Dispense di informatica

Emanuele Nardi

9 aprile 2019

Indice

1 Ripasso del linguaggio C **2**

1.1 Programma minimale 2

1.2 Comandi di base per Linux 2

1.3 Eseguire il programma 3

1.4 Automatizzare il processo di esecuzione 3

1 Ripasso del linguaggio C

1.1 Programma minimale

Come ogni buon tutorial che si rispetti iniziamo costruendo ed analizzando il *programma minimale* (ovvero quello più semplice possibile) del linguaggio C:

```
1 #include <iostream>
2 using namespace std;

3 int main() {
4     cout << "Hello, World!";
5     return 0;
6 }
```

Il codice stampa semplicemente la stringa di testo `"Hello, World!"` sul terminale. Iniziamo spiegandone le prime righe:

- `#include <iostream>` importa la libreria standard che contiene tutte le funzioni principali. Nota che le librerie vengono importate all'inizio del file;
- la nuova libreria `iostream` richiede l'uso di uno spazio dei nomi per identificare univocamente le classi, qui ci viene in aiuto `using namespace std;` che indica al compilatore quale spazio dei nomi bisogna utilizzare (in questo caso quello della `standard library`);
- `int` indica che il valore di ritorno della funzione è di tipo intero. In questo caso restituisce il valore `0` che — per convenzione — significa che la funzione è terminata correttamente;
- il `int main() {...}` è la funzione principale che viene eseguita dal compilatore come prima funzione. In questo caso non ha argomenti (indicato dalle parentesi tonde prive di parametri al loro interno).
- `cout` è una funzione speciale che ci permette di scrivere sul terminale (il quale viene definito lo `standard output`). Nota che se non avessimo impostato lo spazio dei nomi avremmo dovuto specificare `std::cout`.

1.2 Comandi di base per Linux

Abbiamo visto diversi comandi a lezione:

`whoami`	stampa l'utente attualemente loggato
`pwd`	stampa la cartella corrente
`cd`	permette di navigare fra le cartelle
`mkdir`	permette di creare cartelle
`touch`	permette di creare file
`mv`	permette di spostare e rinominare file
`chmod`	permette di modificare i permessi per un file

1.3 Eseguire il programma

Per eseguire il programma dobbiamo prima compilarlo con `g++`

```
g++ hello-world.cc -o hello-world.out
```

che crea un file oggetto chiamato `hello-world.out`, dopodiché lo eseguiamo

```
./hello-world.out
```

che stampa sullo schermo la scritta `"Hello, World!"`.

Notiamo che anche un programma molto semplice come questo può contenere al suo interno tanti piccoli dettagli: a prima vista possono sembrare sottigliezze ma, in un linguaggio complesso come il C, vanno ad accumularsi e, con il passare del tempo e della complessità dei programmi, rendono di difficile comprensione gli stessi.

In futuro non analizzeremo così nel dettaglio gli esercizi proposti ma solo la *business logic*, ossia la logica di fondo di un determinato programma.

1.4 Automatizzare il processo di esecuzione

Apportando varie al modifiche al file ed avendo compilato numerose volte tendiamo a notare come il processo di compilazione e di esecuzione diventi noioso e ripetitvo. La programmazione ci viene (anche stavolta) in aiuto. Risolviamo un problema scrivendo un programma (in gergo informatico uno *script*) che esegua un numero arbitrario di comandi.

```
1 #!/bin/bash               # indico il tipo di file
2 g++ hello-world.cc        # compilo
3 mv a.out hello-world.out  # rinomino
4 ./hello-world.out         # eseguo
```