Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni degli esercizi del 14 Marzo 2012

Esercizio 1. *Creare un file di nome* `prova.txt` *contenente alcune righe di testo. I seguenti comandi che effetto producono? Perché?*

1. `cat < prova.txt > prova2.txt`
   Crea un file di nome `prova2.txt` con lo stesso contenuto di `prova.txt`
2. `cat < prova.txt >> prova2.txt`
   Aggiunge il contenuto di `prova.txt` alla fine di `prova2.txt`. Il file `prova2.txt` contiene quindi il testo di `prova.txt` ripetuto due volte in sequenza.
3. `chmod u-w prova2.txt`
   Toglie il permesso di scrittura per il proprietario di `prova2.txt`
4. `cat < prova.txt > prova2.txt`
   Si ottiene il messaggio d'errore `prova2.txt: Permission denied`, perché non si possiede il permesso di scrittura sul file `prova2.txt`. Il suo contenuto rimane invariato
5. `chmod 000 prova.txt`
   Toglie tutti i permessi al file `prova.txt`
6. `cat prova.txt`
   Si ottiene il messaggio d'errore `prova.txt: Permission denied`, perché non si possiede il permesso di lettura sul file `prova.txt`.
7. `chmod 644 prova.txt`
   Assegna i permessi di lettura/scrittura al proprietario del file `prova.txt`, solo lettura per tutti gli altri.
8. `cat < prova.txt > prova.txt`
   L'effetto del comando è quello di cancellare il contenuto del file `prova.txt`. La ridirezione dello standard output su `prova.txt` fa in modo che il contenuto di quest'ultimo venga cancellato ancor prima che venga letto e processato dal comando `cat`.

Esercizio 2. *Eseguire i seguenti comandi. Che effetto producono? Perché?*

1. `cd`
2. `mkdir d1`
   crea la directory d1 nella directory corrente (a meno che non esista già una directory con tale nome);
3. `chmod 444 d1`
   imposta i permessi di `d1` a `r--r--r--`;
4. `cd d1`
   fallisce provocando la visualizzazione del messaggio di errore `d1: Permission denied`. Per "entrare" in una directory è necessario possedere il permesso di esecuzione su di essa.

Esercizio 3. *Usare il comando* `find` *per:*

1. *elencare tutte le sottodirectory contenute nella propria home.*
   ```
   $ cd ; find . -type d
   ```
2. *trovare tutti i file contenuti nella propria home che sono stati modificati meno di due giorni fa.*
   ```
   $ cd ; find . -mtime -2
   ```

Esercizio 4. *Spostarsi nella cartella* `/usr/include`*.*

```
$ cd /usr/include
```

Usare il comando `grep` *per:*

1. *elencare tutti i file che contengono la stringa* `ioctl.h`
   ```
   $ grep -l ioctl.h *
   ```

2. ~~*elencare tutti i file che **non** contengono la stringa* `ioctl.h`~~
3. *visualizzare tutte le occorrenze della parola singola* `scanf`
   `$ grep -w scanf *`

Esercizio 5. *Qual è la differenza tra i seguenti comandi?*

1. `ls`
   fornisce la lista dei file contenuti nella directory corrente (ordinati alfabeticamente);
2. `ls | cat`
   fornisce la lista dei file contenuti nella directory corrente (ordinati alfabeticamente) in modo che ogni linea contenga il nome di un solo file;
3. `ls | more`
   fornisce la lista dei file contenuti nella directory corrente (ordinati alfabeticamente) in modo che ogni linea contenga il nome di un solo file e organizzando l'output su schermo in pagine.

Esercizio 6. *Quale effetto producono i seguenti comandi?*

1. `ls | wc -w`
   Fornisce il numero di file e directory contenuti nella cartella corrente
2. `who | wc -l`
   Il comando `who` elenca gli utenti collegati al sistema, mostrandoli uno per riga. Il comando fornisce quindi il numero di utenti collegati al sistema. Si noti che lo stesso utente può risultare collegato più volte nello stesso momento: tipicamente, il primo collegamento corrisponde all'interfaccia grafica, gli altri alle varie finestre di terminale che sono state aperte.

Esercizio 7. *Scrivete un unico comando (pipeline) per:*

1. *fornire il numero di file (e directory) contenuti ricorsivamente nella propria home (si può utilizzare* `ls -R`*? ed il comando* `find`*?)*

   L'idea di base per risolvere l'esercizio è quella di produrre una lista dei file e directory contenuti ricorsivamente nella propria home, contando le linee in output (supponendo che ogni linea corrisponda ad un file/directory). Utilizzando `ls -R` non si ottiene il conteggio esatto in quanto tale comando produce anche delle linee vuote ed altre di intestazione, che falsano il computo.
   La pipeline corretta è quindi la seguente:
   `$ find . 2>/dev/null | wc -l`
   Si noti che è opportuno ridirigere lo standard error per non conteggiare eventuali linee con messaggi d'errore (dovuti alla mancanza dei permessi necessari per l'accesso a qualche file/directory).

2. *elencare i file (e directory) contenuti ricorsivamente nella propria home il cui percorso relativo **non** contiene la lettera "a" o "A"*

   Anche il questo caso per risolvere l'esercizio si deve prima produrre la lista dei file e directory contenuti nella propria home con il comando `find`. Si noti che il comando restituisce il percorso relativo di ogni file. Quindi si può utilizzare `grep` per filtrare l'elenco eliminando dalla lista le righe dove appare una "a" (maiuscola o minuscola):
   `$ find . 2>/dev/null | grep -iv a`

3. *visualizzare solamente la quinta riga del file* `/usr/share/vim/vimrc`

   Per risolvere l'esercizio è necessario utilizzare i filtri `head` e `tail` in sequenza. Con `head` si ottengono le prime 5 righe del file, per poi visualizzare solamente l'ultima usando `tail`:
   `$ head -n 5 /usr/share/vim/vimrc | tail -n 1`

Esercizio 8.

1. *Creare un file di nome* `script.sh` *con il seguente contenuto:*

```
#!/bin/bash
cd /bin
ls -l
```

Usare un editor di testo come `gedit`, o `cat > script.sh`

2. *descrivere l'effetto dei comandi `./script.sh` e `source script.sh`*

   Nel primo caso si ottiene un messaggio d'errore perché non si possiede il permesso di esecuzione sul file. Nel caso di `source`, i comandi contenuti nello script vengono eseguiti. L'effetto è quello di produrre la lista di tutti i file contenuti in `/bin`, completa di tutti gli attributi. Notare che al termine ci si è spostati nella directory `/bin`.

3. *rendere il file `script.sh` eseguibile da parte di tutti*
   `$ chmod a+x script.sh`, oppure
   `$ chmod 755 script.sh`

4. *provare nuovamente l'effetto dei comandi `./script.sh` e `source script.sh`, e descrivere le differenze con il punto precedente.*

   In questo caso il comando `./script.sh` esegue lo script perché ora si possiede il permesso di esecuzione. Per entrambi i comandi l'effetto è quello di produrre la lista di tutti i file contenuti in `/bin`, completa di tutti gli attributi. Si noti che nel caso di `source`, al termine dell'esecuzione ci si è spostati nella directory `/bin`, mentre nel caso di `./script.sh` si rimane nella directory dove si trova lo script.