

# Introduzione al C

Corso di Programmazione di Sistema

Nicola Bicocchi

DIEF/UNIMORE

Aprile 2021

## Libro di testo “Programmare in C”

- Prefazione
- Cap. 1.0, 1.1, 2.1 (Introduzione)
- Cap. 10 (Makefile)

# Il linguaggio C

- È un linguaggio di programmazione general-purpose progettato inizialmente da Dennis Ritchie dei Bell Laboratories e implementato nel 1972. Inventato per sopperire ai limiti del linguaggio B e BCPL
- Nel 1983, National Standard Institute (ANSI) sviluppò il cosiddetto ANSI C o C standard
- Lo standard è stato poi aggiornato nel 1999 con l'introduzione di alcune varianti e chiarificazioni. Ulteriori aggiornamenti nel 2011 (C11) e nel 2018 (C18).
- Il linguaggio è di *alto livello* perché non ha una traduzione «diretta», ovvero 1 a 1, ovvero biunivoca, in linguaggio macchina, a differenza dell'assembly.
- Il linguaggio è di *basso livello* perché lo si può utilizzare come una specie di assembly poco più evoluto. Questo modo di usarlo è però poco lungimirante e decisamente sconsigliato.

# Ambito di utilizzo del C

- Sistemi Operativi e.g., Kernel Linux
- Programmi che interagiscono a basso livello con l'hardware o con il sistema operativo (e.g., driver, kernel Python)
- Sistemi embedded
- Librerie e routine ad alte performance

# Caratteristiche del C

- *Procedurale*: il programma è un insieme di *procedure* (funzioni). Non esiste supporto a strutture modulari più complesse come classi ed oggetti.
- *Compilato*: il codice sorgente deve essere trasformato in linguaggio macchina da un compilatore (e.g., gcc) *prima di essere eseguito*.
- *Tipizzato*: ogni variabile ha un tipo associato, lo sviluppatore deve sempre dichiarare il tipo prima di usare la variabile. E' però possibile utilizzare tipi alternativi per accedere al dato (i.e., lascamente tipizzato).

# Caratteristiche del C

- Il linguaggio è pensato per essere efficiente: lo sviluppatore ha il controllo completo su quello che succede.
- Commettere errori è più facile e subdolo: il linguaggio non permette al compilatore di rilevare gli errori con la completezza con cui lavorano interpreti come Java o Python.
- Gli errori possono produrre conseguenze gravi non esistendo una virtual machine (*sandbox*).

# Processo di compilazione

- Il compilatore è un programma apposito per convertire linguaggi arbitrari in codice macchina. Esempi di compilatori popolari:
  - GCC (the GNU Compiler Collection)
  - Microsoft Visual C(++)
  - ARM-GCC (ambito architetture proprietarie)
- L'insieme dei programmi utilizzati per gestire tutta la fase di compilazione è detta *toolchain*.

# Keywords

- **Codice sorgente:** file di testo che contiene il software scritto dallo sviluppatore
- **Programma eseguibile:** file binario che contiene il codice pronto per esecuzione



- I linguaggi sono creature vive e vengono arricchiti periodicamente con nuove funzionalità:
  - 1973: invenzione del linguaggio C da parte di Rennis Ritchie
  - 1990: definizione dello standard C89 o C90
  - 1999: definizione dello standard C99 (versione molto portabile e diffusa)
  - 2011: definizione dello standard C11
  - 2018: definizione dello standard C18 (versione recente, poco diffusa, stringhe unicode)
- La possibilità di utilizzare certe funzionalità del C dipende strettamente dal supporto del compilatore.

# Hello World!

```
1 #include <stdio.h>
2
3 int main(){
4     printf("Hello, World!\n");
5     return 0;
6 }
```

# Hello World! Direttiva include

Linea 1: **#** introduce una direttiva del pre-processore che **include (importa)** un file (**stdio.h**) da un percorso standard (<>)

```
1  #include <stdio.h>
2
3  int main(){
4      printf("Hello, World!\n");
5      return 0;
6  }
```

# Hello World! Funzione main()

Linea 3: **int** tipo del valore di ritorno della funzione, **main** nome della funzione, { inizio del corpo della funzione.

```
1  #include <stdio.h>
2
3  int main(){
4      printf("Hello, World!\n");
5      return 0;
6  }
```

# Hello World! Funzione main()

Linea 4: **printf** invocazione della funzione di libreria printf(), **Hello, World!** costante che contiene una stringa C terminata con carattere a capo **\n**.

Linea 5: **return** istruzione che termina la funzione e ritorna un valore (**0**).

```
1  #include <stdio.h>
2
3  int main(){
4      printf("Hello, World!\n");
5      return 0;
6  }
```

## \$ gcc -Wall -g -o helloworld helloworld.c

- Il comando compila il codice sorgente *helloworld.c* in un programma eseguibile di nome *helloworld*
  - -Wall attiva tutti i warnings (Warnings All)
  - -g mantiene i simboli (debug)
  - -o specifica il nome del file compilato (default=a.out)
- Se generato nella directory corrente, è possibile eseguire il programma con il comando `./helloworld`

# Ambienti di sviluppo

- CLion
- Microsoft Visual Studio
- Eclipse
- Visual Studio Code
- Sublime text, vim