Laboratorio di Programmazione 1

Lezione Introduttiva

Matteo Bianchi

Dipartimento di Informatica "Giovanni degli Antoni"
Università degli Studi di Milano

matteo.bianchi@unimi.it

27 Febbraio, 2025

Laboratorio - introduzione

- In totale vi saranno 12 lezioni di laboratorio, da 3 ore ciascuna.
- Il laboratorio del corso è finalizzato all'apprendimento pratico del linguaggio di programmazione C.
- Durante le lezioni, svolgerete degli esercizi, descritti all'interno di un file pdf, reperibile su sito ARIEL del corso.
- Tali esercizi (a partire dalla prossima lezione), consisteranno nella descrizione di specifiche di programmi che dovrete implementare, in linguaggio C.

Laboratorio - introduzione

- La difficoltà degli esercizi tenderà ad aumentare durante il corso, ma è del tutto normale.
- Tuttavia, è importante che studiate regolarmente, dato che gli esercizi sono sincronizzati con i temi presentati nelle lezioni frontali.
- Come vedrete, ogni esercizio riporterà una indicazione temporale, espressa in minuti.
- Tale valore è utile ai fini della vostra autovalutazione, nel senso che dovreste riuscire a risolvere l'esercizio (implementando il programma) entro il tempo indicato.
- La quantità di esercizi è sovrabbondante, in relazione alla durata delle lezioni di laboratorio.
- Tale scelta è deliberata, così che abbiate del materiale per esercitarvi anche al di fuori del laboratorio.

Libro di testo

il libro di testo ufficiale del corso (nelle slides del corso viene indicato come K&R) è il seguente:

> *B. W. Kernighan, D. M. Ritchie. Il linguaggio C. Principi di programmazione e manuale di riferimento. Seconda edizione. Pearson Italia, 2004.*

Tale libro, nelle appendici contiene lo standard ANSI C89, per cui può essere usato anche per consultare i dettagli sulle funzioni della libreria standard.

E' un testo la cui lettura può risultare difficoltosa, soprattutto per chi non ha precedenti esperienze di programmazione, per cui raccomando di utilizzarlo come complemento alle slides delle lezioni.

Sulla pagina ARIEL del corso trovate ulteriori riferimenti bibliografici.

Linguaggio C

- Il linguaggio C venne sviluppato da Dennis Ritchie tra il 1969 e il 1972.
- Il nome C fu scelto in quanto tale linguaggio era il successore del linguaggio B.
- Nel corso degli anni si sono succeduti diversi standard ANSI del linguaggio C.
- Il primo standard ANSI fu il C89, mentre il più recente è il C23.
- Per quanto concerne il corso, ci fermeremo allo standard C99.
- Dato che il linguaggio C venne sviluppato in ambiente Unix, nel corso del laboratorio utilizzeremo un sistema Unix-like, specificamente un sistema operativo Linux (di cui esistono diverse distribuzioni).

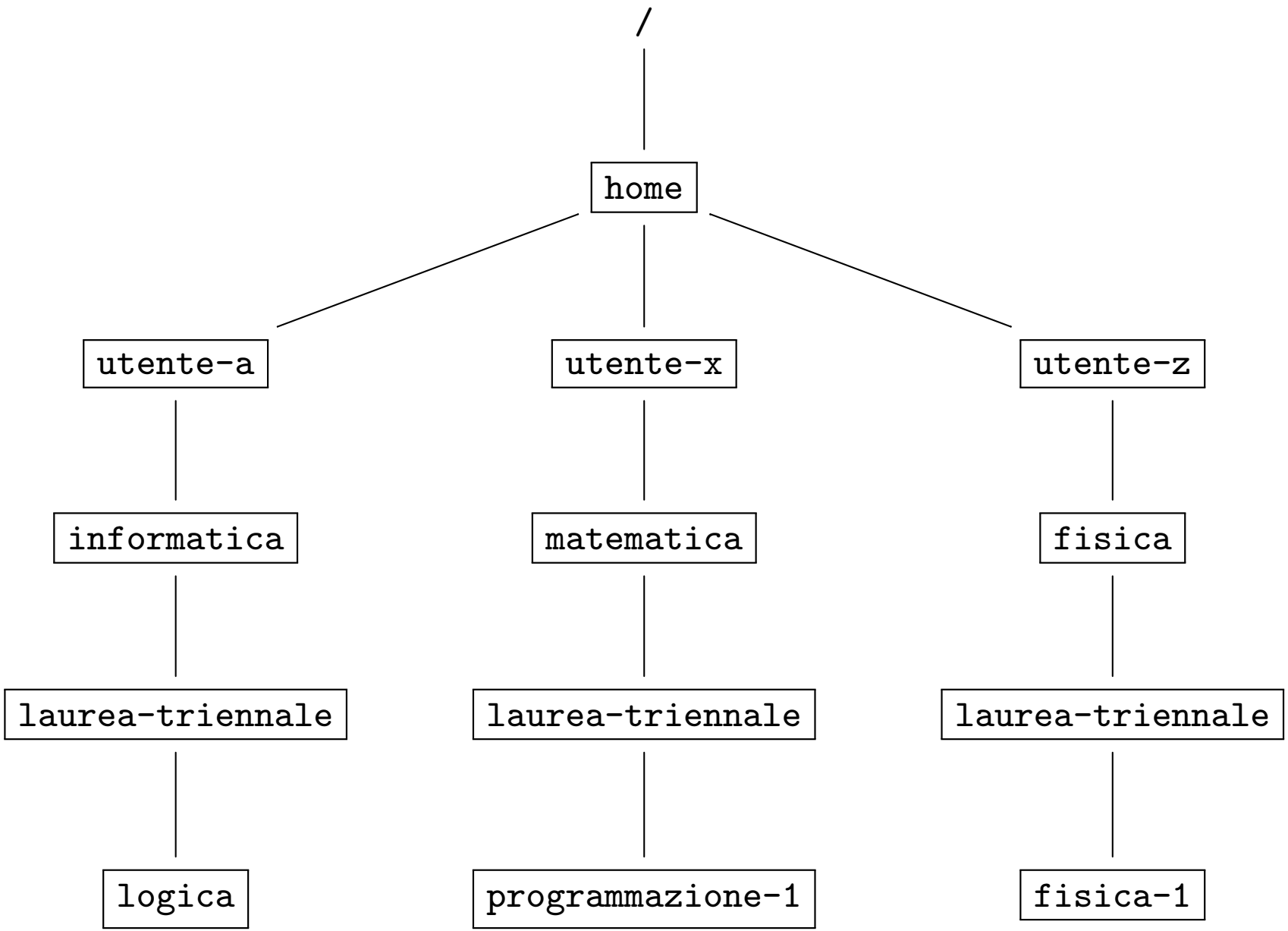
Shell di Linux

- In un sistema operativo, le interfacce principali sono due.
- CLI (command line interface), interfaccia da line di comando.
- GUI (graphical user interface), interfaccia grafica.
- Per l'uso del compilatore sfrutteremo l'interfaccia da linea di comando di Linux, nota anche come *shell*.
- Esistono diversi tipi di shell Linux: bash, csh, zsh, sh. . . .

File e Directory

- Le informazioni salvate su un supporto di memoria permanente, in un computer, vengono gestite da un File System (esempi: FAT32, exFAT, NTFS, EXT3, EXT4, ecc.), attraverso file e directory.
- Potete considerare le directory come delle sorte di contenitori, al cui interno possono esservi altre directory o file (pensate a uno schedario cartaceo).
- In ambiente Linux, esiste una directory radice (root directory) "/" che contiene tutte le altre directory e file. File e directory sono organizzati in una struttura ad albero (un particolare tipo di grafo: lo vedrete in altri corsi).

Esempio


/
home
utente-a
utente-x
utente-z
informatica
matematica
fisica
laurea-triennale
laurea-triennale
laurea-triennale
logica
programmazione-1
fisica-1

Percorso assoluto e relativo directory

Sulla shell il percorso di una directory - o file - può essere *assoluto* o *relativo*.

- Il percorso assoluto inizia dalla root directory "/", e viene indicato con `/directory_1/.../directory_k`.
- Il percorso relativo, parte invece dalla directory corrente (in cui vi trovate), e ha la forma `./directory_1/.../directory_k`.

Ogni utente Linux ha una *home directory*: il nome e il percorso in cui si trova tale directory cambiano a seconda del sistema.

Percorso assoluto e relativo, esempi

- Supponiamo che l'utente X abbia una home directory con percorso assoluto `/home/utente-x`
- Supponiamo ora che l'utente X abbia una directory `programmazione-1` con il seguente percorso assoluto:
  `/home/utente-x/matematica/laurea-triennale/programmazione-1`
- Se ci troviamo nella home directory di X (ovvero `/home/utente-x`), il percorso relativo di `programmazione-1` è il seguente:
  `./matematica/laurea-triennale/programmazione-1`
- Invece, se ci troviamo in `/home/`, il percorso relativo di `programmazione-1` diventa:
  `./utente-x/matematica/laurea-triennale/programmazione-1`
- Pertanto, il percorso relativo di una directory dipende dalla directory in cui ci troviamo. Il percorso assoluto, invece, rimane invariato.
- Si noti che ha senso parlare di percorso relativo di una directory `dir1`, rispetto a una directory `dir2`, solo nel caso in cui `dir1` sia una sottodirectory di `dir2`.
  Ad esempio, se `dir1` ha percorso assoluto `/home/utente-x/dir1` e `dir2` ha percorso assoluto `/home/utente-x/dir2`, allora non esiste un percorso relativo di `dir1` rispetto a `dir2`.

Shell di Linux, alcuni comandi di base - 1

- `man` *nome_comando*.
  Richiama la pagina del manuale (inteso come documentazione) relativa a *nome_comando*.
  Premete `q` per uscire.
- `ls` *nome_percorso*.
  Restituisce l'elenco dei file e delle directory presenti nel percorso specificato. Se nome_percorso è vuoto, prende la directory corrente come percorso.

  Sui computer del laboratorio, `ls` vi mostra i nomi delle directory e di alcuni file con colori diversi. Questa opzione non è attiva di default, su altre distribuzioni Linux.
- `ls -l` *nome_percorso* mostra delle informazioni aggiuntive, rispetto a `ls` *nome_percorso*, su file e directory (permessi, nome del proprietario ecc.).

  Sui computer del laboratorio, invece di `ls -l` potete scrivere l'abbreviazione (tecnicamente è un alias, ma non entriamo nei dettagli) `ll`.
  Si noti che `ll` non è un comando standard della Shell di Linux, per cui su altri sistemi Linux potrebbe non funzionare (usate `ls -l`, nel caso).

Shell di Linux, alcuni comandi di base - 2

- `pwd`
  Mostra il percorso assoluto della directory in cui vi trovate.
- `cd` *nome_percorso*.
  Cambia la directory in *nome_percorso*. Se nome_percorso è vuoto (ovvero scrivete solo `cd`), vi porta nella vostra home directory.
  Scrivendo
  `cd ..`
  passate alla directory che contiene quella in cui vi trovate, se esiste (ad esempio, passate da */percorso1/directory* a */percorso1*). Fa eccezione la root directory /: se vi trovate in questa e scrivete `cd ..` rimanete in /.
- `touch` *nome_file*
  Crea un file vuoto denominato *nome_file*, nella directory corrente.
- `clear`
  Elimina le informazioni presenti sulla schermata del terminale, riportando il prompt dei comandi in alto a sinistra. Combinazione di tasti alternativa: ctrl + L.

Shell di Linux, alcuni comandi di base - 3

- `mkdir` *percorso/nome_directory*.
  crea la directory *nome_directory* in percorso. Se *percorso* è vuoto, crea la directory *nome_directory* nel percorso in cui vi trovate.
- `cp` *percorso/file_1 percorso_destinazione*
  Copia il file *file_1* in *percorso_destinazione*.
- `mv` *percorso1/file1 percorso2/*.
  Sposta *file1* da *percorso1* a *percorso2*. E' anche possibile utilizzare mv per rinominare file: `mv` *percorso/file1 percorso/file2*.
- `rm` *percorso/nome_file*
  Elimina *nome_file* da *percorso*. Qualora *nome_file* sia una directory, dovete utilizzate l'opzione -r.
- `echo` *stringa*
  Stampa su standard output *stringa*.
- less *nome_file*
  Mostra il contenuto del file *nome_file*. Potete scorrere all'interno del file utilizzando le frecce $\uparrow$ e $\downarrow$: premete q per uscire.

Esercizi con la shell di Linux

- visualizzate il percorso della directory corrente (quale comando dovete usare?).
- spostatevi nella vostra home directory (scrivete `cd`).
- visualizzate l'elenco dei file e delle directory (testate `ls`,`ll`,`ls -l`).
- create le directory *test1* e *test2*.
- entrate nella directory *test1*.
- create il file di testo vuoto *filetest*, nella directory *test1* (usate il comando `touch`).
- visualizzate il contenuto del file *filetest* utilizzando il comando `less` (quale pulsante dovete premere per uscire da `less`?)
- rinominate il file *filetest* in *filetest2*.
- Tornate indietro di un livello, nel filesystem (`cd ..`).
- spostate il file *filetest2* nella directory *test2*.
- eliminate il file *filetest2* dalla directory *test2*.

Compilatori - 1

- Il C è un linguaggio compilato.
- Il ruolo del compilatore è quello di ottenere un file eseguibile, a partire da uno o più file sorgenti.
- Non vedremo in dettaglio il processo di compilazione, ma nella prima fase entra in gioco il *preprocessore*, sul quale faremo solo qualche cenno, nel corso.
- Un file eseguibile è un file che contiene delle istruzioni che sono direttamente eseguibili dalla macchina, senza la necessità di un interprete (nota: i comandi che abbiamo visto per la shell di Linux, vengono interpretati).
- A seconda del sistema operativo, le estensioni dei file eseguibili possono cambiare.
- Ad esempio, in ambiente Microsoft, vedrete file eseguibili con estensione `.exe` (in programmi molto datati potreste trovare degli eseguibili con estensione `.com`).

Compilatori - 2

- Il compilatore verifica la *correttezza sintattica* del programma.
- La *correttezza semantica* del programma non viene verificata dal compilatore.
- Pertanto, anche se il programma "compila", questo non garantisce la correttezza semantica dello stesso.
- In particolare, il compilatore può fornire due tipi di informazioni, come output di testo (se non vengono riscontrati problemi, il compilatore non mostra alcun messaggio, e vi appare il prompt dei comandi).
- **Errori di compilazione**: sono errori che bloccano l'operazione di compilazione, e che vanno assolutamente corretti. In caso di errori di compilazione, il file eseguibile NON viene generato.
- **Warnings**: sono avvertimenti che non bloccano la compilazione, ma vanno controllati con attenzione, dato che potrebbero essere la prefigurazione di possibili problemi in fase di esecuzione del programma.

Compilatori - 3

- Nel corso utilizzeremo il compilatore GCC (GNU Compiler Collection) della GNU, su ambiente Linux.
- Esistono altri compilatori, per vari sistemi operativi.
- I vari standard del C fissano il comportamento dei compilatori sotto certi aspetti, ma lasciano libertà implementativa sotto altri.
- Pertanto, se il programma non viene scritto tenendo conto di tali aspetti, è possibile che abbia un comportamento diverso a seconda del compilatore.
- Nel corso, se seguirete le indicazioni che vi daremo (a lezione e a laboratorio), non avrete tali problemi.

Un primo esercizio (guidato) di programmazione

- Creare una directory *laboratorio*, nella vostra home directory.
- Aprire un editor (ad esempio kate).
- scrivere un programma in C che stampi "Ciao Mondo" sullo schermo, e va a capo.
- Salvare il file sorgente come `esercizio1.c`, nella directory *laboratorio*.
- compilare il programma in modo che l'eseguibile abbia nome *eseg1*.

```
gcc esercizio1.c -o eseg1
```

- eseguire il programma.

```
./eseg1
```

Esercizio da eseguire in autonomia

- accedete alla directory *laboratorio*, nella vostra home directory.
- Aprire un editor (ad esempio kate).
- scrivere un programma in C che stampi "Ciao Terra" sullo schermo, e va a capo.
- Salvare il file sorgente come `esercizio2.c`, nella directory *laboratorio*.
- compilare il programma in modo che l'eseguibile abbia nome *eseg2*.
- eseguire il programma.