Sesta Esercitazione

File
comandi
Unix

Passi di sostituzione: esempi

```
echo `pwd` > "f1"
```

R: crea il file di nome **f1**, poi collega stdout di echo a **f1**

1: espande i backquote `` ➔ **echo /usr/bin**

2: sostituisce variabili: nessuna operazione di parsing

3: sostituisce metacaratteri: nessuna operazione di parsing

```
test -f `pwd`/$2 -a -d "$HOME/docs"
```

R: redirezione IO: nessuna operazione di parsing

1: **test -f /temp/$2 -a -d "$HOME/docs"**

2: **test -f /temp/pluto -a -d "/home/utente/docs"**

3: sostituisce metacaratteri: nessuna operazione di parsing

Esempio di file comandi

scrivere un file comandi che ogni 5 secondi controlli se sono stati ***creati o eliminati file in una directory***.

- In caso di cambiamento, si deve **visualizzare un messaggio su stdout** (quanti file sono presenti nella directory)
- il file comandi deve poter essere invocato con ***uno e un solo parametro***, la directory da porre sotto osservazione (fare opportuno controllo dei parametri)

Suggerimento: uso di un file temporaneo, in cui tenere traccia del numero di file presenti al controllo precedente

Esempio di file comandi

```
#!/bin/bash
if [ $# -ne 1 ] ; then echo Sintassi! ; exit; fi
echo 0 > loop.$$.tmp
OK=0
while [ $OK -lt 10 ]
do
   new=`ls "$1"|wc -w`
   old=`cat loop.$$.tmp`
   if [ $new -ne $old ]
   then
      echo $new > loop.$$.tmp
      echo in $1 ci sono $new file
   fi
   OK=`expr $OK + 1`
   sleep 5s
done
```

numero di parametri, $0 escluso

pid del processo in esecuzione

[] sono equivalenti a **test**
Gli spazi sono significativi!

"" evitano problemi in caso di parametro **$1** con spazi

i nomi di file in **$1** potrebbero contenere spazi. Meglio:

```
new=`ls -l "$1"|wc -l`
new=`expr $new - 1`
```

Esercizio 1

- Creare uno script che abbia la sintassi

`./save_dir_structure`

Lo script deve:

- richiedere all'utente e leggere da standard input il path assoluto di una directory.
- controllare che si tratti effettivamente di un path assoluto e di una directory
- utilizzare un'opportuna opzione di **`ls`** per scandire ricorsivamente tutto il contenuto della cartella.
- stampare l'elenco così ottenuto su un file dentro la HOME directory dell'utente che ha invocato lo script.
- il nome del file di output deve avere la seguente forma: **dir_structure_<uname>.out**

  dove <uname> è il nome dello USER che ha invocato lo script

Esercizio 2

- Realizzare un file comandi che preveda la seguente sintassi:

```
toplines D
```

dove **D** è una directory. Il file comandi deve:

- controllare il corretto passaggio dei parametri (un solo parametro corrispondente ad una dir esistente)
- scandire tutti e soli i file del direttorio **D**
- per ogni file contare il numero di caratteri
- individuare il file col maggior numero di caratteri e stamparne a video il nome.

Esercizio 2: suggerimenti

Ciclo su un elenco di file con path assoluto:

```
for file in /path/to/file1.txt
      /path/to/file2.txt /path/to/file3.txt; do
  # do something on $file
done
```

Ciclo su un elenco di file con path relativo alla dir corrente:

```
for file in file1.txt file2.txt file3.txt
```

Oppure usando il metacarattere `?`

```
for file in file?.txt
```

E se voglio tutti i file nella directory `D` passata come parametro quale **metacarattere** devo usare?

Altri suggerimenti

Provare i comandi a linea di comando prima di scriverli nello script bash!

posso provare i comandi semplici:

```
studente@debian:~$ grep -c file1.txt
```

ma anche i comandi più complessi come condizioni, if e cicli:

```
studente@debian:~$ if test -f pippo ; then echo yes ; else echo no; fi
studente@debian:~$ for fname in *; do echo $fname ; done
```

Esercizio 3

- Creare uno script che abbia la sintassi

`./occorrenze dir S [save_file]`

Dove: **`dir`** è una directory esistente, **`S`** è una stringa e **`[save_file]`** è il nome assoluto di un file di output che può essere passato o meno allo script.

Dopo aver controllato i parametri in ingresso, lo script deve:

- portarsi in **`dir`**
- per ciascun file in **`dir`** usare l'apposita opzione di grep per contare il numero ***N*** di linee con almeno un'occorrenza di **`S`** e comporre una stringa con formato

<nomefile> has <N> occurrences

- stamparla su **`[save_file]`** se presente. Altrimenti su standard output.
- riportarsi nella directory da cui era stato originariamente invocato lo script.