Concatenazione di comandi

Tutti i comandi che abbiamo visto nelle lezioni precedenti, nel mondo *NIX sono tutti trattati come "filtri", dove, un filtro è un programma che ha un input, un output ed eventualmente (nel caso si verifichi) un output di errore.
Su ogni riga è possibile dare più di un comando alla volta, basta intervallarli con un punto e virgola (es: ls; cat nome_file)
I comandi generalmente prevedono l'input da tastiera e l'output sul video, o eventualmente l'output di errore sempre sul video, tranne nel caso della ridirezione.

La ridirezione: è quella funzione che mi "dice" o dove depositare l'output del comando appena digitato, o da quale file o comando prelevare l'input.
I comandi della ridirezione sono:

- **<**, dove il file preleva il proprio INPUT dal file che stà a destra e lo stampa a video;
  Esempio:
  cat < nome_file
- **>** esegue il comando che vi è a sinistra e stampa l'output nel file che noi indichiamo;
  Esempio:
  echo "Ciao sono il tuo file di prova" > prova_file
- **>>** esegue il comando che vi è a sinistra e stampa l'output nel file che noi indichiamo, è essenzialmente uguale alla ridirezione semplice illustrata sopra, ma ha il vantaggio di continuare il file, ossia se noi lanciamo per 10 volte dietro fila il comando [echo "Ciao sono il tuo file di prova" > prova_file] ogni volta il file viene cancellato e riscritto e se alla fine proviamo ad aprire il file con un editor di testo, ci troveremo solamente una riga con scritto "Ciao sono il tuo file di prova", mentre se lanciassimo per 10 volte di seguito il comando [echo "Ciao sono il tuo file di prova" >> prova_file] ci troveremo un file di 10 righe con scritto in tutte: "Ciao sono il tuo file di prova"

Quando lo standard di output di un comando coincide con lo standard input di un altro comando, è possibile sfruttare questo meccanismo attraverso la **pipe** (|).
Ad esempio se vogliamo trovare tutte le directory presenti all'interno di un'altra directory, dovremo dare in ordine i comandi:
ls -lh > file_prova; grep ^d < file_prova
INVECE sfruttando il meccanismo di piping, ci basterà dare:
ls -lh | grep ^d

Infine è possibile concatenare i comandi oltre che con il ; anche con il simbolo **&**

I comandi che abbiamo spiegato fino ad ora è possibile inserirli anche all'interno di un file, tutti uno di seguito all'altro, oppure inserendo un comando per riga, mettere un comando per riga, equivale a dividere i comandi con il ;
Proviamo ora a creare un file che:

A) Crei una directory chiamata “conoscerelinux”;
B) Entri nella directory appena creata;
C) Crei un file chiamato “primo”;
D) Ci scriva dentro “Eccomi sono il tuo primo file creato”;
E) Crei una directory chiamata “seconda_directory”;
F) Entri nella directory appena creata;
G) Crei un file chiamato “secondo”;
H) Ci scriva dentro “Eccomi sono il tuo secondo file creato”;
I) Ritorni nella propria directory /home;
J) Stampi a video la scritta “Cartelle e file creati correttamente”;

Soluzione:

```
mkdir conoscerelinux
cd conoscerelinux
touch primo
echo “Eccomi sono il tuo primo file creato” > primo
mkdir seconda_directory
cd seconda_directory
touch secondo
echo “Eccomi sono il tuo secondo file creato” > secondo
cd ../..
echo “Cartelle e file creati correttamente”
```

Ora per eseguire questo file, dovremo cambiargli i permessi e settargli i permessi di esecuzione per gli utenti che desideriamo:

chmod u+x file_script

A questo punto lo possiamo eseguire e vedere il risultato ottenuto. Per eseguirlo ci basterà dare il comando:

./file_script