Sistemi Operativi

AAF - Secondo anno - 3CFU
A.A. 2023/2024
Corso di Laurea in Matematica

Bash: comandi e script

Annalisa Massini

Dipartimento di Informatica
Sapienza Università di Roma

Argomenti trattati

1. Generalità sui comandi
   - Informazioni sui comandi
   - Parentesi

Shell Programming

Informazioni sui comandi

Informazioni sui comandi

- Ogni comando genera un processo
- Il processo generato, quando termina, restituisce un **return code** o **exit code** alla bash
  - Return code uguale a 0 indica *tutto ok*
  - Se il return code assume un valore tra 1 e 255 vuol dire che c'è stato un *errore*
  - ci possono essere molte cause di errore (non più di 255...)
  - per vedere il codice dell'errore: `echo $?`
  - se un comando non è riconosciuto, viene restituito 127
  - se un comando è costituito da una sequenza di comandi separati da `;` allora l'exit code è quello dell'ultimo comando eseguito
  - se un comando viene eseguito in background, l'exit code è 0; per prendere il suo vero exit code, occorre usare il comando builtin `wait`

Esercizi

- **Esercizio:** capire se l'exit code di `ls` è 0 o diverso da 0 nei seguenti casi:
  - nessun argomento
  - un file esistente come argomento
  - un file non esistente come argomento
  - un file esistente e uno non esistente come argomento
  - un file esistente ma non accessibile in lettura come argomento
  - una directory esistente ma non accessibile in lettura come argomento

Informazioni sui comandi

- La bash permette l'**esecuzione condizionale** dei comandi
- Gli operatori && e || rappresentano i connettori logici **and** e **or**, rispettivamente, e possono essere utilizzati per sfruttare il codice di ritorno restituito dai comandi
- Un comando viene eseguito solo se una data condizione è vera
- Il modo più semplice per farlo è condizionare un comando all'esecuzione di un comando precedente usando l'exit code
- Se l'esecuzione del comando precedente è:
  - *corretta* allora l'*exit code è 0*
  - *sbagliata* allora l'*exit code è diverso da 0*

Informazioni sui comandi

- Tenendo conto del significato dei connettori logici *and* e *or*, l'interprete esegue i comandi di una lista interpretandoli da sinistra verso destra
- Si ha il seguente comportamento:
  - `comando1 && comando2`: `comando2` viene eseguito solo se `comando1` è corretto, in caso contrario l'esecuzione viene interrotta
  - `comando1 || comando2`: se `comando1` restituisce il valore vero, l'elaborazione del comando composto viene interrotta perché è già possibile stabilire il valore restituito dall'intero comando, quindi `comando2` viene eseguito solo se `comando1` è sbagliato

Informazioni sui comandi

- Si possono concatenare più di 2 comandi, sia con && che con || e anche usare le parentesi, sia tonde che graffe (vedremo tra poco la differenza tra queste 2 opzioni)
- Ad esempio, con il comando composto

  ```
  (comando1 && comando2) || comando3
  ```

  se `comando1` restituisce il valore vero allora viene eseguito anche `comando2`

  altrimenti, se `comando1` restituisce il valore falso, viene eseguito `comando3`

Shell Programming

Parentesi

Parentesi graffe

- Qualsiasi comando o sequenza di comandi può essere inserito tra **parentesi graffe** (*group command*)
- Quel comando o sequenza di comandi viene eseguito nella bash corrente, cioè dalla stessa bash in cui viene eseguito lo script
  - I comandi devono essere separati da un punto e virgola
  - Anche l'ultima istruzione della lista deve essere seguita da un punto e virgola
  - Tra le parentesi graffe e la prima e l'ultima istruzione deve essere presente uno spazio: infatti le parentesi graffe sono parole riservate del linguaggio Bash e devono essere separate da uno spazio dalle parole chiave adiacenti

Parentesi graffe

- Se il comando è unico, mettere le parentesi graffe è inutile (anzi, può complicare le cose)
- Utilità delle parentesi graffe:
  - raggruppare più comandi in **esecuzione condizionale** (vedere sotto), facendo sì che l'effetto sia sulla bash corrente
  - raggruppare tutte le **redirezioni** in una volta sola (vedere sotto)
  - raggruppare tutto l'input/output da mandare in **pipelining** in una volta sola (vedere sotto)

Parentesi tonde

- Qualsiasi comando o sequenza di comandi può essere inserito tra **parentesi tonde**
- Il comando composto viene eseguito in una *sottoshell* della shell in cui viene eseguito,
  - se il comando è unico, allora il nuovo processo è costituito da quel comando
  - se è una lista di comandi (separati ad esempio da ;), allora il nuovo processo è una bash che esegue quei comandi
  - il comando composto eredita tutte le variabili definite nello script, ma lo scope delle assegnazioni e delle definizioni effettuate nell'ambito del comando composto è limitato al comando stesso
- Le parentesi tonde che delimitano il comando composto non sono delle parole riservate del linguaggio (come le graffe) e non è necessario separarle dalle altre istruzioni con degli spazi

Parentesi tonde

- Utilità delle parentesi tonde:
  - raggruppare più comandi in **esecuzione condizionale** (vedere sotto), facendo sì che non ci sia effetto sulla bash corrente
  - raggruppare tutte le **redirezioni** in una volta sola (vedere sotto)
  - raggruppare tutto l'input/output da mandare in **pipelining** in una volta sola (vedere sotto)
- **Esercizio** provare a dare i comandi `(cd ..)` e `{ cd ..; }`

Parentesi graffe e tonde

Esempio parentesi graffe

- Consideriamo lo script:

```
a=1; b=2
echo "Prima del comando composto:  A = $a, B = $b"
{ b=3; echo "Durante il comando composto:  A = $a, B =
$b"; }
echo "Dopo il comando composto:  A = $a, B = $b"
```

  - Vengono definiti e visualizzati i valori delle variabili `a` e `b`
  - La sequenza di istruzioni nelle parentesi graffe viene eseguita come un singolo comando, nella stessa shell dello script
  - Il valore della variabile `b` viene modificato, mentre il valore di `a`, definita fuori dal comando composto, rimane lo stesso
- L'output è:

```
Prima del comando composto:  A = 1, B = 2
Durante il comando composto:  A = 1, B = 3
Dopo il comando composto:  A = 1, B = 3
```

Parentesi graffe e tonde

Esempio parentesi tonde

- Consideriamo lo script:

```
a=1; b=2
echo "Prima del comando composto:  A = $a, B = $b"
(b=3; echo "Durante il comando composto:  A = $a, B =
$b")
echo "Dopo il comando composto:  A = $a, B = $b"
```

  - Vengono definiti e visualizzati i valori delle variabili `a` e `b`
  - Il comando composto può utilizzare le variabili definite in precedenza e può modificarne il valore, ma le modifiche effettuate **non** sono visibili al termine del comando

- L'output è:

```
Prima del comando composto:  A = 1, B = 2
Durante il comando composto:  A = 1, B = 3
Dopo il comando composto:  A = 1, B = 2
```