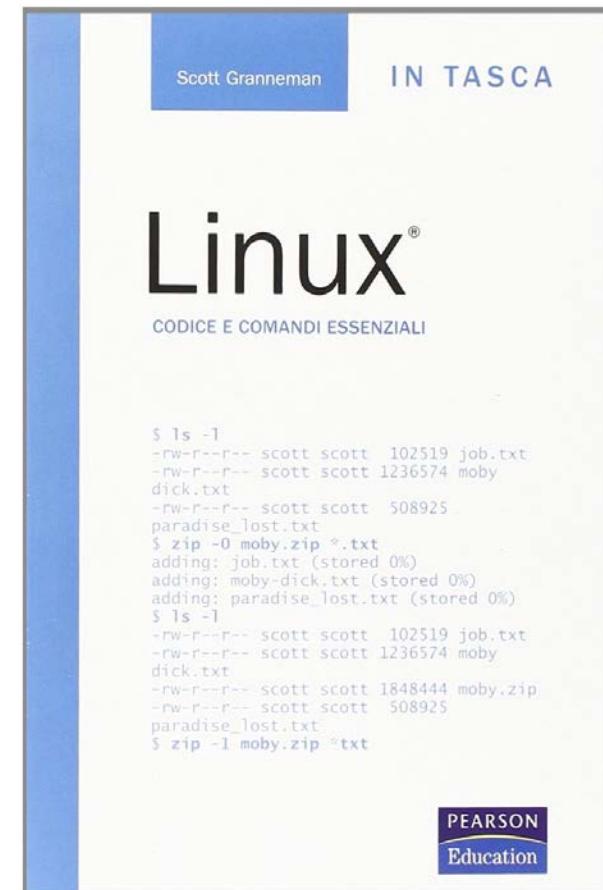
SYNOPSIS
    man [-acdfHkKtwW] [--path] [-m system] [-p string] [-C config_file] [-M pathlist]
        [-P pager] [-B browser] [-H htmlpager] [-S section_list] [section] name ...

DESCRIPTION
    man formats and displays the on-line manual pages. If you specify section, man
    only looks in that section of the manual. name is normally the name of the manual
    page, which is typically the name of a command, function, or file. However, if
    name contains a slash (/) then man interprets it as a file specification, so that
    you can do man ./foo.5 or even man /cd/foo/bar.1.gz.

    See below for a description of where man looks for the manual page files.

OPTIONS
    -C config_file
        Specify the configuration file to use; the default is /private/etc/man.conf.
        (See man.conf(5).)

    -M path
```



man

\$ man [1-9] *nomecomando*

man è il paginatore dei manuali del sistema

- Riceve in input il nome di un programma, di una utility o di una funzione
- Suddiviso in sezioni, ognuna delle quali contiene più di pagine differenti:
 1. Programmi eseguibili e comandi della shell
 2. Chiamate al sistema (funzioni fornite dal kernel)
 3. Chiamate alle librerie (funzioni all'interno delle librerie di sistema)
 4. File speciali (di solito trovabili in /dev)
 5. Formati dei file e convenzioni (es. /etc/passwd)
 6. Giochi
 7. Pacchetti di macro e convenzioni (es. man(7), groff(7))
 8. Comandi per l'amministrazione del sistema (solo per root)
 9. Routine del kernel [Non standard]

man - esempi

\$ man man

il manuale del manuale!

\$ man sleep

cerca in tutte le sezioni, restituisce la pagina della prima sezione trovata

\$ man 1 sleep

cerca nella sezione 1, “Programmi eseguibili e comandi della shell”

\$ man 3 sleep

cerca nella sezione 3, “Chiamate alle librerie (funzioni all’interno delle librerie di sistema)”

man - esempi

\$ man man

il manuale del manuale!

\$ man sleep

cerca in tutte le sezioni, restituisce la pagina della prima sezione trovata

\$ man 1 sleep

cerca nella sezione 1, “Programmi eseguibili e comandi della shell”

\$ man 3 sleep

cerca nella sezione 3, “Chiamate alle librerie (funzioni all’interno delle librerie di sistema)”

man - esempi

```
$ man man(1) man(1)
il man(1) man(1)
NAME man - format and display the on-line manual pages
SYNOPSIS
$ man [-acdfFhkKtwW] [--path] [-m system] [-p string] [-C config_file]
[-M pathlist] [-P pager] [-B browser] [-H htmlpager] [-S section_list]
[section] name ...
cerceri
DESCRIPTION
$ man formats and displays the on-line manual pages. If you specify sec-
cerceri
tion, man only looks in that section of the manual. name is normally
the name of the manual page, which is typically the name of a command,
function, or file. However, if name contains a slash (/) then man
interprets it as a file specification, so that you can do man ./foo.5
or even man /cd/foo/bar.1.gz.
cerceri
See below for a description of where man looks for the manual page
files.
cerceri
OPTIONS
di si : -C config_file
erie
```

man - esempi

\$ man man

il manuale del manuale!

\$ man sleep

cerca in tutte le sezioni, restituisce la pagina della prima sezione trovata

\$ man 1 sleep

cerca nella sezione 1, “Programmi eseguibili e comandi della shell”

\$ man 3 sleep

cerca nella sezione 3, “Chiamate alle librerie (funzioni all’interno delle librerie di sistema)”

man - esempi

```
$ man sleep
marcomorana — less - man 1 sleep — 80x24
$ man SLEEP(1)
BSD General Commands Manual          SLEEP(1)
il man
NAME
    sleep -- suspend execution for an interval of time
SYNOPSIS
    sleep seconds
cerca
DESCRIPTION
    The sleep command suspends execution for a minimum of seconds.
    If the sleep command receives a signal, it takes the standard action.
IMPLEMENTATION NOTES
cerca
    The SIGALRM signal is not handled specially by this implementation.
    The sleep command will accept and honor a non-integer number of specified
    seconds (with a '.' character as a decimal point). This is a non-portable
    extension, and its use will nearly guarantee that a shell script will
    not execute properly on another system.
cerca
EXIT STATUS
di si :|
```

man - esempi

\$ man man

il manuale del manuale!

\$ man sleep

cerca in tutte le sezioni, restituisce la pagina della prima sezione trovata

\$ man 1 sleep

cerca nella sezione 1, “Programmi eseguibili e comandi della shell”

\$ man 3 sleep

cerca nella sezione 3, “Chiamate alle librerie (funzioni all’interno delle librerie di sistema)”

man - esempi

```
$ man sleep
marcomorana — less - man 1 sleep — 80x24
SLEEP(1)                               BSD General Commands Manual      SLEEP(1)
il manuale di man(1)
NAME
    sleep -- suspend execution for an interval of time

SYNOPSIS
    sleep seconds

DESCRIPTION
    The sleep command suspends execution for a minimum of seconds.
    If the sleep command receives a signal, it takes the standard action.

IMPLEMENTATION NOTES
    The SIGALRM signal is not handled specially by this implementation.
    The sleep command will accept and honor a non-integer number of specified
    seconds (with a '.' character as a decimal point). This is a non-portable
    extension, and its use will nearly guarantee that a shell script will
    not execute properly on another system.

EXIT STATUS
    di serie:
```

man - esempi

\$ man man

il manuale del manuale!

\$ man sleep

cerca in tutte le sezioni, restituisce la pagina della prima sezione trovata

\$ man 1 sleep

cerca nella sezione 1, “Programmi eseguibili e comandi della shell”

\$ man 3 sleep

cerca nella sezione 3, “Chiamate alle librerie (funzioni all’interno delle librerie di sistema)”

man - esempi

```
$ man man
```

The screenshot shows a terminal window titled "marco@marco-VirtualBox: ~". The window title bar also displays "File Modifica Visualizza Cerca Terminale Aiuto" and the subtitle "Linux Programmer's Manual". The main content of the window is the man page for the sleep function:

```
SLEEP(3)                               Linux Programmer's Manual                               SLEEP(3)

NAME
    sleep - sleep for a specified number of seconds

SYNOPSIS
    #include <unistd.h>

    unsigned int sleep(unsigned int seconds);

DESCRIPTION
    sleep() causes the calling thread to sleep either until the number of
    real-time seconds specified in seconds have elapsed or until a signal
    arrives which is not ignored.

RETURN VALUE
    Zero if the requested time has elapsed, or the number of seconds left
    to sleep, if the call was interrupted by a signal handler.

Manual page sleep(3) line 1 (press h for help or q to quit)
```

man - esempi

\$ man 2 sleep

cerca nella sezione 2, nessun risultato trovato

\$ man -a sleep

cerca in sequenza in tutte le sezioni e non completa la successiva

\$ man 2 chmod

cerca nella sezione 2

\$ man 5 passwd

cerca nella sezione 5

\$ man 8 shutdown

cerca nella sezione 8

No manual entry for sleep in section 2
See 'man 7 undocumented' for help when
manual pages are not available.

man - esempi

\$ man 2 sleep

cerca nella sezione 2, nessun risultato trovato

\$ man -a sleep

cerca in sequenza in tutte le sezioni, q per completare una sezione, enter per la successiva

\$ man 2 chmod

cerca nella sezione 2, chiamate di sistema di basso livello fornite dal kernel

\$ man 5 passwd

cerca nella sezione 5, formati dei file e convenzioni es. /etc/passwd

\$ man 8 shutdown

cerca nella sezione 8, Comandi per l'amministrazione del sistema (solo root)

man - esempi

```
marco@marco-VirtualBox: ~/Scrivania/esercitazione1
File Modifica Visualizza Cerca Terminale Aiuto
SLEEP(1) User Commands SLEEP(1)

NAME
sleep - delay for a specified amount of time

SYNOPSIS
sleep NUMBER[SUFFIX]...
sleep OPTION

DESCRIPTION
Pause for NUMBER seconds. SUFFIX may be 's' for seconds (the default),
'm' for minutes, 'h' for hours or 'd' for days. Unlike most implemen-
tations that require NUMBER be an integer, here NUMBER may be an arbi-
trary floating point number. Given two or more arguments, pause for
the amount of time specified by the sum of their values.

--help display this help and exit

Manual page sleep(1) line 1 (press h for help or q to quit)
```

man - esempi

```
marco@marco-VirtualBox: ~/Scrivania/esercitazione1
File Modifica Visualizza Cerca Terminale Aiuto
SLEEP(3)                               Linux Programmer's Manual      SLEEP(3)
NAME
    sleep - sleep for a specified number of seconds

SYNOPSIS
    #include <unistd.h>

    unsigned int sleep(unsigned int seconds);

DESCRIPTION
    sleep() causes the calling thread to sleep either until the number of
    real-time seconds specified in seconds have elapsed or until a signal
    arrives which is not ignored.

RETURN VALUE
    Zero if the requested time has elapsed, or the number of seconds left
    to sleep, if the call was interrupted by a signal handler.
Manual page sleep(3) line 1 (press h for help or q to quit)
```

man - esempi

\$ man 2 sleep

cerca nella sezione 2, nessun risultato trovato

\$ man -a sleep

cerca in sequenza in tutte le sezioni, q per completare una sezione, enter per la successiva

\$ man 2 chmod

cerca nella sezione 2, chiamate di sistema di basso livello fornite dal kernel

\$ man 5 passwd

cerca nella sezione 5, formati dei file e convenzioni es. /etc/passwd

\$ man 8 shutdown

cerca nella sezione 8, Comandi per l'amministrazione del sistema (solo root)

man - esempi

```
marco@marco-VirtualBox: ~/Scrivania/esercitazione1
File Modifica Visualizza Cerca Terminale Aiuto
CHMOD(2)          Linux Programmer's Manual          CHMOD(2)

NAME
    chmod, fchmod, fchmodat - change permissions of a file

SYNOPSIS
#include <sys/stat.h>

int chmod(const char *pathname, mode_t mode);
int fchmod(int fd, mode_t mode);

#include <fcntl.h>           /* Definition of AT_* constants */
#include <sys/stat.h>

int fchmodat(int dirfd, const char *pathname, mode_t mode, int flags);

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):
Manual page chmod(2) line 1 (press h for help or q to quit)
```

man - esempi

\$ man 2 sleep

cerca nella sezione 2, nessun risultato trovato

\$ man -a sleep

cerca in sequenza in tutte le sezioni, q per completare una sezione, enter per la successiva

\$ man 2 chmod

cerca nella sezione 2, chiamate di sistema di basso livello fornite dal kernel

\$ man 5 passwd

cerca nella sezione 5, formati dei file e convenzioni es. /etc/passwd

\$ man 8 shutdown

cerca nella sezione 8, Comandi per l'amministrazione del sistema (solo root)

man - esempi

```
marco@marco-VirtualBox: ~/Scrivania/esercitazione1
File Modifica Visualizza Cerca Terminale Aiuto
PASSWD(5)           Formati di file e conversioni          PASSWD(5)

NOME
passwd - il file delle password

DESCRIZIONE
/etc/passwd contiene una riga per ogni account, con sette campi
delimitati da due punti («::»). Questi campi sono:

    • nome di login
    • password cifrata opzionale
    • ID utente numerico
    • ID gruppo numerico
    • nome utente o commento

Manual page passwd(5) line 1 (press h for help or q to quit)
```

man - esempi

\$ man 2 sleep

cerca nella sezione 2, nessun risultato trovato

\$ man -a sleep

cerca in sequenza in tutte le sezioni, q per completare una sezione, enter per la successiva

\$ man 2 chmod

cerca nella sezione 2, chiamate di sistema di basso livello fornite dal kernel

\$ man 5 passwd

cerca nella sezione 5, formati dei file e convenzioni es. /etc/passwd

\$ man 8 shutdown

cerca nella sezione 8, Comandi per l'amministrazione del sistema (solo root)

man - esempi

```
marco@marco-VirtualBox: ~/Scrivania/esercitazione1
File Modifica Visualizza Cerca Terminale Aiuto
SHUTDOWN(8)           shutdown           SHUTDOWN(8)

NAME
  shutdown - Halt, power-off or reboot the machine

SYNOPSIS
  shutdown [OPTIONS...] [TIME] [WALL...]

DESCRIPTION
  shutdown may be used to halt, power-off or reboot the machine.

  The first argument may be a time string (which is usually "now").
  Optionally, this may be followed by a wall message to be sent to all
  logged-in users before going down.

  The time string may either be in the format "hh:mm" for hour/minutes
  specifying the time to execute the shutdown at, specified in 24h clock
  format. Alternatively it may be in the syntax "+m" referring to the
  Manual page shutdown(8) line 1 (press h for help or q to quit)
```

Metacaratteri della shell

- `#`: considera ciò che segue come un commento

```
$ #prova di commento
```

- `var=val` assegna il valore `val` alla variabile `var`

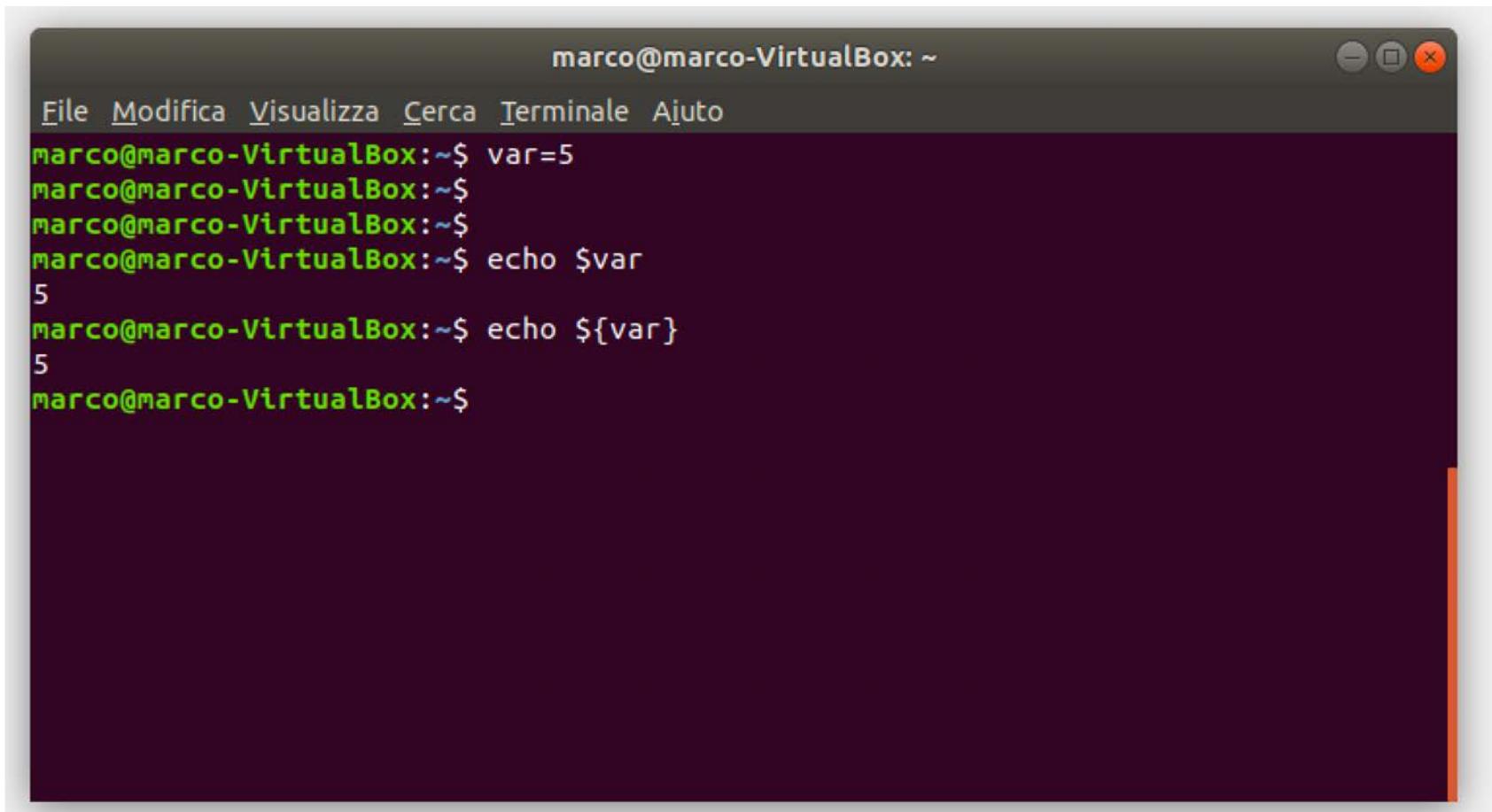
```
$ var=5
```

- `$var` restituisce il valore della variabile di shell `var`

```
$ echo $var
```

- `${var}` come sopra, ma utile per migliorare la leggibilità

Metacaratteri della shell



A screenshot of a terminal window titled "marco@marco-VirtualBox: ~". The window has a dark purple background and a light gray header bar. The terminal shows the following command-line session:

```
File Modifica Visualizza Cerca Terminale Aiuto
marco@marco-VirtualBox:~$ var=5
marco@marco-VirtualBox:~$ echo $var
5
marco@marco-VirtualBox:~$ echo ${var}
5
marco@marco-VirtualBox:~$
```

Metacaratteri della shell

- ` ... ` (il backquote si ottiene con alt+') esegue eventuali comandi contenuti tra gli apici inversi e li sostituisce con i relativi output

```
$ echo `date "+%Y-%m-%d"`          (Y=anno4cifre, m=[01-12], d=[01-31])  
$ echo `date "+%D"`                (same as %m/%d/%y)
```

- ' ... ' la stringa contenuta tra gli apici non viene interpretata dalla shell

```
echo 'date "+%D"'
```

- " ..." solo alcuni metacaratteri vengono interpretati (in particolare: \$, \ e gli apici inversi)

```
echo 'date $var'                  (non viene interpretato)  
echo "date $var"                (viene interpretato soltanto il $)
```

Metacaratteri della shell

```
marco@marco-VirtualBox: ~
File Modifica Visualizza Cerca Terminale Aiuto
marco@marco-VirtualBox:~$ echo `date "+%Y-%m-%d"`
2020-04-06
marco@marco-VirtualBox:~$ echo `date "+%D"`
04/06/20
marco@marco-VirtualBox:~$ echo 'date "+%D"'
date "+%D"
marco@marco-VirtualBox:~$ echo 'date $var'
date $var
marco@marco-VirtualBox:~$ echo "date $var"
date 5
marco@marco-VirtualBox:~$
```

...' la stringa contenuta tra gli apici non viene interpretata dalla shell

"..." solo alcuni metacaratteri vengono interpretati, in particolare \$, \ e gli apici inversi

Operatori

;

Eseguire più comandi in sequenza:

```
$ sleep 4 ; echo "Hello"
```

&&

Eseguire i comandi solo se i precedenti hanno successo:

```
$ mkdir a && cd a
```

||

Eseguire un comando solo se il precedente ha fallito:

```
$ (mv a.txt b.txt && echo 'fatto') || echo 'file inesistente'
```

Redirect

- Di default i comandi Linux prendono l'input da tastiera (**standard input**) e mandano l'output ed eventuali messaggi di errore su video (**standard output, standard error**)

REDIRECT DI STANDARD OUTPUT

- `command > filename`
memorizza l'output di command nel file filename

```
$ echo 'ciao' > a.txt  
$ cat a.txt
```

- `command >> filename`
aggiunge l'output di command in coda al file filename (append)

```
$ ls -l >> a.txt  
$ cat a.txt
```

Redirect

- Di default i comandi Linux prendono l'input da tastiera (**standard input**) e mandano l'output ed eventuali messaggi di errore su video (**standard output**, **standard error**)

REDIRECT SELETTIVO DI STANDARD OUTPUT / STANDARD ERROR

Identificativo	Nome	Default
0	Standard input	Tastiera
1	Standard output	Terminale
2	Standard error	Terminale

Redirect

```
#include <stdio.h>

int main() {
    printf("send to stdout\n");
    fprintf(stderr,"send to stderr\n");
    return 0;
}
```

→ \$ gcc -o test test.c
\$./test > out.txt 2> err.txt (redirect di stdout e stderr su 2 file diversi)

stdout stderr

Redirect

```
#include <stdio.h>

int main() {
    printf("send to stdout\n");
    fprintf(stderr,"send to stderr\n");
    return 0;
}
```

→ \$ gcc -o test test.c
\$./test > out.txt 2> err.txt (redirect di stdout e stderr su 2 file diversi)
\$ cat out.txt
send to stdout
\$ cat err.txt
send to stderr

Redirect

```
#include <stdio.h>

int main() {
    printf("send to stdout\n");
    fprintf(stderr,"send to stderr\n");
    return 0;
}
```

```
$ gcc -o test test.c
$ ./test > out.txt 2> err.txt      (redirect di stdout e stderr su 2 file diversi)
$ cat out.txt
send to stdout
$ cat err.txt
send to stderr

$ ./test &> both.txt                  (redirect di stdout e stderr su uno stesso file)
```

Redirect

```
#include <stdio.h>

int main() {
    printf("send to stdout\n");
    fprintf(stderr,"send to stderr\n");
    return 0;
}
```

```
$ gcc -o test test.c
$ ./test > out.txt 2> err.txt      (redirect di stdout e stderr su 2 file diversi)
$ cat out.txt
send to stdout
$ cat err.txt
send to stderr
```

→ \$./test &> both.txt (redirect di stdout e stderr su uno stesso file)

```
$ cat both.txt
send to stderr
send to stdout
```

Redirect

- Di default i comandi Linux prendono l'input da tastiera (**standard input**) e mandano l'output ed eventuali messaggi di errore su video (**standard output, standard error**)

REDIRECT DI STANDARD INPUT

- `command < filename`

l'input di command è letto dal file filename

```
$ ls > a.txt
```

```
$ wc -l < a.txt (stampa il numero di newline in a.txt interpretandolo come stdin)
```

```
$ wc -l a.txt (stampa il numero di newline in a.txt interpretandolo come file,  
quindi aggiunge anche il nome del file dopo il risultato [provare ad  
esempio wc -l a.txt b.txt per vedere più output seguiti dal filename])
```

Pipe

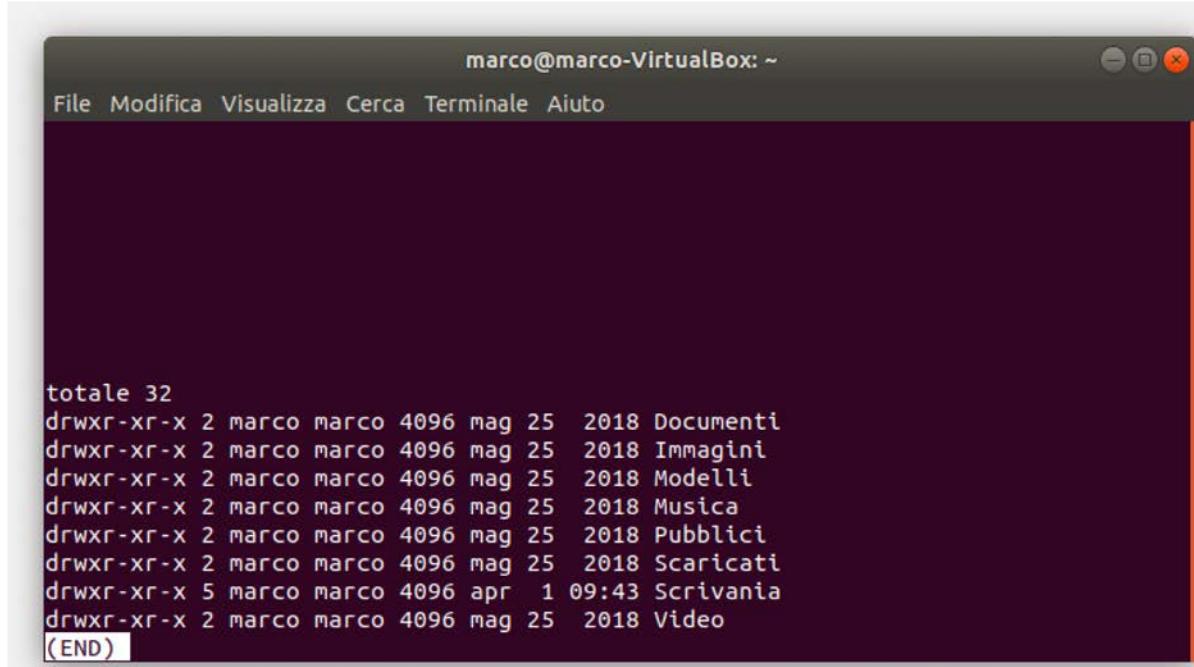
Operatore | connette l'output di un programma all'input di un altro

Il flusso di esecuzione è da sinistra verso destra

```
$ ls -l | less
```

equivale a:

```
$ ls -l > tmp  
$ less < tmp
```



The screenshot shows a terminal window titled "marco@marco-VirtualBox: ~". The window has a dark background and a light-colored title bar. The menu bar includes "File", "Modifica", "Visualizza", "Cerca", "Terminale", and "Aiuto". The main area of the terminal displays the output of the "ls -l" command, which lists the contents of the current directory. The output is as follows:

```
total 32
drwxr-xr-x 2 marco marco 4096 mag 25 2018 Documenti
drwxr-xr-x 2 marco marco 4096 mag 25 2018 Immagini
drwxr-xr-x 2 marco marco 4096 mag 25 2018 Modelli
drwxr-xr-x 2 marco marco 4096 mag 25 2018 Musica
drwxr-xr-x 2 marco marco 4096 mag 25 2018 Pubblici
drwxr-xr-x 2 marco marco 4096 mag 25 2018 Scaricati
drwxr-xr-x 5 marco marco 4096 apr  1 09:43 Scrivania
drwxr-xr-x 2 marco marco 4096 mag 25 2018 Video
(END)
```

Pipe - esempi

```
$ ls > elenco.txt          (file contenente i nomi dei file nella dir corrente)
$ ls >> elenco.txt        (duplico i nomi appendendo in coda)
$ sort elenco.txt          (ordine alfabetico)
$ sort elenco.txt | uniq   (ordino e rimuovo le ripetizioni)

$ cat elenco.txt | head -2 (mostro le prime due righe)
$ cat elenco.txt | head -2 | tail -1 (ultima riga delle prime due)
```

Pipe - esempi

\$ >> f1	(creo 1 file)
\$ cp f1 f2	(lo duplico)
\$ tar cf A.tar f1 f2	(creo un archivio, non compresso)
\$ tar tvf A.tar	(verifico il contenuto)
\$ gzip A.tar	(comprimo)

oppure con un unico comando, reindirizzando i flussi SENZA creare file intermedi:

Pipe - esempi

- \$ >> f1 (creo 1 file)
- \$ cp f1 f2 (lo duplico)
- \$ tar cf A.tar f1 f2 (creo un archivio, non compresso)
- \$ tar tvf A.tar (verifico il contenuto)
- \$ gzip A.tar (comprimo)

oppure con un unico comando, reindirizzando i flussi SENZA creare file intermedi:

```
$ >> f1
```

Pipe - esempi

- \$ >> f1 (creo 1 file)
- \$ cp f1 f2 (lo duplico)
- \$ tar cf A.tar f1 f2 (creo un archivio, non compresso)
- \$ tar tvf A.tar (verifico il contenuto)
- \$ gzip A.tar (comprimo)

oppure con un unico comando, reindirizzando i flussi SENZA creare file intermedi:

```
$ >> f1 && cp f1 f2
```

Pipe - esempi

- \$ >> f1 (creo 1 file)
- \$ cp f1 f2 (lo duplico)
- \$ tar cf A.tar f1 f2 (creo un archivio, non compresso)
- \$ tar tvf A.tar (verifico il contenuto)
- \$ gzip A.tar (comprimo)

oppure con un unico comando, reindirizzando i flussi SENZA creare file intermedi:

```
$ >> f1 && cp f1 f2 && tar cvf - f1 f2
```

Pipe - esempi

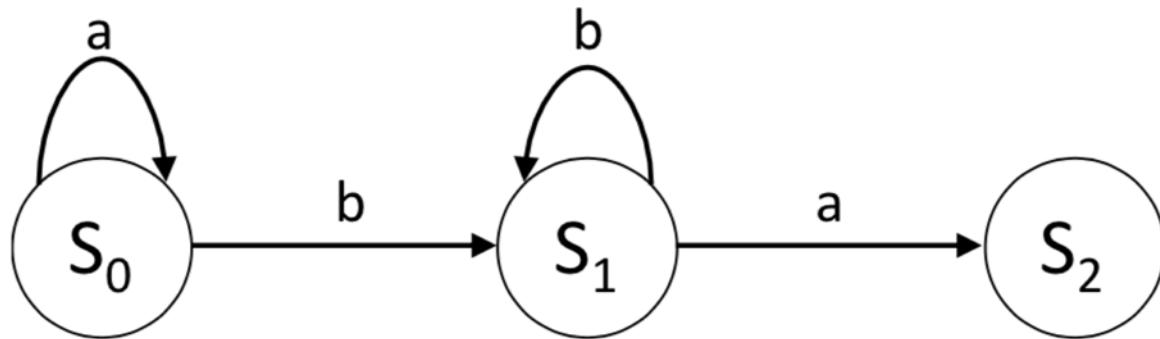
```
$ >> f1                                (creo 1 file)
$ cp f1 f2                               (lo duplico)
$ tar cf A.tar f1 f2                     (creo un archivio, non compresso)
$ tar tvf A.tar                            (verifico il contenuto)
● $ gzip A.tar                            (comprimo)

oppure con un unico comando, reindirizzando i flussi SENZA creare file intermedi:
$ >> f1 && cp f1 f2 && tar cvf - f1 f2 | gzip > A.tar.gz
```

grep: Global Regular Expression Print

Espressioni regolari

- Le espressioni regolari definiscono pattern
- Consentono di rappresentare sequenze di stringhe che possono essere descritte attraverso un “Automa a stati finiti”



- Definisce stringhe costituite da: zero o più ‘a’ seguite da una (prima freccia) o più ‘b’ (loop) e completate da una ‘a’
 - Esempio: “aba”, “aaba”, “ba”, “aaaaaabbbbbbbba” etc.

grep: Global Regular Expression Print

grep cerca dei pattern all'interno del file/stream specificato in input

```
==== heroes.txt ===
```

```
Catwoman
Batman
The Tick
Spider Man
Black Cat
Batgirl
Danger Girl
Wonder Woman
Luke Cage
The Punisher
Ant Man
Dead Girl
Aquaman
SCUD
Spider Woman
Blackbolt
Martian Manhunter
=====
```

grep: Global Regular Expression Print

grep cerca dei pattern all'interno del file/stream specificato in input

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -i man heroes.txt
```

Cerca riga per riga, occorrenze di **man** (una **m**, seguita da una **a**, seguita da una **n**) all'interno del file heroes.txt, ignorando (**-i**) la differenza tra maiuscole/minuscole

Restituisce tutte le righe in cui la ricerca ha successo

```
Catwoman  
Batman  
Spider Man  
Wonder Woman  
Ant Man  
Aquaman  
Spider Woman  
Martian Manhunter
```

grep: Global Regular Expression Print

grep cerca dei pattern all'interno del file/stream specificato in input

```
==== heroes.txt ====  
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -v -i man heroes.txt
```

L'opzione **-v** esclude le righe che matchano i criteri di ricerca

Stampa tutte le righe, eccetto quelle che contengono **man**

```
The Tick  
Black Cat  
Batgirl  
Danger Girl  
Luke Cage  
The Punisher  
Dead Girl  
SCUD  
Blackbolt
```

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -E '^Bat' heroes.txt
```

L'opzione -E specifica l'utilizzo di una regular expression

Il carattere ^ indica di fare il match all'inizio della riga

Usiamo gli apici per forzare la shell a non interpretare
ciò che è contenuto tra essi

```
Batman  
Batgirl
```

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -E 'er$' heroes.txt
```

Il carattere \$ indica di fare il match alla fine della riga

```
The Punisher  
Martian Manhunter
```

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -E 'er' heroes.txt
```

Senza il \$ avremmo trovato qualsiasi istanza di er

```
Spider Man  
Danger Girl  
Wonder Woman  
The Punisher  
Spider Woman  
Martian Manhunter
```

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -E '^$' heroes.txt
```

Combinando ^ e \$ potremmo ad esempio cercare una linea vuota (stringa che inizia e finisce con nessun carattere intermedio)

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -E -i '^^(bat|cat)' heroes.txt
```

Con l'operatore | possiamo cercare una istanza, oppure un'altra, all'interno della stessa regexp

Tutte le righe che iniziano con **bat** o con **cat**, sia maiuscole che minuscole

```
Catwoman  
Batman  
Batgirl
```

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -E '^ [BbCc]at' heroes.txt
```

Con le parentesi [] possiamo specificare un insieme di caratteri da matchare

```
$ grep -E '^ [A-Z]at' heroes.txt
```

[**A-Z**] *tutti i caratteri alfabetici maiuscoli*
[**a-z**] *tutti i caratteri alfabetici minuscoli*
[**A-Za-z**] oppure [**A-z**]
 tutti i caratteri alfabetici maiuscoli e minuscoli
[**A-zA-Z0-9_**]
 come sopra, più i numeri
[**A-MXYZ**]
 tutti i caratteri maiuscoli tra A ed M, e anche X, Y e Z

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ====
Catwoman
Batman
The Tick
Spider Man
Black Cat
Batgirl
Danger Girl
Wonder Woman
Luke Cage
The Punisher
Ant Man
Dead Girl
Aquaman
SCUD
Spider Woman
Blackbolt
Martian Manhunter
=====
=====
```

```
$ grep -E '^ [BbCc]at' heroes.txt
```

Con le parentesi [] possiamo specificare un insieme di caratteri da matchare

```
$ grep -E '^ [A-Z]at' heroes.txt
```

- [**A-Z**] tutti i caratteri alfabetici maiuscoli
- [**a-z**] tutti i caratteri alfabetici minuscoli
- [**A-Za-z**] oppure [**A-z**]
tutti i caratteri alfabetici maiuscoli e minuscoli
- [**A-zA-Z0-9_**]
come sopra, più i numeri
- [**A-MXYZ**]
tutti i caratteri maiuscoli tra A ed M, e anche X, Y e Z

grep: Global Regular Expression Print

grep: Matching per posizione

```
==== heroes.txt ===
```

```
Catwoman  
Batman  
The Tick  
Spider Man  
Black Cat  
Batgirl  
Danger Girl  
Wonder Woman  
Luke Cage  
The Punisher  
Ant Man  
Dead Girl  
Aquaman  
SCUD  
Spider Woman  
Blackbolt  
Martian Manhunter  
=====
```

```
$ grep -E -i '[^b]at' heroes.txt
```

Se all'interno delle parentesi utilizziamo ^ possiamo escludere i caratteri che seguono

'[^b]at' : Tutte le righe in cui compare **at**, non preceduto da **B** o **b** : esclude **Bat** o **bat**

```
Catwoman  
Black Cat
```

grep: Global Regular Expression Print

grep: Caratteri ripetuti

Supponiamo di volere individuare username di al più 8 caratteri, che iniziano con una lettera e contengono lettere o numeri.

Una possibile espressione regolare potrebbe essere:

[a-zA-Z][a-zA-Z0-9]{7}

Ma è complessa da scrivere, e non individuerebbe username con un numero di caratteri minore di 8

grep: Global Regular Expression Print

grep: Caratteri ripetuti

Attraverso le parentesi { } possiamo specificare il numero di volte in cui vogliamo che un carattere possa ripetersi

Ad esempio: ^ [A-z] [A-z0-9] {2,7} \$

grep: Global Regular Expression Print

grep: Caratteri ripetuti

Attraverso le parentesi { } possiamo specificare il numero di volte in cui vogliamo che un carattere possa ripetersi

Ad esempio: ^ [A-z] [A-z0-9] {2,7} \$

Fa il matching di tutte le stringhe che:

- **iniziano con una lettera**
- seguita tra 2 e 7 volte da una lettera o un numero
- prima della fine della riga

grep: Global Regular Expression Print

grep: Caratteri ripetuti

Attraverso le parentesi { } possiamo specificare il numero di volte in cui vogliamo che un carattere possa ripetersi

Ad esempio: ^ [A-z] [A-z0-9] {2,7} \$

Fa il matching di tutte le stringhe che:

- iniziano con una lettera
- **seguita tra 2 e 7 volte da una lettera o un numero**
- prima della fine della riga

grep: Global Regular Expression Print

grep: Caratteri ripetuti

Attraverso le parentesi { } possiamo specificare il numero di volte in cui vogliamo che un carattere possa ripetersi

Ad esempio: ^ [A-z] [A-z0-9] {2,7} \$

Fa il matching di tutte le stringhe che:

- iniziano con una lettera
- seguita tra 2 e 7 volte da una lettera o un numero
- **prima della fine della riga**

grep: Global Regular Expression Print

grep: Caratteri ripetuti

In generale, `X{n,m}` fa il matching di almeno `n` e non più di `m` ripetizioni del carattere `X`

Omettendo `m`, l'espressione fa il matching di almeno `n` ripetizioni di `X`

`{2}` esattamente due occorrenze del carattere precedente
`^G[o]{2}gle` descrive Google

`{2,}` almeno due occorrenze del carattere precedente
`^G[o]{2,}gle` descrive Google, Gooogle, Goooole ...

`{0,1}` zero o una occorrenza del carattere precedente, si abbrevia con ?
`Goo?gle` identifica sia Gogle che Google

`{1,}` almeno una occorrenza del carattere precedente, si abbrevia con +
`Goo+gle` identifica sia Google che Gooole che Goooole...

`{0,}` zero o più occorrenze del carattere precedente, si abbrevia con *
`Goo*gle` identifica sia Gogle che Google che Google...

grep: Global Regular Expression Print

Altri operatori di uso comune

.	<i>Match any single character.</i>
^	<i>Match the empty string that occurs at the beginning of a line or string.</i>
\$	<i>Match the empty string that occurs at the end of a line.</i>
A	<i>Match an uppercase letter A.</i>
a	<i>Match a lowercase a.</i>
\d	<i>Match any single digit.</i>
\D	<i>Match any single non-digit character.</i>
\w	<i>Match any single alphanumeric character; a synonym is [:alnum:].</i>
[A-E]	<i>Match any of uppercase A, B, C, D, or E.</i>
[^A-E]	<i>Match any character except uppercase A, B, C, D, or E.</i>
X?	<i>Match no or one occurrence of the capital letter X.</i>
X*	<i>Match zero or more capital Xs.</i>
X+	<i>Match one or more capital Xs.</i>
X{ n }	<i>Match exactly n capital Xs.</i>
X{ n , m }	<i>Match at least n and no more than m capital Xs. If you omit m, the expression tries to match at least n Xs.</i>
(abc def)+	<i>Match a sequence of at least one abc and def; abc and def would match.</i>

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep 'purchase' demo.txt
```

Tutti i file contenuti all'interno di demo.txt il cui nome inizia con purchase

```
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
```

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep 'purchase.' demo.txt
```

Tutti i file contenuti all'interno di demo.txt il cui nome inizia con purchase seguito da un altro qualsiasi carattere (.)

```
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
```

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep 'purchase.db' demo.txt
```

Tutti i file contenuti all'interno di demo.txt il cui nome inizia con purchase seguito da un altro qualsiasi carattere(.) e poi da db

```
purchase.db
```

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep 'purchase..db' demo.txt
```

Tutti i file contenuti all'interno di demo.txt il cui nome inizia con purchase seguito da due qualsiasi altri caratteri(..) e poi da db

```
purchase1.db
purchase2.db
purchase3.db
```

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep 'purchase.\.' demo.txt
```

Tutti i file contenuti all'interno di demo.txt il cui nome inizia con purchase seguito da un qualsiasi altro carattere e poi da un punto (il carattere \ funge da escape)

```
purchase1.db
purchase2.db
purchase3.db
```

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep -i -E '^purchase[0-9]?\.+' demo.txt
```

Tutti i file contenuti all'interno di demo.txt il cui nome inizia con purchase seguito da zero o più occorrenze di un numero, e poi da un punto

```
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
```

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep -i -E '^purchase[0-9]?\.+' demo.txt
```

Tutti gli esempi precedenti possono essere riscritti utilizzando la pipe

grep: Global Regular Expression Print

```
= demo.txt =
foo.txt
bar.txt
foo1.txt
bar1.doc
foobar.txt
foo.doc
bar.doc
dataset.txt
purchase.db
purchase1.db
purchase2.db
purchase3.db
purchase.idx
foo2.txt
=====
```

```
$ grep -i -E '^purchase[0-9]?\.+' demo.txt
```

Tutti gli esempi precedenti possono essere riscritti utilizzando la pipe

```
$ cat demo.txt | grep -i -E '^purchase[0-9]?\.+'
```

grep: Global Regular Expression Print

Tutte le opzioni

- **-i**: ignore the case of your search term
- **-v**: show lines that *don't* match, instead of those that do
- **-c**: instead of returning matches, return the *number* of matches
- **-x**: return only an exact match
- **-E**: interpret search as an extended regular expression
- **-F**: interpret search as a list of fixed strings, including newlines, dots, etc
- **-f**: get the search patterns from this file
- **-H**: print the filename with each match
- **-m**: stop reading file after *n* number of matches
- **-n**: print the line number of where matches were found
- **-q**: don't output anything, but exit with status 0 if any match is found (check that status with `echo $?.`)
- **-A**: print *n* number of lines after the match
- **-B**: print *n* number of lines before the match
- **-o**: print only the matching part of the line
- **-e**: search literally, and protects patterns starting with a hyphen
- **-w**: find matches surrounded by space
- **--color**: add color to the matched output
- **--help**: get some help
- **-v**: get grep's version

grep: Global Regular Expression Print

```
Stewart
Christina
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

```
$ cat file.txt | grep -i jill
```

Cerca jill all'interno di file.txt ignorando maiuscole/minuscole

```
Jill
```

grep: Global Regular Expression Print

```
Stewart
Christina
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

```
$ cat file.txt | grep -i -E '^.*?jill'
```

Cerca jill all'inizio di una riga di file.txt ignorando maiuscole/minuscole, ma specifica che prima di jill possono essere presenti 0,1 caratteri

```
Jill
```

grep: Global Regular Expression Print

```
Stewart
Christina
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

```
$ cat file.txt | grep -v Christina
```

Restituisce tutte le righe che non matchano
l'espressione specificata

```
Stewart
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

grep: Global Regular Expression Print

```
Stewart
Christina
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

```
$ cat file.txt | grep -E '[:digit:]'
```

Restituisce tutte le righe che contengono un numero

```
- 441
54r4h
Shazbot123
221
Item 1, Item 2, Item 3
23.44.124.67
172.16.23.1
```

grep: Global Regular Expression Print

```
Stewart
Christina
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

```
$ cat file.txt | grep -E '^[[[:digit:]]]'
```

Equivale a `cat file.txt | grep -E '^[0-9]'`
Restituisce tutte le righe che iniziano con un numero:

```
54r4h
221
23.44.124.67
172.16.23.1
```

grep: Global Regular Expression Print

```
Stewart
Christina
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

```
$ cat file.txt | grep -E '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
```

Restituisce tutte le righe che contengono un indirizzo IP

```
23.44.124.67
172.16.23.1
```

grep: Global Regular Expression Print

```
Stewart
Christina
- 441
<a href="https://www.google.com">Google</a>
Jill
54r4h
Shazbot123
111
221
Item 1, Item 2, Item 3
TABS    TABS    TABS
23.44.124.67
172.16.23.1
```

```
$ cat file.txt | grep -n -E '[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}'
```

Restituisce tutte le righe che contengono un indirizzo IP
e specifica il numero di riga

```
12:23.44.124.67
13:172.16.23.1
```

AWK - pattern-directed scanning and processing language

`grep` individua dei pattern all'interno di un file/stream

AWK consente di individuare pattern all'interno di file/stream e modificare i contenuti di file/stream

- Consente di definire delle **action** da effettuare quando viene individuato un pattern
- Come `grep` opera per righe di input
- Ogni linea di input viene processata come se fosse formata da un insieme di campi separati da spazio, o da un field separator FS che può essere specificato con il parametro -F

AWK - pattern-directed scanning and processing language

```
CREDITS, EXPDATE, USER, GROUPS  
99,01 jun 2018,sylvain,team:::admin  
52,01 dec 2018,sonia,team  
52,01 dec 2018,sonia,team  
25,01 jan 2019,sonia,team  
10,01 jan 2019,sylvain,team:::admin  
8,12 jun 2018,öle,team:support
```

La coppia '1 {print}'
è uno statement del tipo
'pattern {action}'

```
$ awk '1 {print}' file2.txt
```

1 = true; l'action print è anche l'action di default
Il comando equivale quindi a: awk 1 file2.txt

Se per una certa riga il pattern
restituisce TRUE, allora viene
eseguita l'action

AWK - pattern-directed scanning and processing language

```
CREDITS,EXPDATE,USER,GROUPS  
99,01 jun 2018,sylvain,team:::admin  
52,01 dec 2018,sonia,team  
52,01 dec 2018,sonia,team  
25,01 jan 2019,sonia,team  
10,01 jan 2019,sylvain,team:::admin  
8,12 jun 2018,öle,team:support
```

stampa tutte le righe

```
$ awk '1 {print}' file2.txt
```

1 = true; l'action print è anche l'action di default
Il comando equivale quindi a: awk 1 file2.txt

```
CREDITS,EXPDATE,USER,GROUPS  
99,01 jun 2018,sylvain,team:::admin  
52,01 dec 2018,sonia,team  
52,01 dec 2018,sonia,team  
25,01 jan 2019,sonia,team  
10,01 jan 2019,sylvain,team:::admin  
8,12 jun 2018,öle,team:support
```

AWK - pattern-directed scanning and processing language

\$1

CREDITS,EXPDATE,USER,GROUPS

```
99,01 jun 2018,sylvain,team:::admin
52,01 dec 2018,sonia,team
52,01 dec 2018,sonia,team
25,01 jan 2019,sonia,team
10,01 jan 2019,sylvain,team:::admin
8,12 jun 2018,öle,team:support
```

Per ciascuna riga, ciascun campo viene identificato con \$1, \$2, ... \$NF

L'intera riga corrisponde al campo \$0

```
awk '{print $1}' file2.txt
```

Stampa il primo campo di tutte le righe

```
CREDITS,EXPDATE,USER,GROUPS
99,01
52,01
52,01
25,01
10,01
8,12
```

AWK - pattern-directed scanning and processing language

\$1

```
CREDITS EXPDATE,USER,GROUPS  
99,01 jun 2018,sylvain,team:::admin  
52,01 dec 2018,sonia,team  
52,01 dec 2018,sonia,team  
25,01 jan 2019,sonia,team  
10,01 jan 2019,sylvain,team:::admin  
8,12 jun 2018,öle,team:support
```

```
awk -F , '{print $1}' file2.txt
```

Stampa per tutte le righe il primo campo delimitato dal nuovo carattere separatore ,

```
CREDITS  
99  
52  
52  
25  
10  
8
```

AWK - pattern-directed scanning and processing language

\$1 \$3

CREDITS	EXPDATE	USER	GROUPS
99	01 jun 2018	sylvain	team:::admin
52	01 dec 2018	sonia	team
52	01 dec 2018	sonia	team
25	01 jan 2019	sonia	team
10	01 jan 2019	sylvain	team:::admin
8	12 jun 2018	öle	team:support

```
awk -F , '{print $1,$3}' file2.txt
```

Stampa per tutte le righe il primo ed il terzo campo
delimitato dal carattere separatore ,

```
CREDITS USER
99 sylvain
52 sonia
52 sonia
25 sonia
10 sylvain
8 öle
```

AWK - pattern-directed scanning and processing language

\$NF

CREDITS, EXPDATE, USER, GROUPS

```
99,01 jun 2018,sylvain,team:::admin
52,01 dec 2018,sonia,team
52,01 dec 2018,sonia,team
25,01 jan 2019,sonia,team
10,01 jan 2019,sylvain,team:::admin
8,12 jun 2018,öle,team:support
```

```
awk -F , '{print $NF}' file2.txt
```

Stampa per tutte le righe l'ultimo campo delimitato dal carattere separatore ,

```
GROUPS
team:::admin
team
team
team
team:::admin
team:support
```

AWK - pattern-directed scanning and processing language

AWK può essere utilizzato con una pipe per filtrare l'output di altri comandi

```
$ ls -la
```

```
total 40
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

```
$ ls -la | awk '{print $1}'
```

```
total
drwxr-xr-x
drwxr-xr-x
-rw-r--r--@
-rw-r--r--@
-rw-r--r--@
-rw-r--r--@
-rw-r--r--@
```

AWK - pattern-directed scanning and processing language

```
total 40
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

Se il pattern è individuato da una espressione regolare, si utilizza la sintassi
`awk '/regex/' filename`

AWK - pattern-directed scanning and processing language

```
total 40
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

Se il pattern è individuato da una espressione regolare, si utilizza la sintassi
`awk '/regex/' filename`

```
$ ls -la | awk '/^d/'
```

Stampa tutti i campi (action di default) delle righe che iniziano con una d (le directory)

```
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
```

AWK - pattern-directed scanning and processing language

```
total 40
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

```
$ ls -la | awk '/^[-d]/'
```

Stampa tutti i campi (action di default) delle righe che iniziano con – (file) o d (directory)

```
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

AWK - pattern-directed scanning and processing language

```
total 40
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

```
$ ls -la | awk '/^[-]/ {print $9}'
```

Stampa il campo numero 9 (il filename) delle righe che iniziano con – (file)

```
demo.txt
example.txt
file.txt
file2.txt
heroes.txt
```

AWK - pattern-directed scanning and processing language

```
total 40
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

Per indicare il pattern deve essere soddisfatto su un campo specifico si usa la tilde

```
$ ls -la | awk '$7 ~ /^A/ {print $9}'
```

Stampa il campo numero 9 (il filename) di tutte le righe il cui campo numero 7 (mese di modifica) inizia per A

```
.
```

```
..
```

```
demo.txt
```

```
example.txt
```

```
file.txt
```

```
file2.txt
```

```
heroes.txt
```

AWK - pattern-directed scanning and processing language

```
total 40
drwxr-xr-x  7 user  staff  224 28 Apr  2019 .
drwxr-xr-x 10 user  staff  320 30 Apr  2019 ..
-rw-r--r--@  1 user  staff   153 26 Apr  2019 demo.txt
-rw-r--r--@  1 user  staff    77  6 Apr  2019 example.txt
-rw-r--r--@  1 user  staff   180 26 Apr  2019 file.txt
-rw-r--r--@  1 user  staff   209 26 Apr  2019 file2.txt
-rw-r--r--@  1 user  staff   173  6 Apr  2019 heroes.txt
```

Possiamo applicare più espressioni regolari su diversi campi contemporaneamente

```
$ ls -la | awk '$7 ~ /^A/ && $1 ~ /^-/ {print $NF}'
```

Come l'esempio precedente, ma usiamo \$NF invece di \$9 e stampiamo soltanto i file (non . e ..)

```
demo.txt
example.txt
file.txt
file2.txt
heroes.txt
```