Lezione 2: Comandi avanzati della Shell di Unix

Laboratorio di Elementi di Architettura e Sistemi Operativi

14 Marzo 2012

Parte 1: Soluzione degli esercizi

Statistiche

- Ho ricevuto 21 soluzioni
- Tutte le soluzioni hanno ricevuto voto positivo
- Le soluzioni complete e l'elenco dei voti sono disponibili sulla pagina del corso

Esercizio 5

3. Copiare il file `/usr/share/vim/vimrc` nella directory `prova` attribuendo al nuovo file il nome `filecopia`

Soluzione:
```
$ cp /usr/share/vim/vimrc filecopia
```
???
```
$ cp /usr/share/vim/vimrc prova ; mv prova filecopia
$ cp vimrc /home/.../prova
$ cd /home/.../prova ; mv vimrc filecopia
```

Esercizio 5

4. Visualizzare il file `vimrc` sullo schermo

Soluzione:
```
$ cat vimrc
```
Soluzioni alternative:
```
$ gedit vimrc
$ more vimrc
$ less vimrc
```

Visualizzazione per pagine

Esistono tre comandi quasi equivalenti:

- `pg file1 file2 ...`
- `more file1 file2 ...`
- `less file1 file2 ...`

Durante la visualizzazione è possibile dare dei comandi interattivi:

`spazio` prossima pagina

`invio` prossima riga

`b` pagina precedente

`/testo` prossima pagina contenente `testo`

`?testo` pagina precedente contenente `testo`

`q` termina programma

`v` edita file corrente

Esercizio 7

Alcuni modi per creare un file:

- creare il file con un editor di testo come `gedit`: `$ gedit fileprova.txt`
- creare una copia di un file esistente: `$ cp fileprova.txt fileprova2.txt`
- usare il comando `touch`: `$ touch fileprova3.txt`
- usare il comando `cat`, usando la redirezione dell'output: `$ cat > fileprova4.txt` (premere `ctrl-D` per terminare l'inserimento del file)

Modifica degli attributi di un file

`touch [-opzioni] [data] file ...`
aggiorna data e ora dell'ultimo accesso/modifica di un file

- se `data` non è specificata, usa data e ora corrente
- se il file non esiste lo crea vuoto
- opzioni:
  - `-a` cambia data di accesso del file
  - `-m` cambia data di modifica del file

Esercizio 9

Un link simbolico può puntare ad un altro link che a sua volta punta ad un file? Se è possibile, c'è un limite al numero di link simbolici che si possono avere in catena?

Sì, un link simbolico può puntare ad un altro link:
`$ ln -s file_name link1 $ ln -s link1 link2`
eseguendo il comando `cat link2` viene stampato a video il contenuto del file `file_name`.

Esercizio 9

Il limite al numero di link simbolici in catena dipende dal sistema operativo. Nel caso dei computer del laboratorio è 8:

```
$ ln -s file_name link1
$ ln -s link1 link2
$ ln -s link2 link3
$ ln -s link3 link4
$ ln -s link4 link5
$ ln -s link5 link6
$ ln -s link6 link7
$ ln -s link7 link8
$ ln -s link8 link9
```

Eseguendo il comando `cat link9` si ottiene il messaggio d'errore `cat : link9: Too many symbolic links encountered.`

Parte 2: Permessi e proprietari dei file

Ripartiamo da `ls`

Eseguendo il comando `ls -l /bin` si ottiene il seguente output:

```
...
lrwxrwxrwx  1 root root      4 Dec 5  2000 awk -> gawk
-rwxr-xr-x  1 root root   5780 Jul 13 2000 basename
-rwxr-xr-x  1 root root 512540 Aug 22 2000 bash
...
```

da sinistra a destra abbiamo:

1. tipo di file (`-` file normale, `d` directory, `l` link, `b` block device, `c` character device),
2. permessi,
3. numero di hard link al file,
4. proprietario del file,
5. gruppo del proprietario del file,
6. grandezza del file in byte,
7. data di ultima modifica,
8. nome del file.

I permessi dei file

- Unix è un sistema *multiutente*.
- L'amministratore del sistema (`root`) ha tutti i permessi
- Per gli altri utenti l'accesso ai file è regolato dai permessi:
  `-rw-r--r-- 1 root root 981 Sep 20 16:32 passwd`
- Il blocco di caratteri `rw-r--r--` rappresenta i permessi di accesso al file:
  - I primi 3 (`rw-`) sono riferiti al proprietario.
  - Il secondo blocco (`r--`) è riferito al gruppo.
  - L'ultimo blocco (`r--`) a tutti gli altri utenti.
  - La prima posizione rappresenta il permesso di lettura (`r`),
  - la seconda il permesso di scrittura (`w`)
  - e la terza il permesso di esecuzione (`x`).
  - Un trattino (`-`) indica l'assenza del permesso corrispondente.

Cambio di proprietario

- cambia il gruppo del file
  `chgrp [-R] gruppo file`
- cambia proprietario [e gruppo] del file
  `chown [-R] utente[:gruppo] file`
- l'opzione `-R` indica di propagare il comando alle sottodirectory

Cambio dei permessi

- assegna i permessi al file

  `chmod [-R] permessi file`

- permessi assoluti: un numero di quattro cifre corrispondenti alla *codifica binaria* dei permessi:

	proprietario	gruppo	altri
4 2 1	4 2 1	4 2 1	4 2 1
s S t	r w x	r w x	r w x

  Il primo numero si può omettere

- permessi simbolici: una stringa di tre caratteri a scelta tra quelli tra parentesi quadre

  `[ugoa] [+-=] [rwxst]`

Simboli dei permessi

`u` user

`g` group

`o` other

`a` all (user, group, other)

`+` assegna il permesso

`-` toglie il permesso

`=` assegna i permessi specificati, toglie gli altri

`r` read permission

`w` write permission

`x` execution permission

`S` setgid (vedi dopo)

`s` setuid (vedi dopo)

`t` sticky bit (vedi dopo)

Esempi di modifica dei permessi

- `chmod 640 prova.txt`
  - Lettura/scrittura per proprietario: $6 \rightarrow 4 + 2 \rightarrow 110 \rightarrow$ `rw-`
  - Lettura per gruppo: $4 \rightarrow 100 \rightarrow$ `r--`
  - Nessun permesso per altri: $0 \rightarrow 000 \rightarrow$ `---`
- `chmod 755 dir`
  - `r/w/x` per proprietario: $7 \rightarrow 4 + 2 + 1 \rightarrow 111 \rightarrow$ `rwx`
  - Lettura/esecuzione per gruppo: $5 \rightarrow 4 + 1 \rightarrow 101 \rightarrow$ `r-x`
  - Lettura/esecuzione per altri: $5 \rightarrow 4 + 1 \rightarrow 101 \rightarrow$ `r-x`
- `chmod u+x filename`
  - Aggiungi permesso di esecuzione per l'utente

- `chmod o-w filename`
  - Togli il permesso di scrittura agli altri
- `chmod g=rx`
  - Assegna il permesso di lettura ed esecuzione, *ma non di scrittura* al gruppo

Lo sticky bit

- Lo *sticky bit* è identificato dal simbolo `t`
- Non usato sui file
- Nelle directory, solo il proprietario del file o root possono cancellare o rinominare i file contenuti

Esempio: la directory `/tmp`

```
$ ls -ld /tmp

drwxrwxrwt 6 root root 1024 Aug 10 01:03 /tmp
```

Setuid e Setgid

- Setuid (`s`): chi esegue il file diventa temporaneamente il proprietario
- Setgid (`S`): chi esegue il file diventa temporaneamente dello stesso gruppo del file

Esempio: il comando `passwd`

```
$ ls -l /usr/bin/passwd

-r-s--x--x 1 root root 17700 Jun 25 2004 /usr/bin/passwd
```

Parte 3: Ricerca dei file

Ricerca di file per nome o attributi

- Visita tutto l'albero sotto la directory specificata e ritorna i file che rendono vera `espressione`:
  `find directory espressione`
- espressioni:

  `-name pattern` (usare gli apici `" "` se si usano metacaratteri)
  `-type tipo` (`b c d l f`)
  `-user utente`
  `-group gruppo`
  `-newer file`
  `-atime [+/-] giorni` ultimo accesso
  `-mtime [+/-] giorni` ultima modifica

Esempi di uso di `find`

- Tutti i file con estensione `.txt`: `$ find . -name "*.txt"`
- Tutti i link della directory corrente: `$ find . -type l`
- Tutti i file posseduti da `pippo` in `/tmp`: `$ find /tmp -user pippo`
- Tutti i file modificati da meno di un giorno in `/var/log`: `$ find /var/log -mtime -1`
- Tutti i file modificati da più di due giorni in `/var/log`: `$ find /var/log/ -mtime +2`

Ricerca di file per contenuto

- Il comando `grep` restistuisce le linee dei file che contengono un pattern specificato
  `grep [options] pattern [file1 file2 ...]`
- Opzioni:

  `-i` ignora la distinzione fra lettere maiuscole e minuscole
  `-l` fornisce la lista dei file che contengono il pattern
  `-n` le linee in output sono precedute dal numero di linea
  `-v` vengono restituite solo le linee che *non* contengono il pattern
  `-w` vengono restituite solo le linee che contengono il pattern come parola completa

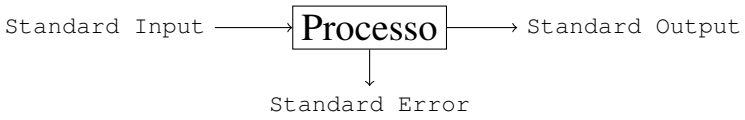
Esempi di uso di `grep`

- visualizza le linee del file `/etc/passwd` che contengono la stringa `rossi`: `$ grep rossi /etc/passwd`
- visualizza le linee del file `relazione.txt` che contengono la stringa `Nel Fiume`, ignorando le maiuscole/minuscole: `$ grep -i "Nel Fiume" relazione.txt`
- visualizza le linee del file `relazione.txt` che non contengono il carattere "a", precedute dal numero di riga: `$ grep -nv a relazione.txt`
- visualizza i nomi dei file con estensione `.c` che contengono la parola intera `print`: `$ grep -lw print *.c`

Parte 4: Redirezione dell'Input/Output

I canali di Input/Output

- Ogni processo di un sistema operativo Unix ha tre canali associati:



- Ogni canale può essere rediretto:
  - su file
  - su altro canale
  - verso un altro processo

Redirezione su file

- Standard input da file `comando < file`
- Standard output in file (se il file esiste già viene sovrascritto) `comando > file`
- Standard output aggiunto in coda al file `comando >> file`
- Standard error in file (se il file esiste già viene sovrascritto) `comando 2> file`
- Standard output e standard error su due file diversi `comando > fileout 2> fileerr`
- Standard output e standard error sullo stesso file `comando > file 2>&1` `comando &> file`

Esempi

- redirezione dell'output:

```
$ echo ciao a tutti > file
$ more file
ciao a tutti
```

- redirezione dell'output con append:

```
$ echo ciao a tutti >> file
$ more file
ciao a tutti
ciao a tutti
```

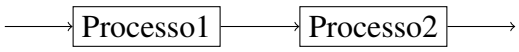
- Il comando `wc` fornisce numero di linee, parole, caratteri:

```
$ wc < progetto.txt
21 42 77
```

- Redirezione dei messaggi d'errore:

```
$ man 2 passwd 2> errori.txt
$ less errori.txt
No entry for passwd in section 2 of the manual.
```

La “pipe”: redirezione tra processi



- Standard output del primo comando nello standard input del secondo:
  `comando1 | comando2`
- Permette di combinare più comandi semplici per eseguire compiti complessi

I comandi filtro

- I *filtri* sono una particolare classe di comandi che possiedono i seguenti requisiti:
  - leggono l’input dallo standard input,
  - effettuano delle operazioni sull’input ricevuto,
  - inviano il risultato delle operazioni allo standard output.
- I filtri sono degli ottimi strumenti per costruire pipeline che svolgano compiti complessi.
- Alcuni esempi di filtri che abbiamo già incontrato:
  `cat, head, tail, more, less, find, grep, wc`

Esempi

- La sequenza di comandi

  ```
  $ ls /usr/bin > temp
  $ wc -w temp
  459
  ```

  ha lo stesso effetto della pipeline:

  ```
  $ ls /usr/bin | wc -w
  459
  ```

  I comandi ls e wc sono eseguiti in parallelo: l’output di `ls` è letto da `wc` mano a mano che viene prodotto.
- Visualizzare l’output di `ls` ricorsivo pagina per pagina `$ ls -R | more`
- Elencare i file con permessi `rw-r--r--`: `$ ls -l | grep "rw-r--r--"`