



# Miscellaneous

---

*Roberto De Virgilio*

*Sistemi operativi - 23 Ottobre 2017*

# Canali di comunicazione e ridirezione

- ◆ *I programmi dispongono di 3 canali di comunicazione standard (standard file descriptor):*
  - ▶ **0 Standard input** (tastiera)
  - ▶ **1 Standard output** (terminale)
  - ▶ **2 Standard error** (terminale).
- ◆ *La shell consente il reindirizzamento dei 3 canali standard all'interno di file con la sintassi:*  
*comando operatore file.*

# Operatori di ridirezione dello standard output

- ◆ *Si può ridirigere lo standard output del comando su file. Se il file non esiste, viene creato; se il file esiste, viene sovrascritto (>) oppure il nuovo output viene accodato (>>):*
- ▶ *SINTASSI: comando > file  
> ridirige lo standard output in modalità truncate*
- ▶ *SINTASSI: comando >> file  
>> ridirige lo standard output in modalità append.*

# Operatori di ridirezione dello standard error

- ✿ *Si può ridirigere lo standard error del comando su file. Se il file non esiste, viene creato; se il file esiste, viene sovrascritto (2>) oppure il nuovo output viene accodato (2>>):*
- ▶ *SINTASSI: comando 2> file  
> ridirige lo standard error in modalità truncate*
- ▶ *SINTASSI: comando 2>> file  
>> ridirige lo standard error in modalità append.*

# Operatori di ridirezione dello standard input

- ◆ *Si può ridirigere lo standard input del comando su file, cioè fa sì che l'input di un comando provenga, anziché da tastiera, da un file:*
  - ▶ *SINTASSI: comando < file*  
*< ridirige lo standard input*
  - ▶ *SINTASSI: comando << file*  
*<< redirezione doppia dell'input con accodamento fino alla digitazione di un carattere di escape o buffer pieno.*

# Operatori di ridirezione dello standard input

- ▶ **SINTASSI:** comando << file  
*<< redirezione doppia dell'input con accodamento fino alla digitazione di un carattere di escape o buffer pieno.*

```
grep 'ciao' << eof
> ciao
> ciao
> ciao
> eof
ciao
ciao
ciao
```

# Liste di comandi

- ◆ *Sulla stessa linea di comando è possibile eseguire una lista di comandi separandoli con caratteri speciali. Tipi di esecuzione: sequenza*
  - ▶ *SINTASSI: comando1 ; comando2  
Indipendentemente dal loro exit status, con ; i due comandi vengono eseguiti in sequenza (prima comando1 e poi comando2) .*

# Liste di comandi

- ◆ *Sulla stessa linea di comando è possibile eseguire una lista di comandi separandoli con caratteri speciali. Tipi di esecuzione:*

*concorrenza*

- ▶ *SINTASSI: comando1 & comando2*  
*Il comando1 viene eseguito in background, il comando2 in foreground*

# Liste di comandi

- ◆ *Sulla stessa linea di comando è possibile eseguire una lista di comandi separandoli con caratteri speciali. Tipi di esecuzione: condizione*
  - ▶ **SINTASSI:** *comando1 && comando2*  
*Il comando1 viene eseguito sempre;*  
*il comando2 soltanto se il primo è terminato con successo (exit status 0).*

# Liste di comandi

- ◆ *Sulla stessa linea di comando è possibile eseguire una lista di comandi separandoli con caratteri speciali. Tipi di esecuzione:*

*condizione*

- ▶ *Supponiamo che il file /tmp/file1 non esista  
cp /tmp/file1 . && cat file1 file2 > file3  
In questo caso cat non viene eseguito  
perche' cp non ha trovato il file da copiare.  
Una volta generato il file da copiare,  
eseguendo lo stesso comando il cat verrà  
eseguito perché cp terminerà senza errori.*

# Liste di comandi

- ◆ *Sulla stessa linea di comando è possibile eseguire una lista di comandi separandoli con caratteri speciali. Tipi di esecuzione:*

*esclusione*

- ▶ *SINTASSI: comando1 || comando2*  
*Il comando1 viene sempre eseguito; il comando2 soltanto se il primo è terminato con exit status diverso da 0*

# Liste di comandi

- ◆ *Sulla stessa linea di comando è possibile eseguire una lista di comandi separandoli con caratteri speciali. Tipi di esecuzione:*

## *esclusione*

- ▶ ***chown rob /tmp/file1 || rm -f file2***  
*In questo caso il secondo rm viene eseguito perché l'utente corrente non può cambiare il proprietario di un file.*
- ▶ ***rm -f /tmp/file1 || rm -f file2***  
*In questo caso il secondo rm non viene eseguito perché il primo rm termina senza errori.*

# Comando time

- ◆ Il comando *time* avvia un programma e, quando esso termina, visualizza sullo standard error il tempo impiegato per eseguirlo, diviso in tre valori:
  - ▶ **real** il tempo di esecuzione reale (il tempo trascorso dall'avvio al termine del programma);
  - ▶ **user** il tempo di CPU utente (il tempo impiegato dalla CPU per eseguire le istruzioni non di sistema del programma)
  - ▶ **sys** il tempo di CPU di sistema (il tempo impiegato dalla CPU per eseguire le istruzioni di sistema del programma)

*real = user + sys + waiting*

*waiting = I/O waiting time + idle time (running other tasks)*

# Comando time

- ◆ Il comando *time* avvia un programma e, quando esso termina, visualizza sullo standard error il tempo impiegato per eseguirlo, diviso in tre valori:
  - ▶ **SINTASSI:** *time [opzioni] [--] comando [arg1 [arg2 ...]]*  
I parametri *comando* e *arg* specificano il comando di cui misurare il tempo di esecuzione, insieme ai suoi eventuali parametri. Il doppio trattino -- (facoltativo) indica che i parametri successivi non sono da considerarsi opzioni. L'unica opzione di rilievo è *-p* che indica di mostrare i valori in un formato standard.

# Comando time

- ◆ Il comando *time* avvia un programma e, quando esso termina, visualizza sullo standard error il tempo impiegato per eseguirlo, diviso in tre valori:
  - ▶ **esempio:** `time sort file.txt > file_ordinato.txt`  
`real 0m0.507s`  
`user 0m0.492s`  
`sys 0m0.008s`

# Comando alias

- ✿ *Il comando alias è un comando di shell che permette di definire altri comandi:*
  - ▶ *SINTASSI: alias nome\_alias = 'comando'*  
*Ad esempio alias ll='ls -l'*

# Misurare ‘disk usage’

- ◆ *Il comando du misura l'uso del disco da parte di un file:*
  - ▶ *SINTASSI: du parametri file*
  - ▶ *du <file> restituisce l'uso del disco da parte del file in termini di numero di blocchi*
  - ▶ *du -h <file> restituisce l'uso del disco da parte del file in termini di K (kilobytes), M (megabytes) o G (gigabytes).*
  - ▶ *du -sh <dir> restituisce la somma degli usi dei vari file contenuti nella directory*

# Misurare ‘disk usage’

- ◆ Il comando *df* misura l’uso del disco e lo spazio libero da parte di una directory:
  - ▶ *SINTASSI:* *df parametri <dir>*
  - ▶ *df -h <dir>* restituisce l’uso del disco da parte della directory in termini di K (kilobytes), M (megabytes) o G (gigabytes).

```
> df -h .  
Filesystem  
/dev/hda5
```

	Size	Used	Avail	Use%	Mounted on
/dev/hda5	9.2G	7.1G	1.8G	81%	/

# Comprimere e decomprimere

- ◆ *g[un]zip <file>: zip compression utility. Crea .gz file. Mostra performance ordinarie (simile allo Zip).*
- ◆ *b[un]zip2 <file>: Più recente ed efficace. Crea .bz2 file. Di solito ha una compressione migliore del 20-25% rispetto a gzip.*
- ◆ *[un]lzma <file>: Più efficace di bzip2 (dal 10 al 20%).*

# Archiviare

- ◆ *tar (acronimo per tape archive), in informatica è un software che permette di generare dei file utili per l'archiviazione e il backup, sia su memorie di massa che su dispositivi a nastro magnetico utilizzando il formato omonimo.*
- ▶ *SINTASSI [crea]: tar cvf<archive> <files or directories>*
  - ▶ *c: create*
  - v: verbose. Useful to follow archiving progress.*
  - f: file. Archive created in file (tape used otherwise).*
- ▶ *SINTASSI [estrai]: tar xvf<archive>*

