Esercitazione 6
Gruppo LZ

File comandi Unix

Esempio di file comandi

Scrivere un file comandi da invocare come segue:

```
./esempio D
```

dove D è il nome di una directory esistente.

Dopo un opportuno controllo sugli argomenti, lo script dovrà controllare ogni 5 secondi se sono stati **creati o eliminati file nella directory D:**

- In caso di cambiamento, si deve **visualizzare un messaggio su stdout** che comunichi quanti file sono presenti nella directory.

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

Esempio: soluzione

```
#!/bin/bash
if [ $# -ne 1 ] ; then echo Sintassi! ; exit; fi
if [  -d $1 ]; then echo $1 è una directory esistente
else  echo $1 non è una directory!; 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.$$.tmp
        echo in $1 ci sono $new file
      fi
      OK=`expr $OK + 1`
      sleep 5s
done
rm loop.$$.tmp
```

numero di parametri, **$0** escluso

pid del processo in esecuzione

"" 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

`./testdim F B`

dove **`F`** è una stringa e **`B`** è un intero positivo.

Lo script deve:

- **controllare** che siano stati passati due parametri
- **controllare** che **`F`** sia il **basename** di un file esistente e leggibile posizionato nella home directory dell'utente che ha invocato lo script.
- **controllare** che **`B`** sia un intero positivo

Qualora i controlli sui parametri in ingresso diano esito positivo, lo script deve controllare che la dimensione ***D*** del file **`F`** sia divisibile per **`B`** e, qualora questa condizione sia verificata, stampare a video il messaggio seguente:

Il file ***<absNameF>*** *ha dimensione <D>, divisibile per <**`B`**>*

dove ***<absNameF>*** è il nome assoluto di **`F`** e ***<D>*** è la sua dimensione

Esercizio 1: Suggerimenti (1/2)

Test **`F`** sia basename (i.e., non deve contenere nessun '/')

- Quale metacarattere per fare pattern matching?
- Serve un costrutto che supporti l'uso dei metacaratteri nel confronto, ovvero che impedisca l'espansione di tali metacaratteri coi nomi di file che fanno match (**`[[…]]`** oppure **`switch-case`**)

Test di file:

- **`test -f <path>`** Esistenza del file. Alternativa **`[ -f <path> ]`**
- **`test -d <path>`** Esistenza del direttorio
- **`test -r <path>`** Diritto di lettura (allo stesso modo, **`-w`** e **`-x`**)

Test di **`B`**:

- **`[[ ]]`** Più comodo di **`test`** per testare regular expressions

Esercizio 1: Suggerimenti (2/2)

`F` deve trovarsi nella **home directory** dell'utente che ha invocato lo script: utilizzare un'opportuna **variabile di ambiente**

Come reperire la dimensione *D* di un file? Due possibilità:

- Utilizzare **`ls -l`** e **`awk`** per filtrare solo la dimensione
- Utilizzare il comando **`stat`**: stampa lo "stato" del file (**`man stat`** per individuare come stampare la dimensione)

Come verificare se *D* è divisibile per **`B`**? Servirà effettuare un'opportuna operazione (**`man expr`**)

Altri suggerimenti

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

posso provare i comandi semplici:

```
studente@debian:~$ grep stringa 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 2 (1/2)

Realizzare un file comandi che preveda la seguente sintassi:

`trova M D1 D2 .. DN`

- **`M`** è un intero positivo.
- **`D1, D2,… DN`** sono nomi assoluti di directory esistenti.

Il file comandi deve:

- richiedere all'utente e **leggere da standard input** la stringa **`Fout`** corrispondente al path assoluto di un file non esistente
- **controllare** il corretto passaggio degli argomenti:
  - siano passati almeno 2 argomenti
  - **`M`** sia un intero positivo.
  - **`D1, D2,… DN`**, siano path assoluti di direttori esistenti

Esercizio 2 (2/2)

Il file comandi deve inoltre:

- **ispezionare** il contenuto di tutte le directory date (`D1, D2,.. DN`) allo scopo di **individuare tutti file di proprietà dell'utente che ha invocato lo script**
- Tra questi dovrà individuare il file col **maggior numero di parole nelle prime `M` righe** e stamparne a video il **nome assoluto**
- Infine dovrà stampare su `Fout` il numero totale di file esaminati (**totale di file di proprietà dell'utente che ha invocato lo script**)

Esercizio 2: Suggerimenti (1/2)

- Lettura da standard input:
- **`read var1 var2`**
  - Le stringhe in ingresso vengono attribuite alle variabili a seconda della corrispondenza posizionale
- Ciclo su un elenco di directory con path assoluto:

```
for dir in /path/to/dir1 /path/to/dir2 /path/to/dir3
do
  # do something on $dir
done
```

- L'esercizio richiede di iterare su un elenco di directory fornite da linea di comando: quale **variabile notevole** devo usare?
- Se ciclo su tutte le variabili fornite da linea di comando, tale lista include anche **`M`** (la stringa di lancio è: **`trova M D1 D2 .. DN`**)
- Come posso "far scorrere" gli argomenti in modo da evitare di ciclare sul primo?

Esercizio 2: Suggerimenti (2/2)

- Per trovare il massimo devo ciclare su tutti i file del direttorio X

  `for dir in *` → cicla su tutti i file del dir corrente.

  Come ciclo su tutti i file in X ?
- Considero solo i file di proprietà dell'utente che ha invocato lo script → utilizzare un'opportuna **variabile di ambiente**
- Selezionare solo le prime `M` righe del file → occorre un opportuno comando shell che **filtri** le prime linee
- Devo poi contare le parole in tali linee:
  - opportuna opzione del comando `wc`
  - occorre collegare l'out del comando che estrae le prime linee all'input di `wc`
- Il numero totale di file di proprietà dell'utente che ha invocato lo script deve essere scritto su file: → redirezioniamo l'output su file

Nota – spazi nei parametri di input

Qualunque script dovrebbe sempre funzionare anche qualora i parametri passati in ingresso includano degli spazi.

- Una volta realizzata la soluzione, testare sempre se funziona anche con file o directory che hanno spazi nel nome
- Usare opportunamente i `" "`
- A questo proposito, verificare la differenza tra **`for i in $*`** e **`for i in "$@"`** quando i parametri in ingresso contengono spazi!

Ulteriore esercizio per continuare a casa..

Esercizio 3

Creare uno script che abbia la sintassi

```
./conteggio M S filedir
```

Dove:

- `M` è un intero positivo,
- `S` è una stringa
- `filedir` è il nome assoluto di un file leggibile esistente contenente una serie di nomi assoluti di directory esistenti. Si supponga per semplicità che i nomi di directory riportati in `filedir` siano tutti privi di spazi.

cercare **nelle directory elencate in `filedir` tutti file con più di `M` occorrenze di `S`**; per ogni file che soddisfa questa condizione, lo script dovrà calcolarne la dimensione in bytes e stampare la stringa seguente:

«Il file <nome file> nella directory <Di> contiene <dim> caratteri.»

Esercizio 3: suggerimenti

Ciclo su un elenco di directory contenute in un file:

- ricordiamo che il comando **`cat`** stampa il contenuto del file dato come arg.
- ricordiamo il significato dei backquote: **`` `cat  FILEDIR` ``**

Come calcolare il numero di occorrenze di una stringa in un file?
Vedere **`grep -o ...`** e **`wc -l`** (consultare il man)