Esercitazione 6
Gruppo AK

File comandi Unix

Esercizio 1

Creare uno script che abbia la sintassi

`./check file1 file2`

dove **`file1`** e **`file2`** sono stringhe.

Lo script deve:

- **controllare** che **`file1`** e **`file2`** siano **basename** file esistenti nella home dell'utente che ha invocato lo script.
- richiedere all'utente e **leggere da standard input** una stringa **`user`**
- controllare che **`file1`** e **`file2`** siano di proprietà dell'utente **`user`**

Qualora i controlli sui parametri in ingresso diano esito positivo, lo script deve controllare quale file tra quelli passati a linea di comando abbia la dimensione maggiore e stampare a video il messaggio seguente:

*Il file **<absNameFin>** ha dimensione maggiore, pari a **<D>** KiB*

dove ***<absNameFin>*** è il nome assoluto del file (**`file1`** o **`file2`**) di dimensione maggiore e ***<D>*** è la dimensione in KiB di tale file

Esercizio 1: Suggerimenti (1/2)

Lettura da standard input:

- **`read var1 var2`**

  Le stringhe in ingresso vengono attribuite alle variabili a seconda della corrispondenza posizionale

Test **`fileX`** sia un «basename» (i.e., non deve contenere nessun '/')

- Quale metacarattere per fare pattern matching?
- Serve un costrutto che supporti l'uso dei metacaratteri (**`[[…]]`** oppure **`switch-case`**)

Test di file:

- **`test -f <path>`** Esistenza del file. Alternativa **`[ -f <path> ]`**

Esercizio 1: Suggerimenti (2/2)

`file1` e **`file2`** devono trovarsi nella **home directory** dell'utente che ha invocato lo script: utilizzare un'opportuna **variabile di ambiente**

Come reperire l'utente proprietario di un file? Due possibilità:

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

Come reperire la dimensione di un file? →Analogamente all'utente proprietario

La dimensione ottenuta potrebbe essere in byte, come la converto in KiB? → **`expr`**. NOTA: trascuriamo il fatto che **`expr`** supporta solo operazioni tra interi

Suggerimenti generali

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

posso provare i comandi semplici:

```
studente@debian:~$ ls -l
```

ma anche i comandi più complessi e loro piping

```
studente@debian:~$ ls –l | awk …
```

oppure 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

Realizzare un file comandi che preveda la seguente sintassi:

`lastLines fout N D1 D2 ... Dn`

- **`fout`** è il path assoluto di un file non esistente
- **`N`** è un intero positivo.
- **`D1, D2,… Dn`** sono nomi assoluti di directory esistenti.

Il file comandi deve innanzitutto:

- **controllare** il corretto passaggio degli argomenti:
  - siano passati almeno 3 argomenti
  - **`N`** sia un intero positivo.
  - **`D1, D2,… Dn`**, siano path assoluti di direttori esistenti

Esercizio 2

Il file comandi deve poi:

- **ispezionare** il contenuto di tutte le directory date (**`D1, D2,.. DN`**) allo scopo di sommare il numero di parole nelle ultime 10 righe di ogni file. Pertanto, per ogni directory **`Di`** passata come parametro, dovrà:
  - **considerare solo i file leggibili** in **`Di`**
  - per ogni file **`fj`** in **`Di`**, contare il numero **`Xj`** di parole nelle **ultime 10 righe** di **`fj`**
  - produrre un numero **`Si`** corrispondente alla **somma** dei valori **`Xj`** di tutti i file in **`Di`**
- Per ogni directory per cui **`Si>N`** (dove **`N`** è passato come parametro a linea di comando), il file comandi dovrà stampare su **`fout`** una riga con il seguente formato:

  <Di> <Si>

- dovrà individuare e stampare a video il nome assoluto della directory per cui **`Si`** è massimo, i.e., la directory in (**`D1, D2,.. Dn`**) per cui è maggiore la somma delle parole nelle ultime 10 righe di tutti i file leggibili

Esercizio 2: Suggerimenti (1/3)

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 `fout` e `N`

```
lastLines fout N D1 D2 .. DN
```

Come posso "far scorrere" gli argomenti in modo da evitare di ciclare sui primi due?

Esercizio 2: Suggerimenti (2/3)

- 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 leggibili: `test -r <path>`
- Selezionare solo le ultime `10` righe del file → occorre un opportuno comando shell che **filtri** le ultime linee
- Devo poi contare le parole in tali linee:
  - opportuna opzione del comando `wc`
  - occorre collegare l'out del comando che estrae le ultime linee all'input di `wc`

Esercizio 2: Suggerimenti (3/3)

- Per ogni directory per cui **`Si>N`**, il file comandi dovrà stampare su **`fout`** una riga ***<Di> <Si>*** → redirezioniamo l'output su file avendo cura di
  - cancellare il contenuto del file a inizio script
  - scrivere in append

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)