La shell di Linux

Elencare i contenuti di una directory

- `$ ls` elenca i contenuti della working directory.
- `$ ls dir_name` elenca i contenuti di `dir_name.`

- ls: opzioni

-a	tutti i files, compresi quelli nascosti
-F	aggiunge / per le directories, * per gli eseguibili, @ per i link simbolici
-l	formato completo – dettagli per i files
-m	elenca i files come se fossero un testo scritto, separandoli con la virgola
-r	inverte l'ordine alfabetico
-R	ricorsivo; comprende le sottodirectories
-s	dimensioni dei files in blocchi
-t	elenca secondo la data dell'ultima modifica
-u	elenca secondo la data dell'ultimo accesso
-i	inode di ciascun file

Creare e cancellare directories

- `$ mkdir dir_name` (crea una directory).
  - `$ mkdir appunti ; ls`
  - `$ mkdir {appunti,lucidi}; ls`
- `$ rmdir dir_name` (cancella directories vuote. Attenzione: no warning!).
  - `$ rmdir appunti`
- Se la directory non è vuota, per cancellarla si deve usare `$ rm -r dir_name`
  - `$ rm -r appunti`

Creare e cancellare files

- `$ cat > file_name` (scrive l'input proveniente dalla tastiera nel file `file_name`).
  - `$ cat > prova`

    `Help! I'm stuck in a Linux program!`

    ^D

    (Il testo in rosso indica ciò che scrive l'utente).
- `$ rm file_name` (cancella il file. Attenzione: no warning!).
  - `$ rm pippo` (Cancella definitivamente il file `pippo`. No warning!).
  - rm: opzioni:

    `-r` (*recursive)* rimuove i contenuti delle directories uno dopo l'altro

    `-i` (*interactive*) avvisa prima di cancellare ciascun file

    `-f` (*force*) forza rm a cancellare i files ignorando errori o avvertimenti

Copiare

- `$ cp options file1 file2` (per copiare il file file1 nel file file2).
  - `$ cp /etc/passwd pass`
  - `$ cp problemi/*  ~/backup`
- Se il file file2 non esiste, allora cp lo crea; altrimenti cp lo sovrascrive.
- Se file2 è una directory, cp fa una copia del file1 nella directory file2.
  - `$ cp pippo /articoli`
  - `$ cp /etc/passwd .`
- cp: opzioni
  - `-i` avvisa prima di sovrascrivere su un file esistente
  - `-p` conserva i permessi
  - `-r` copia i files e le sottodirectories uno dopo l'altro

Spostare

- `$ mv olddirectory newdirectory` (rinomina la directory olddirectory in newdirectory).
  - Se `newdirectory` esiste già, mv sposta `olddirectory` dentro quella nuova.
- `$ mv oldname newname` (rinomina il file oldname in newname).
  - Se `newname` esiste già, mv scrive `oldname` su `newname`.
- mv: opzioni
  - `-i` avvisa prima di sovrascrivere su un file esistente
  - `-f` forza mv ad agire indipendentemente dai permessi (non sempre…)
- `$ mv file path` (sposta il file `file` dalla current directory alla nuova directory, indicata in `path`).
  - `$ mv chap[1,3,7] book` (sposta i files chap1, chap3, e chap7 nella directory book).
  - `$ mv chap[1-5] book` (sposta i files da chap1 a chap5 nella directory book).

Mostrare i contenuti di un file (cat)

- Sintassi: cat [OPZIONE...] [FILE...]
- cat concatena i file indicati e li scrive sullo standard output.
  - Se vengono specificati dei file, cat considera questi; altrimenti viene preso lo standard input.

- `$ cat filename` (mostra i contenuti di filename).
- cat: opzioni
  - `-n` precede ogni linea con un numero , le righe vengono numerate
  - `-v` visualizza i caratteri non stampabili (--show nonprinting)
  - `-e` visualizza $ alla fine di ogni riga (--show-ends)

Mostrare i contenuti di un file (cat)

- Può svolgere una funzione di concatenazione di file in uno solo (da cui il nome cat) :
  - $ cat file1.txt file2.txt file3.txt

    Concatena i file nell'ordine in cui vengono proposti e li visualizza

Mostrare i contenuti di un file (**more**)

- Sintassi: more [OPZIONE...] [NOME_FILE...]
- more riporta in uscita gli ingressi specificati una schermata alla volta.
  - `$ more filename` (mostra la prima schermata di filename)

- more:opzioni:

-c	mostra le schermate successive dall'alto della pagina;
-s	sostituisce linee vuote consecutive con un'unica linea vuota
+/pattern	mostra il file a partire dalla prima occorrenza di "pattern";

  - `$ more [-cs] [+startline] [+/pattern] [filename]`

`startline:` numero di linea da cui si vuole iniziare a visualizzare;
`pattern:` pattern iniziale da cercare;

Mostrare i contenuti di un file (`more`)

- <space>: mostra la schermata successiva;
- <invio>: visualizza la riga di testo successiva
- si chiude automaticamente dopo l'ultima schermata
- h: informazioni aggiuntive;
- q: esce dal programma;
- v: apre il file con l'editor vi;
- /pattern Cerca il pattern indicato all'interno del testo
- :n Salta al file successivo (quando si visualizzano più file
- :p Salta al file precedente

Mostrare i contenuti di un file (`less`)

- Sintassi: less [OPZIONE...] [NOME_FILE...]
- less riporta in output i file specificati una schermata alla volta.
  - utilizzato quando si vuole visionare velocemente un lungo file di testo oppure in abbinamento con altri comandi, quando essi forniscono in uscita informazioni che non hanno spazio su di una singola schermata.
- `$ less filename` (mostra la prima schermata di filename)
- less: opzioni

  -o copia l'output sul file specificato nel caso in cui l'input provenga da una **pipe**.

  -p mostra il file a partire dalla prima occorrenza di "pattern";

Mostrare i contenuti di un file (`less`)

- <space>: mostra la schermata successiva;
- <invio> : visualizza la riga di testo successiva
- frecce direzionali per spostarsi su e giù di una linea per volta
- q o Q per uscire dal comando less
- /pattern Cerca il pattern indicato all'interno del testo
- :n Salta al file successivo (quando si visualizzano più file)
- :p Salta al file precedente

Mostrare i contenuti di un file (**`head`**)

- Sintassi: head [OPZIONE...] [FILE...]
- head fornisce la parte iniziale dei file in ingresso.
  - Se non viene specificato altrimenti l'ingresso viene considerato semplice testo e ne vengono date le prime 10 righe.

- `$ head -n filename` (mostra le prime n linee di filename. Se n non è specificato, di default è 10).

Mostrare i contenuti di un file (`tail`)

- Sintassi: tail [OPZIONE...] [FILE...]
- tail fornisce la parte finale dei file in ingresso.
  - Se non viene specificato altrimenti l'ingresso viene considerato semplice testo e ne vengono date le ultime 10 righe.

- `$tail -n filename` (mostra le ultime n linee di filename. Se n non è specificato, di default è 10).

Il comando man

[lferrari@homelinux lferrari]$ man cat

CAT(1) User Commands CAT(1)

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 nonblank output lines

-e equivalent to -vE

-E, --show-ends
display $ at end of each line

-n, --number
number all output lines

-s, --squeeze-blank
never more than one single blank line

-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.

AUTHOR
Written by Torbjorn Granlund and Richard M. Stallman.

REPORTING BUGS
Report bugs to <bug-textutils@gnu.org>.

COPYRIGHT

SEE ALSO

The full documentation for cat is maintained as a Texinfo manual. If the info and cat programs are properly installed at your site, the command *info cat* should give you access to the complete manual.

Ricerca dei files (**find**)

- L'utility find è un comando molto versatile, senza corrispondente in DOS. Si può usare find per cercare files usando una grande varietà di condizioni di ricerca, quindi eseguire svariate azioni coi risultati della ricerca. La sintassi è:
  - `$ find <path> <search-condition(s)> <action>`
- L'utility find:
  - Discende ricorsivamente attraverso il path,
  - Applica le condizioni di ricerca a ogni file.
- Condizioni di ricerca:

  -atime n files visitati n giorni fa

  -mtime n files modificati n giorni fa

  -size n[bckw] files di dimensione esattamente n (n può essere data in blocchi da 512 byte [b], caratteri da 1 byte [c], Kilobytes [k] o parole da due bytes[w])

  -type c tipo di file (ad es., f=file, d=directory, l=link)

  -name "name" trova tutti i files di nome "name" (es., "*.c")

Ricerca dei files (**find**)

- Come troviamo un file chiamato **kernel** in **/usr/src**?

```
$ find /usr/src -name "kernel"
```

- Come troviamo i files con estensione pdf?

```
$ find / -name "*.pdf"
```

- Come facciamo a trovare quali files sono stati modificati nella nostra home negli ultimi due giorni?

```
$ find ~ -mtime -2
```

- Come facciamo a trovare i files della nostra home visitati nelle ultime due ore?

```
$ find ~ -amin -120
```

- Come troviamo files di dimensione superiore a 1 MB, ma inferiore a 2 MB?

```
$ find / -size +1024 -size -2048
```

- Come troviamo i files nella current directory più recenti del file "test"?

```
$ find . -newer test
```

Ricerca dei files (**<action>**)

- <action>:
  - -exec *command [options]* { } \;

  esegue il comando *command* usando come input il file trovato; { } rappresenta il percorso del file trovato, mentre \; termina la linea di comando.
  - *-ok command [options]* { } \;

  come –exec ma richiede conferma.
  - -print

  mostra i files trovati sullo schermo; non è necessario se non dopo un –exec, oppure se si vuole redirezionare l'output.

- Come facciamo a trovare i files con estensione tmp e poi cancellarli con richiesta di conferma?

```
$ find / -name "*.tmp" -ok rm {} \;
```

- Come facciamo a cancellare tutti i files più vecchi di un mese (modificati più di un mese fa) nella directory /tmp?

```
$ find /tmp -mtime +30 -exec rm {} \;
```

- Come facciamo a trovare i files dell'utente lferrari (che hanno cioè l'utente lferrari come owner)?

```
$ find / -user lferrari
```

- Come facciamo a trovare tutti i files che cominciano con "pr" nella current directory la cui dimensione è minore di 10Kbyte, quindi registrare l'output nel file "risultato"?

```
$ find . -name "pr*" -maxdepth 1 -size -10k
  -print > risultato
```

Filtrare i files (**grep**)

- grep sta per get regular expression. Si può usare grep quando cerchiamo files contenenti un motivo specifico. grep è stato ulteriormente esteso con comandi come egrep e fgrep. La sintassi è:
  - `$ grep <options> <search-pattern> <file(s)>`
- Grep mostra le linee
  - che contengono il motivo `<search-pattern>`
  - in ciascuno dei files `<file(s)>`.
- Esempio:

```
$ grep Italy /usr/src/linux/CREDITS
$ grep pippo ~/prova.txt
```

Opzioni (**grep**)

- -n visualizza anche il numero di riga

```
$ grep –n italy /usr/src/linux/CREDITS
```

- -c mostra il numero di linee in cui la sequenza viene trovata, ma non le linee stesse

```
$ grep –c Italy /usr/src/linux/CREDITS
```

- -i non distingue tra maiuscole e minuscole

```
$ grep –i Italy /usr/src/linux/CREDITS
```

- -w trova solo parole intere

```
$ grep –w Italy /usr/src/linux/CREDITS
```

- -q dà 0, se il testo è stato trovato, e 1 altrimenti

```
$ grep –q Italy /usr/src/linux/CREDITS
```

- -l visualizza solo i nomi dei file che contengono le righe corrispondenti all'espressione regolare, ma non le righe stesse

```
$ grep –l Italy /usr/src/linux/CREDITS
```

Filtrare i files (**sort**)

- sort
  - È un filtro che ordina un *flusso di testo*, o un file, in senso crescente o decrescente, o secondo le diverse interpretazioni o posizioni dei caratteri.
  - `$ sort file` ordina alfanumericamente le linee del *file* e le mostra
  - `$ sort -r file` (lo stesso, ma in ordine inverso)
  - sort:opzioni
    - -u elimina dal risultato le linee duplicate.
    - -f non distingue tra caratteri maiuscoli e minuscoli.
    - -r inverte il senso di ordinamento.
    - -n ordina numericamente .
    - -k *chiave_di_ordinamento* Indica una porzione della linea da usare come chiave per l'ordinamento.

Filtrare i files (**uniq**)

- uniq
  - È un filtro che elimina le righe duplicate di un file che è stato ordinato. È spesso usato in coppia con sort.
    - `$ uniq file`
  - uniq:opzioni

    -c Precede ogni linea con un conteggio del numero di volte consecutive in cui è ripetuta.

    -d Scarta le linee che non fanno parte di gruppi di linee identiche consecutive.

Filtrare i files (**wc**)

- wc
  - Fornisce il "numero di parole (word count)"presenti in un file o in un flusso I/O
    - `$ wc file`
  - wc:opzioni
    - -w fornisce solo il numero delle parole.
    - -l fornisce solo il numero di righe.
    - -c fornisce solo il numero dei byte.
    - -m fornisce solo il numero dei caratteri.
    - -L fornisce solo la dimensione della riga più lunga.

Altri comandi Utili

- history
  - permette di visualizzare l'elenco completo dei comandi che abbiamo digitato in precedenza
- ps
  - **p**rocess **s**tate ci mostra lo stato dei processi in corso all'interno del sistema
- kill
  - Termina un processo in base al suo PID o gli invia un segnale a piacimento
- wget
  - utility gratuita per il download non interattivo di file da Internet , supporta il download via HTTP/S,FTP

Standard Input e Standard Output

- Ogni programma eseguito dalla shell apre tre files:
  - standard input ← 0
  - standard output ← 1
  - standard error ← 2
- I files forniscono i principali mezzi di comunicazione tra i programmi, e rimangono in vita per tutta la durata del processo.
- Il file standard input fornisce un modo per inviare dati a un processo. Di default, lo standard input viene letto dalla *tastiera del terminale*.
- Lo standard output fornisce al programma un mezzo per rendere disponibili i dati. Di default, lo standard output viene visualizzato sullo *schermo del terminale*.
- Lo standard error è dove il programma registra ogni eventuale errore incontrato durante l'esecuzione. Di default, anche lo standard error viene indirizzato sullo *schermo del terminale.*

Standard Input e Standard Output

- 1>pippo invia lo standard output al file pippo.
- 2>pippo invia lo standard error al file pippo.
- Es. $ ls 1> pippo scrive l'output di ls in pippo,
  come ls > pippo
  $ list 2> pippo scrive l'errore
  "-bash: list: command not found" nel file pippo.

Redirezionare Input e Output

- È possibile dire a un programma:
  - dove cercare l'input
  - dove inviare l'output,

  usando la tecnica di redirezione dell'input/output. In UNIX si usano i caratteri speciali **<** e **>** per significare redirezionamento di input e output, rispettivamente.
- Redirezionare l'input: usando **<** con il nome di un file (as es., < file1) in un comando di shell, si comunica alla shell di leggere l'input da un file chiamato "file1" invece che dalla tastiera.
  - `$ more < /etc/passwd`
- Redirezionare l'output: usando **>** con il nome di un file (per es., > file 2), si impone alla shell di memorizzare l'output del comando in un file chiamato "file2" invece che sullo schermo. Se il file "file2" esiste già, la versione vecchia verrà sovrascritta.
  - `$ ls /tmp > ~/ls.out`
  - `$ sort pippo > pippo.ordinato`

Redirezionare Input e Output

- L'uso di **>>** per completare un file esistente (per es., >> file2) impone alla shell di accodare l'output del comando alla fine di un file chiamato "file2".
  - Se il file "file2" non esiste già, verrà creato.
- Esempio

1. `$ ls /bin  > ~/bin;  wc -l ~/bin`

   `$ ls /usr/sbin > ~/bin ; wc -l ~/bin`

2. `$ ls /bin  > ~/bin;`

   `$ ls /usr/sbin >> ~/bin; wc -l ~/bin`

Redirezionare l'errore

- L'uso di **>&** con il nome di un file (ad es., >& file1) impone alla shell di inserire lo standard error e lo standard output del comando in un file detto "file1".
  - Se il file "file1" esiste già, la versione vecchia verrà sovrascritta.
- Esempio
  - `$ ls abcdef`
  - `$ ls abcdef >& lserror`
  - `cat lserror`
  - `$ abcdef >& command`
  - `cat command`
  - `$ mkdir /bin/miei >& ~/miei; cat ~/miei`
  - `$ rm /bin/perl >& ~/errperl; cat ~/errperl`

Pipes (tubi)

- UNIX offre la possibilità di connettere processi, permettendo allo standard output di un processo di venire usato come standard input di un altro processo. Questo meccanismo viene detto una **pipe (|)**.
- `$ command1 | command2` fa sì che lo standard output di `command1` venga utilizzato come standard input di `command2`.
- Una sequenza di comandi concatenati in questo modo viene detto una pipeline. Connettere processi semplici in una pipeline permette di eseguire compiti complessi senza scrivere programmi troppo complessi.
  - `$ cat /etc/passwd | sort > ~/pass_ord`
  - `$ sort < pippo | lpr`

Esercizi

1. Determinare il numero di files nella directory /bin la cui prima lettera è “c”.
2. Creare un file contenente i nomi dei primi 7 files della directory /etc.
3. Determinare il numero dei files della current directory nel cui nome compare la stringa *“string”.*
4. Creare un file contenente una lista col nome di 10 comandi di /bin ordinati secondo il momento dell’ultimo accesso.
5. Creare un file contenente i nomi dei primi 7 files e gli ultimi 6 files (in ordine alfabetico) della directory /etc.
6. Creare un file contenente una lista coi nomi di 8 files in /usr/sbin ordinati secondo il momento dell’ultima modifica.