

II Software

Moreno Marzolla

<http://www.moreno.marzolla.name/>

Copyright © 2011, Gianluca Amato
<http://fad.unich.it/course/view.php?id=12>

Copyright © 2011, Moreno Marzolla

<http://www.moreno.marzolla.name/teaching/LabInf2011/>



This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 (CC BY-SA 3.0) License. To view a copy of this license, visit

<http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Hardware & Software

Hardware e Software

- **Hardware**: tutto ciò che **ha** una consistenza fisica. Monitor, stampante, tastiera, mouse, etc. sono esempi di hardware.
- **Software**: tutto ciò che **non ha** consistenza fisica, quindi i programmi (Windows, Linux, Word, ...)
- Quando comprate dell'hardware, pagate un prezzo che comprende
 - Il costo per produrre fisicamente l'oggetto in questione.
 - Questo costo è irrisorio se si parla di software.
 - Il costo sostenuto per la fase di ricerca che ha portato allo sviluppo del prodotto.

Computer e Programmi

- Un **programma per computer** è una serie di istruzioni che istruiscono il computer su come svolgere determinate operazioni:
 - Cosa fare quando l'utente fa un click su una icona sullo schermo?
 - Cosa fare quando viene premuto il tasto Stampa?
 - Come fare a interpretare il contenuto di un DVD e visualizzarlo sullo schermo?
- Analogia “culinaria”
 - **Un programma è l'equivalente di una ricetta.**
 - **Il computer equivale a un cuoco.**
 - Così come il computer **esegue** un programma e ci da dei risultati sullo schermo o sulla stampante, il cuoco **esegue** la ricetta per produrre una pietanza

Programmi e linguaggi

- Per specificare le istruzioni si usano i **linguaggi di programmazione**.
 - C, Java, Pascal, etc..
- Problema: i computer eseguono un solo linguaggio di programmazione: il **linguaggio macchina**.
- Scrivere “a mano” programmi in linguaggio macchina è troppo laborioso
- Per tale motivo sono stati inventati i linguaggi di programmazione “di alto livello”

Compilatori



- **Codice binario:** (o anche codice oggetto) quando è scritto in linguaggio macchina
- **Codice sorgente:** quando è scritto in un linguaggio di programmazione ad altro livello.

Hello, world!

Programma in linguaggio C che stampa "Hello world"

```
#include <stdio.h>

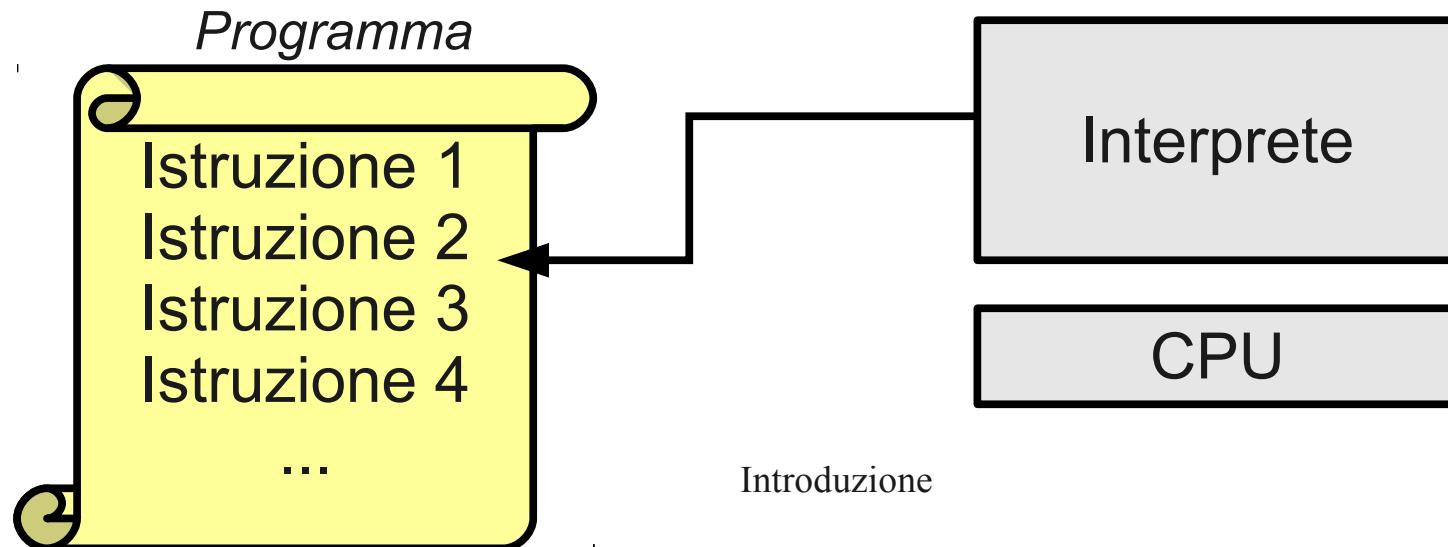
int main( void )
{
    printf("Hello, world\n");
    return 0;
}
```

Lo stesso programma in Assembler (versione "leggibile" del linguaggio macchina)

```
.file "prova.c"
.section .rodata
.LC0:
.string "Hello, world"
.text
.globl main
.type main, @function
main:
    pushl   %ebp
    movl   %esp, %ebp
    andl   $-16, %esp
    subl   $16, %esp
    movl   $.LC0, (%esp)
    call   puts
    movl   $0, %eax
    leave
    ret
    .size  main, .-main
    .ident "GCC: (Ubuntu 4.4.3-4ubuntu5) 4.4.3"
    .section .note.GNU-stack,"",@progbits
```

Interpreti

- Un linguaggio **interpretato** non richiede la fase di compilazione prima di essere eseguito
- Un interprete legge ed esegue le istruzioni ad alto livello
 - Pensate agli *interpreti* (umani!) che traducono da una lingua ad un'altra man mano che lo speaker parla



Il software libero

Licenze e codice sorgente

- I programmi sono generalmente sviluppati con linguaggi compilati, e vengono solitamente distribuiti soltanto sotto forma di **codice binario**.
- Il **codice sorgente** è tenuto segreto dal produttore
 - Per impedire che altri possano copiare le idee utilizzate nella scrittura del programma.
 - Anche quando la modifica del programma non è espressamente vietata, senza il codice sorgente è quasi impossibile farlo. Capire il funzionamento di un programma dal codice binario è estremamente difficile

Il software libero

- Nel 1984 Richard M. Stallman crea il **progetto GNU**: l'obiettivo è produrre un sistema operativo completamente libero.
- Poco dopo nasce la **Free Software Foundation**, una società senza fini di lucro il cui scopo è promuovere la diffusione del software libero.
- **Cosa vuol dire software libero?**
 - Stallman identifica quattro libertà principali che a un utente devono essere garantite.
 - Un programma la cui licenza garantisce all'utente queste libertà è un **software libero**.
 - L'opposto del software libero è il **software proprietario**.



Le quattro libertà fondamentali

<http://www.gnu.org/philosophy/free-sw.it.html>

- **Libertà 0: Eseguire il programma, per qualsiasi scopo**
 - Senza nessuna restrizione che ne impedisce, ad esempio, l'utilizzo per scopi commerciali.
- **Libertà 1: Studiare come funziona il programma e adattarlo alle proprie necessità**
- **Libertà 2: Distribuire copie del programma**
 - Un software libero può essere copiato, regalato a un amico
- **Libertà 3: Modificare il programma e distribuire le modifiche, in modo tale che tutta la comunità ne tragga beneficio.**

Il software libero è gratuito?

- **Software libero ≠ software gratis**
 - Se dispongo di un software libero, posso cederlo a chi voglio gratuitamente...
 - ... ma posso anche venderlo.
- Esistono molte **raccolte di software libero** su CD/DVD
 - Es., GNU/Linux è un software libero, si può scaricare da Internet senza pagare nulla...
 - ...ma se volete evitare la fatica di scaricarlo, potete comprare i DVD con il programma.
- Vendere software libero è un modo per ottenere fondi per finanziare lo sviluppo di altro software libero.

Altre categorie di software

- **Software Freeware**. Di solito si intende del software gratuito non modificabile. Non si ha accesso ai sorgenti e talvolta neanche possibilità di redistribuzione.
 - Esempio: Internet Explorer
- **Software Shareware**: software di cui è permessa la distribuzione ma che bisogna pagare per l'uso. Spesso non si ha accesso ai sorgenti.
 - Esempio: Winzip
- **Software commerciale**: è un software sviluppato da una azienda per trarre profitto dal suo uso. Può essere libero oppure proprietario.
 - Esempio di software commerciale ma libero: OpenOffice
 - Esempio di software commerciale proprietario: Microsoft Office

Vantaggi pratici

- **Il prezzo** (ma non è detto)
 - Posso anche pagarlo caro se richiedo un contratto di assistenza
- **Indipendenza dal fornitore**
 - Se un altro fornitore offre una soluzione migliore, sono libero di cambiare, senza perdere i dati.
- **Verificabilità del codice**
 - Posso essere sicuro che il programma non contenga nulla di insolito..
 - ...altrimenti, come posso essere sicuro che il programma che sto utilizzando non diffonda documenti privati a persone estranee?
- **Riutilizzabilità del codice**
 - Se il programma non fa esattamente quello che dovrebbe, posso adattarlo alle mie esigenze senza riscriverlo da capo!

Vantaggi sociali

- **Patrimonio pubblico**
 - Il software libero è un bene pubblico a disposizione di tutti. Costituisce una infrastruttura al servizio della società.
- **Accesso alla tecnologia**
 - Il software libero consente di superare il divario digitale che divide paesi ricchi e poveri.
- **Valore formativo**
 - La libertà di studiare e modificare il codice sorgente mette tutti in grado di imparare con software allo stato dell'arte.
- **Supporto alle economie locali**
 - La competizione si sposta dal software (in mano alle multinazionali) a quelli dei servizi (e le realtà locali competere).

In realtà...

- La cosa *realmente* importante per la maggior parte degli utenti sono i **dati** che vengono creati/manipolati con i vari programmi
 - Foto, documenti di testo, fogli elettronici, cartelle cliniche, risultati di esami medici...
- La domanda da porsi è la seguente: *“Che cosa succede dei miei dati se il programma X che li ha creati non è più disponibile?”*
 - Decido di cambiare sistema operativo, e il programma X non è disponibile sul nuovo sistema operativo
 - L'azienda che sviluppa il programma X fallisce
 - Le nuove versioni del programma X non riescono più a leggere i dati prodotti da quelle precedenti