Esercizi Lezione bash

Esempi slide lezione 1

`ls -l /bin`

tipo&permessi	hard link	proprietario	gruppo	dimensione byte	ultima modifica	nome file
lrwxrwxrwx	1	root	root	7	apr 22 2024	/bin -> usr/bin

soluzioni lezione 1

qual è il pathname della home directory? `~` oppure `/home/user`, per spostarsi basta `$cd`

visualizzare i file della home directory ordinati in base alla data di ultima modifica: `ls -t`

Che differenza c'è tra i comandi `cat`, `more`, `tail`?

- `cat` serve a concatenare i file
- `more` è un filtro che serve a visualizzare su schermo un flusso di testo una pagina per volta
- `tail` stampa su standard output le ultime 10 righe dei file forniti come argomenti

esercizio link (...)

Elenco delle subdirectory contenute ricorsivamente nella vostra home:

```
cd
ls -R
```

modi per creare un file

- usare un editor
- `touch [nome_file]`
- `cat > nome_file` oppure `cat - > nome_file`

commenta i seguenti comandi

```
$ cd
$ mkdir d1
$ chmod 444 d1
$ cd d1
```

`cd` : ti porta in /home
`mkdir d1` : crea la cartella d1
`chmod 444 d1` : setta i permessi -r--r--r--
`cd d1` : provo a entrare nella cartella ma fallisco perché non ho permesso esecuzione

Soluzioni lezione 2

pipeline per copiare il contenuto della directory *dir1* in *dir2*:

- dir2 non esiste: `cp -r dir1 dir2`
- dir2 esiste: `cp /r dir1/* dir2`

contare numero file ricorsivamente in una cartella: `$ find /home | wc -l`
nota: se usassi `$ ls -R | wc -l` non funzionerebbe perché `ls -R` genera anche linee vuote. Inoltre meglio usare `-l` rispetto a `-w` perché file con uno spazio nel nome altererebbero il conteggio.

lista dei file della home directory il cui nome è una stringa di 3 caratteri seguita da una cifra:
`$ ls ~/???[0/9]`, per evitare le directory posso usare opzione `/d` : `$ ls -d ~/???[0/9]`

differenza dei comandi
`ls` : ordine alfabetico dei file della cartella
`ls | cat` : lista in ordine alfabetico con un nome file per linea
`ls | more` : lista in ordine alfabetico con un nome file per linea diviso in pagine

effetto dei comandi
`uniq < file` : stampa a schermo il contenuto del file, rimuovendo righe multiple adiacenti
`who | wc -l` : numero di utenti collegati a sistema
`ps -e | wc -l` : numero di tutti i processi anche non collegati al terminale +1 (intestazione)

Soluzioni lezione 3

ridefinisci rm in modo che non sia chiesta conferma prima della cancellazione dei file:
`$ alias rm='rm -f'`
definire rmi (rm interattivo) che chiede conferma prima di rimuovere un file: `$ alias rmi='rm -i'`
pipeline per numero di tutti processi in esecuzione : `ps -el --no-header | wc -l`
salvare file di testo output ultimo evento: `$ !?ls? > file.txt`
comando che fornisce numero comandi history list: `history | wc -l` , `wc-l ~/.bash_history`

comando che fornisce primi 15 comandi history list: `history | head -15`
comandi unix che iniziano con `lo` : `lo<tab><tab>`
lista dei file della home che iniziano con -al : `ls -d ~/al*` e `ls ~/al<Tab><Tab>`. (`find ~/al*`)

scrivere una pipeline che fornisca in output il numero di processi appartenenti all'utente root:
`ps -U root -u root --no-headers | wc -l`

```
$ emacs &
$ emacs
```

il primo comando provoca l'avvio di un processo in background, il secondo l-avvio in foreground

Esempi lezione 4

`$ find . -name '*.c' -print` :
`$ find . -name '*.bak' -ls -exec rm {} \;`
`$ find /etc -type d -print`

`$ fgrep rossi /etc/passwd` : in output le linee del file /etc/passwd che contengono la stringa *fissata* rossi
`$ ls -al . | grep '^d.......w.'` : in output le linee di relazione.txt che *non* contengono i caratteri a,g,t
`$ grep-w print *.c` : in output le linee di tutti i file con .c che contengolo la parola *intera* print
`$ egrep ’[a-c]+z’ doc.txt` : in output le linee di doc.txt che contengono stringa con prefisso di lunghezza non nulla e costruito solo da caratteri a,b,c seguito da una z

`sort -t: -k3,3 /etc/passwd` : riordina le righe del file /etc/passwd in base al terzo campo (user id). non funziona in quanto ordine di default è alfabetico

`$ sort -t: -k3,3 -n /etc/passwd` : funziona in quanto con l'opzione `-n` l'ordine è numerico. Il separatore `:` è impostato con `-t`

`$ tr a-z A-Z` : converte le minuscole in maiuscole
`$ tr -c A-Za-z0-9 ' '` : sostituisce tutti i caratteri NON (`-c` è il complemento) alfanumerici con degli spazi
`$ tr -cs A-Za-z0-9 ' '` : comprime (`-s`) gli spazi adiacenti in un unico spazio e sostituisce
`$ tr -d str` : cancella i caratteri contenuti nella stringa *str*

`$ cut -d: -f1 /etc/passwd` : estrae il campo 1 delimitato dal carattere : , i campi partono da 1

```
$ cd; cut -d: -f1 /etc/passwd > p1.txt; cut -d: -f6 /etc/passwd > p6.txt
$ paste p1.txt p6.txt
```

ti sposta in home (`cd`), estrae il primo campo delimitato da ':' del file /etc/passwd e lo stampa in p1.txt (`cut -d: -f1 /etc/passwd > p1.txt`),
estrae il sesto campo delimitato da ':' del file /etc/passwd e lo stampa in p6.txt (`cut -d: -f6 /etc/passwd > p6.txt`), infine combina le righe dei due file nello standard output (`paste p1.txt p6.txt`)

Soluzioni lezione 4

quanto spazio occupa il contenuto della propria home directory, visualizzando soltanto il numero di blocchi:
`(du ~ | tail -1) | cut -f1` , vale lo stesso senza parentesi . Soluzione del prof `du -s ~ | cut -f1`

l'effetto di `$ sort file >file` è quello di sovrascrivere il file prima che `sort` venga effettuato

gli argomenti di `tr` sono due stringhe, e la conversione avviene carattere per carattere, ovvero:

- str1 > str2, caratteri eccedenti di str1 convertiti nell'ultimo di str2
- str1 < str2, i caratteri di str1 verranno convertiti soltanto nei primi n (= len str1) caratteri di str2

comando per sostituire tutti i caratteri alfanumerici nell'input con un carattere <Tab> in modo che non sia ripetuto: `$ tr -s a-zA-Z0-9 '\t'`

pipeline per estrarre soltanto i minuti dal comando `date` (il cui output è: lun 23 dic 2024, 17:19:59, CET): `$ date | cut -d' ' -f5 | cut -d: -f2` . Soluzione del prof `date | cut -d: -f2`

scrivere una pipeline che permetta di scoprire se ci sono linee ripetute in un file: `$ sort file | uniq` poiché `$ uniq -d file` seleziona soltanto le righe ripetute adiacenti, devo prima riordinarle con `sort`
La soluzione del prof è
`cat nome_file | sort | uniq -c | sort -k1,1 -ns | tail -1 | tr -s ' ' | cut -d' ' -f2`

visualizzare su standard output, senza ripetizioni, lo UID degli utenti che hanno almeno un processo attivo nel sistema: `ps -el --no-header | tr -s ' ' : | cut -d: -f3 | sort -n | uniq`
osservazione: `ps` genera una tabella con molti spazi tra i campi, rendendo impossibile l'uso del comando `cut` con qualsiasi delimitatore. Per questo è necessario usare `tr -s ' ' :` per comprimere gli spazi in un unico carattere utilizzabile per delimitare i campi. In questo caso è stato scelto :, rendendo poi possibile operare con `cut` , `sort` e `uniq` . Poichè `uniq` funziona solo su linee adiacenti, era necessario riordinarle con `sort` , in questo caso in ordine numerico (`sort -n`).

Esempi lezione 5

`$ sed '4,$d' /etc/passwd`: stampa a video soltanto le prime 3 righe del file /etc/passwd, `d` (delete) cancella le righe a partire dalla quarta
`$ sed 3q /etc/passwd`: sed esce `q` (quit) dopo aver elaborato la 3 riga
`$ sed /sh/y/:0/_%/ /etc/passwc`: in tutte le righe che contengono sh sostituisce : con _ e 0 con %
`$ sed ’/sh/!y/:0/_%/’ /etc/passwd`: lo stesso in tutte le righe che non contengono sh, uso del quoting per !

sostituzione con sed: `s/expr/new/flags`

`$ sed ’/^root/,/^bin/s/:x:/::/w disabled.txt’ /etc/passwd`: sostituisce la stringa x con la stringa vuota nelle righe in input comprese fra quella che inizia con root e quella che inizia con bin, tali righe poi accodate nel file disabled.txt

`$ cat /etc/passwd | sed 's?/bin/.*sh$?/usr/local&?'` cerca tutte le righe in input in cui compare la stringa corrispondente all’espressione regolare `/bin/.*sh$` (ad esempio /bin/bash) e sostituisce quest’ultima con la stringa corrispondente a `/usr/local/bin/.*sh$` (ad esempio /usr/local/bin/bash). Si noti che, siccome il carattere separatore di sed compare nella stringa da cercare, si è usato il carattere ? come separatore. Inoltre il carattere & viene rimpiazzato automaticamente da sed con la stringa cercata (corrispondente a /bin/.*sh$)

Shell scripting

```
$ cat >dirsize
ls /usr/bin | wc -w
Ctrl-d

$ chmod 700 dirsize

$ ./dirsize
    1620
```

```
$ cat >data
set-x
echo the date today is:
date
Ctrl-d

$ chmod u+x data
$ ./data
++ echo the date today is:
the date today is:
++ date
Tue Oct 25 17:37:52 CEST 2005
```

```
$ cat >sost
set -v
cd TEXT
ls*.txt
sed s/’#’/’;;;’/ file.txt
Ctrl-d

$ more TEXT/file.txt
# questo è un commento
# in un programma

$ chmod u+x sost
$ ./sost
cd TEXT
ls *.txt
file.txt
sed d/'#'/';;;'/ file.txt
;;; questo è un commento
;;; in un programma
```

variabili

```
$ x=variabile
$ echo $x
variabile
```

Soluzioni lezione 5

creare sottocartella bin della home directory in modo che possano essere invocati da qualunque directory con il nome del file senza specificare l'intero pathname:

```
mkdir ~/bin
export  PATH=$PATH:~/bin
```

commentare effetti del seguente script

```
$ cat >chdir            #creo lo script
cd ..                   #comando cd ..
Ctrl-d

$ chmod 700 chdir       #rendo eseguibili
$ ./chdir               #eseguo
$pwd                    #noto che NON ho cambiato directory -> salto viene fatto in u
```

creare un alias permanente `lo` per il comando `exit`

```
$ nano ~/.bashrc
alias lo='exit'
```

script che prende come parametri una stringa e un file e controlla se la stringa compare nel file:

```
grep "$1" "$2" 2>/dev/null | wc -l
```

script che estrae i commenti da file .java sostituiendo // con la stringa linea di commento del file <nome file>, inoltre i commenti estratti devono essere salvati nel file fornito come secondo argomento: `sed "s//?linea di commento del file $1:?w $2" -n $1?` (va commentata)

Esempi lezione 6

comando condizionale

```
if grep "^$1:" /etc/passwd >/dev/null 2>/dev/null
then
    echo $1 is a valid login name
else
    echo $1 is not a valid login name
fi

exit 0
```

esempio di uso di espressione numerica complessa:

```
    $ num1=2
    $ num1=$[$num1*3+1]
    $ echo $num1
    7
```

esempio ciclo while

```
while test -e $1
do
    sleep 2
done

echo file $1 does not exist
exit 0
```

esempio ciclo until

```
until false
    do
    read firstword restofline
    if test $firstword = end
    then
        exit 0
    else
        echo $firstword $restofline
    fi
done
```

esempio ciclo for

```
for i in 1 2 3 4 5
do
    echo the value of i is $i
done

exit 0
```

esempio di case selection

```
case $# in
1)
    cat >>$1
    ;;
2)
    cat >>$1 <$2
    ;;
*)
    echo "usage: append out_file [in_file]"
    ;;
esac

exit 0
```

se il numero degli argomenti forniti (variabile $#) è 1, accoda testo da standard input in $1, se sono 2 accoda $2 in $1, altrimenti stampa un messaggio che illustra l'utilizzo dello script.

Esempio di command substitution

```
$ basefile=`basename /usr/bin/man`
$ echo $basefile
man
```

Esempio di scripting: script lisfiles che data una directory$(1)$$e la dimensione$(2) in byte restituisce tutti i file della directory più piccoli. Con controllo parametri

```
if test $# -ne 2
then
    echo 'usage: listfiles <dirpath> <dimensione>'
    exit 1
fi
if ! test -d $1             #testo se $1 è una cartella
then
    echo 'usage: listfiles <dirpath> <dimensione>'
    exit 1
fi

for i in $1/*
do
    if test -r $i -a -f $i
    then
        size='wc -c <$i'
        if test $size -lt $2
        then
            echo 'basename $i 'has size $size bytes
        fi
    fi
done

exit 0
```

Osservazioni interessanti

```
    file1.txt
    Ciao, come stai? Io tutto bene. Spero valga lo stesso anche per te

    file2.txt
    Bella, io sto una meraviglia.
```

`$ cat file1.txt file2.txt`

```
    Ciao, come stai? Io tutto bene. Spero valga lo stesso anche per te
    Bella, io sto una meraviglia.
```

differenze set -v, set -x

```
$ ./differenze              # file eseguibile
imposto set -x
++ ps -el --no-header
++ tr -s ' ' :
++ cut -d: -f3
++ uniq
++ sort -n
++ tail -2
996
1000
++ set -
ho tolto set -x
ora imposto set -v
ps -el --no-header | tr -s ' ' : | cut -d: -f3 | sort -n | uniq | tail -2
996
1000
set -
fine
```