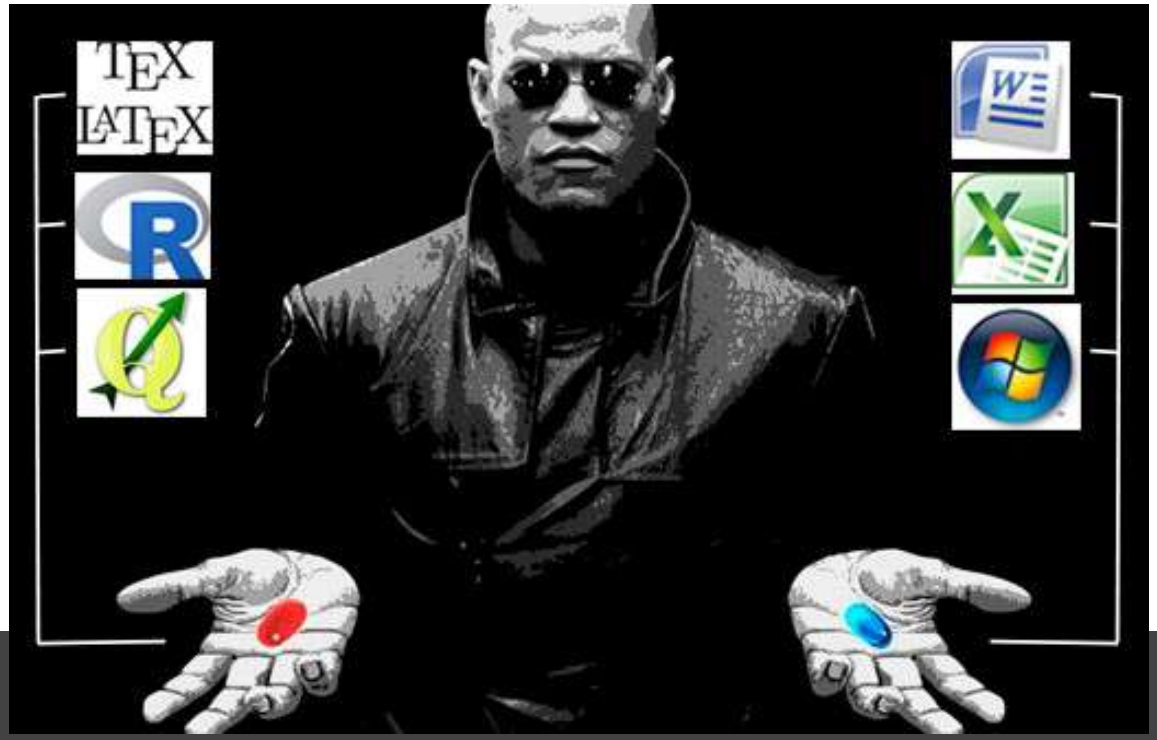


Parte VI: Gli approcci WYSIWYGe WYSIWYM, Una introduzione a LATEX

Introduzione a Latex

LaTeX è un sistema per la preparazione di testi basato sul programma di composizione tipografica TEX.

Fornisce funzioni di desktop publishing programmabili e mezzi per l'automazione della maggior parte della composizione tipografica, inclusa la numerazione, i riferimenti incrociati, tabelle e figure, organizzazione delle pagine, bibliografie e molto altro.

Introduzione

LATEX venne creato nel 1985 da Leslie Lamport ed è divenuto il principale metodo di utilizzo di TEX — poche persone usano ancora direttamente TEX base per la redazione di documenti. La versione attuale è denominata LATEX2ε.

ha trovato un'ampia diffusione nel mondo accademico, grazie all'ottimagestione dell'impaginazione delle formulematematiche ed alla gestione dei riferimenti bibliografici, resa possibile dal progetto gemello BibTeX.

Intruduzione

È distribuito con una licenza di software libero e questo lo ha reso disponibile per praticamente qualsiasi architettura: ne esistono pertanto versioni funzionanti per tutti i sistemi operativi, tra cui anche Microsoft Windows e MacOS X.

Introduzione

Al contrario di editor (meglio parlare di word processors) più conosciuti quali ad esempio Microsoft Word, WordPerfect, Works, StarOffice, che si basano sull'editoria WYSIWYG (*What You See Is What You Get*), **con LATEX si scrive un testo quale lo si pensa**. Scrivendo il *codice sorgente* (o più semplicemente, *sorgente*) sullo schermo, **il testo del documento risulta frammisto ad istruzioni**: per ottenere l'output finale è necessario che tale sorgente sia poi compilato. Questo approccio viene anche definito WYSIWYM (*What You See Is What You Mean*): con LATEX cioè **lo scrittore è anche tipografo** e deve occuparsi, all'inizio, soltanto delle convenzioni da usare. Fissate queste si concentra sul contenuto del testo, non curando invece ad esempio l'impaginazione, l'indice (generale e analitico), l'inserimento delle figure, delle tabelle, che sarà invece semi-automaticamente curata da LATEX.

WISIWIT, what I see is what I type
WYCIWYG, what you cache is what you get
WYGIWYG; what you get is what you get
WYGIWYS, what you get is what you see
WYSIAWYG; what you see is almost what you get
WYSIAYG, what you see is all you get
WYSIMOLWYG, what you see is more or less what you get
WYSINWYW, what you see is not what you want
WYSIPWYG, what you see is probably what you get
WYSIWYD, what you see is what you deserve
WYSIWYM, what you see is what you mean
WYSIWYN, what you see is what you need
WYSIWYS, what you see is what you sign
WYSIWYW, what you see is what you want
WYSYHYG, what you see you hope you get
YAFIYGI, you asked for it you got it

Computer Languages

- Machine
- Assembly
- Compiled
- Interpreted

- Low Level
- High Level

- 1GL
- 2GL
- 3GL
- 4GL
- …

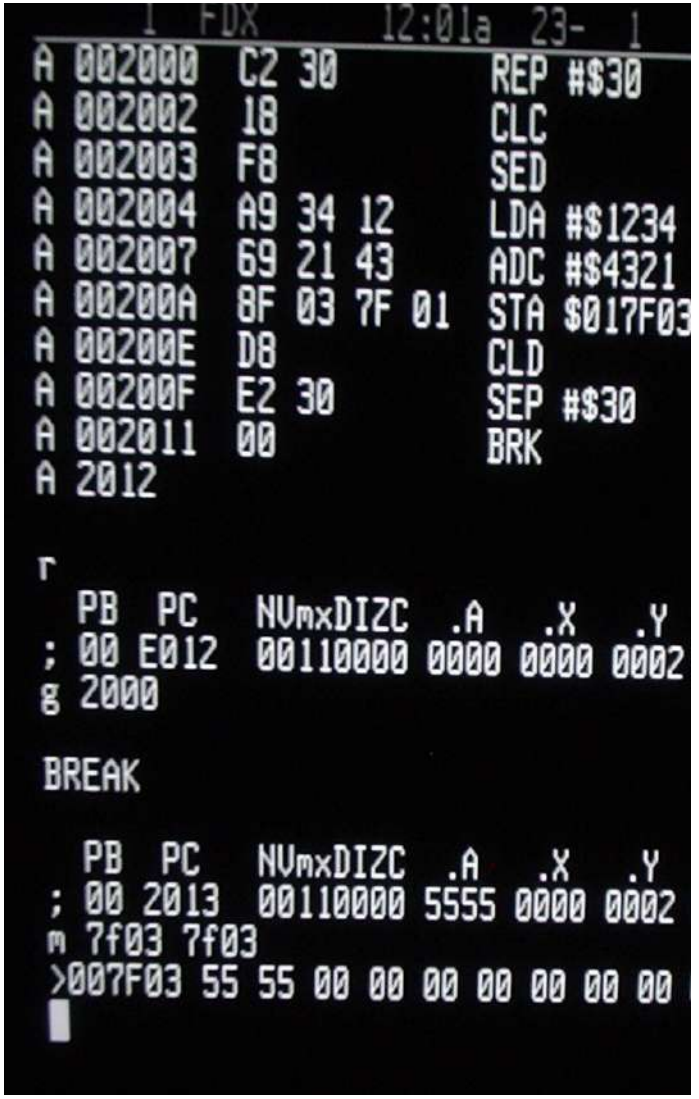
Computer Languages – Low Level

A **low-level programming language** is a programming language that provides little or no abstraction from a computer's instruction set architecture—commands or functions in the language map closely to processor instructions. Generally this refers to either machine code or assembly language. The word "low" refers to the small or nonexistent amount of **abstraction** between the language and machine language; because of this, low-level languages are sometimes described as being "close to the hardware". Programs written in low-level languages tend to be relatively non-portable

Computer Languages - Machine

Machine code or **machine language** is a set of instructions executed directly by a computer's central processing unit(CPU). Each instruction performs a very specific task, such as a load, a jump, or an ALU operation on a unit of data in a CPU register or memory. Every program directly executed by a CPU is made up of a series of such instructions. (The phrase 'directly executed' needs some clarification; machine code is by definition the lowest level of programming detail visible to the programmer, but internally many processors use microcode or optimise and transform machine code instructions into sequences of micro-ops in a sophisticated way.)

Cognitive science professor Douglas Hofstadter has compared machine code to genetic code, saying that "Looking at a program written in machine language is vaguely comparable to looking at a DNA molecule atom by atom."



Computer Languages - Machine

True machine code is a stream of raw, usually binary, data. A programmer coding in "machine code" normally codes instructions and data in a more readable form such as decimal, octal, or hexadecimal which is translated to internal format by a program called a loader or toggled into the computer's memory from a front panel.

A function in hexadecimal representation of
32-bit x86 machine code to calculate
the nth Fibonacci number:

8B542408 83FA0077 06B80000 0000C383
FA027706 B8010000 00C353BB 01000000
B9010000 008D0419 83FA0376 078BD989
C14AEBF1 5BC3

Computer Languages – Assembly

An **assembly** (or **assembler**) **language**, often abbreviated **asm**, is a low-level programming language for a computer, or other programmable device, in which there is a very strong (but often not one-to-one) correspondence between the language and the architecture's machine code instructions. Each assembly language is specific to a particular computer architecture.

Assembly language uses a mnemonic to represent each low-level machine instruction or opcode, typically also each architectural register, flag, etc. Many operations require one or more operands in order to form a complete instruction and most assemblers can take expressions of numbers and named constants as well as registers and labels as operands, freeing the programmer from tedious repetitive calculations.

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

                        ************************************
                        * FUNCTION: INITA - Initialize ACIA
                        * INPUT: none
                        * OUTPUT: none
                        * CALLS: none
                        * DESTROYS: acc A

0013            RESETA  EQU     %00010011
0011            CTLREG  EQU     %00010001

C003 86 13      INITA   LDA A   #RESETA   RESET ACIA
C005 B7 80 04           STA A   ACIA
C008 86 11              LDA A   #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A   ACIA

C00D 7E C0 F1           JMP     SIGNON    GO TO START OF MONITOR

                        ************************************
                        * FUNCTION: INCH - Input character
                        * INPUT: none
                        * OUTPUT: char in acc A
                        * DESTROYS: acc A
                        * CALLS: none
                        * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04   INCH    LDA A   ACIA      GET STATUS
C013 47                 ASR A             SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC     INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A   ACIA+1    GET CHAR
C019 84 7F              AND A   #$7F      MASK PARITY
C01B 7E C0 79           JMP     OUTCH     ECHO & RTS

                        ************************************
                        * FUNCTION: INHEX - INPUT HEX DIGIT
                        * INPUT: none
                        * OUTPUT: Digit in acc A
                        * CALLS: INCH
                        * DESTROYS: acc A
                        * Returns to monitor if not HEX input

C01E 8D F0      INHEX   BSR     INCH      GET A CHAR
C020 81 30              CMP A   #'0       ZERO
C022 2B 11              BMI     HEXERR    NOT HEX
C024 81 39              CMP A   #'9       NINE
C026 2F 0A              BLE     HEXRTS    GOOD HEX
C028 81 41              CMP A   #'A
C02A 2B 09              BMI     HEXERR    NOT HEX
C02C 81 46              CMP A   #'F
C02E 2E 05              BGT     HEXERR
C030 80 07              SUB A   #7        FIX A-F
C032 84 0F      HEXRTS  AND A   #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF   HEXERR  JMP     CTRL      RETURN TO CONTROL LOOP
```

Computer Languages – Assembly

The same Fibonacci number but in x86 assembly language using MASM syntax:

```
fib:
        mov edx, [esp+8]
        cmp edx, 0
        ja @f
        mov eax, 0
        ret

@@:
        cmp edx, 2
        ja @f
        mov eax, 1
        ret

@@:
        push ebx
        mov ebx, 1
        mov ecx, 1

@@:
        lea eax, [ebx+ecx]
         cmp edx, 3
         jbe @f
        mov ebx, ecx
        mov ecx, eax
        dec edx
        jmp @b

@@:
        pop ebx
        ret
```

Computer Languages – High Level

In computer science, a **high-level programming language** is a programming language with strong abstraction from the details of the computer. It may use natural language elements, be easier to use, or may automate (or even hide entirely) significant areas of computing systems (e.g. memory management), making the process of developing a program simpler and more understandable relative to a lower-level language. The amount of abstraction provided defines how "high-level" a programming language is.

"High-level language" refers to the higher level of abstraction from machine language. Rather than dealing with registers, memory addresses and call stacks, high-level languages deal with variables, arrays, objects, complex arithmetic or boolean expressions, subroutines and functions, loops, threads, locks, and other abstract computer science concepts, with a focus on usability over optimal program efficiency.

Computer Languages – Compiled

A **compiled language** is a programming language whose implementations are typically compilers (translators that generate machine code from source code), and not interpreters (step-by-step executors of source code, where no pre-runtime translation takes place). In principle, any language can be implemented with a compiler or with an interpreter.[citation needed] A combination of both solutions is also common: a compiler can translate the source code into some intermediate form (often called p-code or bytecode), which is then passed to an interpreter which executes it.

Programs compiled into native code at compile time tend to be faster than those translated at run time

Low-level programming languages are typically compiled, especially when efficiency is the main concern, rather than cross-platform support.

For such languages, there are more one-to-one correspondences between the programmed code and the hardware operations performed by machine code, making it easier for programmers to control use of central processing unit (CPU) and memory in fine detail.

Computer Languages – Compiled

Fibonacci function in C:

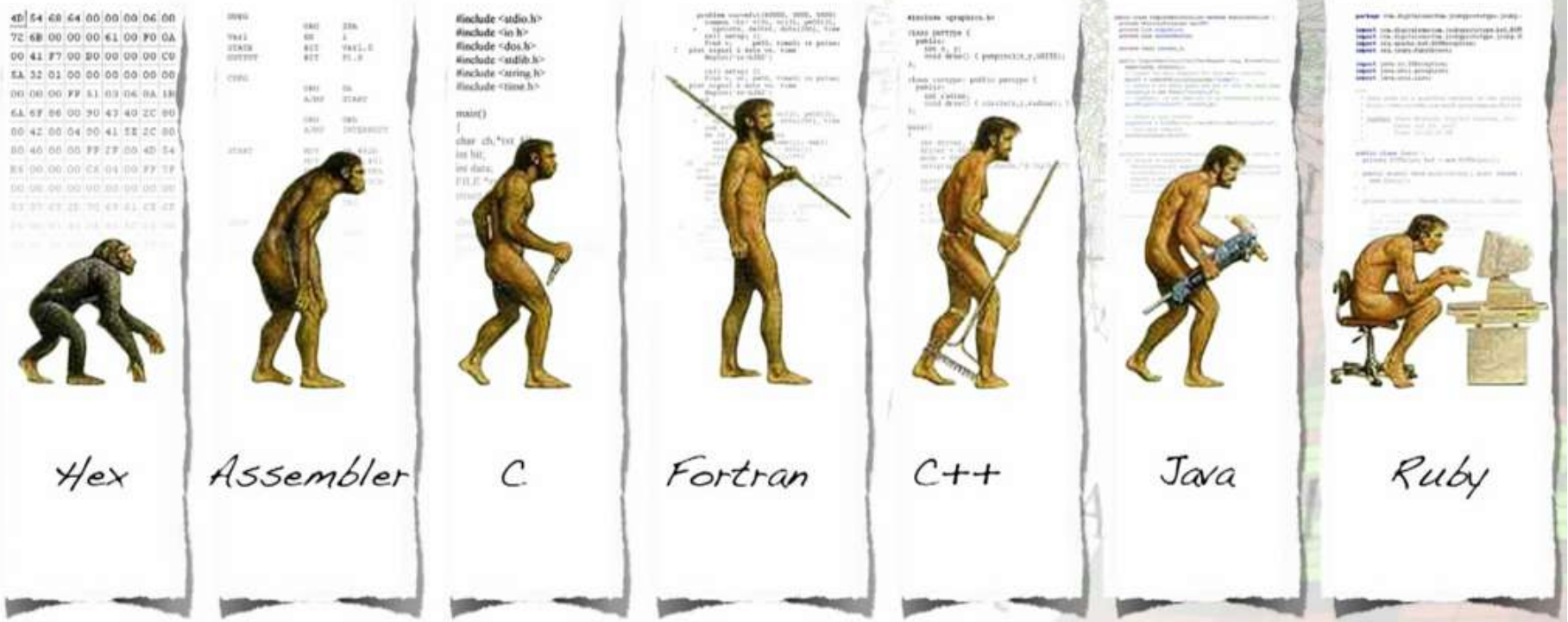
```
unsigned int fib(unsigned int n) {
        if (n <= 0) return 0;
        else if (n <= 2) return 1;
        else {
                unsigned int a,b,c; a = 1; b = 1;
                while (1) {
                        c = a + b;
                        if (n <= 3) return c;
                        a = b; b = c; n--;
                        }
        }
}
```

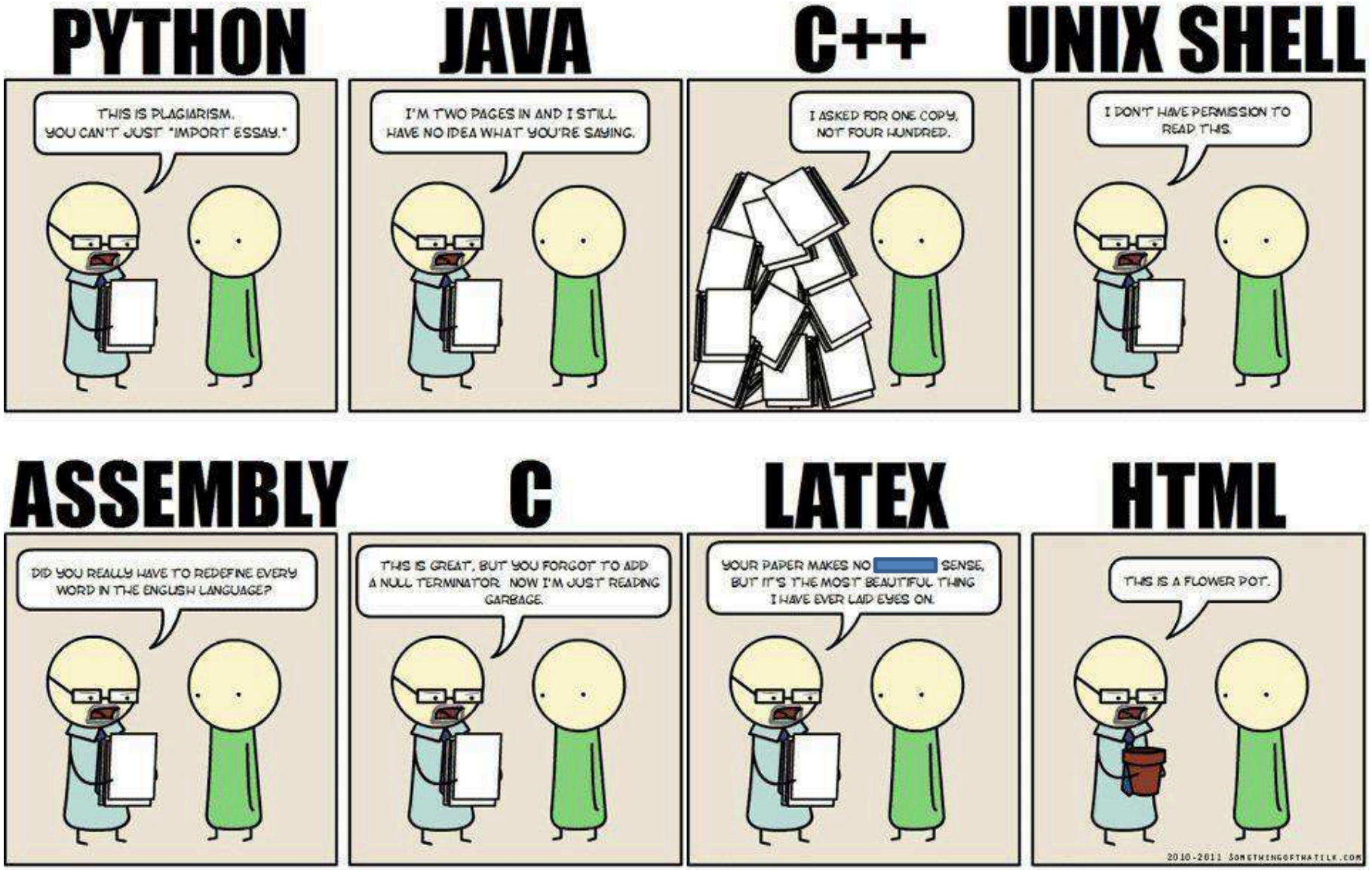
Computer Languages – Interpreted

An **interpreted language** is a type of programming language for which most of its implementations execute instructions directly and freely, without previously compiling a program into machine-language instructions. The interpreter executes the program directly, translating each statement into a sequence of one or more subroutines already compiled into machine code.
The terms interpreted language and compiled language are not well defined because, in theory, any programming language can be either interpreted or compiled. In modern programming language implementation it is increasingly popular for a platform to provide both options.
Interpreted languages can also be contrasted with machine languages. Functionally, both execution and interpretation mean the same thing — fetching the next instruction/statement from the program and executing it.

In principle, programs in many languages may be compiled or interpreted, emulated or executed natively, so this designation is applied solely based on common implementation practice, rather than representing an essential property of a language

The Evolution Of Computer Programming Languages




PYTHON
THIS IS PLAGIARISM. YOU CAN'T JUST "IMPORT ESSAY."
JAVA
I'M TWO PAGES IN AND I STILL HAVE NO IDEA WHAT YOU'RE SAYING.
C++
I ASKED FOR ONE COPY, NOT FOUR HUNDRED.
UNIX SHELL
I DON'T HAVE PERMISSION TO READ THIS.
ASSEMBLY
DID YOU REALLY HAVE TO REDEFINE EVERY WORD IN THE ENGLISH LANGUAGE?
C
THIS IS GREAT, BUT YOU FORGOT TO ADD A NULL TERMINATOR. NOW I'M JUST READING GARBAGE.
LATEX
YOUR PAPER MAKES NO [redacted] SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.
HTML
THIS IS A FLOWER POT.
2010-2011 SOMETHINGOFTHATILK.COM

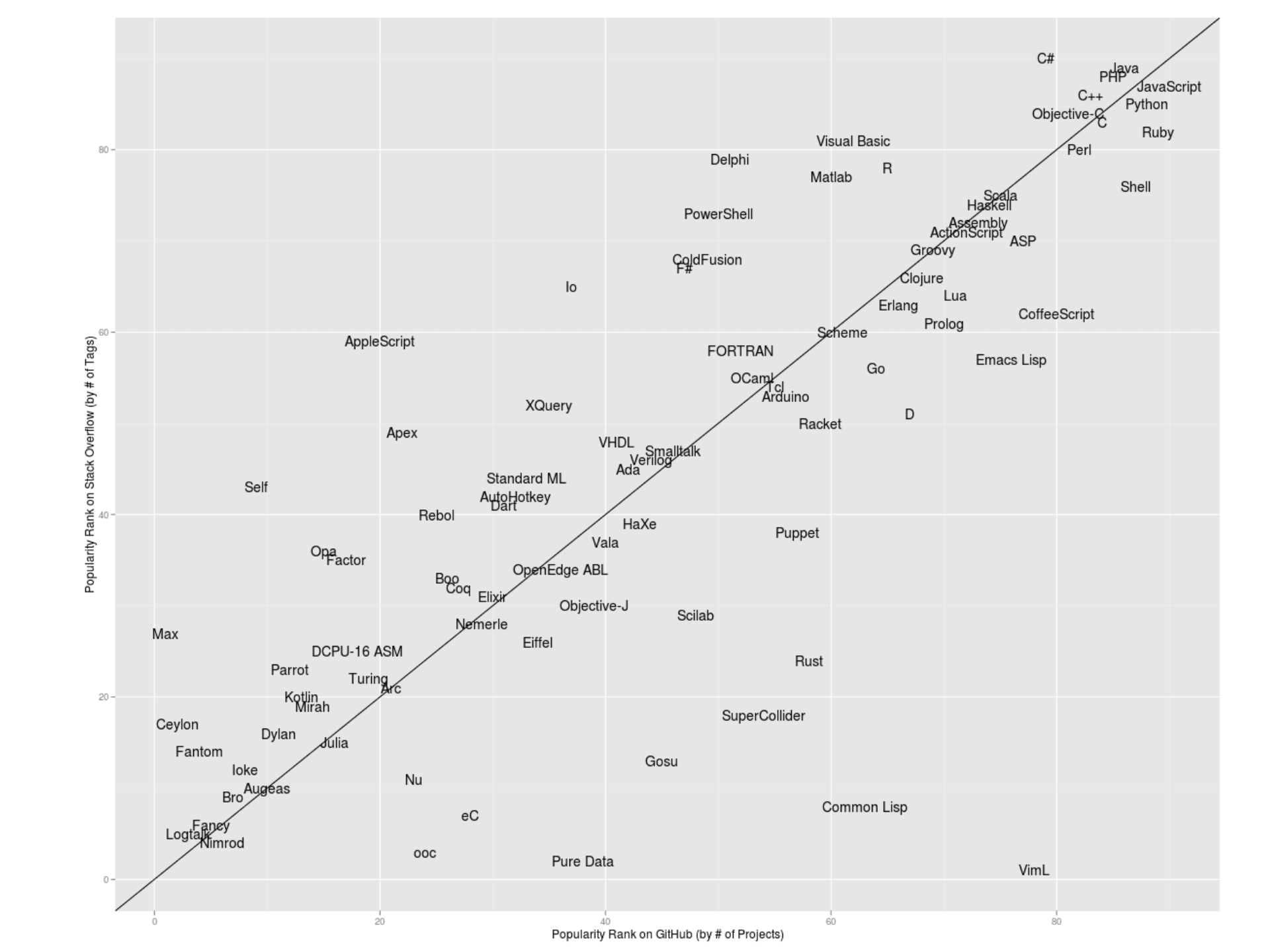
Computer Languages

A **programming language** is a formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.

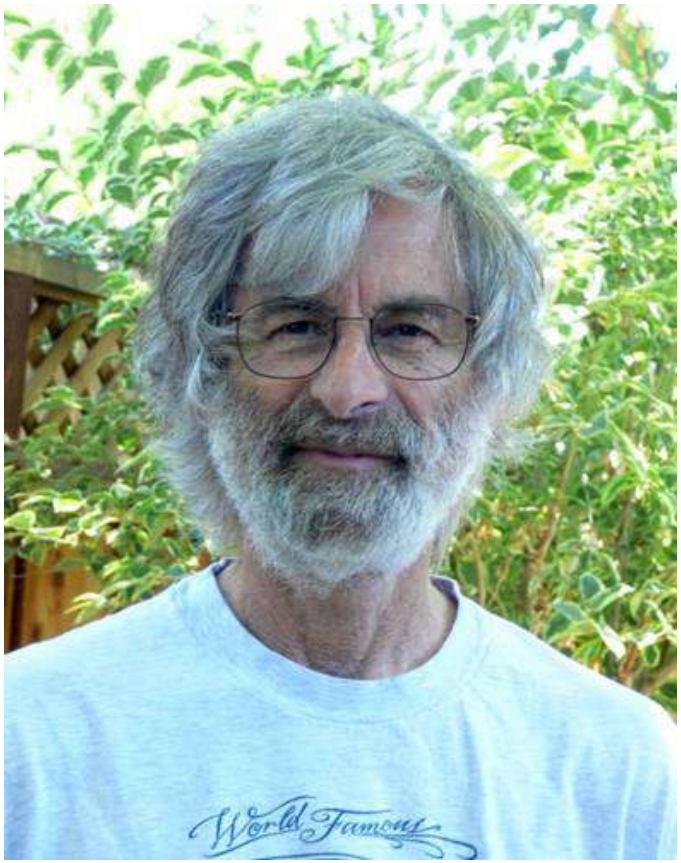
A **markup language** is a system for annotating a document in a way that is syntactically distinguishable from the text. The idea and terminology evolved from the "marking up" of paper manuscripts, i.e., the revision instructions by editors, traditionally written with a blue pencil on authors' manuscripts.[citation needed] In digital media, this "blue pencil instruction text" was replaced by tags, that is, instructions are expressed directly by tags or "instruction text encapsulated by tags."

Query languages or **data query languages** (DQLs) are computer languages used to make queries in databases and information systems.

A **style sheet language**, or **style language**, is a computer language that expresses the presentation of structured documents. One attractive feature of structured documents is that the content can be reused in many contexts and presented in various ways. Different style sheets can be attached to the logical structure to produce different presentations.


C#
Java
PHP
JavaScript
C++
Python
Objective-C
C
Ruby
Perl
Visual Basic
Delphi
R
Matlab
Shell
Scala
Haskell
PowerShell
Assembly
ActionScript
ASP
Groovy
ColdFusion
F#
Clojure
Io
Lua
Erlang
CoffeeScript
Prolog
Scheme
AppleScript
FORTRAN
Emacs Lisp
Go
OCaml
Tcl
Arduino
XQuery
D
Racket
Apex
VHDL
Smalltalk
Verilog
Ada
Standard ML
Self
AutoHotkey
Dart
Rebol
HaXe
Puppet
Vala
Opa
Factor
OpenEdge ABL
Boo
Coq
Elixir
Objective-J
Scilab
Nemerle
Max
Eiffel
DCPU-16 ASM
Rust
Parrot
Turing
Arc
Kotlin
Mirah
SuperCollider
Ceylon
Dylan
Julia
Fantom
Gosu
Ioke
Nu
Augeas
Bro
Common Lisp
eC
Fancy
Logtalk
Nimrod
ooc
Pure Data
VimL
0
20
40
60
80
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)

Tex e LaTex



- Tex: Donald E. Knuth realizza un programma per la composizione di testo e formule matematiche (1977).
- LaTex: Leslie Lamport scrive un pacchetto di macro che permette agli autori di impaginare e stampare documenti con elevata qualità tipografica (1985).

Pronouncing and writing "LaTeX"

The final consonant of TeX (on which LaTeX is based) is intended by its developer to be pronounced similar to 'loch' or 'Bach'. However, English speakers often pronounce it /ˈtɛk/, like the first syllable of technical.
The characters T, E, X in the name come from the Greek capital letters tau, epsilon, and chi, as the name of TeX derives from the Greek: τέχνη (skill, art, technique); for this reason, TeX's creator Donald Knuth promotes a pronunciation of /tɛx/ (tekh) (that is, with a voiceless velar fricative as in Modern Greek, similar to the ch in loch). Lamport writes "TeX is usually pronounced tech, making lah-teck, lah-teck, and lay-teck the logical choices; but language is not always logical, so lay-tecks is also possible."
The name is traditionally printed in running text with a special typographical logo: LATEX. In media where the logo cannot be precisely reproduced in running text, the word is typically given the unique capitalization LaTeX. The TeX, LaTeX and XeTeX logos can be rendered via pure CSS and XHTML for use in graphical web browsersfollowing the specifications of the internal \LaTeX macro.

Introduzione

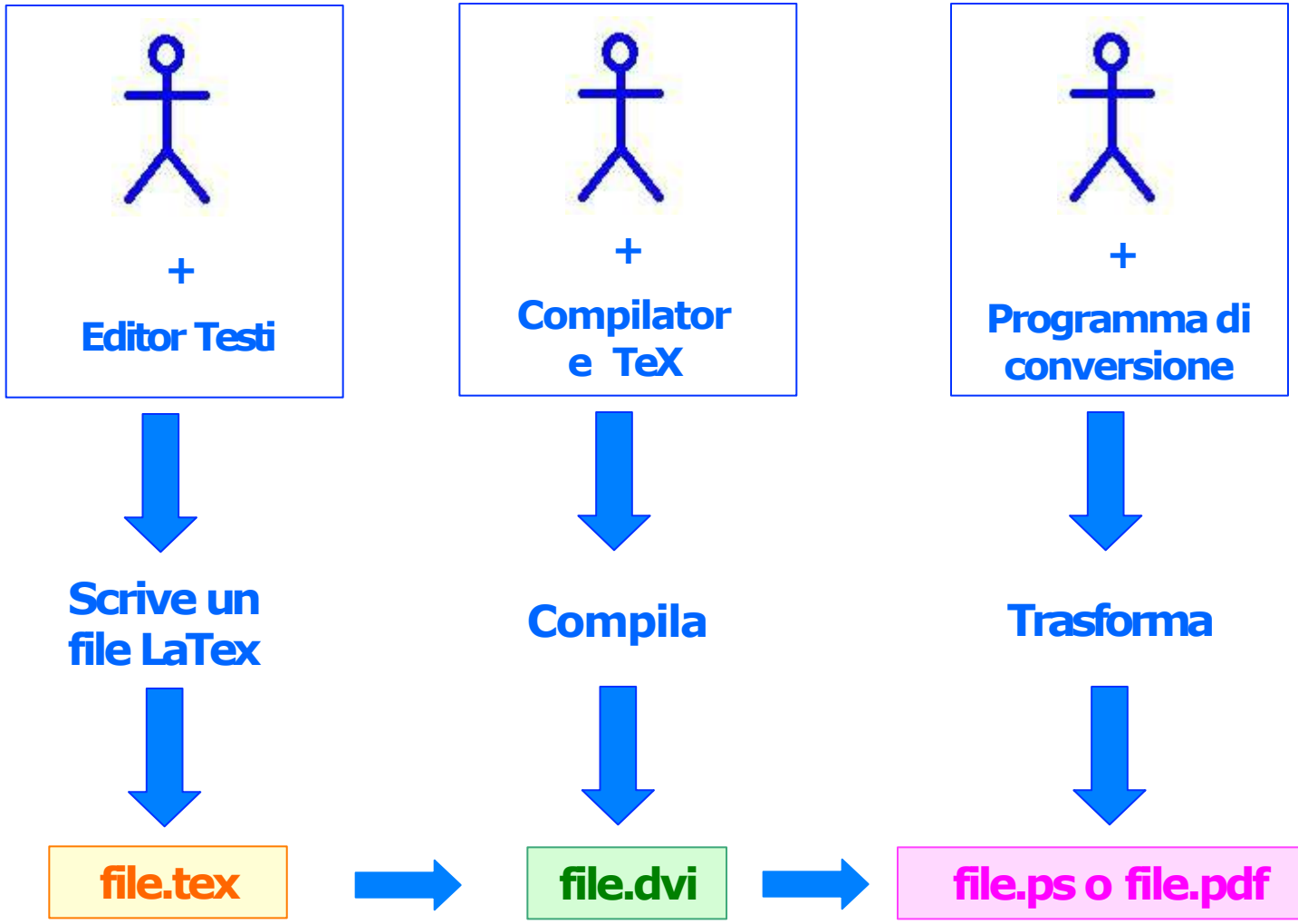
Il file prodotto da LATEX era, in passato, esclusivamente in formato DVI. Grazie al contributo degli sviluppatori della comunità *open source* ora si può ottenere un file nel più comune e diffuso standard Portable Document Format (PDF) ed anche in HTML: in quest'ultimo caso però le eventuali formule matematiche presenti verranno incluse in formato grafico, come se fossero immagini. È anche possibile, partendo da un file compilato con LATEX, ottenere un qualsiasi altro formato, anche .doc di Microsoft Word.

Introduzione

- LaTex è un sistema di composizione di testo che garantisce alta qualità tipografica
  - Documenti scientifici
  - Documenti matematici
  - Editoria elettronica

- Contenuto del documento e presentazione del contenuto vengono separati
  - Uso di marcatori

Passi principali



File di input in LaTex: struttura

```
\documentclass{...}
\usepackage{...}

\begin{document}

    Corpo del
    documento
    Testo +
    Marcatori

\end{document}
```

```
\documentclass{article}
\usepackage{amsmath}
\title{\LaTeX}
\begin{document}
\maketitle
\LaTeX{} is a document preparation system for the
\TeX{} typesetting program. It offers
programmable desktop publishing features and
extensive facilities for automating most aspects of
typesetting and desktop publishing, including
numbering and cross-referencing, tables and
figures, page layout, bibliographies, and much
more. \LaTeX{} was originally written in 1984 by
Leslie Lamport and has become the dominant
method for using \TeX; few people write in plain
\TeX{} anymore. The current version is \LaTeXe. %
This is a comment, not shown in final output. %
The following shows typesetting power of LaTeX:
\begin{align} E_0 &= mc^2 \\ E &=
\frac{mc^2}{\sqrt{1-\frac{v^2}{c^2}}} \end{align}
\end{document}
```

LaTeX

LaTeX is a document preparation system for the TeX typesetting program. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout, bibliographies, and much more. LaTeX was originally written in 1984 by Leslie Lamport and has become the dominant method for using TeX; few people write in plain TeX anymore. The current version is LaTeX 2_ε.

$$E_0 = mc^2 \tag{1}$$

$$E = \frac{mc^2}{\sqrt{1-\frac{v^2}{c^2}}} \tag{2}$$

File di input in LaTex: spazi

- Spazio e tabulazione sono trattati indifferentemente come spazio.
- Più caratteri consecutivi di spazio sono considerati come *un solo* spazio.
- Una riga vuota tra due righe di testo delimita un paragrafo.

```
Tanti spazi      dopo una
parola    non vengono
considerati.

Una riga vuota fa iniziare
un nuovo paragrafo.
```

input:
file.tex

Tanti spazi dopo una parola non vengono considerati.

Una riga vuota fa iniziare un nuovo paragrafo.

output:
file.dvi

File di input in LaTex: caratteri speciali

- Iseguenti simboli sono caratteri riservati
  - `$ & % # _ { } ~ ^ \`
- Alcuni possono essere usati nei documenti se preceduti da un *backslash*

```
\$ \& \% \# \_ \{ \}
```

input: file.tex

$ & % # _ { }

output: file.dvi

- Gli altri simboli e molti altri possono essere stampati con comandi speciali.
- La sequenza `\\` si usa per le interruzioni di riga.

File di input in LaTex: comandi

- I comandi LaTex sono *case sensitive*
- Iniziano con `\` e poi hanno un nome composto da sole lettere che termina con:
  - uno spazio
  - un numero
  - un carattere "non lettera"
- Sono costituiti da `\` ed un carattere speciale

```
Luogo: Verona.\\
Data: \today
```

input: file.tex

Luogo: Verona.
Data: 30 agosto 2006

output: file.dvi

File di input in LaTex: comandi (2)

- Alcuni comandi necessitano di un parametro che deve essere fornito tra parentesi graffe { }
- Alcuni comandi accettano parametri opzionali che si aggiungono dopo il nome del comando tra parentesi quadre [ ]

```
Per scrivere in
\textit{corsivo} o
\textbf{grassetto} uso
un comando con
parametro.\newline Posso
scrivere anche
{\small piccolo} o
{\LARGE grande}.
```

input: file.tex

Per scrivere in *corsivo* o **grassetto** uso un comando con parametro.
Posso scrivere anche piccolo o grande.

output: file.dvi

Sintassi comandi LaTex

- Sintassi generale comandi LaTex

  `\comando[opzione]{parametro}`

- Comando senza parametri:

  `\appendix`

- Comando con parametri:

  `\chapter{titolo}`

- Comando con parametri opzionali:

  `\item[nome]`

- Comando con parametri opzionali ed obbligatori:

  `\cite[pag. 150]{lamport-latex}`

File di input in LaTex: commenti

- `%` ignora il resto della riga attuale, l'interruzione di riga e tutti gli spazi bianchi all'inizio della riga successiva.

```
Prova a dire: % se
% riesci
Supercal%
       ifragilist%
   ichespiralidoso.
```

Prova a dire: Super-califragilistichespiralidoso.

```
Ecco come inserire un commento
\begin{comment}
utile
\end{comment}
in un documento.
```

Ecco come inserire un commento in un documento.

- Ambiente `comment` per commenti lunghi.

File di input in LaTex: esempio 2

```
\documentclass[11pt]{article}

\begin{document}
 Giro giro tondo,\\
 casca il mondo,\\
 casca la terra;\\
 tutti gi\`u per terra.
\end{document}
```

Giro giro tondo,
casca il mondo,
casca la terra;
tutti giù per terra.

La classe di un documento

- `\documentclass{classe}`

  la classe definisce il tipo di documento:

  - **`article`**

    classe utilizzata per i documenti più corti
  - **`book`**

    utilizzata per i libri: gestisce la suddivisione in capitoli
  - **`letter`**

    permette di scrivere lettere
  - **`slides`**

    serve per comporre lucidi

Le opzioni di classe

- `\documentclass[opzioni]{classe}`

  le opzioni modificano l'impaginazione di un documento:

  - **`a4paper`**

    permette di impostare i margini del documento per la stampa su A4
  - **`11pt,12pt,...`**

    imposta la dimensione del font principale (default 10pt)
  - **`twocolumn`**

    per impaginare il documento su 2 colonne

Le estensioni

- **`\usepackage[opzione]{estensione}`**

  le opzioni modificano l'impaginazione di un documento:

  - **`fontenc`**

    utilizzata con l'opzione **`T1`** permette di utilizzare la codifica di caratteri T1 (standard LaTex)

  - **`inputenc`**

    con l'opzione **`latin1`** permette di utilizzare dei caratteri ISO-8859-1 nel documento (lettere accentate)

  - **`babel`**

    opzione non standard che, combinata con l'opzione **`italian`** adatta LeTex alle convenzioni tipografiche italiane

Creare un file LaTex

- Usare l'editor per creare:

```
\documentclass[a4paper,11pt]{article}
\usepackage[latin1]{inputenc}
\usepackage[italian]{babel}

\begin{document}
 Il mio primo documento \LaTeX.
\end{document}
```

- Salvare il file con il nome **prova.tex**
- Compilare con il comando **latex prova.tex**
- **Visualizzare con xdvi prova.dvi**
- **Convertire in PostScript con**

  dvips prova.dvi -o

Le lettere accentate

Input	Abbreviazione	Output
\`{e}	\`e	è
\'{e}	\'e	é
\^{e}	\^e	ê
\"{e}	\"e	ë
\~{a}	\~a	ã
...	...	...

Attenzione!!!

\`i \`e leggermente diverso da \`{i}.

Verificate.

Dimensione del carattere

Input	Output
`\tiny`	dimensione carattere
`\scriptsize`	dimensione carattere
`\footnotesize`	dimensione carattere
`\small`	dimensione carattere
`\normalsize`	dimensione carattere
`\large`	dimensione carattere
`\Large`	dimensione carattere
`\LARGE`	dimensione carattere
`\huge`	dimensione carattere
`\Huge`	dimensione carattere

Titoli, capitoli e sezioni

- Per la classe **`article`** sono disponibili i seguenti comandi di sezionamento del testo:
  - `\section{titolo}`
  - `\subsection{titolo}`
  - `\subsubsection{titolo}`
  - `\appendix`
- Per la classe **`book`** oltre ai precedenti, sono disponibili **anche** i seguenti comandi di sezionamento del testo:
  - `\part{titolo}`
  - `\chapter{titolo}`

Chapters and Sections

The command \section{} marks the beginning of a new section, inside the braces is set the title. Section numbering is automatic and can be disabled by including a * in the section command as \section*{}. We can also have \subsection{}s, and indeed \subsubsection{}s. The basic levels of depth are listed below:

-1	\part{part}
0	\chapter{chapter}
1	\section{section}
2	\subsection{subsection}
3	\subsubsection{subsubsection}
4	\paragraph{paragraph}
5	\subparagraph{subparagraph}

Note that **\part** and **\chapter** are only available in *report* and *book* document classes.

\chapter{First Chapter}

\section{Introduction}
This is the first section. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortisfacilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales…

\section{Second Section}
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante…

\subsection{First Subsection}
Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales…

\section*{Unnumbered Section}
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem

Chapter 1

First Chapter

1.1 Introduction

This is the first section.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortisfacilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales...

1.2 Second Section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi necante...

1.2.1 First Subsection

Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales...

Unnumbered Section

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisissem

Indice

- LaTex crea l'indice prendendo i titoli delle sezioni e i numeri di pagina dell'ultima compilazione.
- Il comando **`\tableofcontents`** produce l'indice nel punto in cui è inserito.
- Per ottenere un indice corretto il documento deve essere compilato due volte.
- Per generare titoli che non compaiono nell'indice (e non sono numerati) si usa le versione "asterisco" del comando

  Esempio: **`\section*{titolo}`**

Riferimenti incrociati

- **\label{segnalibro}** e **\ref{segnalibro}**

  LaTex sostituisce a **\ref** il numero della sezione, sottosezione, figura, tabella o teorema dopo il quale si trova il corrispondente comando **\label**.

- **\label{segnalibro}** e **\pageref{segnalibro}**

  Il comando **\pageref** stampa il numero di pagina del corrispondente comando **\label**.

```
\section{Introduzione}\label{Intro}
bla bla bla ... come vedremo in
Sezione~\ref{RelWork}.

\section{Related Work}\label{RelWork}
bla bla bla ...
```

1 Introduzione
bla bla bla ... come vedremo in Sezione 2.

2 Related Work
bla bla bla

Ambienti

- Un generico ambiente si invoca con

```
\begin{ambiente}
  ...
\end{ambiente}
```

dove **ambiente** è il nome dell'ambiente (environment).

- Gli ambienti possono essere chiamati l'uno all'interno dell'altro

```
\begin{ambienteAAA}
...
   \begin{ambienteBBB}
     ...
   \end{ambienteBBB}
...
\end{ambiente}
```

Elenchi puntati e numerati

- L'ambiente **itemize** consente di comporre liste puntate
- L'ambiente **enumerate** consente di comporre liste numerate

```
\begin{enumerate}
 \item La lista della spesa:
  \begin{itemize}
   \item Pane
   \item Pasta
   \item Latte
  \end{itemize}
 \item Ricordare:
  \begin{itemize}
   \item[-] Chiamare dentista
   \item[+] Pagare bollo
  \end{itemize}
 \item ....
\end{enumerate}
```

1. La lista della spesa:
   - Pane
   - Pasta
   - Latte
2. Ricordare:
   - Chiamare dentista
   + Pagare bollo
3. ...

Tabelle

```
\begin{center}
\begin{tabular}{ c c c }

cell1 & cell2 & cell3 \\
cell4 & cell5 & cell6 \\
cell7 & cell8 & cell9

\end{tabular}
\end{center}
```

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

Tabelle

```
\begin{center}
\begin{tabular}{| |c c c c| |}
\hline
Col1 & Col2 & Col2 & Col3 \\ [0.5ex]
\hline
\hline
1 & 6 & 87837 & 787 \\
\hline
2 & 7 & 78 & 5415 \\
\hline
3 & 545 & 778 & 7507 \\
\hline 4 & 545 & 18744 & 7560 \\
\hline 5 & 88 & 788 & 6344 \\ [1ex]
\hline
\end{tabular}
\end{center}
```

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Tabelle

```
\begin{tabular}{|r|l|}
 \hline
 {\bf Studente} & {\it Voto} \\
 \hline
 Pippo     &  30  \\
 \hline
 Pluto     &  28  \\
 \hline
\end{tabular}
```

Studente	*Voto*
Pippo	30
Pluto	28

L'ambiente **verbatim**

- Per stampare un testo come se fosse battuto a macchina, con tutti gli spazi e le interruzioni di linea
- I comandi latex e i caratteri speciali non vengono interpretati

```
\begin{verbatim}
Questo
   ambiente
    funziona
 cos \`{\i}.
\end{verbatim}

Oppure:
\verb+Cosi'+
```

```
Questo
   ambiente
    funziona
 cos \`{\i}.

Oppure:
Cosi'
```

Oggetti mobili: figure e tabelle

- `\begin{figure}[posizionamento]` e

  `\begin{table}[posizionamento]`

  il parametro **`posizionamento`** permette di dire a LaTex dove gli oggetti possono essere spostati
  - **`h`** (here): nel punto esatto in cui compare il testo
  - **`t`** (top): in cima ad una pagina
  - **`b`** (bottom): in fondo ad una pagina
  - **`p`** (page): in una pagina speciale contenente solo oggetti mobili

  Esempio:
  ```
  \begin{figure}[htpb]
      ...
  \end{figure}
  ```

Inserimento figura

```
\documentclass{article}
...
\usepackage{epsfig}

\begin{document}

\begin{figure}[htpb]
 \begin{center}
  \epsfig{file=./Images/figura1.eps,width=0.8\textwidth}
  \caption{Descrizione figura1}
  \label{segnalibro_figura1}
 \end{center}
\end{figure}

... come mostrato in Figura~\ref{segnalibro_figura1} ...

\end{document}
```

Inserimento figura (2)

```
\documentclass{article}
...
\usepackage[dvips]{graphicx}

\begin{document}

\begin{figure}[htpb]
 \begin{center}
  \includegraphics[width=0.8\textwidth]{figura2}
  \caption{Descrizione figura2}
  \label{segnalibro_figura2}
 \end{center}
\end{figure}

... come mostrato in Figura~\ref{segnalibro_figura2} ...

\end{document}
```

Inserimento tabella

```
\documentclass{article}
...

\begin{document}

\begin{table}[htpb]
 \begin{center}
  \begin{tabular}
  ...
  \end{tabular}
  \caption{Descrizione tabella1}
  \label{segnalibro_tabella1}
 \end{center}
\end{table}

... come mostrato in Tabella~
\ref{segnalibro_tabella1} ...

\end{document}
```

Formule matematiche

```
La somma di $a$ al quadrato
e $b$ al quadrato per avere
$c$ al quadrato, si indica:
$c^{2}=a^{2}+b^{2}$
```

La somma di a al quadrato e b al quadrato per avere c al quadrato si indica: $c^{2}=a^{2}+b^{2}$

```
La somma di $a$ al quadrato
e $b$ al quadrato per avere
$c$ al quadrato, si indica
anche:
\begin{displaymath} c^{2}
=a^{2}+b^{2}
\end{displaymath}
Dove $c$ ... \\
Oppure anche:
$$c^{2}=a^{2}+b^{2}$$
```

La somma di a al quadrato e b al quadrato per avere c al quadrato si indica anche:

$$c^{2}=a^{2}+b^{2}$$

dove c ...
Oppure anche:

$$c^{2}=a^{2}+b^{2}$$

The well known Pythagorean theorem \(x^2 + y^2 = z^2\) was proved to be invalid for other exponents. Meaning the next equation has no integer solutions:

\[x^n + y^n = z^n \]

The well known Pythagorean theorem $x^2 + y^2 = z^2$ was proved to be invalid for other exponents. Meaning the next equation has no integer solutions:

$$x^n + y^n = z^n$$

Testo in una formula

```
$ f(y) > 1 $ se $ y < 3 $
```

$f(y) > 1$ se $y < 3$

```
$$
f(y) > 1 \mbox{ se } y < 3
$$
```

$$f(y) > 1 \text{ se } y < 3$$

```
$$
f(y) > 1 \mbox{se} y < 3
$$
```

$$f(y) > 1 \text{se} y < 3$$

To put your equations in *inline* mode use one of these delimiters: \(\), $ $ or \begin{math} \end{math}. They all work and the choice is a matter of taste.

In physics, the mass-energy equivalence is stated by the equation **$E=mc^2$**, discovered in 1905 by Albert Einstein.

In physics, the mass-energy equivalence is stated by the equation $E = mc^2$, discovered in 1905 by Albert Einstein.

To print your equations in *display* mode use one of these delimiters: \[\], $$ $$, \begin{displaymath} \end{displaymath} or \begin{equation} \end{equation}

The mass-energy equivalence is described by the famous equation **$$E=mc^2$$** discovered in 1905 by Albert Einstein. In natural units (**c** = 1), the formula expresses the identity

```
\begin{equation}
E=m
\end{equation}
```

The mass-energy equivalence is described by the famous equation

$$E = mc^2$$

discovered in 1905 by Albert Einstein. In natural units ($c = 1$), the formula expresses the identity

$$E = m \tag{1}$$

Equazioni

```
\usepackage{amsfonts,amssymb}

Vale l'equazione:
\begin{equation}\label{eq1}
\epsilon > 0
\end{equation}
Dall'equazione~\ref{eq1} ...
```

Vale l'equazione:

$$\varepsilon > 0 \qquad (1.1)$$

Dall'equazione (1.) ...

```
\usepackage{amsfonts,amssymb}

\begin{equation}\label{eq2}
\forall x \in \mathbf{R}:
\qquad x^{2} \geq 0
\end{equation}
```

$$\forall x \in \mathbf{R}: \quad x^2 \geq 0 \qquad (1.2)$$

Bibliografia

- Per realizzare la bibliografia si usa l'ambiente **thebibliography**
- Ciascuna voce viene inserita con il comando **\bibitem{nome-rif}** e il riferimento alla voce viene fatto con **\cite{nome-rif}**

```
\documentclass{article}
...
\begin{document}

... come descritto in~\cite{biblio1}

\begin{thebibliography}{99}
\bibitem{biblio1} B. Oliboni. {\it Informatica di base}
\bibtem{biblio2}
...
\end{thebibliography}
\end{document}
```

Bibliografia (2): file.bib

```
@book{libro1,
    author = "Autore del libro",
    title = "Titolo libro",
    Series = "Serie libro",
    publisher = "Editore",
    volume = 1,
    year = 2000
}

@article{articolo1,
    author = "Autore1 and Autore2",
    title = "Titolo articolo",
    journal = "Rivista",
    volume = 10,
    number = 1,
    pages = "2--20",
    year = 2006
}
```

biblio.bib

Bibliografia (2): file.tex

```
\documentclass{article}
\usepackage[latin1]{inputenc}
\usepackage[italian]{babel}
...
\title{Titolo del documento}
\author{Autore del documento}
\begin{document}
\maketitle

... come descritto in~\cite{libro1} e
in~\cite{articolo1}...

\bibliographystyle{plain}
\bibliography{file-biblio}

\end{document}
```

documento.tex

Creare un documento LaTex

- Usare l'editor per creare e salvare il file **documento.tex** e il file **biblio.bib**
- Compilare con il comando

  latex documento.tex
- Eseguire BibTex sul documento

  bibtex documento
- Ricompilare due volte con

  latex documento.tex
- **Visualizzare con xdvi documento.dvi**
- **Convertire in PostScript con**

  dvips documento.dvi -o

Risorse e Riferimenti:

- Il materiale di questa lezione è stato assemblato utilizzzando le seguenti risorse disponili online:
  - http://www.astro.unipd.it/cosmo/carlo/latex_personal/LaTeX%20%28Esempi%20&%20Presentazione%29/LaTeX.ppt
  - www.diegm.uniud.it/~fusiello/teaching/.../slides/ Lez6_Latex.ppt