Sommario

introduzione

Materiale di approfondimento reperibile sù:

http://linuxcommand.org/superman_pages.php

appena effettuato l'accesso ci troviamo di fronte ad una schermata simile a questa:

```
[patrizio.bellan@dominio Patrizio]$
```

Questa è la nostra shell, il nostro ambiente di lavoro. Da qui possiamo effettuare qualsiasi azione all'interno del sistema operativo.

Per prima cosa andiamo ad identificare il sistema operativo su cui stiamo lavorando, per fare questo utilizzeremo il comando ***uname***

IL COMANDO UNAME

```
[patrizio.bellan@dominio Patrizio]$ uname
Linux
```

Come si può vedere, nel nostro caso, il sistema operativo utilizzato è Linux.

Per verificare esattamente quale sistema operativo stiamo utilizzando, postponiamo l'opzione parametrale –o.

```
[patrizio.bellan@dominio Patrizio]$ uname -o
GNU/Linux
```

Ora che sappiamo quale sistema operativo stiamo utilizzando (che nel nostro caso è GNU/Linux), andiamo a verificarne quale versione stiamo utilizzando, per fare ciò andiamo ad aggiungere l'opzione –r.

```
[patrizio.bellan@dominio Patrizio]$ uname -r
2.6.32-279.14.1.el6.x86_64
```

Nel nostro caso la versione del kernel utilizzata è la ***2.6.32-279.14.1.el6.x86_64***

Ora andiamo a esplorare con quale versione hardware dell'elaboratore stiamo lavorando, per scoprire ciò utilizziamo l'opzione –m.

```
[patrizio.bellan@dominio Patrizio]$ uname -m
x86_64
```

che nel nostro caso risulta esser un elaboratore basato su tecnologia a 64bit.

Si riporta, a scopo di completezza, il riepilogo delle opzioni disponibili per il comando ***uname***

La sintassi del comando

uname –[opzioni]

Riepilogo opzioni del comando uname

```
NAME
       uname - print system information
SYNOPSIS
       uname [OPTION]...
DESCRIPTION
       Print certain system information.  With no OPTION, same as -s.
       -a, --all
              print all information, in the following order, except omit -p and -i if unknown:
       -s, --kernel-name
              print the kernel name
       -n, --nodename
              print the network node hostname
       -r, --kernel-release
              print the kernel release
       -v, --kernel-version
              print the kernel version
       -m, --machine
              print the machine hardware name
       -p, --processor
              print the processor type or "unknown"
       -i, --hardware-platform
              print the hardware platform or "unknown"
       -o, --operating-system
              print the operating system
       --help display this help and exit
```

Ora possiamo iniziare ad esplorare tutto il nostro ambiente.

Iniziamo vedendo dove ci troviamo e cosa c'è intorno a noi. Andiamo quindi a scoprire quali file e cartelle sono presenti nella nostra directory

Per assolvere questo compito utilizziamo il comando ***ls***

IL COMANDO LS

```
[patrizio.bellan@dominio Patrizio]$ ls
alice                    PaisaSentsExtractor  testi   venv27sistem_site_packages  virtualenv-15.0.1.tar.gz
paisa.annotated.CoNLL.utf8  testerTokenizers     venv27  virtualenv-15.0.1
```

Come possiamo vedere, una volta digitato il comando ci appare una lista di file e cartelle. Probabilmente sul vostro terminale le cartelle e i file avranno i colori dei fonts differenti per aiutarvi subito a identificare di che tipologia si tratta.

Nel nostro studio dei comandi non possiamo ignorare i files nascosti (tutti i files nascosti hanno il nome del file che inizia con il carattere "**.**"; utilizzeremo l'opzione ***–a***

```
[patrizio.bellan@dominio Patrizio]$ ls -a
.  alice     paisa.annotated.CoNLL.utf8  testerTokenizers  venv27                  virtualenv-15.0.1
.. commandLs  PaisaSentsExtractor        testi             venv27sistem_site_packages  virtualenv-15.0.1.tar.gz
```

Come sappiamo ogni file possiede degli attributi che indicano quale siano i permessi di accesso e di modifica, per scoprire questi attributi utilizziamo l'opzione parametrale ***–l***

```
[patrizio.bellan@dominio Patrizio]$ ls -l
total 9029140
-rw-r--r-- 1 patrizio.bellan domain users     175251 Apr  1 16:38 alice
-rw-r--r-- 1 patrizio.bellan domain users      10088 Apr  2 16:21 commandLs
-rwxrwxrwx 1 luca.ducceschi  domain users 9243746112 Dec  2  2011 paisa.annotated.CoNLL.utf8
drwxr-xr-x 3 patrizio.bellan domain users       4096 Mar 30 15:35 PaisaSentsExtractor
drwxr-xr-x 9 patrizio.bellan domain users      36864 Mar 30 15:52 testerTokenizers
drwxr-xr-x 2 patrizio.bellan domain users       4096 Apr  1 16:31 testi
drwxr-xr-x 6 luca.ducceschi  domain users       4096 Mar 30 17:56 venv27
drwxr-xr-x 5 luca.ducceschi  domain users       4096 Mar 30 17:50 venv27sistem_site_packages
drwxr-xr-x 9 luca.ducceschi  domain users       4096 Mar 30 17:50 virtualenv-15.0.1
-rw-r--r-- 1 luca.ducceschi  domain users    1842776 Mar 17 16:18 virtualenv-15.0.1.tar.gz
```

Una volta eseguito questo comando, è possibile reperire le informazioni più svariate riguardo ad ogni singolo file o cartella; dai permessi di accesso e modifica (la prima colonna), alla data di creazione (penultima colonna).

Possiamo ottenere anche l'elenco in ordine invertito tramite l'opzione ***-r***

```
[patrizio.bellan@dominio Patrizio]$ ls
alice     paisa.annotated.CoNLL.utf8  testerTokenizers  venv27                  virtualenv-15.0.1
commandLs  PaisaSentsExtractor        testi             venv27sistem_site_packages  virtualenv-15.0.1.tar.gz
```

una delle opportunità più comode è la possibilità di concatenare i parametri; ad esempio per per ottenere un elenco con tutte le informazioni in ordine inverso utilizzeremo l'opzione ***–lr***

```
[patrizio.bellan@dominio Patrizio]$ ls –lr
total 9029140
-rw-r--r-- 1 luca.ducceschi  domain users    1842776 Mar 17 16:18 virtualenv-15.0.1.tar.gz
drwxr-xr-x 9 luca.ducceschi  domain users       4096 Mar 30 17:50 virtualenv-15.0.1
drwxr-xr-x 5 luca.ducceschi  domain users       4096 Mar 30 17:50 venv27sistem_site_packages
drwxr-xr-x 6 luca.ducceschi  domain users       4096 Mar 30 17:56 venv27
drwxr-xr-x 2 patrizio.bellan domain users       4096 Apr  1 16:31 testi
drwxr-xr-x 9 patrizio.bellan domain users      36864 Mar 30 15:52 testerTokenizers
drwxr-xr-x 3 patrizio.bellan domain users       4096 Mar 30 15:35 PaisaSentsExtractor
-rwxrwxrwx 1 luca.ducceschi  domain users 9243746112 Dec  2  2011 paisa.annotated.CoNLL.utf8
-rw-r--r-- 1 patrizio.bellan domain users      10088 Apr  2 16:21 commandLs
-rw-r--r-- 1 patrizio.bellan domain users     175251 Apr  1 16:38 alice
```

La sintassi del comando

ls –[opzioni] [nome file]

Riepilogo opzioni del comando ls

```
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.
       Mandatory arguments to long options are  mandatory  for  short  options
       too.
       -a, --all
              do not ignore entries starting with .
       -A, --almost-all
              do not list implied . and ..
       --author
              with -l, print the author of each file
       -b, --escape
              print octal escapes for nongraphic characters
       --block-size=SIZE
              use SIZE-byte blocks.  See SIZE format below
       -B, --ignore-backups
              do not list implied entries ending with ~
       -c     with -lt: sort by, and show, ctime (time of last modification of
              file status information) with -l: show ctime and  sort  by  name
              otherwise: sort by ctime
       -C     list entries by columns
      --color[=WHEN]
              colorize  the  output.  WHEN  defaults  to  'always'  or can be
              'never' or 'auto'.  More info below
       -d, --directory
              list directory entries instead of contents, and do not  derefer-
              ence symbolic link
       -D, --dired
              generate output designed for Emacs' dired mode
       -f     do not sort, enable -aU, disable -ls --color
       -F, --classify
              append indicator (one of */=>@|) to entries
       --file-type
              likewise, except do not append '*'
       --format=WORD
              across  -x, commas -m, horizontal -x, long -l, single-column -1,
              verbose -l, vertical -C
       --full-time
              like -l --time-style=full-iso
       -g     like -l, but do not list owner
       --group-directories-first
              group directories before files.
              augment with a --sort option, but any use  of  --sort=none  (-U)
              disables grouping
       -G, --no-group
              in a long listing, don't print group names
       -h, --human-readable
              with -l, print sizes in human readable format (e.g., 1K 234M 2G)
      --si    likewise, but use powers of 1000 not 1024

       -H, --dereference-command-line
```

```
       follow symbolic links listed on the command line
  --dereference-command-line-symlink-to-dir
       follow each command line symbolic link that points to  a  direc-
       tory
  --hide=PATTERN
       do  not  list implied entries matching shell PATTERN (overridden
       by -a or -A)
  --indicator-style=WORD
       append indicator with style WORD to entry names: none (default),
       slash (-p), file-type (--file-type), classify (-F)
  -i, --inode
       print the index number of each file
  -I, --ignore=PATTERN
       do not list implied entries matching shell PATTERN
  -k     like --block-size=1K
  -l     use a long listing format
  -L, --dereference
       when showing file information for a symbolic link, show informa-
       tion for the file the link references rather than for  the  link
       itself
  -m     fill width with a comma separated list of entries
  -n, --numeric-uid-gid
       like -l, but list numeric user and group IDs
  -N, --literal
       print  raw entry names (don't treat e.g. control characters spe-
       cially)
  -o     like -l, but do not list group information
  -p, --indicator-style=slash
       append / indicator to directories
  -q, --hide-control-chars
       print ? instead of non graphic characters
  --show-control-chars
       show non graphic characters as-is  (default  unless  program  is
       'ls' and output is a terminal)
  -Q, --quote-name
       enclose entry names in double quotes
  --quoting-style=WORD
       use  quoting style WORD for entry names: literal, locale, shell,
       shell-always, c, escape
  -r, --reverse
       reverse order while sorting
  -R, --recursive
       list subdirectories recursively
  -s, --size
       print the allocated size of each file, in blocks
  -S     sort by file size
  --sort=WORD
       sort by WORD instead of name: none -U, extension  -X,  size  -S,
       time -t, version -v
  --time=WORD
       with  -l,  show time as WORD instead of modification time: atime
       -u, access -u, use -u, ctime -c, or  status  -c;  use  specified
       time as sort key if --sort=time
  --time-style=STYLE
       with  -l, show times using style STYLE: full-iso, long-iso, iso,
       locale, +FORMAT.  FORMAT is interpreted like 'date';  if  FORMAT
       is  FORMAT1<newline>FORMAT2, FORMAT1 applies to non-recent files
       and FORMAT2 to recent files; if STYLE is prefixed with 'posix-',
       STYLE takes effect only outside the POSIX locale
  -t     sort by modification time
```

```
       -T, --tabsize=COLS
              assume tab stops at each COLS instead of 8
       -u     with  -lt:  sort  by, and show, access time with -l: show access
              time and sort by name otherwise: sort by access time
       -U     do not sort; list entries in directory order
       -v     natural sort of (version) numbers within text
       -w, --width=COLS
              assume screen width instead of current value
       -x     list entries by lines instead of by columns
       -X     sort alphabetically by entry extension
       -1     list one file per line
       SELinux options:
       --lcontext
              Display security context.   Enable -l. Lines  will  probably  be
              too wide for most displays.
       -Z, --context
              Display  security context so it fits on most displays.  Displays
              only mode, user, group, security context and file name.
       --scontext
              Display only security context and file name.
       --help display this help and exit
       --version
              output version information and exit
       SIZE may be (or may be an integer optionally followed by) one  of  fol-
       lowing: KB 1000, K 1024, MB 1000*1000, M 1024*1024, and so on for G, T,
       P, E, Z, Y.
       Using color to distinguish file types is disabled both by  default  and
       with  --color=never.  With --color=auto, ls emits color codes only when
       standard output is connected to a terminal.  The LS_COLORS  environment
       variable can change the settings.  Use the dircolors command to set it.
     Exit status:
       0      if OK,
       1      if minor problems (e.g., cannot access subdirectory),
       2      if serious trouble (e.g., cannot access command-line  argument).
```

Ora siamo pronti a creare la nostra prima cartella per organizzare i files.

IL COMANDO MKDIR

Per creare una nuova cartella nella posizione in cui ci troviamo utilizziamo il comando ***mkdir***

```
    [patrizio.bellan@dominio Patrizio]$ mkdir cartella
```

Per verificare che la cartella è stata creata correttamente, dato che il comando non restituisce nessun tipo di output, utilizziamo il comando ***ls***

```
    [patrizio.bellan@dominio Patrizio]$ ls
    alice     commandLs              PaisaSentsExtractor  testi   venv27sistem_site_packages  virtualenv-
15.0.1.tar.gz
    cartella  paisa.annotated.CoNLL.utf8  testerTokenizers     venv27  virtualenv-15.0.1
```

come possiamo vedere dall'elemento evidenziato, la directory precedentemente creata è ora presente nella nostra attuale posizione.

Se volessimo ricevere in output in feedback del processo di creazione della cartella possiamo utilizzare il parametro **–*v***, come nell'esempio

```
[patrizio.bellan@dominio Patrizio]$ mkdir cartella3 -v
mkdir: created directory `cartella3'
```

La sintassi del comando

mkdir –[opzioni] [nome cartella]

Riepilogo opzioni del comando ls

```
NAME
       mkdir - make directories
SYNOPSIS
       mkdir [OPTION]... DIRECTORY...
DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.
       Mandatory arguments to long options are mandatory for short options too.
       -m, --mode=MODE
              set file mode (as in chmod), not a=rwx - umask
       -p, --parents
              no error if existing, make parent directories as needed
       -v, --verbose
              print a message for each created directory
       -Z, --context=CTX
              set the SELinux security context of each created directory to CTX
       --help display this help and exit
       --version
              output version information and exit
```

mettiamo il caso che abbiamo la necessità di rimuovere/eliminare la cartella appena creata, per soddisfare questa necessità utilizzeremo il comando di rimozione cartelle ***rmdir***

IL COMANDO RMDIR

Quindi per rimuovere la cartella appena creata, scriveremo a terminale

```
[patrizio.bellan@dominio Patrizio]$ rmdir cartella
```

Come per il comando precedente, anche questo comando non restituisce nessun tipo di feedback. Per verificare che la cartella sia stata effettivamente cancellata, andiamo a richiamare il comando di list files ***ls***

```
[patrizio.bellan@dominio Patrizio]$ ls
alice      paisa.annotated.CoNLL.utf8  testerTokenizers  venv27                    virtualenv-15.0.1
commandLs  PaisaSentsExtractor         testi             venv27sistem_site_packages  virtualenv-15.0.1.tar.gz
```

Come possiamo vedere, la folder *cartella* non è più inclusa nel nostro elenco, ciò significa che è stata effettivamente rimossa

Nota sull'utilizzo del comando:

questo comando permette di eliminare una cartella solo ed esclusivamente se la cartella è vuota, ossia se non vi sono files all'interno di essa. Nel caso in cui ci fossero presenti dei files, otterremo un output simile a questo

```
[patrizio.bellan@dominio Patrizio]$ rmdir cartella
rmdir: failed to remove `cartella': Directory not empty
```

che ci informa che non è stato possibile portare a termine il comando con successo

La sintassi del comando

rmdir–[opzioni] [nome cartella]

Riepilogo opzioni del comando rmdir

```
rm NAME
       rmdir - remove empty directories
SYNOPSIS
       rmdir [OPTION]... DIRECTORY...
DESCRIPTION
       Remove the DIRECTORY(ies), if they are empty.
       --ignore-fail-on-non-empty
              ignore each failure that is solely because a directory
              is non-empty
       -p, --parents
              remove DIRECTORY and its ancestors; e.g., 'rmdir -p a/b/c' is similar to 'rmdir a/b/c a/b a'
       -v, --verbose
              output a diagnostic for every directory processed
       --help display this help and exit
       --version
              output version information and exit
```

ora ricreiamo la folder di nome cartella (mkdir cartella).

Il passo successivo consiste nell'accedere alla cartella appena creata; per fare ciò utilizzeremo il comando di change directory ***cd***

IL COMANDO CD

```
[patrizio.bellan@dominio Patrizio]$ cd cartella
[patrizio.bellan@dominio cartella]$
```

Come possiamo vedere, una volta inviato il comando ci troviamo direttamente nella cartella appena indicata. È facile accorgersi di questo perché nel nostro prompt, subito dopo il nome utente e il dominio troviamo il nome della folder in cui siamo entrati (come evidenziato)

Questo comando ci permette di spostarci all'interno delle cartelle del nostro sistema. Per ritornare alla cartella precedente, ovverosia quella di livello superiore in cui ci trovavamo prima, utilizzeremo al posto del nome della cartella, due punti successivi "***..***"

```
[patrizio.bellan@dominio cartella]$ cd ..
[patrizio.bellan@dominio Patrizio]$
```

Come possiamo vedere ora ci troviamo nella cartella precedente.

Per accedere direttamente alla nostra cartella di home, invece di ".." utilizziamo il carattere tilde oppure "***--***"

```
[patrizio.bellan@dominio Patrizio]$ cd --
[patrizio.bellan@dominio ~]$
```

La sintassi del comando

cd [nome cartella] –[opzioni]

Riepilogo opzioni del comando cd

```
cd [-L|-P] [dir]
    Change the current directory to dir.  The variable HOME  is  the
    default  dir.   The  variable CDPATH defines the search path for
    the directory containing dir.  Alternative  directory  names  in
    CDPATH  are  separated by a colon (:).  A null directory name in
    CDPATH is the same as the current directory,  i.e.,  ‘‘.’’.   If
    dir  begins  with  a  slash (/), then CDPATH is not used. The -P
    option says to use the physical directory structure  instead  of
    following  symbolic  links  (see  also  the -P option to the set
    builtin command); the -L option forces symbolic links to be fol-
    lowed.   An  argument  of - is equivalent to $OLDPWD.  If a non-
    empty directory name from CDPATH is used, or if - is  the  first
    argument,  and  the directory change is successful, the absolute
    pathname of the new working directory is written to the standard
    output.   The return value is true if the directory was success-
    fully changed; false otherwise.
```

Ora ci troviamo nella folder di nome *cartella*, siamo pronti per creare il nostro primo file. Andremo a creare un file vuoto; usiamo il comando ***touch***

IL COMANDO TOUCH

```
[patrizio.bellan@dominio cartella]$ touch file_prova
```

Anche questo comando non ci restituisce un feedback, per verificare che il file sia stato effettivamente creato, possiamo utilizzare il comando ***ls***

```
[patrizio.bellan@dominio cartella]$ ls
file_prova
```

è anche possibile creare più file con una sola chiamata del comando, come nel seguente esempio:

```
[patrizio.bellan@dominio cartel1]$ touch f1 f2
[patrizio.bellan@dominio cartel1]$ ls
f1  f2
[patrizio.bellan@dominio cartel1]$ touch f3 f4 f5
[patrizio.bellan@dominio cartel1]$ ls
f1  f2  f3  f4  f5
```

una delle opzioni più interessanti che ci mette a disposizione questo comando è la possibilità di aggiornare la data di creazzione di un file tramite il parametro opzionale –m. ne riportiamo un brevissimo esempio

```
[patrizio.bellan@masterclic4 cartella]$ date
Mon Apr  4 14:08:28 CEST 2016
[patrizio.bellan@masterclic4 cartella]$ ls -l
total 172
-rwxrwxr-x 1 patrizio.bellan domain users 175251 Apr  4 14:00 alice
[patrizio.bellan@masterclic4 cartella]$ touch -m alice
[patrizio.bellan@masterclic4 cartella]$ ls -l
total 172
-rwxrwxr-x 1 patrizio.bellan domain users 175251 Apr  4 14:08 alice
```

Come si può notare dalla schermata del terminale, subito dopo aver eseguito il comando, la data di creazione del file è stata modificata (come evidenziato)

La sintassi del comando

touch –[opzioni] [nome file]

Riepilogo opzioni del comando touch

```
NAME
       touch - change file timestamps
SYNOPSIS
       touch [OPTION]... FILE...
DESCRIPTION
       Update the access and modification times of each FILE to the current time.
       A FILE argument that does not exist is created empty, unless -c or -h is supplied.
       A FILE argument string of - is handled specially and causes touch to change the times of the file
associated with standard
       output.
       Mandatory arguments to long options are mandatory for short options too.
       -a     change only the access time
       -c, --no-create
              do not create any files
       -d, --date=STRING
              parse STRING and use it instead of current time
       -f     (ignored)
       -h, --no-dereference
```

```
              affect each symbolic link instead of any referenced file (useful only on systems that can change the timestamps  of
              a symlink)
       -m     change only the modification time
       -r, --reference=FILE
              use this file's times instead of current time
       -t STAMP
              use [[CC]YY]MMDDhhmm[.ss] instead of current time
       --time=WORD
              change  the  specified time: WORD is access, atime, or use: equivalent to -a WORD is modify or mtime: equivalent to
              -m
       --help display this help and exit
       --version
              output version information and exit
       Note that the -d and -t options accept different time-date formats.
   DATE STRING
       The --date=STRING is a mostly free format human readable date string  such  as  "Sun,  29  Feb  2004 16:21:42  -0800"  or
       "2004-02-29  16:21:42"  or  even  "next Thursday".  A date string may contain items indicating calendar date, time of day,
       time zone, day of week, relative time, relative date, and numbers.  An empty string indicates the beginning  of  the  day.
       The date string format is more complex than is easily documented here but is fully described in the info documentation.
```

Il passo successivo è quello di imparare a eliminare un singolo file, per fare questo utlizzeremo il comando ***rm***

IL COMANDO RM

```
[patrizio.bellan@dominio cartella]$ rm file_prova
```

Anche questo comando non ci restituisce un output di feedback, qualora lo volessimo, dobbiamo utilizzare il parametro ***–v***

```
[patrizio.bellan@dominio cartella]$ rm file_prova -v
removed `file_prova'
```

come possiamo vedere dall'esempio sopra, questa volta il sistema ci informa che il processo è giunto al termine con successo (come evidenziato).

Ora la nostra folder è vuota, difatti se digitiamo il comando di list files otteniamo il seguente output

```
[patrizio.bellan@dominio cartella]$ ls
[patrizio.bellan@dominio cartella]$
[patrizio.bellan@dominio cartella]$ ls -l
total 0
```

nella prima esecuzione di ***ls*** (quella senza parametri) vediamo che non è comparsa nessuna lista; nella seconda (quella con il parametro long ***–l***), il sistema ci informa che ci sono **0** dati in questa cartella.

È possibile dare una conferma esplicita dell'eliminazione del file semplicemente aggiungendo il parametro opzionale –i

```
[patrizio.bellan@dominio cartella]$ ls
file_prova  file_prova2  file_prova3
```

```
[patrizio.bellan@dominio cartella]$ rm file_prova -i
rm: remove regular empty file `file_prova'? y
[patrizio.bellan@dominio cartella]$ ls
file_prova2  file_prova3
[patrizio.bellan@dominio cartella]$ rm file_prova2 -i
rm: remove regular empty file `file_prova2'? no
[patrizio.bellan@dominio cartella]$ ls
file_prova2  file_prova3
[patrizio.bellan@dominio cartella]$ rm file_prova2 -i
rm: remove regular empty file `file_prova2'? n
[patrizio.bellan@dominio cartella]$ ls
file_prova2  file_prova3
[patrizio.bellan@dominio cartella]$
```

La vera potenza di questo comando risiede nella possibilità di eliminare ricorsivamente i files presenti in una cartella. Per verificare questo per prima cosa dalla nostra posizione attuale creaiamo una nuova cartella che chiameremo cartel1, andremo a spostarci all'interno di essa tramite il comando cd cartel1, creeremo cinque files diversi chiamati f1,f2,f3,f4,f5 tramite il comando touch. Una volta terminate queste operazioni andremo a cancellare tutto il contenuto in modo ricorsivo, come nel prossimo esempio

```
[patrizio.bellan@dominio cartella]$ mkdir cartel1
[patrizio.bellan@dominio cartella]$ cd cartel1
[patrizio.bellan@dominio cartel1]$ touch f1 f2 f3 f4 f5
[patrizio.bellan@dominio cartel1]$ ls
f1  f2  f3  f4  f5
[patrizio.bellan@dominio cartel1]$ cd ..
[patrizio.bellan@dominio cartella]$ rm cartel1 -r
[patrizio.bellan@dominio cartella]$ ls
file_prova2  file_prova3
[patrizio.bellan@dominio cartella]$
```

Tutto questo è stato possibile grazie al parametro ***–r*** che informa il comando di andare ad eliminare in modo ricorsivo tutti i file presenti nella cartella ***cartel1***, ed in ultimo rimuove la stessa.

Come si ricorda dai comandi precedenti, è possibile concatenare le opzioni; in questo esempio siamo andati a concatenare le opzioni di eliminazione ricorsiva e conferma di eliminazione (***-ri***)

```
[patrizio.bellan@dominio cartella]$ rm cartel1 -ri
rm: descend into directory `cartel1'? n
rm: remove regular empty file `cartel1/f3'? n
rm: remove regular empty file `cartel1/f2'? n
rm: remove regular empty file `cartel1/f4'? y
rm: remove regular empty file `cartel1/f5'? y
rm: remove regular empty file `cartel1/f1'? y
rm: remove directory `cartel1'? y
rm: cannot remove `cartel1': Directory not empty
[patrizio.bellan@dominio cartella]$
[patrizio.bellan@dominio cartella]$ rm cartel1 -ri
rm: descend into directory `cartel1'? y
rm: remove regular empty file `cartel1/f3'? y
rm: remove regular empty file `cartel1/f2'? y
rm: remove directory `cartel1'? y
```

La sintassi del comando

Rm –[opzioni] [nome file]

Riepilogo opzioni del comando rm

```
NAME
       rm - remove files or directories
SYNOPSIS
       rm [OPTION]... FILE...
DESCRIPTION
       This  manual page documents the GNU version of rm.  rm removes each specified file.  By default, it
does not remove direc-
       tories.
       If the -I or --interactive=once option is given, and there are more than three files or the -r,  -R,  or  --
recursive  are
       given, then rm prompts the user for whether to proceed with the entire operation.  If the response is not
affirmative, the
       entire command is aborted.
       Otherwise, if a file is unwritable, standard input is a terminal, and the -f or --force option is not given, or
the -i  or
       --interactive=always option is given, rm prompts the user for whether to remove the file.  If the response
is not affirma-
       tive, the file is skipped.
OPTIONS
       Remove (unlink) the FILE(s).
       -f, --force
              ignore nonexistent files, never prompt
       -i     prompt before every removal
       -I     prompt once before removing more than three files, or when removing recursively.  Less  intrusive
than  -i,  while
              still giving protection against most mistakes
       --interactive[=WHEN]
              prompt according to WHEN: never, once (-I), or always (-i).  Without WHEN, prompt always
       --one-file-system
              when  removing a hierarchy recursively, skip any directory that is on a file system different from
that of the cor-
              responding command line argument
       --no-preserve-root
              do not treat '/' specially
      --preserve-root
              do not remove '/' (default)
       -r, -R, --recursive
              remove directories and their contents recursively
       -v, --verbose
              explain what is being done
       --help display this help and exit
       --version
              output version information and exit
       By default, rm does not remove directories.  Use the --recursive (-r or -R) option to remove each listed
directory,  too,
       along with all of its contents.
       To remove a file whose name starts with a '-', for example '-foo', use one of these commands:
              rm -- -foo
              rm ./-foo
       Note  that  if you use rm to remove a file, it is usually possible to recover the contents of that file.  If you
want more
       assurance that the contents are truly unrecoverable, consider using shred.
```

Giunti a questo punto, vogliamo iniziare a scrivere qualche informazione dentro un file. Il sistema operativo ci viene incontro grazie ad un semplicissimo editor di test chiamato ***nano***

L'EDITOR DI TESTI NANO

Per accedere a questo semplice programm, basta digitarne il nome nel propt dei comandi

```
[patrizio.bellan@dominio cartella]$ nano
```

Qui ci troviamo difronte ad una schermata molto minimalista, in cui sono riportati nella prima riga le informazioni riguardo a questo editor, nella parte inferiore invece si trova un riepilogo dei comandi e il loro scopo.

Ora andiamo a scrivere qualche semplice informazione nel file, lo salviamo e chiudiamo l'editor. Basta iniziare a digitare sulla tastiera…

```
Hello World
I'm Patrizio Bellan
```

Ora dobbiamo salvare quello che abbiamo scritto su file, per fare questo premiamo la combinazione di tasti:

[crtl] + [o]

```
File Name to Write: provaNano
```

Decidiamo di chiamare il nostro file *provaNano*. Come possiamo notare, una volta premuta la combinazione di tasta in basso l'editor ci chiede come desideriamo chiamare il file (attirando la nostra attenzione grazie all'evidenziazione della riga).

Per uscire dall'editor digitiamo la combinazione di tasti:

[crtl] + [x]

Se ci fossimo dimenticati di salvare prima di uscire, l'editor ci chiederà la conferma di uscita

```
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES ) ?
```

Ora che siamo usciti, proviamo ad aprire il nostro file appena creato per aggiungere qualche informazione in più, per fare queste usiamo nano [nome file], come nell'esempio

```
[patrizio.bellan@dominio cartella]$ nano provaNano
```

A questo punto ci ritroviamo nell'editor precedente, con la differenza che il testo precedentemente salvato è presente nella nostra schermata.

Nota:

tutti i comandi utilizzabili nell'editor (quelli presenti nella parte bassa) si eseguono grazie alla combinazione di tasti:

[crtl] + [lettera associata al comando]

Per proseguire nell'esplorazione dei comandi e iniziare a lavorare sui testi, andiamo a scaricare un file da internet. Andremo a scaricare il file di Alice nel paese delle meraviglie, gratuitamente scaricabile dal sito del progetto gutenberg all'indirizzo:

http://www.gutenberg.org/cache/epub/28371/pg28371.txt

questo formidabile sistema operativo ci viene incontro, semplificandoci il questo compito, mettendoci a disposizione l'utility ***wget***

UTILITY WGET

Quindi andiamo sulla schermata del nostro terminale e digitiamo wget [indirizzo file]

```
[patrizio.bellan@dominio cartella]$ wget http://www.gutenberg.org/cache/epub/28371/pg28371.txt
--2016-04-02 19:33:48--  http://www.gutenberg.org/cache/epub/28371/pg28371.txt
Resolving www.gutenberg.org... 152.19.134.47
Connecting to www.gutenberg.org|152.19.134.47|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 175251 (171K) [text/plain]
Saving to: "pg28371.txt"

100%[==========================================================================
==========================>] 175,251      220K/s   in 0.8s

2016-04-02 19:33:50 (220 KB/s) - "pg28371.txt" saved [175251/175251]

[patrizio.bellan@dominio cartella]$ ls
f1  f2  f3  f4  f5  file_prova2  file_prova3  pg28371.txt  provaNano
[patrizio.bellan@dominio cartella]$
```

Come possiamo vedere, il nostro file evidenziato è quello che abbiamo appena scaricato.

Dato che il nome *pg28371.txt* non è molto chiarificatore riguardo al suo contenuto, sarebbe stato più comodo scaricare il file e chiamarlo alice, per fare questo aggiungiamo dopo l'indirizzo i due parametri:

–O [*nome da dare al file*]

Come nell'esempio:

```
[patrizio.bellan@dominio cartella]$ wget http://www.gutenberg.org/cache/epub/28371/pg28371.txt -O alice
--2016-04-02 19:38:09--  http://www.gutenberg.org/cache/epub/28371/pg28371.txt
Resolving www.gutenberg.org... 152.19.134.47
Connecting to www.gutenberg.org|152.19.134.47|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 175251 (171K) [text/plain]
Saving to: "alice"

100%[==========================================================================
==========================>] 175,251      214K/s   in 0.8s

2016-04-02 19:38:10 (214 KB/s) - "alice" saved [175251/175251]
```

La sintassi del comando

wget [indirizzo del file] –[opzioni] [parametri opzione]

Riepilogo opzioni del comando wget

```
NAME
       Wget - The non-interactive network downloader.

SYNOPSIS
       wget [option]... [URL]...
DESCRIPTION
       GNU Wget is a free utility for non-interactive download of files from the Web.  It supports HTTP,
HTTPS, and FTP
       protocols, as well as retrieval through HTTP proxies.
       Wget is non-interactive, meaning that it can work in the background, while the user is not logged on.
This allows you to
       start a retrieval and disconnect from the system, letting Wget finish the work.  By contrast, most of the
Web browsers
       require constant user's presence, which can be a great hindrance when transferring a lot of data.
       Wget can follow links in HTML, XHTML, and CSS pages, to create local versions of remote web sites,
fully recreating the
       directory structure of the original site.  This is sometimes referred to as "recursive downloading."
While doing that,
       Wget respects the Robot Exclusion Standard (/robots.txt).  Wget can be instructed to convert the links
in downloaded files
       to point at the local files, for offline viewing.

       Wget has been designed for robustness over slow or unstable network connections; if a download fails
due to a network
       problem, it will keep retrying until the whole file has been retrieved.  If the server supports regetting, it
will
       instruct the server to continue the download from where it left off.
OPTIONS
   Option Syntax
       Since Wget uses GNU getopt to process command-line arguments, every option has a long form along
with the short one.  Long
       options are more convenient to remember, but take time to type.  You may freely mix different option
styles, or specify
       options after the command-line arguments.  Thus you may write:
              wget -r --tries=10 http://fly.srk.fer.hr/ -o log
       The space between the option accepting an argument and the argument may be omitted.  Instead of -o
log you can write
       -olog.
       You may put several options that do not require arguments together, like:
              wget -drc <URL>
      This is completely equivalent to:
              wget -d -r -c <URL>
       Since the options can be specified after the arguments, you may terminate them with --.  So the following
will try to
       download URL -x, reporting failure to log:
              wget -o log -- -x
       The options that accept comma-separated lists all respect the convention that specifying an empty list
clears its value.
       This can be useful to clear the .wgetrc settings.  For instance, if your .wgetrc sets "exclude_directories"
to /cgi-bin,
       the following example will first reset it, and then set it to exclude /~nobody and /~somebody.  You can
also clear the
       lists in .wgetrc.
              wget -X '' -X /~nobody,/~somebody

   Basic Startup Options
       -V
```

--version
Display the version of Wget.
-h
--help
Print a help message describing all of Wget's command-line options.
-b
--background
Go to background immediately after startup. If no output file is specified via the -o, output is redirected to wget-
log.
-e command
--execute command
Execute command as if it were a part of .wgetrc. A command thus invoked will be executed after the commands in
.wgetrc, thus taking precedence over them. If you need to specify more than one wgetrc command, use multiple
instances of -e.

Logging and Input File Options

-o logfile
--output-file=logfile
Log all messages to logfile. The messages are normally reported to standard error.

-a logfile
--append-output=logfile
Append to logfile. This is the same as -o, only it appends to logfile instead of overwriting the old log file. If
logfile does not exist, a new file is created.
-q
--quiet
Turn off Wget's output.
-v
--verbose
Turn on verbose output, with all the available data. The default output is verbose.
-nv
--no-verbose
Turn off verbose without being completely quiet (use -q for that), which means that error messages and basic
information still get printed.
-i file
--input-file=file
Read URLs from a local or external file. If - is specified as file, URLs are read from the standard input. (Use ./-
to read from a file literally named -.)
-F
--force-html
When input is read from a file, force it to be treated as an HTML file. This enables you to retrieve relative links
from existing HTML files on your local disk, by adding "<base href="url">" to HTML, or using the --base command-line option.
-O file
--output-document=file
The documents will not be written to the appropriate files, but all will be concatenated together and written to file.
If - is used as file, documents will be printed to standard output, disabling link conversion. (Use ./- to print to a file literally named -.)
-c
--continue
Continue getting a partially-downloaded file. This is useful when you want to finish up a download started by a
previous instance of Wget, or by another program. For instance:

```
wget -c ftp://sunsite.doc.ic.ac.uk/ls-lR.Z
```

tornando all'esempio precedente, il nostro file si chiamava *pg28371.txt* invece di scaricarlo nuovamente e assegnarli un nome durante la fase di scaricamento, avremmo potuto semplicemente rinominare il file. Per rinominare un file si utilizza il comando ***mv***

IL COMANDO MV

```
[patrizio.bellan@dominio cartella]$ mv pg28371.txt alice_rinominato
[patrizio.bellan@dominio cartella]$ ls -l
total 348
-rw-r--r-- 1 patrizio.bellan domain users 175251 Apr  2 19:38 alice
-rw-r--r-- 1 patrizio.bellan domain users 175251 Apr  2 19:35 alice_rinominato
-rw-r--r-- 1 patrizio.bellan domain users     78 Apr  2 19:13 provaNano
```

Come si evince dall'esempio appena mostrato, il nostro file è stato rinominato in alice_rinominato, proprio come volevamo.

Il comando mv serve a spostare file tra le cartelle, ma grazie al suo uso "in locale", ovverosia all'interno della cartella in cui siamo, è possibile rinominare i file.

```
[patrizio.bellan@dominio cartella]$ ls
alice  alice_rinominato  provaNano
[patrizio.bellan@dominio cartella]$ mkdir cartellaA
[patrizio.bellan@dominio cartella]$ mv alice_rinominato cartellaA/alice
[patrizio.bellan@dominio cartella]$ ls
alice  cartellaA  provaNano
[patrizio.bellan@dominio cartella]$ cd cartellaA
[patrizio.bellan@dominio cartellaA]$ ls
alice
```

come possiamo vedere dal riepilogo dell'output del terminale, il file è stato correttamente spostato e rinominato allo stesso tempo.

Se non avessimo voluto rinominare il file, bastava non digitare il nome del file dopo il nome della cartella

```
[patrizio.bellan@dominio cartella]$ mv alice_rinominato cartellaA/
```

Nota sulle cartelle:

è possibile copiare il file nella cartella di livello superiore senza passare tutto l'indirizzo in modo assoluto ma utilizzando come nome cartella della prima carte il nome "..". i due punti successivi rappresentano proprio il collegamento tra la cartella e il loro contenitore

se ad esempio avessimo dovuto copiare (sempre il nostro file *alice*_rinominato) da *cartellaA* a *cartella* (si ricorda che *cartellaA* è una sottocartella di *cartella*), avremmo eseguito il comando con questa forma:

```
[patrizio.bellan@dominio cartellaA]$ mv alice_rinominato ../cartella
```

La sintassi del comando

Mv –[opzioni] [nome file source] [nome file dest]

Riepilogo opzioni del comando

```
NAME
       mv - move (rename) files

SYNOPSIS
       mv [OPTION]... [-T] SOURCE DEST
       mv [OPTION]... SOURCE... DIRECTORY
       mv [OPTION]... -t DIRECTORY SOURCE...
DESCRIPTION
       Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.
       Mandatory arguments to long options are mandatory for short options too.
       --backup[=CONTROL]
              make a backup of each existing destination file
       -b     like --backup but does not accept an argument
       -f, --force
              do not prompt before overwriting
       -i, --interactive
              prompt before overwrite
       -n, --no-clobber
              do not overwrite an existing file
       If you specify more than one of -i, -f, -n, only the final one takes effect.
       --strip-trailing-slashes
              remove any trailing slashes from each SOURCE argument
       -S, --suffix=SUFFIX
              override the usual backup suffix
       -t, --target-directory=DIRECTORY
              move all SOURCE arguments into DIRECTORY
             treat DEST as a normal file
       -u, --update
              move only when the SOURCE file is newer than the destination file or when the destination file is
missing
       -v, --verbose
              explain what is being done
       --help display this help and exit
       --version
              output version information and exit
       The  backup  suffix  is '~', unless set with --suffix or SIMPLE_BACKUP_SUFFIX.  The version control
method may be selected
       via the --backup option or through the VERSION_CONTROL environment variable.  Here are the
values:
       none, off
              never make backups (even if --backup is given)
       numbered, t
              make numbered backups
       existing, nil
              numbered if numbered backups exist, simple otherwise
       simple, never
              always make simple backups
```

ora che sappiamo come scaricare un file, rinominarlo e spostarlo, impariamo a copiare un file. Per copiare un file o una cartella utilizziamo il comando ***cp***

IL COMANDO CP

```
[patrizio.bellan@dominio cartella]$ ls
alice  cartellaA
[patrizio.bellan@dominio cartella]$ cp alice cartellaA
[patrizio.bellan@dominio cartella]$ ls
alice  cartellaA
[patrizio.bellan@dominio cartella]$ cd cartellaA
[patrizio.bellan@dominio cartellaA]$ ls
alice
[patrizio.bellan@dominio cartellaA]$
```

Come possiamo vedere, il file alice è stato copiato nella folder *cartellaA*.

Anche questo comando è dotato del parametro –r di ricorsività che ci permette di copiare più files con una sola chiamata. Se ne fornisce un esempio chiarificativo

```
[patrizio.bellan@dominio Patrizio]$ cd cartella
[patrizio.bellan@dominio cartella]$ ls
alice  cartellaA
[patrizio.bellan@dominio cartella]$ mkdir cartellaB
[patrizio.bellan@dominio cartella]$ touch cartellaB/f1 cartellaB/f2 cartellaB/f3
[patrizio.bellan@dominio cartella]$ cd cartellaB
[patrizio.bellan@dominio cartellaB]$ ls
f1  f2  f3
[patrizio.bellan@dominio cartellaB]$ cd ..
[patrizio.bellan@dominio cartella]$ mkdir cartellaDst
[patrizio.bellan@dominio cartella]$ cp cartellaB cartellaDst -r
[patrizio.bellan@dominio cartella]$ ls
alice  cartellaA  cartellaB  cartellaDst
[patrizio.bellan@dominio cartella]$ cd cartellaDst
[patrizio.bellan@dominio cartellaDst]$ ls
cartellaB
[patrizio.bellan@dominio cartellaDst]$ cd cartellaB
[patrizio.bellan@dominio cartellaB]$ ls
f1  f2  f3
[patrizio.bellan@dominio cartellaB]$ cp * ../
[patrizio.bellan@dominio cartellaB]$ ls
f1  f2  f3
[patrizio.bellan@dominio cartellaB]$ cd ..
[patrizio.bellan@dominio cartellaDst]$ ls
cartellaB  f1  f2  f3
[patrizio.bellan@dominio cartellaDst]$
```

La sintassi del comando

cp –[opzioni] [nome file source] [nome file destinazione]

Riepilogo opzioni del comando

```
NAME
       cp - copy files and directories
SYNOPSIS
       cp [OPTION]... [-T] SOURCE DEST
       cp [OPTION]... SOURCE... DIRECTORY
       cp [OPTION]... -t DIRECTORY SOURCE...
DESCRIPTION
```

```
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.
Mandatory arguments to long options are mandatory for short options too.
-a, --archive
      same as -dR --preserve=all
--backup[=CONTROL]
      make a backup of each existing destination file
-b    like --backup but does not accept an argument
--copy-contents
      copy contents of special files when recursive
-d    same as --no-dereference --preserve=links
-f, --force
      if an existing destination file cannot be opened, remove it and try again (redundant if the -n option is
used)
-i, --interactive
      prompt before overwrite (overrides a previous -n option)
-H    follow command-line symbolic links in SOURCE
-l, --link
      link files instead of copying
-L, --dereference
      always follow symbolic links in SOURCE
-n, --no-clobber
      do not overwrite an existing file (overrides a previous -i option)
-P, --no-dereference
      never follow symbolic links in SOURCE
-p    same as --preserve=mode,ownership,timestamps
--preserve[=ATTR_LIST]
      preserve the specified attributes (default: mode,ownership,timestamps), if possible additional
attributes: context,
      links, xattr, all
-c    same as --preserve=context
--no-preserve=ATTR_LIST
      don't preserve the specified attributes
--parents
      use full source file name under DIRECTORY
-R, -r, --recursive
      copy directories recursively
--reflink[=WHEN]
      control clone/CoW copies. See below.
--remove-destination
      remove each existing destination file before attempting to open it (contrast with --force)
--sparse=WHEN
      control creation of sparse files. See below.
--strip-trailing-slashes
      remove any trailing slashes from each SOURCE argument
-s, --symbolic-link
      make symbolic links instead of copying
-S, --suffix=SUFFIX
      override the usual backup suffix
-t, --target-directory=DIRECTORY
      copy all SOURCE arguments into DIRECTORY
-T, --no-target-directory
      treat DEST as a normal file
-u, --update
      copy only when the SOURCE file is newer than the destination file or when the destination file is
missing
-v, --verbose
      explain what is being done
-x, --one-file-system
      stay on this file system
-Z, --context=CONTEXT
```

```
      set security context of copy to CONTEXT
  --help display this help and exit
  --version
      output version information and exit
```

ora possiamo iniziare a guardare all'interno dei files. L'editor di testo nano è solo una delle possibilità per farlo. Possiamo andare a vedere l'intero contenuto di un file grazie al comando ***cat***

IL COMANDO CAT

Per prima cosa possiamo iniziare aprendo nano e scrivendo qualche riga; salviamo il file, chiamando ad esempio "*prova_per_cat*"

```
[patrizio.bellan@masterclic4 cartella]$ nano prova_per_cat
[patrizio.bellan@masterclic4 cartella]$ cat prova_per_cat
questo è un file di prova
scrivo giusto qualche riga per
provare il comando cat
```

come possiamo vedere dall'esempio, questo comando ci permette di ottenere tutto il contenuto di un file sullo standard output (nel nostro caso è lo schermo del nostro terminale)

una delle opzioni utilizzabili tramite questo comando è sicuramente quella che ci offre la possibilità di numerare le righe; per fare questo aggiungiamo il parametro ***-b***

```
[patrizio.bellan@masterclic4 cartella]$ cat -b prova_per_cat
     1  questo è un file di prova
     2  scrivo giusto qualche riga per
     3  provare il comando cat
```

```
[patrizio.bellan@masterclic4 cartella]$ cat prova_per_cat -b
     1  questo è un file di prova
     2  scrivo giusto qualche riga per
     3  provare il comando cat
```

Come possiamo vedere, l'output ottenuto presenta il numero di riga come primo carattere. Com'è possibile notare dai due esempi precedenti, il comando risulta molto flessibile e ci permette di utilizzare il parametro anche in fordo alla chiamata del comando, per quanto questa non sia una buona usanza.

Un output simile a questo lo si ottiene con il parametro ***–n,*** proprio come il parametro ***–b*** precedente anchesso numera le righe ma con la differenza che ***–n*** numera tutte le righe mentre ***–b*** solo le righe non vuote. Per mostrarvi questa differenza si è modificato il file *prova_per_cat*

```
[patrizio.bellan@masterclic4 cartella]$ nano prova_per_cat
[patrizio.bellan@masterclic4 cartella]$ cat -b prova_per_cat
     1  questo è un file di prova
     2  scrivo giusto qualche riga per
     3  provare il comando cat

     4  questa è una riga dopo una riga vuota
     5  questa è un'altra riga
```

```
     6  dopo 3 righe vuote smetto di scrivere
[patrizio.bellan@masterclic4 cartella]$ cat -n prova_per_cat
     1  questo è un file di prova
     2  scrivo giusto qualche riga per
     3  provare il comando cat
     4
     5  questa è una riga dopo una riga vuota
     6  questa è un'altra riga
     7
     8
     9
    10  dopo 3 righe vuote smetto di scrivere
```

La sintassi del comando

cat –[opzioni] [nome file]

Riepilogo opzioni del comando

```
NAME
       cat - concatenate files and print on the standard output
SYNOPSIS
       cat [OPTION]... [FILE]...
DESCRIPTION
       Concatenate FILE(s), or standard input, to standard output.
       -A, --show-all
              equivalent to -vET
       -b, --number-nonblank
              number nonempty output lines
       -e     equivalent to -vE
       -E, --show-ends
              display $ at end of each line
       -n, --number
              number all output lines
       -s, --squeeze-blank
              suppress repeated empty output lines
       -t     equivalent to -vT
       -T, --show-tabs
              display TAB characters as ^I
       -u     (ignored)
       -v, --show-nonprinting
              use ^ and M- notation, except for LFD and TAB
       --help display this help and exit
       --version
              output version information and exit

       With no FILE, or when FILE is -, read standard input.
EXAMPLES
       cat f - g
              Output f’s contents, then standard input, then g’s contents.
       cat    Copy standard input to standard output.
```

Ora assumiano che il nostro scopo sia visualizzare solo le prime n righe del file e non l'interno contenuto. Per assolvere questo compito, ancora una volta, è stato creato un comando specifico che ci permette di farlo tramite una chiamata concisa ed efficiente; il comando ***head***

IL COMANDO HEAD

```
[patrizio.bellan@masterclic4 cartella]$ head prova_per_cat
questo è un file di prova
scrivo giusto qualche riga per
provare il comando cat

questa è una riga dopo una riga vuota
questa è un'altra riga



dopo 3 righe vuote smetto di scrivere
```

è possibile specificare la quantità di output che si desidera visualizzare; possiamo decidere di visualizzare le prime n righe, tramite l'opzione –n oppure i primi m caratteri, tramite l'opzione –c [dim].

```
[patrizio.bellan@masterclic4 cartella]$ head -5 prova_per_cat
questo è un file di prova
scrivo giusto qualche riga per
provare il comando cat

questa è una riga dopo una riga vuota
[patrizio.bellan@masterclic4 cartella]$ head -1 prova_per_cat
questo è un file di prova
```

```
[patrizio.bellan@masterclic4 cartella]$ head -c 10 prova_per_cat
questo è [patrizio.bellan@masterclic4 cartella]$ head -c 50 prova_per_cat
questo è un file di prova
scrivo giusto qualche r[patrizio.bellan@masterclic4 cartella]$ head -c 100 prova_per_cat
questo è un file di prova
scrivo giusto qualche riga per
provare il comando cat

questa è una rig[patrizio.bellan@masterclic4 cartella]$
```

La sintassi del comando

head –[opzioni] [nome file]

Riepilogo opzioni del comando

```
NAME
       head - output the first part of files

SYNOPSIS
       head [OPTION]... [FILE]...
```

```
DESCRIPTION
       Print  the first 10 lines of each FILE to standard output.  With more than one FILE, precede each with a header giving the
       file name.  With no FILE, or when FILE is -, read standard input.

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

       -c, --bytes=[-]K
              print the first K bytes of each file; with the leading '-', print all but the last K bytes of each file

       -n, --lines=[-]K
              print the first K lines instead of the first 10; with the leading '-', print all but the last K lines of each file

       -q, --quiet, --silent
              never print headers giving file names

       -v, --verbose
              always print headers giving file names

       --help display this help and exit

       --version
              output version information and exit

       K may have a multiplier suffix: b 512, kB 1000, K 1024, MB 1000*1000, M 1024*1024, GB 1000*1000*1000,  G  1024*1024*1024,
       and so on for T, P, E, Z, Y.
```

La bash ci offre anche la possibilità di utilizzare un comando speculare a ***head*** che ci permette di visualizzare l'ultima parte di un file. Il comando che utilizzeremo sarà ***tail***

IL COMANDO TAIL

```
[patrizio.bellan@masterclic4 cartella]$ tail -5 prova_per_cat
questa è un'altra riga



dopo 3 righe vuote smetto di scrivere
[patrizio.bellan@masterclic4 cartella]$ tail -c 5 prova_per_cat
vere
[patrizio.bellan@masterclic4 cartella]$ tail -c 50 prova_per_cat
tra riga



dopo 3 righe vuote smetto di scrivere
[patrizio.bellan@masterclic4 cartella]$
```

Come possiamo vedere a questo esempio, anche il comando ***tail***, come il comando ***head***, accetta come parametri ***–n*** e ***–c*** che ci permettono di specificare la quantità di output desiderata

La sintassi del comando

tail –[opzioni] [nome file]

Riepilogo opzioni del comando

```
NAME
       tail - output the last part of files
SYNOPSIS
       tail [OPTION]... [FILE]...
DESCRIPTION
       Print  the  last 10 lines of each FILE to standard output.  With more than one FILE, precede each with a header giving the
       file name.  With no FILE, or when FILE is -, read standard input.
       Mandatory arguments to long options are mandatory for short options too.
       -c, --bytes=K
              output the last K bytes; alternatively, use -c +K to output bytes starting with the Kth of each file
       -f, --follow[={name|descriptor}]
              output appended data as the file grows; -f, --follow, and --follow=descriptor are equivalent
       -F     same as --follow=name --retry
       -n, --lines=K
              output the last K lines, instead of the last 10; or use -n +K to output lines starting with the Kth
       --max-unchanged-stats=N
              with --follow=name, reopen a FILE which has not changed size after N (default 5) iterations to see if it  has  been
              unlinked or renamed (this is the usual case of rotated log files).  With inotify, this option is rarely useful.
       --pid=PID
              with -f, terminate after process ID, PID dies
       -q, --quiet, --silent
              never output headers giving file names
       --retry
              keep  trying  to  open  a  file  even when it is or becomes inaccessible; useful when following by name, i.e., with
              --follow=name
       -s, --sleep-interval=N
              with -f, sleep for approximately N seconds (default 1.0) between iterations.
              With inotify and --pid=P, check process P at least once every N seconds.
       -v, --verbose
              always output headers giving file names
       --help display this help and exit
       --version
              output version information and exit
```

ma se avessimo bisogno di visualizzare e leggere tutto il contenuto di un file? Se il file fosse più lungo di una schermata, tramite il comando cat e le relative opzioni sarebbe impossibile da assolvere questo compito.

Anche in questo caso la bash ci viene incontro tramite il comando ereditato dal *core unix* *BSD Free* ***more***

IL COMANDO MORE

Grazie a questo comando è possibile scorrere l'interno file. Per scoprirne l'efficacia lo applicheremo sul file precedentemente scaricato *alice*.

```
[patrizio.bellan@masterclic4 cartella]$ more alice
```

Una volta premuto il tasto invio sullo schermo apparirà la prima pagina del nostro file.

A questo punto possiamo scorrere le pagine premendo il tasto *barra spaziatrice* oppure scorrere di una riga alla volta premendo il *tasto invio.*

Qualora fossimo giunti alla fine del file, oppure decidessimo per una qualsiasi ragione di voler uscire, basta premere il ***tasto q.***

Ma le potenzialità di questo comando, come di tutti i comandi di bash, risiedono nei parametri opzionali che ci permettono di modificare il comportamento del comando a nostro piacimento.

Possiamo decidere di iniziare a leggere il file da una riga arbitraria

```
[patrizio.bellan@masterclic4 cartella]$ more +55 alice
```

Come possiamo vedere dall'esempio, basta anteporre il parametro +n al nome del file per informare il comando su quale deve essere la riga di partenza all'interno del file. (per eliminare ogni dubbio sull'utilizzo si consiglia di provare il comando more +55 prova_per_cat, l'output è come facilmente presumibile vuoto inquanto il numero di partenza eccedeva il numero totale di righe all'interno del file).

Il parametro –d ci permette di riceve un messaggio di prompt nell'ultima riga del terminale; il parametro –p invece blocca la possibilità di effettuare lo scroll all'interno del file.

La sintassi del comando

more –[opzioni] [nome file]

Riepilogo opzioni del comando

```
NAME
       more - file perusal filter for crt viewing
SYNOPSIS
       more [-dlfpcsu] [-num] [+/pattern] [+linenum] [file ...]
DESCRIPTION
       More is a filter for paging through text one screenful at a time.  This version is especially primitive.
Users should real-
       ize that less(1) provides more(1) emulation and extensive enhancements.
OPTIONS
       Command line options are described below.  Options are also taken from the environment variable
MORE (make sure to precede
       them with a dash (‘‘-’’)) but command line options will override them.
       -num  This option specifies an integer which is the screen size (in lines).
       -d    more will prompt the user with the message "[Press space to continue, ’q’ to quit.]" and will display
"[Press ’h’ for
             instructions.]" instead of ringing the bell when an illegal key is pressed.
       -l    more usually treats ^L (form feed) as a special character, and will pause after any line that contains a
form feed.
             The -l option will prevent this behavior.
       -f    Causes more to count logical, rather than screen lines (i.e., long lines are not folded).
```

```
       -p    Do not scroll.  Instead, clear the whole screen and then display the text.
       -c    Do not scroll.  Instead, paint each screen from the top, clearing the remainder of each line as it is
displayed.
       -s    Squeeze multiple blank lines into one.
       -u    Suppress underlining.
       +/    The +/ option specifies a string that will be searched for before each file is displayed.
       +num  Start at line number num.
```

Il passo successivo che andremo a compiere sarà quello di far stampare a video una stringa (cioè un insieme di caratteri successivi). Questo comando ci permetterà di imparare successivamente come reindirizzare lo *standard output* invece che sullo schermo del nostro terminale direttamente in un file. Per fare questo, iniziamo con l'esplorazione del comando ***echo***.

IL COMANDO ECHO

```
[patrizio.bellan@masterclic4 cartella]$ echo 'hello world'
hello world
```

Come evidenziato, la nostra stringa è stata stampata a video.

Nota

la stringa di testo deve neccessariamente essere racchiusa tra una coppia di apici singoli oppure da una coppia di doppi apici

```
[patrizio.bellan@masterclic4 cartella]$ echo "hello world"
hello world
```

Nota 2

La differente interpretazione data tra apici e doppi apici consiste che nel primo caso la stringa è considerata immutabile mentre nel secondo caso è mutabile. Per lo scopo attuale non ci interessa la differenza tra le due, l'argomento verrà approfondito più anvanti nella guida.

La sintassi del comando

echo –[opzioni] [stringa]

Riepilogo opzioni del comando

```
NAME
       echo - display a line of text
SYNOPSIS
       echo [SHORT-OPTION]... [STRING]...
```

```
      echo LONG-OPTION
DESCRIPTION
      Echo the STRING(s) to standard output.
      -n     do not output the trailing newline
      -e     enable interpretation of backslash escapes
      -E     disable interpretation of backslash escapes (default)
      --help display this help and exit
      --version
            output version information and exit
      If -e is in effect, the following sequences are recognized:
      \\     backslash
      \a     alert (BEL)
      \b     backspace
      \c     produce no further output
      \e     escape
      \f     form feed
      \n     new line
      \r     carriage return
      \t     horizontal tab
      \v     vertical tab
      \0NNN  byte with octal value NNN (1 to 3 digits)
      \xHH   byte with hexadecimal value HH (1 to 2 digits)
      NOTE:  your  shell may have its own version of echo, which usually supersedes the version described here.
Please refer to
      your shell's documentation for details about the options it supports.
```

Ora abbiamo le conoscenze necessarie per capire ed affrontare il nostro primo reindirizzamento dell'output. Per fare questo useremo l'operatore >

L'OPERATORE > PER IL REINDIRIZZAMENTO DELLO STANDARD OUTPUT

La possibilità di effettuare il reindirizzamento dell'output ci permette di ottenere un file come risultato delle operazioni effettuate su altri files. In questo primissimo e semplicissimo esempio, andremo a scrivere nel file la stringa che prima appariva sullo schermo.

```
[patrizio.bellan@masterclic4 cartella]$ echo 'hello world' > hello
[patrizio.bellan@masterclic4 cartella]$ cat hello
hello world
```

La sintassi del comando

[operazione] > [file su cui scrivere il risultato dell'operazione]

LE PRIME OPERAZIONI DI ANALISI SUI TESTI

Ora che abbiamo esplorato i primi comandi di base, siamo pronti per iniziare ad effettuare le prime operazioni di analisi testo.

Il primo comando che andremo ad esplorare è ***rev***

IL COMANDO REV

Grazie a questo comando è possibile inverire l'ordine dei caratteri all'interno di una stringa o di un file di testo.

Per esplorarne il comportamento riprendiamo il file di esempio precedentemente creato *prova_per_cat*

```
[patrizio.bellan@masterclic4 cartella]$ cat prova_per_cat
questo è un file di prova
scrivo giusto qualche riga per
provare il comando cat

questa è una riga dopo una riga vuota
questa è un'altra riga



dopo 3 righe vuote smetto di scrivere
[patrizio.bellan@masterclic4 cartella]$ rev prova_per_cat
avorp id elif nu è otseuq
 rep agir ehclauq otsuig ovircs
tac odnamoc li eravorp

atouv agir anu opod agir anu è atseuq
agir artla'nu è atseuq



erevircs id ottems etouv ehgir 3 opod
[patrizio.bellan@masterclic4 cartella]$
```

Come possiamo vedere dal testo evidenziato, si tratta dello stesso testo visualizzato con il comando cat, questa volta però i caratteri sono tutti invertiti (partendo dall'ultimo per arrivare al primo) per ogni riga

Ora inviamo il risultato del comando ***rev*** ad un file che chiameremo *prova_rev* (si ricorda che possiamo andare a visualizzarne il risultato tramite ***cat prova_rev***)

```
[patrizio.bellan@masterclic4 cartella]$ rev prova_per_cat > prova_rev
[patrizio.bellan@masterclic4 cartella]$ cat rev
cat: rev: No such file or directory
[patrizio.bellan@masterclic4 cartella]$ cat prova_rev
avorp id elif nu è otseuq
```

```
 rep agir ehclauq otsuig ovircs
tac odnamoc li eravorp

atouv agir anu opod agir anu è atseuq
agir artla'nu è atseuq



erevircs id ottems etouv ehgir 3 opod
```

Nota

Questo comando non supporta nessun tipo di parametri opzionali.

Piccola nota

Anche questo comando è un'eredità di DSB Free

La sintassi del comando

rev [dati da invertire]

Riepilogo opzioni del comando

```
NAME
       rev - reverse lines of a file or files

SYNOPSIS
       rev [file ...]

DESCRIPTION
       The rev utility copies the specified files to the standard output, reversing the order of characters in every
line.  If no
       files are specified, the standard input is read.

AVAILABILITY
       The rev command is part of the util-linux-ng package and is available from
ftp://ftp.kernel.org/pub/linux/utils/util-linux-
       ng/.
```

IL COMANDO SORT

La sintassi del comando

echo –[opzioni] [stringa]

Riepilogo opzioni del comando

IL COMANDO WC

La sintassi del comando

echo –[opzioni] [stringa]

Riepilogo opzioni del comando

IL COMANDO UNIQ

La sintassi del comando

echo –[opzioni] [stringa]

Riepilogo opzioni del comando

IL COMANDO TR

La sintassi del comando

echo –[opzioni] [stringa]

Riepilogo opzioni del comando

IL COMANDO GREP

La sintassi del comando

echo –[opzioni] [stringa]

Riepilogo opzioni del comando

IL COMANDO SED

La sintassi del comando

echo –[opzioni] [stringa]

Riepilogo opzioni del comando

LA CONCATENZAZIONE DELL'OUTPUT GRAZIE ALL'OPERATORE | (PIPE)

La sintassi dell'operatore Pipe