Programmazione e Architetture degli Elaboratori - Soluzioni Foglio 3

Luca Manzoni, Michele Rispoli, Pietro Morichetti

Esercizio 01

Si utilizzi il comando `man`, sulla shell Unix, per rispondere alle seguenti domande:

1. Lanciare `man` per un qualsiasi comando (e.g. `man ls`) e consultare l'help di navigazione (i.e. premi `h` da qualsiasi manpage). A cosa serve il comando `/<pattern>`? Ed i comandi `n` ed `N`?
2. Consultare la manpage del comando `ls`. A cosa serve l'opzione `-R`?
3. Consultare la manpage del comando `wc`. A cosa serve l'opzione `-L`?
4. Consultare la manpage del comando `touch`. A cosa serve l'opzione `-d`? Fai un esempio con cui si potrebbe usare questa opzione.
5. Consultare la manpage del comando `more`. A cosa serve? Esiste un comando alternativo che svolge la stessa funzione meglio?
6. Consultare la manpage del comando `cp`. A cosa serve l'opzione `-R`?

Esercizio 02

Utilizzare il set di comandi, fornito qui di seguito, per esplorare la struttura delle directory del file system (FS): `cat cd head less ls tail which`; ricordate che potete consultare le manpage dei comandi per capire come funzionano e quali opzioni accettano.

Note:

Il comando `which` è usato per localizzare la posizione di un dato file eseguibile (a.k.a. programma) passato come argomento presente in uno dei percorsi specificati nel `PATH`; il comando stampa a schermo la posizione dell'eseguibile specificato. Per ulteriori dettagli si consiglia di visitare la manpage del comando.

1. Aprire un terminale, navigare al root del FS (`/`) e visualizzare tutti gli elementi contenuti in esso.

   Sol:

   ```
   cd /
   ls
   ```

2. Navigare fino alla home del proprio user (path "∼") e visualizzare tutti gli elementi contenuti in essa, compresi quelli nascosti.

   Sol:

   ```
   cd ~
   ls -la
   ```

3. Visualizzare sul terminale il contenuto del file *.bashrc* (nota: è un file nascosto) che si trova nella propria home directory, prima con `cat` e poi con `less`.

   Sol:

   ```
   cat ~/.bashrc
   less ~/.bashrc
   ```

4. Visualizzare l'output di `ls -lR` sulla directory */home* con `less` (usate il pipe "|").

   Sol:

   ```
   ls -lR /home | less
   ```

5. Visualizzare le prime 7 e le ultime 7 righe del file *bash.bashrc* che si trova all'interno della sottodirectory *etc* del root.

   Sol:

   ```
   head -n 7 /etc/bash.bashrc
   tail -n 7 /etc/bash.bashrc
   ```

6. **Non abbiamo visto which a lezione** Individuare la posizione sul disco del programma `less` e visualizzare il contenuto della directory che lo contiene con `less` (hint: possiamo ridirezionare l'output di `which` nell'input di `ls`...)

   Sol:

```
which less | ls -l | less
```

Esercizio 03

Utilizzare il set di comandi, fornito qui di seguito, per la creazione e manipolazione di file e/o directory: `cp mkdir mv rm rmdir touch`, e precedenti.

1. Crea la directory *Tutorato_Programmazione* nella home.

   Sol:

   ```
   mkdir ~/Tutorato_Programmazione
   ```

2. Crea un file vuoto chiamato *es_1.txt* nella directory Tutorato_Programmazione.

   Sol:

   ```
   touch ~/Tutorato_Programmazione/es_1.txt
   ```

3. Crea una copia di es_1.txt con nome *copia_es_1.txt*, nella directory Tutorato_Programmazione ed elimina es_1.txt

   Sol:

   ```
   cd ~/Tutorato_Programmazione
   cp es_1.txt copia_es_1.txt
   rm es_1.txt
   ```

4. Crea una sotto-directory di Tutorato_Programmazione con il nome di *Esercizi*, dove sposterai il file copia_es_1.txt una volta rinominato in *es_1.txt*. Disegnare il *Tree-Folder* considerando la home come root.

   Sol:

   ```
   cd ~/Tutorato_Programmazione
   mv ./copia_es_1.txt ./es_1.txt
   mkdir Esercizi
   mv ./es_1.txt ./Esercizi/es_1.txt
   tree ~
   ```

5. Eseguire una “copia profonda” (aka Deep-Copy), ovvero una copia della directory e di tutti i file in essa contenuti della directory Tutorato_Programmazione con nome *Copia_Tutorato_Programmazione* nella home.

   Sol:

   ```
   cd ~
   cp -R Tutorato_Programmazione Copia_Tutorato_Programmazione
   ```

6. Eliminare la directory Copia_Tutorato_Programmazione_Esercizi in maniera ricorsiva (ossia eliminando tutti i file e sotto-directory presenti nella directory Esercizi).

   Sol:

   ```
   cd ~
   rm -R Copia_Tutorato_Programmazione
   ```

Esercizio 04

Utilizzare `grep` per svolgere i seguenti esercizi:

1. Stampare a schermo tutte le linee del file */.bashrc* che contengono il termine ”alias”, corredate di numero di riga.

   Sol:

   ```
   grep alias -n ~/.bashrc
   ```

   Nota: L’output del comando potrebbe variare sulla vostra macchina e potrebbe anche essere vuoto.

2. Stampare a schermo tutte le linee del file */.bashrc* che contengono il termine “uncomment” senza badare al case (i.e. sia maiuscolo che minuscolo, dunque “uncomment” o “Uncomment” o “uNCommEnT” vengono matchati)

   Sol:

   ```
   grep uncomment -i ~/.bashrc
   ```

3. Stampare a schermo tutte le linee del file */.bashrc* che contengono (almeno) un numero (hint: `man grep`, sezione "REGULAR EXPRESSIONS" > "Character Classes and Bracket Expressions")

   Sol:

   ```
   grep "[0-9]" ~/.bashrc
   ```

4. Stampare a schermo tutte le linee del file */.bash_history* che cominciano per "c" (hint: `man grep`, sezione "REGULAR EXPRESSIONS" > "Anchoring")

   Sol:

   ```
   grep "^c" ~/.bash_history
   ```

Esercizio 05

Utilizzare i comandi `cut sed vim wc` (in aggiunta a quelli visti in precedenza) per risolvere i quesiti.

Note:

- `vim` è un editor di testo per linea di comando estremamente potente, ma anche noto per la sua curva d'apprendimento estremamente ripida; è tipicamente presente di default su linux, mentre se usate Cygwin potreste doverlo selezionare esplicitamente durante la procedura d'installazione (o aggiungerlo dopo, rilanciando il setup). Qui trovate una breve lista di alcuni comandi di base, sufficienti a svolgere i compiti che seguono, ma il modo migliore di apprendere le basi per l'utilizzo di vim è seguire il `vimtutor` (inserite il comando in una shell e seguite il tutorial).

- `sed` è un programma che prende in input un testo (eg. un file o l'output di un altro comando, che gli possiamo passare con il pipe "|") e permette di sostituire ogni occorrenza di una data stringa (o regex) con un'altra stringa (o regex). Anche `sed` è molto potente e flessibile, ma per risolvere gli esercizi proposti vi basta sapere che con il comando

  ```
  sed "s/match/substitution/g" filename
  ```

  ogni occorrenza di `match` in *filename* viene sostituita con `substitution` ed il testo modificato viene restituito in output (e non salvato in *filename*!). Ad esempio

```
$ echo "Ciao mondo!" | sed "s/o/!O!/g"
Cia!O! m!O!nd!O!!
```

Per maggiori informazioni potete consultare la manpage di `sed`.

1. Creare un file vuoto chiamato *text1.txt*, aprirlo con Vim, scriverci "Caro diario, oggi è un gran giorno poiché sto imparando ad usare la linea di comando della Shell Unix. Non dimenticherò mai gli insegnamenti del professore e dei tutors!", salvare il file, chiudere Vim, e controllare che il testo sia correttamente salvato nel file con `cat`.

2. Aprite Vim su un nuovo file, chiamato *text2.txt*, e riportare il seguente al suo interno (considerando anche i ritorni a capo):

   ```
   12345:Administration:james:smith
   67891:Staff:Stephan:York
   18230:Staff:Luke:Cutwine
   62913:Sales:Red:Blues
   ```

   salvatelo ed uscite da Vim. A questo punto, stampate su terminale solo i caratteri dal 1° a posizione 5 di ogni linea di text2.txt.

   Sol:

   ```
   vim text2.txt
   ...
   cut -c1-5 text2.txt
   ```

3. Determinare quanti caratteri sono contenuti nel file text2.txt, senza aprire Vim.

   Sol:

   ```
   wc -c ./text2.txt
   ```

4. Visualizzare sul terminale il contenuto di text2.txt, sostituendo però ogni occorrenza di ':' con uno spazio di tabulazione (i.e. '\t'), senza modificare il file e senza utilizzare Vim.

   Sol:

```
sed 's/:/  /g' text2.txt
```

5. Stampare a schermo nome e cognome delle entry in text2.txt corrispondenti alla categoria "Staff" senza modificare il file.

   Sol:

   ```
   cat text2.txt | grep 'Staff' | cut -d':' -f1,4
   ```

6. (Extra, per i PRO) Stampare a schermo esclusivamente il mese, la data ed il nome dei file presenti nella directory corrente (non importa quale). *Hint: potrebbe essere necessario manipolare la spaziatura dell'output di* `ls` *con* `sed` *per poter estrarre i campi richiesti.*

   Sol:

   ```
   ls -l | sed 's/ \+/\t/g' | cut -f6,7,9
   ```

Esercizio 06

Realizzare uno script in bash, chiamato *script.sh* che riproduca tutti i punti dell'Esercizio 03; verificarne poi la correttezza eseguendolo su terminale.

Nota: Non dimenticatevi di configurare correttamente i permessi di esecuzione dello script con `chmod` e includere l'intestazione corretta dello script, come visto durante la lezione 9.

Sol:

1. Aprire un file .sh in Vim, ad esempio con nome *script.sh*;
2. Riportare il seguente codice:

```
#! bin/bash

mkdir ~/Tutorato_Programmazione
touch ~/Tutorato_Programmazione/es_1.txt

cd ~/Tutorato_Programmazione
cp es_1.txt copia_es_1.txt
rm es_1.txt

mv copia_es_1.txt es_1.txt
mkdir Esercizi
```

```
mv es_1.txt Esercizi/es_1.txt

cp -R Tutorato_Programmazione Copia_Tutorato_Programmazione

rm -R Copia_Tutorato_Programmazione
```

3. Salvare e chiudere Vim;
4. Modificare i permessi di script.sh attraverso il seguente comando: `chmod 766 script.sh` (la sequenza numerica è di esempio, l'importante è che si renda il file eseguibile per l'utente);
5. Eseguire lo script digitando su linea di comando: `./script.sh`.