Introduzione all'Informatica

Loriano Storchi

loriano@storchi.org

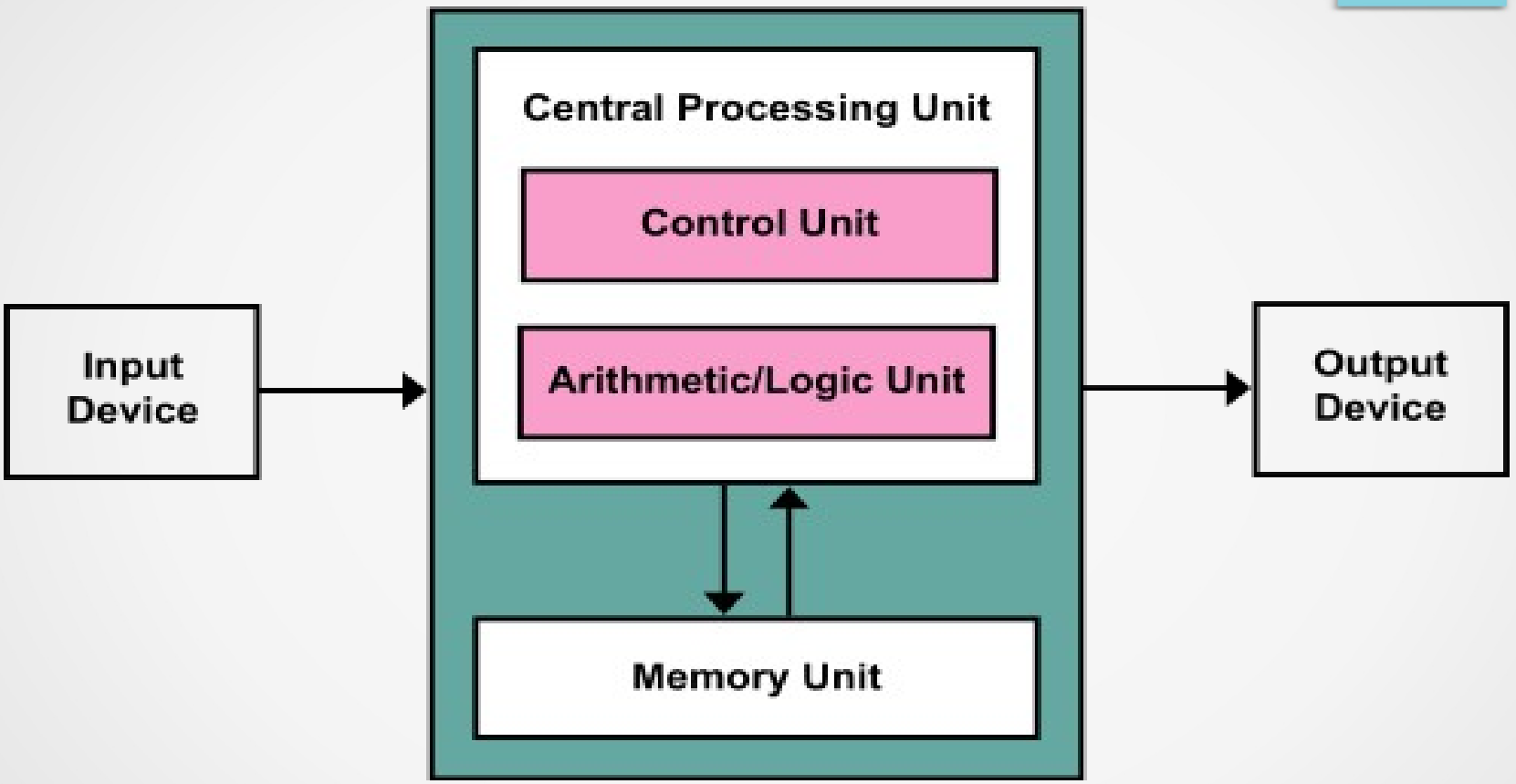
http::://www.storchi.org/

Informatica

- Possiamo usare diverse definizioni come ad esempio: l'informatica e' la scienza della rappresentazione e dell'elaborazione dell'informazione, parafrasando lo studio degli algoritmi che descrivono e trasformano l'informazione

HARDWARE

Macchina di Von Neumann



BUS di SISTEMA , collegamento (architettura Harvard separazione tra memoria dati e memoria contente il programma)

C.P.U.

C.P.U. Central Processing Unit o Unità di Elaborazione Centrale, coordina e gestisce tutti i vari dispositivi hardware per acquisire, interpretare ed eseguire le istruzioni dei programmi

- Unita' di controllo , interpreta ed esegue le istruzioni
- A.L.U. (unita' logico-Aritmetica) svolge le operazioni logiche ed aritmetiche
- REGISTRI “piccole” memorie interne della C.P.U.
- CLOCK: scandisce gli intervalli di tempo in cui agiscono i dispositivi interni della CPU . Ne determina la velocita' espressa come numero di intervalli nell'unita' di tempo

C.P.U.

CPU moderne:

- Pipeline: estensione della struttura interna della CPU, cosi; da poter eseguire in parallelo varie fasi connesse al caricamento, interpretazione ed esecuzione delle istrusioni. Ogni fase e' affidata ad una specifica “unita'”
- Co-processori: processori specilizzati ad eseguire particolari funzioni:
  - Esecuzione di calcoli numerici
  - Elaborazioni grafiche (GPU)
- Parallelismo: processori multicores (CPU replicate), per poter eseguire piu' istruzioni in parallelo

Memoria Centrale

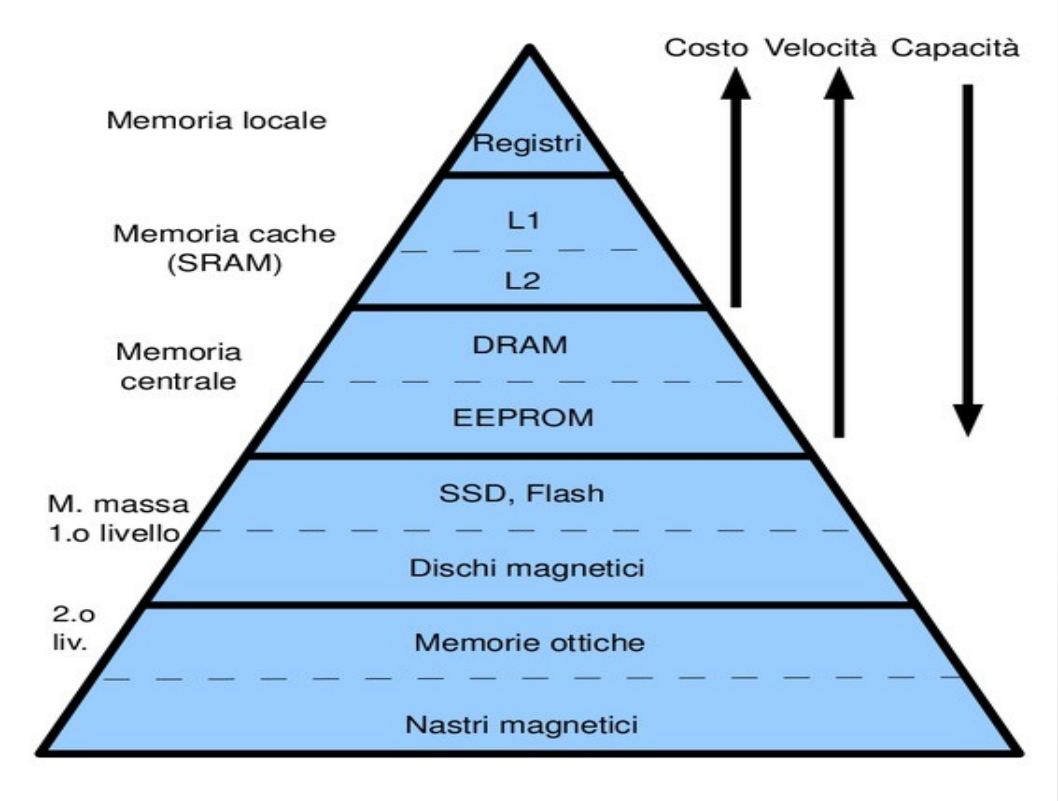
- RAM (Random Access Memory)
- Nella RAM sono contenute le istruzioni (opcode) che verrano eseguite ed i dati su cui tali istruzioni opereranno
- Caratteristiche:
  - Volatile: il contenuto e' perso quanto l'elaboratore e' spento
  - Veloce: 100 cicli di clock circa . Veloce quindi costosa
  - Dimensioni piccole rispetto alla memoria di massa ordine di qualche GiB

Memoria di massa

- Costituita da Dischi Rigidi, CD, DVD, dischi stato solido , nastri
  - Non volatile
  - Lenta rispetto alla RAM (> 10000 cicli)
  - Mediamente economica
  - Di grandi dimensioni (oramai centinaia di GiB o qualche TiB)

Gerarchia di memoria

La gerarchia di memoria nei processori attuali e' compostada diversi livelli, caratterizzati ciascuno da velocita' di accesso ai dati inversamente proporzionali alla dimensione: piu' sono grandi queste aree e maggiore e' il temporichiesto per recuperare i dati in esso contenuti.



Cache

Cache In generale da 1 a 3 livelli di memoria cache sono installati. La memoria cache e' caratterizzata da diversi tempi di accesso e dimensioni, a seconda che essa sia all'interno del chip (on chip) o fuori dal chip (offchip), e dalla tecnologia con cui sono realizzate le celle di memoria. (cat /proc/cpuinfo per vedere le dimensioni della cache)

L'uso e il vantaggio di una cache si basa sul principio di localita' ovvero che un programma tende a riutilizzare dati ed istruzioni usate recentemente Una conseguenza e una regola del pollice che dice che in genere il 90 % del tempo totale viene impiegato ad eseguire il 10% delle istruzioni. Per essere piu precisi l'uso della cache e' vantaggioso quando di un codice si sfrutta sia la localita' spaziale che la localita temporale dei dati.

UNITA' DI MISURA

Unita' di misura dell'informazione

- BIT = è l'unità di misura dell'informazione (dall'inglese "binary digit"), definita come la quantità minima di informazione che serve a discernere tra due possibili eventi equiprobabili. (Wikipedia)
- BYTE = 8 BIT (storicamente i caratteri erano rappresentati da 8 BIT , motivo per cui 1 Byte rimane tutt'oggi l'unita' di memoria minima indirizzabile)
- “KiloByte” meglio “kibibyte” KiB = 2^{10} Byte = 1024 Byte
- “MegaByte” meglio “mebibyte” MiB = 1024 * 1024 Byte
- “GigaByte” meglio “gibibyte” GiB = 1024 * 1024 * 1024 Byte
- “TeraByte” meglio “tebibyte” TiB = 1024 * 1024 * 1024 * 1024 Byte

FLOPS

FLOPS e' un'abbreviazione per Floating Point Operations Per Second, ed indica appunto il numero di operazioni in virgola mobile che una CPU esegue in un secondo.

Ad esempio nel caso di un prodotto classico tra matrici, vengono eseguite $2*N^3$ operazioni, quindi posso valutare i FLOPS esattamente misurando il tempo necessario ad eseguire tale moltiplicazione ed ottenere:

$$[\text{flops}] = 2*N^3 / \text{tempo}$$

TOP500

- Differenza tra susteined perfomace e prestazioni di picco
- Per valutare in modo oggettivo le prestazioni di un computer c'e' bisogno di un test di riferimento, un benachmark standard, ad esempio Linpack
- TOP500 http://www.top500.org/, classifica dei 500 computer piu' potenti al mondo

SOFTWARE

Software

- Software di base: dedicato alla gestione dell'elaboratore stesso, ad esempio il Sistema Operativo

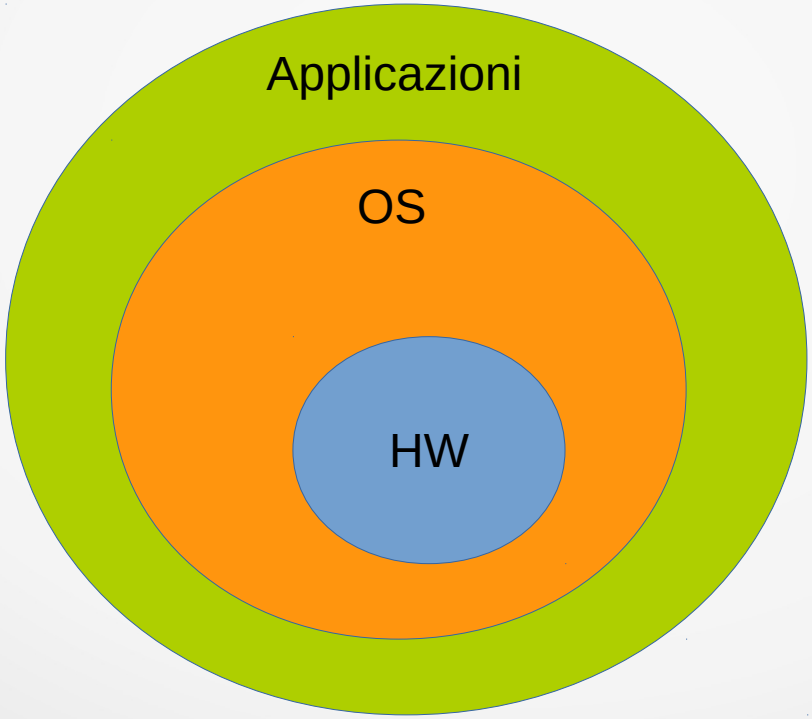
- Software applicativo: dedicato alla realizzazione di specifiche applicazioni, ad esempio navigazione su internet, o videoscrittura o altro.

Sistema Operativo

- Fornisce funzionalita' di alto livello rendendo l'hardware “facile” da usare
- Ad esempio organizza i dati presenti nelle memoria
- Esegue comandi dell'utente
- Esegui programmi e mostra risultati a video
- Nel caso di sistemi multiutente deve gestire le risorse e renderle fruibili a tutti (gestite le risorse parallelismo “virtuale” sistemi multitasking)
- Esempio, Linux, Unix, Mac OS X (Darwin), Microsoft Windows

Sistemi Operativi

Rendono piu' semplice la scrittura di programmi applicativi che non devono preoccuparsi delle specifiche caratteristiche dell'HW.



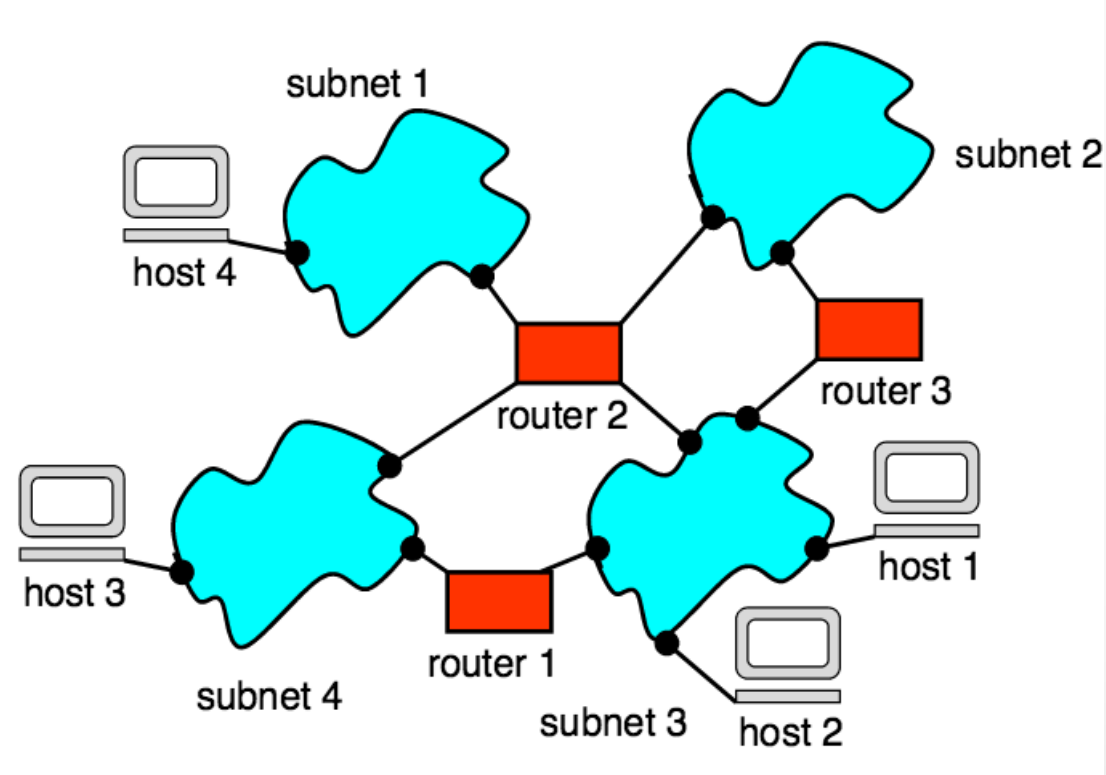
RETI DI COMPUTER ED INTERNET

Reti di Computer

- Una rete di computer e' di fatto un insieme di computer collegati fra loro di modo che possano scambiarsi informazioni, condividere risorse. Ci sono diverse tipologie di reti.
- LAN (Local Area Network): rete su scala locale, si tratta di reti estese a livello di una singola stanza o al massimo di un edificio
- MAN (Metropolitan Area Network): puo' ad esempio collegare piu' LAN
- WAN (Wide Area Network): reti estese su aree geografiche. Conettono assiene LAN e MAN (Internet e' la WAN per eccellenza)

Internet

- Internet nasce negli anni 60/70 essenzialmente con scopi militari
- Internet oggi interconnette migliaia di sottoreti



Host: computer collegato ad internet, puo' essere si un client che un server a livello applicativo

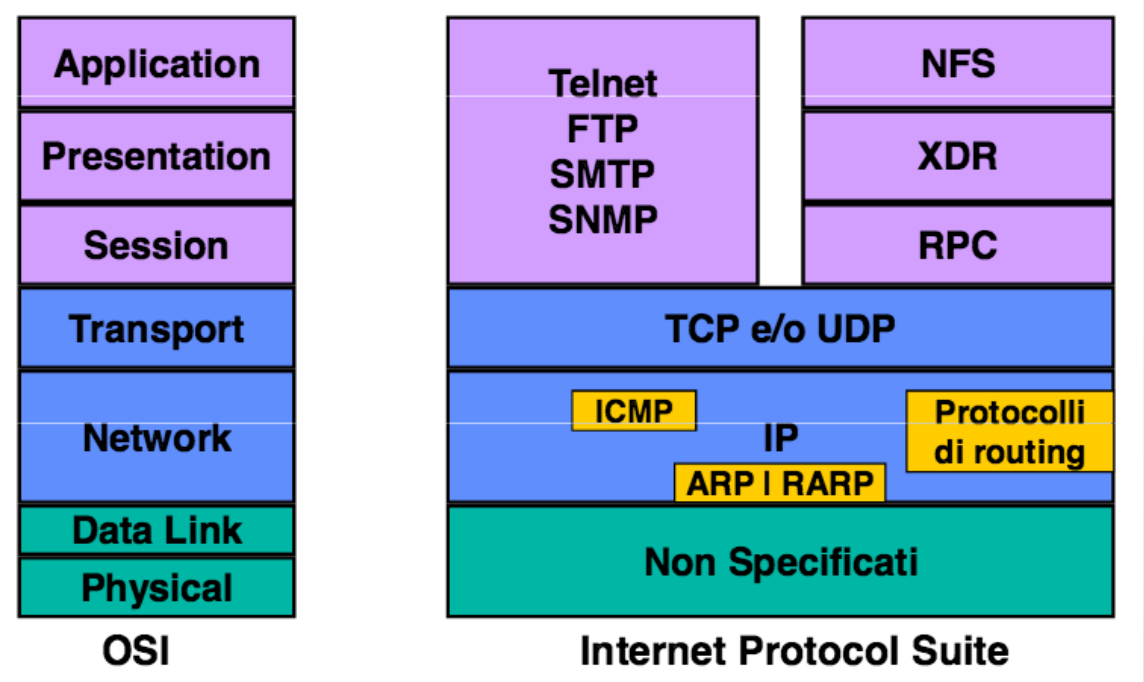
Router: nodo ce serve ad instradare il traffico

Sottorete: insieme di host fra cui c'e' un collegamento di livello 2 (ad esempio una LAN)

Ad oggi ordine di miliardi di Host

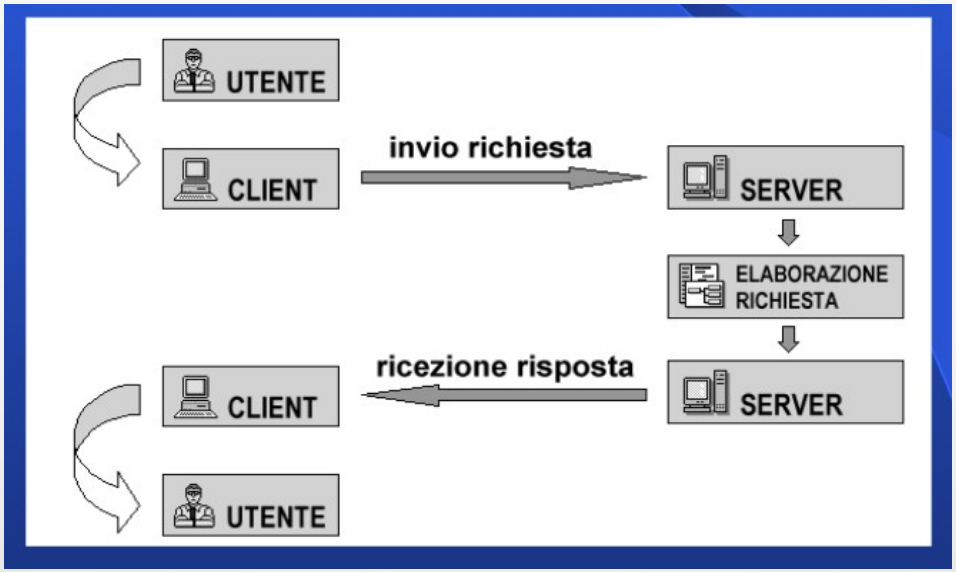
Internet

- Per permettere a diversi host di comunicare sono necessarie regole comuni. Il Protocollo TCP/IP: Transfer Control Protocol/ Internet Protocol
- L'informazione (i dati) viene divisa in pacchetti che poi sono ricomposti a destinazione (ogni pacchetto puo' seguire strade diverse)



Client/Server

- La maggior parte dei servizi telematici offerti da internet si basano sulla modalita' di interazione client/server (diverso ispetto al P2P).
- Il client e' dotato di un particolare software client in grado di inviare richieste di servizio ad un articolare server. Il clint formatta le richieste in modod adeguato e comprensibile al server, usando quindi uno specifico protocollo



Protocolli ad alto livello

- Vengono usati diversi tipi di protocollo ognuno per ogni specifico servizio:
  - HTTP (HyperText Transfer Protocol) Accesso alle pagine ipertestuali (WEB) nell'ambito del WWW
  - FTP (File Transfer Protocol) trasferire e copiare file
  - SMTP (Simple Mail Transfer Protocol) Spedizione di messaggi di posta elettronica (e-mail)

Una risorsa in rete e' quindi “identificata” dall'URL:

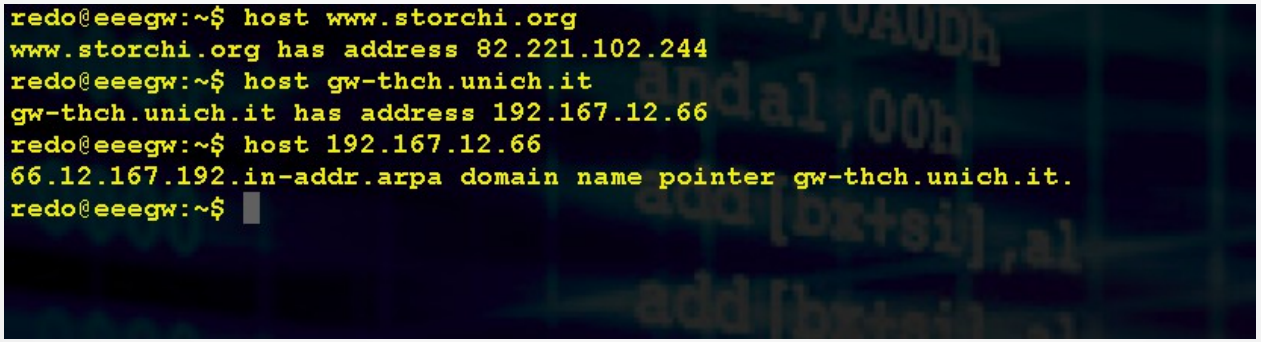
http://nomehost.it/index.html

Indirizzamento

- Ogni computer collegto ad internet e' identificato dal suo indirizzo IP, composto da 4 gruppi di un byte ciascuno (complessivamente 32-bit).
- Ogni gruppo puo' assumere un valore massimo di 255, ad esempio: **192.167.12.66** (IP statici o Dinamici, IP privati …)
- L'ultimo numero identifica solitamente un Host, i numeri precedenti la sottorete a cui questo Host appartiene.
- Il massimo numero di indirizzi IPv4 e' dunque 255*255*255*255

Indirizzamento

- E' difficile per un essere umano memorizzare numeri, molto piu' facile nomi. Ci sono quindi servizi di DNS (Domain Name System). Quindi sistemi utili a tradurre in un verso e nell'altro nomi e indirizzi.



Indirizzamento

- Ogni host e' dunque identificato dall'utente da un nome simbolico:
  - gw-thch.unich.it
- I nomi sono assegnati univocmante e gestiti amminsitratativamente in modo gerarchico
- I nomi identificano in modo univocono un host all'interno di un dominio:
  - it e' il dominio
  - unich e' il sotto-dominio all'interno ad it
- I domini principali sono:
  - .gov .edu .com essenzialmene in USA associati al tipo di organizzazione
  - Le varie nazioni invece hanno domni del tipo: .it, .uk, .fr , .de ….

Unita' di misura

- Volecita' trasmissione dati = quantita' di informazione / tempo di trasferimento
- In generale questa velocita' viene espressa in bit per secondo cioe' **bit/s** (oppure **bps** detta anche **bit rate**). Si usa anche il byte per secondo **byte/s** (oppure **Bps**).
- Si usano poi i prefissi standard k (=kilo 10^3) , M (=mega 10^6), G (=giga 10^9), quini non le approssimazioni basaste sulle potenze di due che si utilizzano in ambito informatico.
- Convertire da bps a Bps e' semplice basta dividere per 8. Ad esempio ADSL 10 Mbps = 10 Mbps/8 = 125 KBps
- Dovendo trasferire un file da 10 MiB con una linea da 5 Mbps impieghero' circa (10 * 1024 * 1024 * 8)/ 5 10^6 = 16.8 s

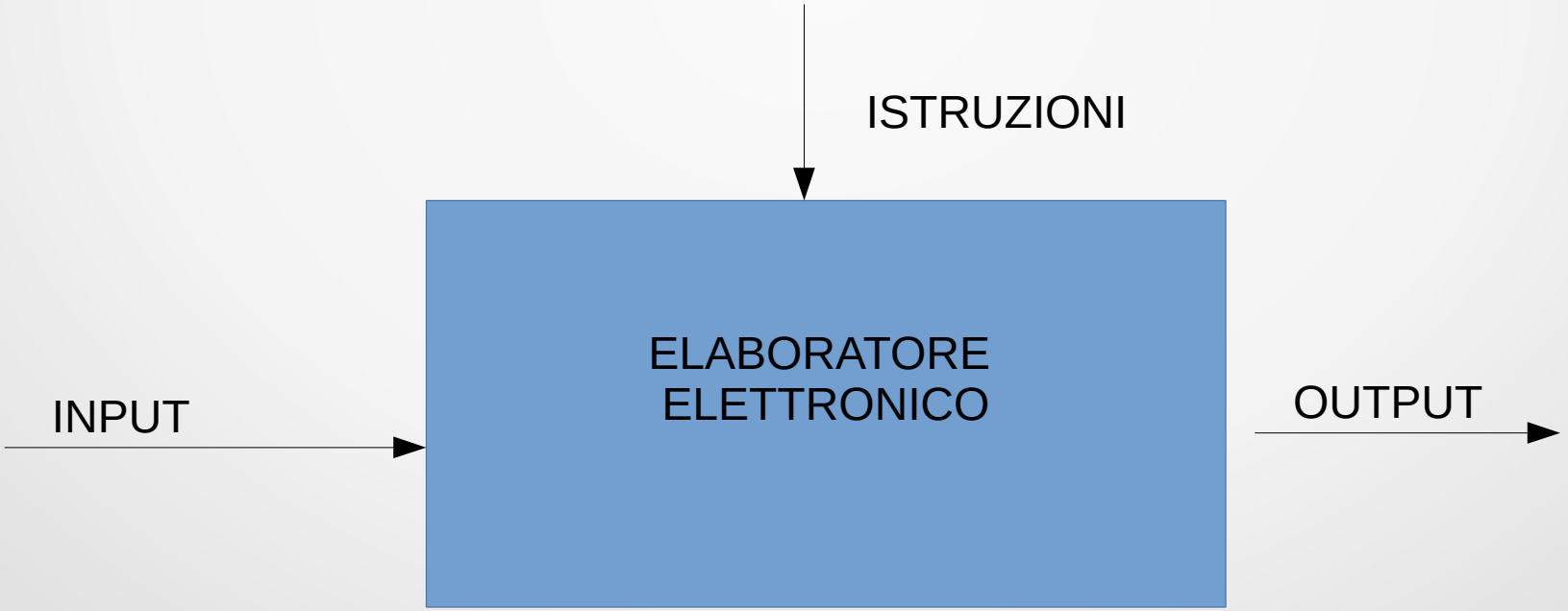
ALGORITMI, PROGRAMMI E LINGUAGGI DI PROGRAMMAZIONE

Algoritmi

- Gli algoritmi descrivono il modo in cui trasformare l'informazione. L'informatica si occupa della loro teoria, analisi, progettazione, della loro efficienza realizzazione ed applicazione.
- Un algoritmo è un procedimento formale che risolve un determinato problema attraverso un numero finito di passi. Il termine deriva dalla trascrizione latina del nome del matematico persiano al-Khwarizmi, che è considerato uno dei primi autori ad aver fatto riferimento a questo concetto. L'algoritmo è un concetto fondamentale dell'informatica, anzitutto perché è alla base della nozione teorica di calcolabilità: un problema è calcolabile quando è risolvibile mediante un algoritmo. (Wikipedia)

Programmazione

- Identifica l'attivita' mediante la quale si “istruisce” un calcolatore ed eseguire un particolare insieme di azioni, che agiscono su dati di ingresso (input), allo scopo di risolvere un problema e dunque produrre opportuni dati di uscita (output). Implementazione di un dato algoritmo.



Linguaggi e livelli

Il livello L0 rappresenta il computer reale ed il linguaggio macchina che esso è in grado di eseguire direttamente. Ogni livello superiori rappresenta una macchina astratta. I programmi (istruzioni) di ogni livello superiore devono essere o tradotti in termini di istruzioni di uno dei livelli inferiori, o interpretati da un programma che gira su di una macchina astratta di livello strettamente inferiore.

Linguaggio Macchina e ASSEMBLY

Ogni calcolatore e' in grado di interpretare un linguaggio di basso livello detto linguaggio macchina, le istruzioni (OPCODE) sono semplici sequenze di bit che il processore interpreta e seguendo una serie precisa di operazioni. Ogni istruzione a questo livello e' costituita da operazione estremamente basilari.

Per rendere piu' immediata la programmazioni il primo passo e' stato l'introduzione dell'ASSEMBLY, dove il codice binario e' sostituito da istruzioni mnemoniche umanamente piu' facilmente utilizzabili.

Linguaggio Macchina:
00000000000000000000001000000
00000000001000000000001000100
00000010000000010000000000000000
00000001000000000000000000111100

PSEUDO-ASSEMBLY:

```
z : INT;
x : INT 8;
y : INT 38;
LOAD R0,x;
LOAD R1,y;
ADD R0,R1;
STORE R0,z;
```

LINGUAGGI

- Oggi sono presenti numerosi linguaggi di programmazione. In generale ogni linguaggio risulta piu' o meno adaguato ad uno scopo specifico.
- **Linguaggi naturali** : sono dato spontaneamente sono estremamnte espressivi ma ambigui: **la vecchia porta la sbarra**
- **Linguaggi artificiali** : sono linguaggi che hanno una data precisa di nascita ed una lista di autori. I linguaggi artificiali possono essere **formali** e non-formali.
- **Linguaggi formali**: non ambigui costituiti da un insieme finito di stringhe costruite a partire da un alfabeto finito. E' un linguaggio per cui la forma delle frasi (**sintassi**) ed il significato dell stesse (**semantica**) sono definite in modo preciso non ambiguo. E' dunque possibile definire una procedura algoritmica in grado di verificare la correttezza **grammaticale** delle frasi.

LINGUAGGI

- Per definire un linguaggio rigorosamente occorrono alcuni strumenti di base:
  - **Alfabeto**: insieme dei simboli di base necessari a costituire le parole
  - **Lessico**: insieme delle regole necessarie a acrivere le parole di un linguaggio (vocabolario)
  - **Sintassi** (regole **grammaticali**): insieme di regole che permettono di stabilire se una frase (insieme di parole) e' corretta
  - **Semantica**: definisce il significato di una “frase” sintatticamente corretta ad esempio: **int a[5];** in linguaggio C permette di riservare spazio in memoria necessario a contenere 5 interi

LINGUAGGI

- Abbiamo gia' accennato a Linguaggi artificiali, Linguaggio macchina e Linguaggio di basso livello (ASSEMBLY)
- Linguaggi di alto livello si allontanano dalla logica del processore e sono costruiti per essere semplici, efficienti e leggibili, oltre che indipendenti dalla macchina
- Sono ad oggi presenti tantissimi linguaggi di programmazione, anche se quelli effettivamente utilizzati sono una decina.
- Di seguito riporteremo una classificazione sommaria di tali linguaggi (C, C++, C#, JAVA, PYTHON, FORTRAN, PASCAL, BASIC, Objectice-C e tantissimi altri). E' chiaro che ogni paradigma di programmazione e' piu' o meno adatto ad uno scopo piu' o meno specifico.

LINGUAGGI IMPERATIVI

- La componente fondamentale del programma e' l'istruzione, ed ogni istruzione indica l'operazione che deve essere eseguita. Le singole istruzioni che operato su i dati del programma.
- Le istruzioni vengono eseguite una dopo l'altra
- Ogni programma e' costituito da due parti fondamentali la dichiarazione dei dati e l'algoritmo inteso come sequenza di operazioni
- Ogni istruzione e' un ordine (programmazione dichiarativa il programma e' una serie di affermazioni)
- Di fatto da un punto di vista sintattico molti linguaggi imperativi utilizzano appunto verbi all'imperativo (i.e. PRINT, READ, ….)

PROGRAMMAZIONE PROCEDURALE

- Possiamo considerarlo un sotto-paradigma della programmazione imperativa.
- Viene introdotto il concetto di sotto programma (subroutine) o funzioni.
- Quindi si introduce la possibilita' di creare porzioni di codice sorgente utili ad eseguire funzioni specifiche.
- Questi sottoprogrammi possono ricevere parametri di input e restituire valori di output.

PROGRAMMAZIONE STRUTTURATA

- Possiamo considerarlo un sotto-paradigma della programmazione imperativa.
- In pratica il programmatore e' vincolato ad usare olo strutture di controllo canoniche che non includono le istruzioni di salto incondizionato (GOTO).
- L'uso dell'istruzione GOTO porta invetabilmente ad una scarsa leggibilita' del codice (spaghetti-code)

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI (1/2)

- Possiamo considerarlo un sotto-paradigma della programmazione imperativa.
- Tale paradigma di programmazione permette di definire **Oggetti** Software in grado di intergire gli uni con gli altri.
- L'organizzazione del software sotto forma di oggetti permette un piu' facile riuso dello stesso. Una migliore organizzazione di progetti di grandi dimensioni.
- L'oggetto e' l'insieme di dati e di operazione sui dati.
- Un linguaggio orientato agli oggetti permette di implementare tre meccanismi di base utilizzando la sintassi nativa del linguaggio : **incapsulamento, polimorfismo, ereditarieta'**.

PROGRAMMAZIONE ORIENTATA AGLI OGGETTI (2/2)

- **Incapsulamento:** separazione precisa fra implementazione ed interfaccia della classe. Chi usa la classe (oggetto) non deve conoscere il dettaglio implementativo.
- **Polimorfismo**: possiamo cercare di esemplificare questo concetto dicendo: molteplici definizione della stessa funzione (overloading), classi e funzioni parametriche rispetto al tipo di dato
- **Ereditarieta'**: una classe puo' ereditare da una classe base ed evolverne o specializzarne le funzionalita'.

PROGRAMMAZIONE FUNZIONALE

- Come dice il nome stesso il flusso di esecuzione assume la forma di valutazione di una serie di valutazioni di funzioni matematiche. Il programma e' quindi un'insieme di funzioni
- Nei linguaggi funzionali puri non esiste il concetto di assegnazione.
- I valori non si trovano cambiando lo stato del programma, non esiste appunto l'assegnazione di valore, ma costruendo il nuovo stato mediante funzioni a partire dallo stato precedente.
- Usanti nell'ambito dell' AI (poco usanti o assenti in ambito industriale)

LINGUAGGI DICHIARATIVI (O LOGICI)

- Rispetto al paradigma imperativo il programma consiste di una serie di affermazioni e non di ordini.
- Nel programma si specifica il COSA si voglia ottenere non il COME. Il come e' lasciato all'esecutore
- In pratica il programma (o la sua esecuzione) si puo' considerare come la dimostrazione della verita' di un'affermazione.

....

- Si possono poi classificare i linguaggi anche secondo il tipo di dato **tipizzato forte** e **tipizzazione dinamica**.
- **Linguaggi o paradigmi di programmazione parallela** (per le moderne architetture)
- **Linguaggi esoterici** : sono linguaggi voltamente complessi e poco chiari. Popolari solo fra gli utenti piu' abili ed usati al solo scopo di mettere alla prova le capacita' di programmazione (scopo essenzialmente ludico)
- **Scripting**: nati inizialmente per essere usati nelle shell Unix. Sono linguaggi usati per automatizzare compiti ripetitivi e lunghi.

Programmazione altri concetti

- Il sorgente viene scritto in file di testo ASCII.Il sorgente esprime l'algoritmo implementato nel linguaggio scelto. Per scrivere il sorgente si possono usare semplici editor di testo (VI, Emacs). Oppure IDE ambineti di sviluppo integrato con altri strumenti, come compilatori , linker e debugger.
- **Compilazione**: il sorgente viene tradotto (dal compilatore) da linguaggio ad alto livello a codice eseguibile. Il vantaggio e' che l'esecuzione e' “veloce” e che il codice viene ottimizzato per la piattaforma specifica. Lo svantaggio e' che si dovra' ri-compilare per ogni diverso sistema operativo o hardware.
- **Linking**: ogni programma generalmente fa uso di una o piu' librerie ed il linker collega assieme librerie e programma di partenza. I linking puo' essere sia statico che dinamico (ad esempio librerie .so in Linux/Unix-like o .dll in windows)

Programmazione altri concetti

- **Interpretazione**: onde evitare il problema della portabilita' dei programmi si e' ricorso al concetto di interpretate In questo caso il codice sorgente non viene compilato “tradotto” ma eseguito appunto da un'interprete. Questo introduce altri problemi come quello delle prestazioni.
- **Bytecode, P-code** : possiamo definirlo come un approccio intermedio in cui il programma sorgente viene “tradotto” in un codice intermedio che viene interpretato da una macchina virtuale. Questo permette di unire due vantaggi una buona velocita' di esecuzione assieme ad una estrema portabilita' del programma. JIT (Just in Time) al momento dell'esecuzione compilano il codice intermedio in codice macchina.

Programmazione altri concetti

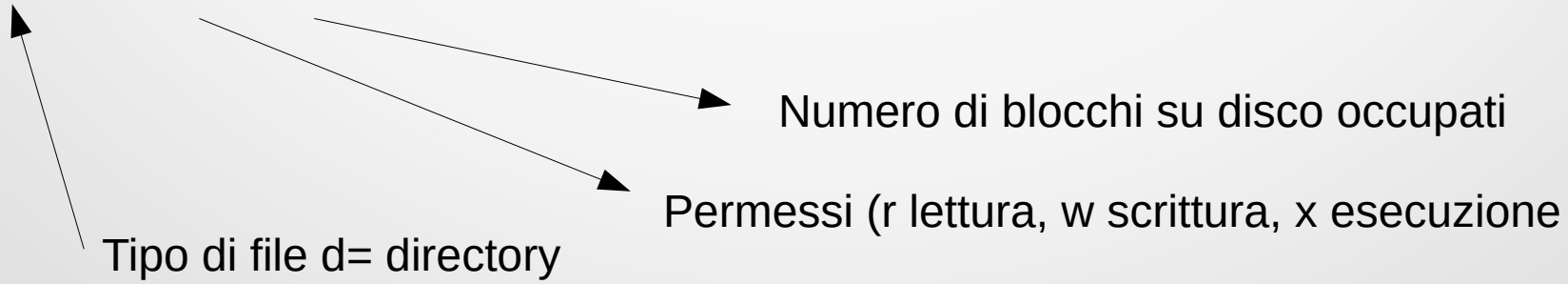
- **Calcolabilita' e complessita'** ……

FILESYSTEM UNIX/UNIX-LIKE E COMANDI DI BASE

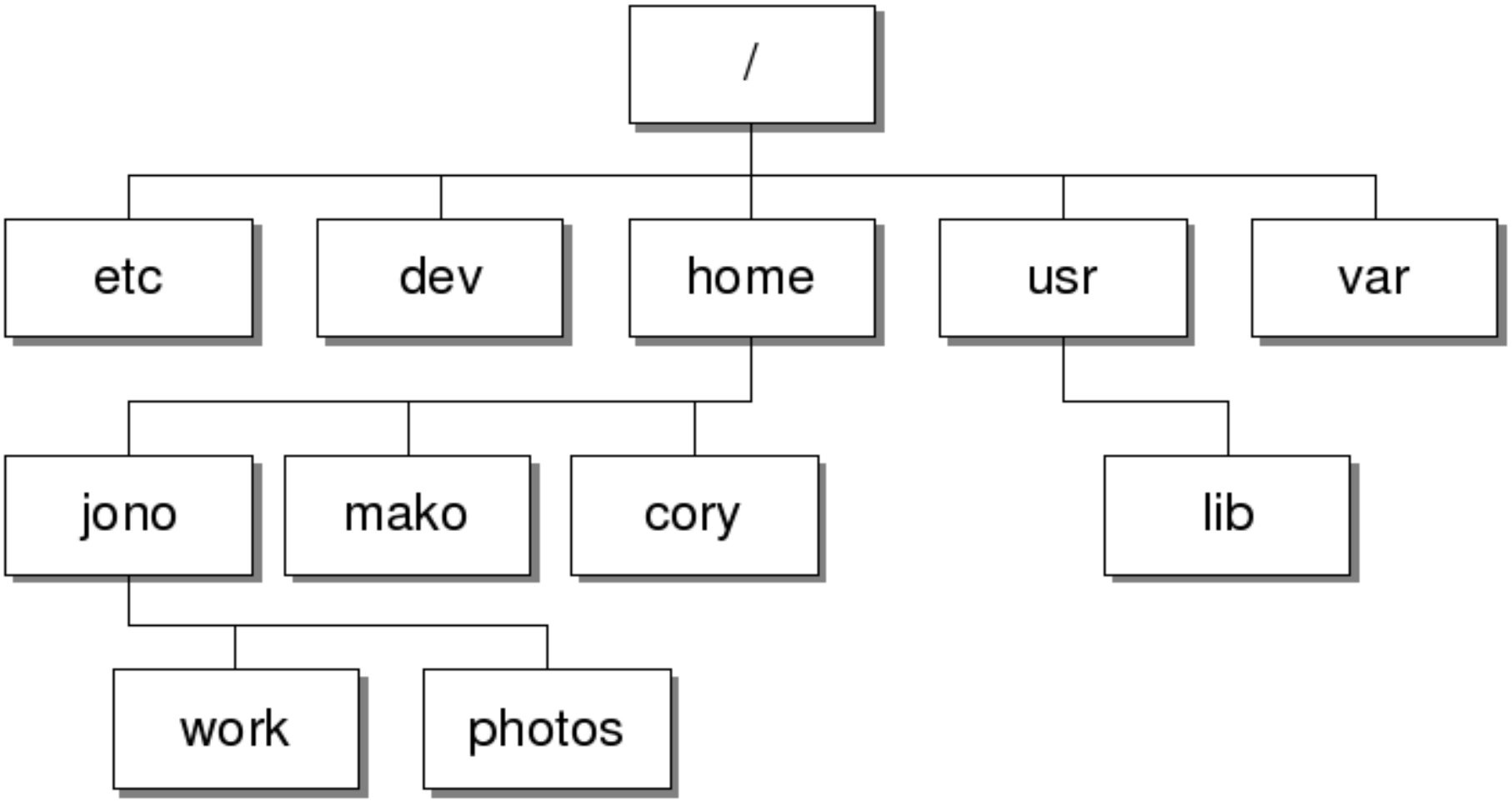
Filesystem

- La parte dell'SO che si occupa di “contenere/mantenere” dati e programmi in maniera persistente
- Fornisce le seguenti astrazioni:
  - File : unita' di informazione memorizzata in modo persistente
  - Directory: permette di raggruppare assieme piu' files
- Attributi dei files in SO Unix e Unix-like (Linux, OS X):

-rw-r--r--. 1 redo redo Oct 27 14:38 test



Filesystem Linux



Shell

```
[redo@banquo ~]$ mkdir test
[redo@banquo ~]$ cd test/
[redo@banquo test]$ ls -al
total 8
drwxrwxr-x.   2 redo redo 4096 Oct 27 14:51 .
drwx------. 103 redo redo 4096 Oct 27 14:51 ..
[redo@banquo test]$ ls
[redo@banquo test]$ > test
[redo@banquo test]$ ls -al
total 8
drwxrwxr-x.   2 redo redo 4096 Oct 27 14:55 .
drwx------. 103 redo redo 4096 Oct 27 14:51 ..
-rw-rw-r--.   1 redo redo    0 Oct 27 14:55 test
[redo@banquo test]$ dd if=/dev/zero of=./test bs=1k count=2
2+0 records in
2+0 records out
2048 bytes (2.0 kB) copied, 0.000215692 s, 9.5 MB/s
[redo@banquo test]$ ls -al
total 12
drwxrwxr-x.   2 redo redo 4096 Oct 27 14:55 .
drwx------. 103 redo redo 4096 Oct 27 14:51 ..
-rw-rw-r--.   1 redo redo 2048 Oct 27 14:55 test
[redo@banquo test]$ pwd
/home/redo/test

[redo@banquo test]$ cd
[redo@banquo ~]$ pwd
/home/redo
[redo@banquo ~]$
```

History dei comandi usando le frecce

Dir ./ corrente
Dir ../ padre

ps o top lista dei processi

....

ESEMPI PROGRAMMAZIONE

Esempio di procedure numeriche

- In generale e' possibile trovare un algoritmo risolutivo per molti problemi, ma non tutti (problema di computabilita'). Ad esempio calcolare le soluzioni di un'equazione di secondo grado:

INSERISCI VALORI DI A, B, C

D = (B*B)-(4 * A * C)

SE D >= 0.0

  SOL1 = (-1,0 * B + SQRT(D)) / (2,0 * A)

  SOL2 = (-1,0 * B - SQRT(D)) / (2,0 * A)

  STAMPA SOL1 E SOL2

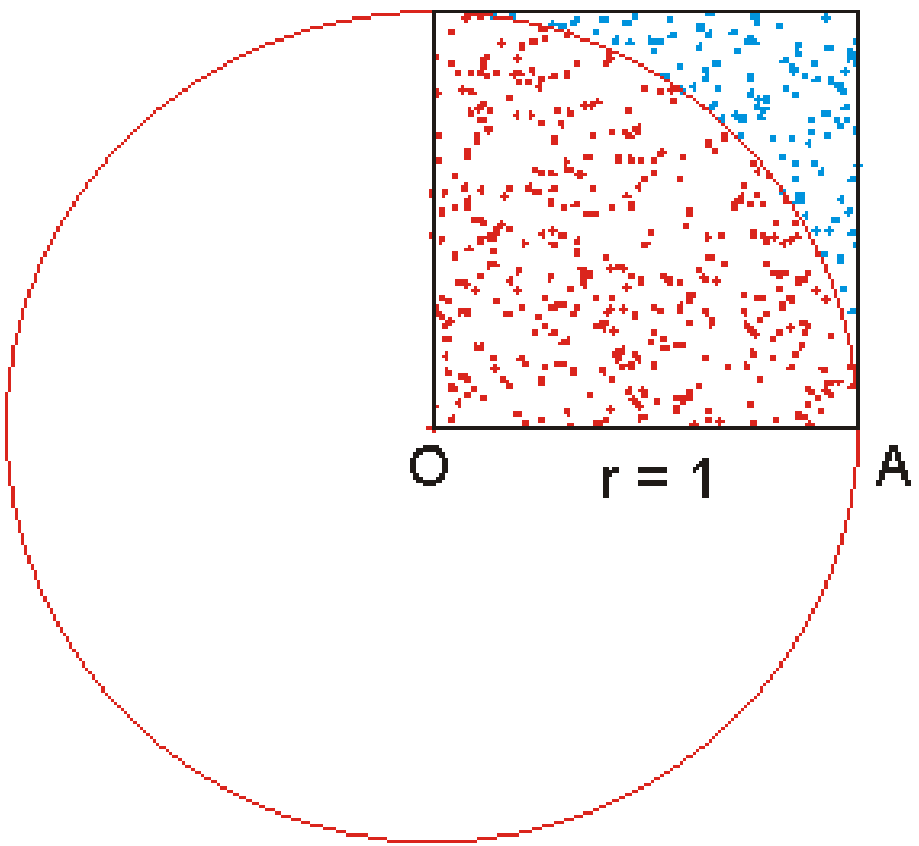
Altrimenti

  STAMPA non ci sono soluzioni reali

Programma in Python

```
Import  math
ai = input("insert a:")
a = float(ai)
bi = input("insert b:")
b = float(bi)
ci = input("insert c:")
c = float(ci)
print "a = ", a, " b = ", b, " c = ", c
delta = math.pow(b, 2.0) - (4.0 * a * c)
if (delta >= 0):
  tn = math.sqrt(delta)
  sol1 = ((-1.0 * b) + tn) / (2.0 * a)
  sol2 = ((-1.0 * b) - tn) / (2.0 * a)
  print sol1, sol2
else:
  print "No real solutions"
```

Calcolo PI con metodo MC



$$A_S = (2r)^2 = 4r^2$$

$$A_C = \pi r^2$$

$$\pi = 4 \times \frac{A_C}{A_S}$$

Codice python

```
import random
import math
DIM = 200000
circle_count = 0
for i in range(0,DIM):
   x = random.uniform(0.0, 1.0)
   y = random.uniform(0.0, 1.0)
   if ((math.pow(x, 2.0) + math.pow(y, 2.0)) < 1.0):  # distanza dall'origine
     circle_count = circle_count + 1
pi = float(circle_count) / float(DIM)
print 4.0 * pi
```

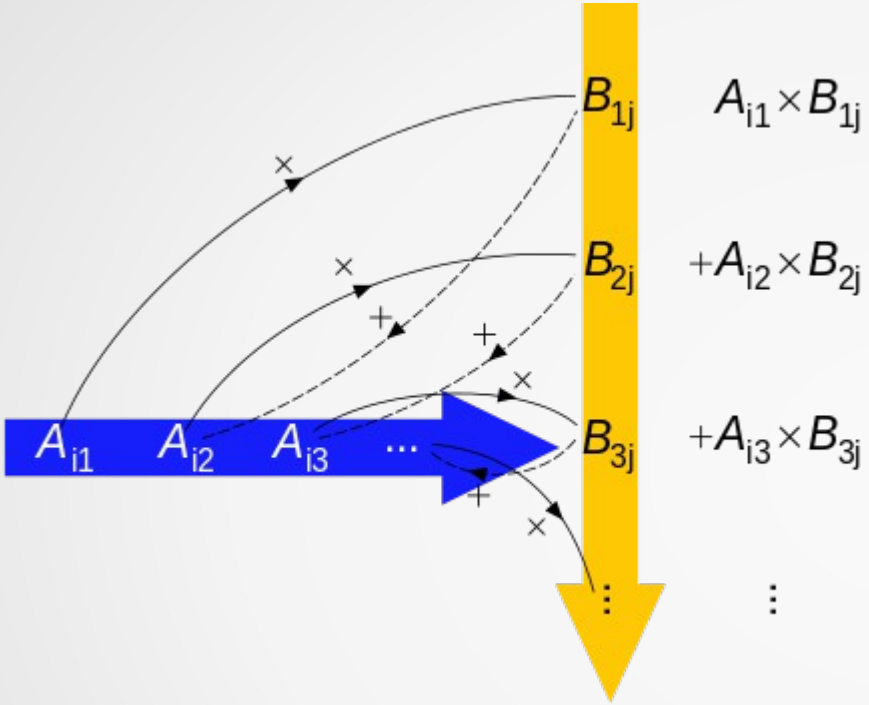
Moltiplicazione matrice matrice

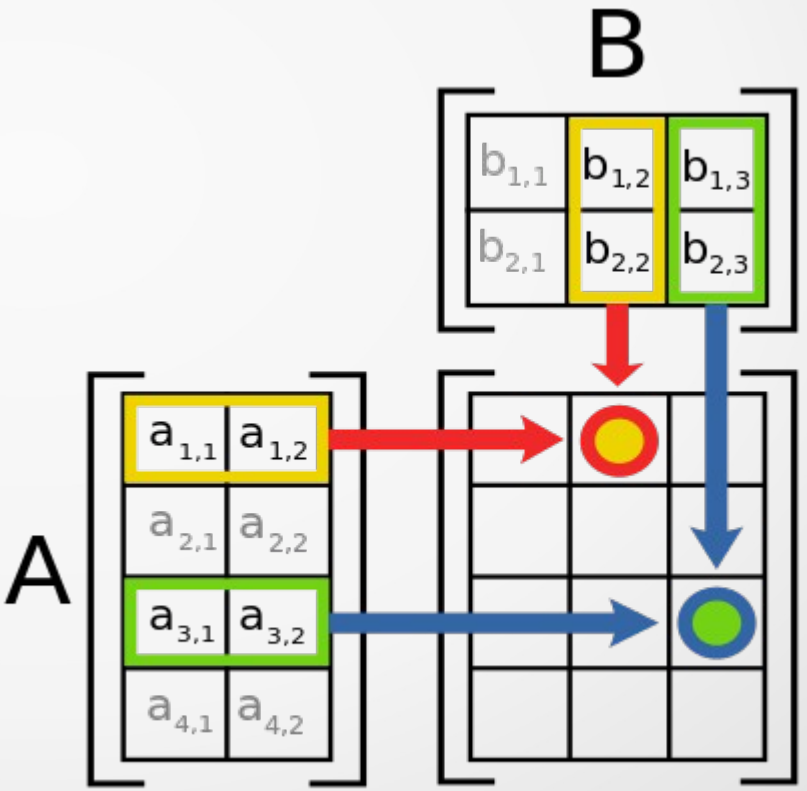
$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} B_{11} & B_{12} & \cdots & B_{1p} \\ B_{21} & B_{22} & \cdots & B_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mp} \end{pmatrix}$$

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj} \ .$$

Il prodotto e' definito solo per matrici con dimensioni compatibili

Moltiplicazione matrice matrice





Source code python

```
import random
import math

A =  [[0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0]]

B =  [[0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0]]

C =  [[0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0],
      [0.0, 0.0, 0.0]]
```

Source code python

```
for i in range(len(A)):
  for j in range(len(A[0])):
    A[i][j] = random.uniform(0.0, 1.0)

for i in range(len(A)):
  for j in range(len(A[0])):
    B[i][j] = random.uniform(0.0, 1.0)

for i in range(len(A)):
  for j in range(len(B[0])):
    for k in range(len(B)):
      C[i][j] = C[i][j] + A[i][k]*B[k][j]
```

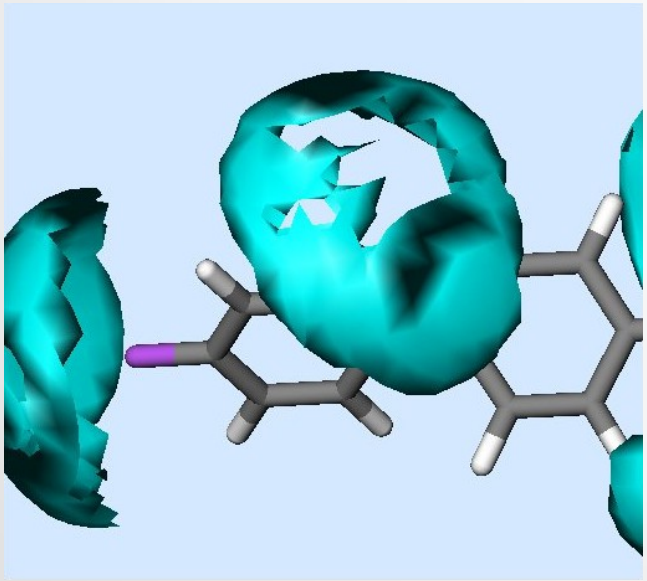
Grid force fields

GRID programme:

a computational procedure for determining energetically favourable binding sites on molecules for functional groups of known structure through the use of PROBES.

The PROBE is moved through a grid of points suporimposed on the target molecule.
Its interaction energy with the target molecule is computed by an empirical energy function.

$$E_{XYZ} = \Sigma[E_{LJ}] + \Sigma[E_{HB}] + \Sigma[E_Q] + [S]$$



E_{LJ}= Lennard-Jones potential
E_{HB}= hydrogen bonding interaction energy
E_Q= electrostatic function
S= entropic term