Sistemi Operativi

AAF - Secondo anno - 3CFU

A.A. 2024/2025

Corso di Laurea in Matematica

Bash script e comandi

Annalisa Massini

Dipartimento di Informatica
Sapienza Università di Roma

Argomenti trattati

1. Bash script
   - Pipelining
2. Comandi sui file
   - Comandi di visualizzazione

Pipelining

Generalità sui comandi

- Finora, abbiamo visto che si possono dare comandi in sequenza nei seguenti modi:
  - con il ; o l'andata a capo (equivalenti)
  - con && e il ||
  - mettendoli dentro parentesi tonde o graffe
- In tutti questi esempi, quello che succede è :
  - che prima si esegue un comando, e poi (eventualmente) un altro
  - input ed output sono scollegati, a meno di redirezioni comuni nel caso di sottoshell o di group command

Generalità sui comandi

- Si possono concatenare tra loro due o più comandi inviando ciò che viene prodotto sul canale di standard output di un comando sul canale di standard input di un altro comando
- Questa operazione di concatenazione di programmi si chiama **pipelining** e si realizza usando il simbolo |
- Quindi il **pipelining** serve a collegare gli `stdout` agli `stdin` per le sequenze non condizionali

Generalità sui comandi

Esempio

- Si può inviare l'output del comando `ls -1` (verificare cosa produce l'opzione `-1`) in input al comando `wc -l` scrivendo:

  ```
  ls -1 | wc -l
  ```

- Così lo `stdout` del comando a sinistra viene rediretto nello `stdin` del comando a destra `wc` che, con l'opzione `-l`, esegue il conteggio delle righe in input e lo stampa in output

- Con `|&` anzichè `|`, anche lo `stderr` viene rediretto nello `stdin`

- Sulla stessa riga possono essere utilizzati più operatori di pipe per costruire sequenze di operazioni che, prima di produrre l'output finale, passano i dati le une alle altre

Generalità sui comandi

- L'operatore **pipe** può essere applicato ad un *group command*, cioè comandi tra parentesi graffe, o ad una *subshell*, cioè comandi tra parentesi tonde
- Così ha effetto sull'input/output combinato di tutti i comandi del gruppo
- Ad esempio, anziché scrivere

  ```
  { cmd1; cmd2; } > out; cmd3 < out; rm out
  ```

  si può scrivere

  ```
  { cmd1; cmd2; } | cmd3
  ```

Informazioni sui comandi

- Ricapitolando, la shell Bash è in grado di elaborare i seguenti tipi di comandi, sia che essi siano forniti dall'utente sulla linea di comando in modalità interattiva, sia che siano inseriti in uno script:
  1. **comandi semplici**: singoli comandi interni o esterni
  2. **liste di comandi**: sequenze di comandi, concatenati con i connettori ;, &, &&, ||
  3. **comandi in pipeline**: sequenze di comandi concatenati dall'operatore di pipe, cioè "|", che consente di trasformare l'output di un comando nell'input per il comando successivo nella sequenza della pipeline

Informazioni sui comandi

- Inoltre, la shell Bash è in grado di elaborare i seguenti tipi di comandi:
  1. **comandi composti**: comandi più complessi, composti da più istruzioni e da strutture di controllo che formano una struttura logico-algoritmica non solo sequenziale
  2. **funzioni**: sotto-programmi, identificati da un nome, che possono essere richiamati nella shell o in uno script

Altri comandi

Comandi di visualizzazione

Comando `cat`

- Abbiamo già visto il comando `cat [nomefile]`
- Scrive a schermo il contenuto di `nomefile`
  - funziona bene solo se il file è di testo (e se usa la codifica riconosciuta da `cat`), altrimenti scrive caratteri incomprensibili
  - si può usare `cat` per leggere più di un file alla volta scrivendo `cat file1 file2... filen`:
    - il comando `cat` stamperà il contenuto del primo file, poi del secondo, e così via
    - l'output sarà quindi la concatenazione del contenuto dei file specificati
  - senza argomenti, resta in attesa: se si scrive qualcosa e poi si preme invio, ripete quanto scritto, finché non si preme CTRL+d, che è il carattere EOF (*end-of-file*)

Comando `tac`

- Il comando `tac [nomefile]` scrive a schermo il contenuto di `nomefile` dall'ultima alla prima riga
  - si può usare `tac` per leggere a rovescio le righe di più di un file alla volta scrivendo `tac file1 file2... filen`:
    - il comando `tac` stampa il contenuto del primo file dall'ultima riga alla prima, poi del secondo sempre dall'ultima riga alla prima, e così via
    - l'output sarà quindi la concatenazione del contenuto dei file specificati ognuno dall'ultima riga alla prima

Comandi `less` e `more`

- Comandi `less {files}` e `more [-num] [+num] [-d] {files}`
  - come `cat`, ma paginano l'output se è troppo lungo
  - `less` è normalmente usato da `man` per mostrare il manuale
  - differiscono in svariati comportamenti:
    - `less` permette di muoversi sempre sia in avanti che all'indietro, `more` solo se usato senza pipelining (cioè il |)
    - si chiudono premendo `q`, ma per `less` sparisce tutto quello che era scritto, con `more` resta l'ultima schermata
    - `less` pagina sempre, `more` solo se l'output è più grande di una pagina
    - opzioni di `more`: `-num` imposta a `num` il numero di righe in una pagina; `+num` comincia la visualizzazione dalla riga `num`, `-d` mostra una sorta di lista di comandi in basso

Comando `head`

- Comando `head [-c car] [-n righe] [file...]`
  - al solito, senza argomenti legge da tastiera (una riga per volta)
  - come `cat`, ma stampa solo i primi `car` caratteri o le prime `righe` righe (ciò che è minore) dei file dati
  - senza opzioni stampa 10 righe, senza limiti sui caratteri
  - **Esercizio** scrivere quanto segue su un file, e poi farsi stampare solo la prima riga, ma usando l'opzione `-c`:
    ```
    ciao
    addio
    ```

Comando `tail`

- Comando `tail [-n righe] [-f] [file...]`
  - al solito, senza argomenti legge da tastiera (ma questa volta occorre premere CTRL+d alla fine dell'input)
  - come `cat`, ma stampa solo le ultime `righe` dei file dati
  - con `-f`, aggiorna di continuo la stampa: utile se si vuole leggere un file cui vengono continuamente aggiunti dati (ad esempio, come risultato di una qualche computazione, *mentre* la computazione stessa è in esecuzione)

Comandi `head` e `tail`

- **Esercizio** trovare il modo per andare avanti ed indietro di k righe e di k pagine con `more` e `less`
- **Esercizio** creare un file con `gedit` (lanciato in background), scriverci dentro almeno 4-5 righe e poi salvare; tornare sulla shell ed eseguire `tail nomefile`. Poi aggiungere qualche altra riga, ed eseguire nuovamente `tail nomefile`. Effettuare nuovamente questi passaggi, ma eseguendo stavolta `tail -f nomefile`. È sufficiente scrivere le modifiche sul file, affinché vengano visualizzate dal `tail`?