Il Software

TECNOLOGIA DIGITALE

CPU, memoria centrale e dispositivi sono realizzati con **tecnologia elettronica digitale**.

Dati ed operazioni vengono codificati a partire da due valori distinti di grandezze elettriche:

- tensione alta (V_H, 5V)
- tensione bassa (V_L, 0V)

A tali valori vengono convenzionalmente **associate le due cifre binarie 0 e 1:**

- **logica positiva: 1 → V_H , 0 → V_L**
- logica negativa: 0 → V_H, 1 → V_L

TECNOLOGIA DIGITALE (segue)

Dati ed operazioni vengono codificati tramite **sequenze di bit**

01000110101 ....

CPU è in grado di operare soltanto in aritmetica binaria, effettuando operazioni *elementari* :

- somma e differenza
- scorrimento (shift)
- ...

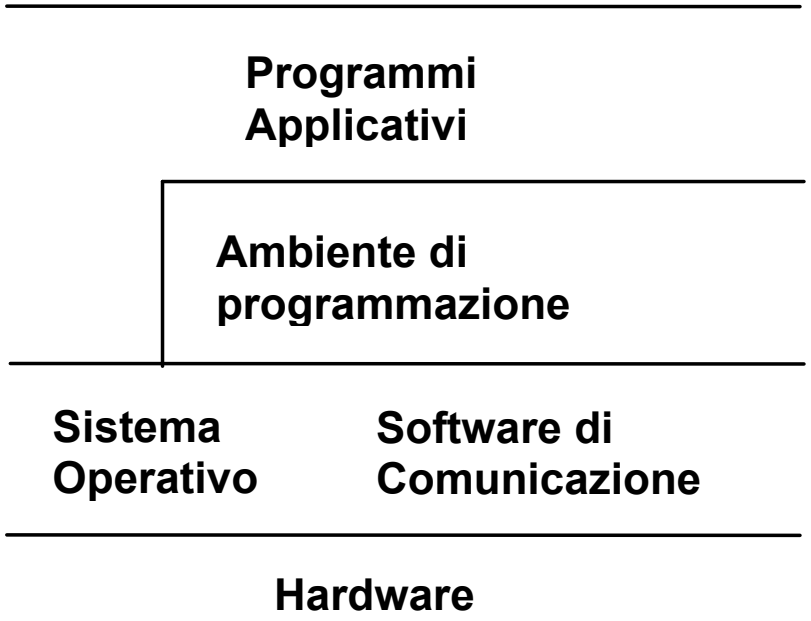
Lavorando direttamente sull'hardware, **l'utente è forzato a esprimere i propri comandi *al livello della macchina*, tramite *sequenze di bit.***

IL SOFTWARE

Software:
insieme di programmi eseguibili dal computer.

Organizzazione a strati, ciascuno con funzionalità di livello più alto rispetto a quelli sottostanti

Concetto di
macchina virtuale



IL FIRMWARE

Firmware:
il confine fra hardware e software.

È uno strato di *micro-programmi*, *scritti dai costruttori dell'hardware*, che agiscono direttamente al di sopra dello strato hardware

Sono memorizzati su una speciale *memoria centrale permanente* (ROM, EPROM, …)

IL SISTEMA OPERATIVO

Programma che opera *al di sopra* dell'hardware fornendo un ambiente di esecuzione per i programmi, nascondendo i dettagli relativi alle caratteristiche fisiche delle componenti e alla loro gestione.

Per lo stesso elaboratore, spesso **si può scegliere tra *diversi sistemi operativi***, con diverse caratteristiche.

Esempi:

- Windows (95 / 98, NT, XP, Vista...)
- Unix
- Linux
- MacOs X...

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
     - esecuzione di programmi
     - gestione dello storage (hard disk, CD, USB stick, ...)
     - interfaccia grafica con il sistema

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
  2. **Astrazione**: fornire una visione astratta delle risorse del calcolatore per i programmatori
     - Ponte tra visione astratta e visione fisica.
     - Esempio: *file system*.

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
  2. **Astrazione**: fornire una visione astratta delle risorse del calcolatore per i programmatori
  3. **Efficienza**: utilizzare in modo efficiente le risorse del calcolatore
     - arbitrare le diverse attività che vengono svolte in un calcolatore
     - promuovendo il parallelismo ed limitando i tempi morti

Cos'è un sistema operativo?

Definizione (Sistema Operativo)

Un sistema operativo è un programma che controlla l'esecuzione di programmi applicativi e agisce come interfaccia tra le applicazioni e l'hardware del calcolatore

- Obiettivi di un sistema operativo:
  1. **Semplicità**: semplificare l'utilizzazione della macchina da parte degli utenti
  2. **Astrazione**: fornire una visione astratta delle risorse del calcolatore per i programmatori
  3. **Efficienza**: utilizzare in modo efficiente le risorse del calcolatore
     - arbitrare le diverse attività che vengono svolte in un calcolatore
     - promuovendo il parallelismo ed limitando i tempi morti

FUNZIONI DEL SISTEMA OPERATIVO

Le funzioni messe a disposizione dal S.O. dipendono dalla complessità del sistema di elaborazione:

- Interazione con l'utente ->interpretazione ed esecuzione di comandi
- gestione delle risorse disponibili:
  - •Cpu: assegnazione delle cpu ai diversi programmi,
  - •Memoria centrale : allocazione della memoria ai programmi
  - •Dispositivi: accesso alle periferiche indipendente dalle caratteristiche hw dei dispositivi; organizzazione e gestione della memoria di massa.
- gestione di un sistema multi-utente/multitasking:
  - •concorrenza delle attivita`
  - •protezione
  - •una macchina astratta (o virtuale) per ogni utente:

Ogni utente "vede" l'elaboratore solo tramite il Sistema Operativo
→ il S.O. realizza una "macchina virtuale"

FUNZIONI DEL SISTEMA OPERATIVO

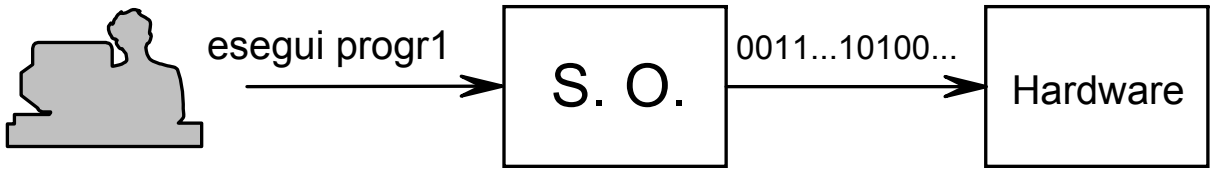
Conseguenza:
S.O. diversi possono realizzare *diverse macchine virtuali* **sullo stesso elaboratore fisico**

Interazione con l'utente:
Attraverso il S.O. il livello di interazione fra utente ed elaboratore viene elevato:

- senza S.O.: sequenze di bit
- con S.O.: comandi, programmi, dati, forniti in modo testuale o grafico

RUOLO DEL SISTEMA OPERATIVO

Il S.O. **traduce le richieste dell'utente** in opportune **sequenze di istruzioni**, a loro volta trasformate in **corrispondenti sequenze di bit** per la macchina fisica.



e viceversa:

RUOLO DEL SISTEMA OPERATIVO

Qualsiasi operazione di accesso a risorse della macchina implicitamente richiesta dal comando di utente **viene esplicitata dal S.O.**

Esempi:

- accesso a memoria centrale
- accesso ai dischi
- I/O verso video, tastiera, …

ESEMPIO

e viceversa:

Utente:	Sistema Operativo:
"esegui progr1"	- input da tastiera - ricerca codice di "progr1" su disco - carica in memoria centrale codice e dati <elaborazione>
Utente:	**Sistema Operativo:**
"stampa 10"	- output su video

Servizi offerti dai sistemi operativi

1. Esecuzione di programmi
   - applicativi e di sistema
   - supporto alla **multi-programmazione**
2. Gestione dell'accesso alla **memoria** del calcolatore
   - principale e secondaria
3. Gestione dell'accesso ai **dispositivi di I/O**
4. Rilevazione e risposta agli **errori**
   - isolare gli effetti degli errori (ambiente protetto di esecuzione)
5. Amministrazione di **utenti** diversi (multi-utenza)
   - occorre separare le attività e i dati di ciascun utente
6. Controllo degli **accessi**
   - l'utente che lancia un programma ha il diritto di eseguirlo?
7. Accounting
   - misura e controllo della **quantità** di risorse usate (es: quota)

Gestione dei Processi

Processi

Definizione (Processo)

Un **processo** *è un'attività controllata da un* **programma** *che si svolge su un* **processore**.

- È il modo in cui un programma viene eseguito **nel tempo**.
  - Programma: entità statica.
  - Processo: entità dinamica.
- Ciascun processo ha uno **stato**, composto da un certo numero di informazioni
  - Immagine di memoria
  - Immagine nel processore
  - Stato di avanzamento

Stato di un processo

Definizione (Immagine di memoria)

*L'***immagine di memoria** *di un processo è l'insieme di informazioni relative al processo mantenute nella memoria principale del calcolatore.*

- **Codice** del programma in esecuzione
- **Dati** su cui sta lavorando
- **Strutture dati del SO** per la gestione del processo

Stato di un processo

Definizione (Immagine nel processore)

*L'**immagine nel processore** di un processo è il contenuto dei registri del processore, che vengono utilizzati come memoria temporanea durante l'esecuzione delle istruzioni.*

- Esempi:
  - **Program counter**: la prossima istruzione da eseguire
  - **Flag**: esito di una operazione della ALU
  - **A**, **B**, ...: operandi

Stato di un processo

Definizione (Stato di avanzamento)

Lo **stato di avanzamento** *descrive l'attuale attività del processo.*

- Esempi di possibili stati di avanzamento
  - **Waiting**: in attesa di qualche evento
  - **Running**: in esecuzione
  - **Ready**: attende di essere eseguito

Multi-programmazione e time-sharing

Definizione (Multi-programmazione)

Tecnica di gestione della CPU secondo la quale durante i **periodi di I/O** *di un processo vengono eseguiti altri processi.*

Definizione (Time-sharing)

Tecnica di gestione della CPU secondo la quale l'esecuzione del processore viene suddivisa in un certo numero di **quanti temporali**; *allo scadere di un quanto, il processo corrente viene interrotto e l'esecuzione passa ad un altro processo.*

- Numerosi vantaggi:
  - Processore non inattivo durante operazioni di I/O (lunghe).
  - Memoria utilizzata al meglio, caricando il maggior numero di programmi possibile.
  - Impressione di esecuzione contemporanea di processi diversi.

Meccanismo degli interrupt

- Implementazione:
  - Tramite un componente del SO detto **scheduler**
  - Invocato ogni volta che una operazione di I/O viene:
    - **iniziata** (da un processo)
    - **terminata** (da un dispositivo)
  - Usa meccanismo degli **interrupt**

→ **richiesta di I/O da parte di un processo**

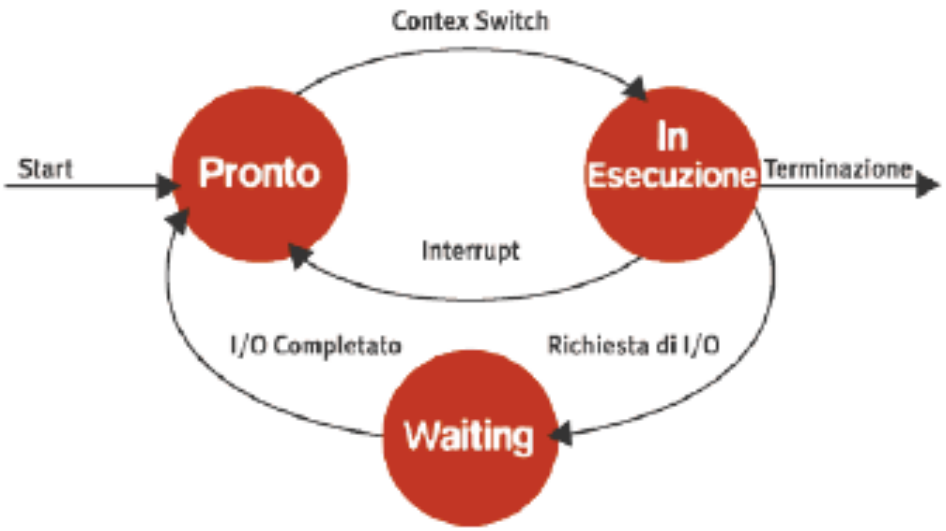
- interrupt software

→ **completamento di tale operazione**

- interrupt hardware generato dal dispositivo associato

- suddivisione **in quanti temporali**
  - tramite un processo **timer**, che è in grado di generare un interrupt periodicamente
- Per passare da un processo a un altro: **context switch**.

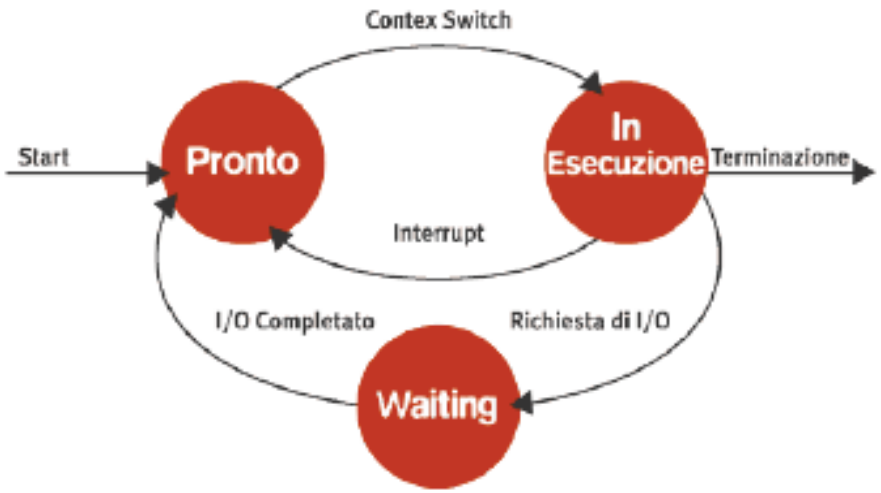
Lo stato di avanzamento dei processi



Lo stato di avanzamento dei processi

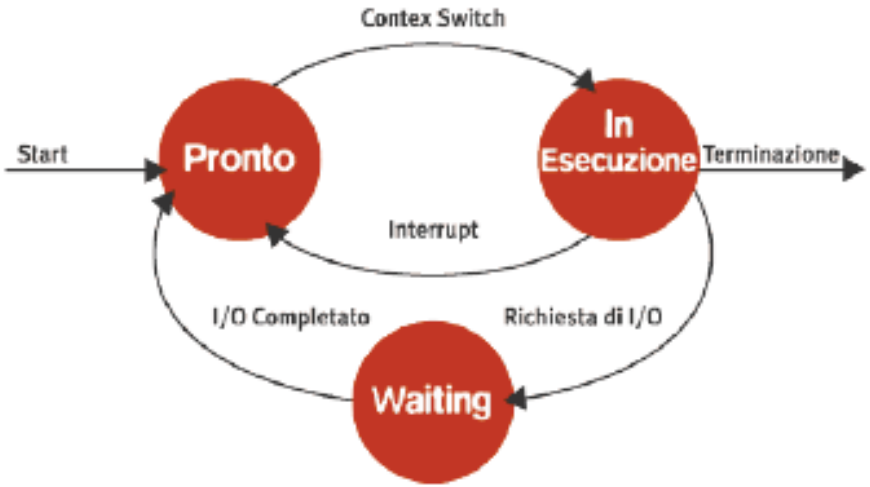
- **In esecuzione (running)**
  - il processo è in esecuzione
    - 1 CPU ⇒ 1 solo processo in esecuzione per volta
    - multi-core: 1 processo per CPU
  - controllo passa al SO se:
    - interrupt hardware ⇒ *ready*
    - interrupt software ⇒ *waiting*

Lo stato di avanzamento dei processi



- **In esecuzione (running)**
- **In attesa (waiting)**
  - il processo è in attesa di evento esterno (tipicamente, I/O)
    - non può essere eseguito
  - quando l'evento si verifica $\Rightarrow$ *ready*

Lo stato di avanzamento dei processi

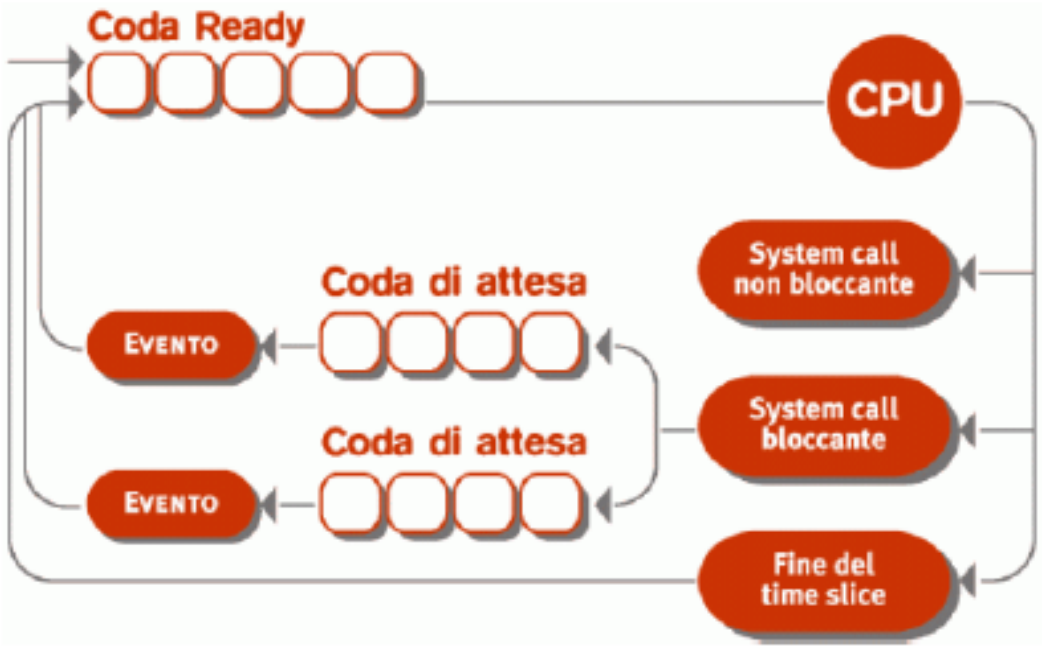


- **In esecuzione (running)**
- **In attesa (waiting)**
- **Pronto (ready)**
  - il processo può essere eseguito, ma
  - la CPU è impegnata da un altro processo
  - il processo attende che il SO lo faccia ripartire ($\Rightarrow$ *running*)

Lo scheduler

Definizione (Scheduler)

Lo **scheduler** *è il componente del sistema operativo che decide di volta in volta quale processo deve essere eseguito.*



Informazioni sui processi attivi (UNIX)

- Il SO tiene traccia di alcune informazioni sui processi, tra cui:
  - Direttorio corrente (**working directory**)
  - Info sui file usati dal processo (**file descriptor table**)
  - ID del processo (**Process ID, PID**)
  - ID del gruppo (**Group ID, GID**)
  - ID del processo "padre" (**Parent Process ID, PPID**)
  - Variabili di ambiente (es: `PATH`)

Come interagire con i processi

- Funzioni del SO per operare con i processi:
  - **Creazione di un processo**: `fork()`
  - **Esecuzione di un programma**: `exec()`
  - **Terminazione**: `exit()`
  - **Sincronizzazione tra processi**: `signal()` e `kill()`
  - **Comunicazione tra processi**: `send()` e `receive()`
- Operazioni comuni per un utente:
  - esecuzione di un programma (es. *double click*)
  - terminazione di un processo (es. *task manager*)

Gestione della Memoria

Gestore della memoria

Definizione (Gestore della memoria)

Il **gestore della memoria** *è il componente del sistema operativo che si occupa di gestire la memoria per conto dei processi.*

- La memoria serve a ogni processo in esecuzione, per contenere **codice** e **dati**.
- La memoria (principale) è una **risorsa limitata** (e costosa):
  - Gerarchia delle memorie
    - Memoria principale (100 €): $\sim$ GB
    - Memoria secondaria (100 €): $\sim$ TB
- Uso di una porzione dell'Hard Disk (**partizione di swap**) per emulare la memoria principale (**memoria virtuale**)

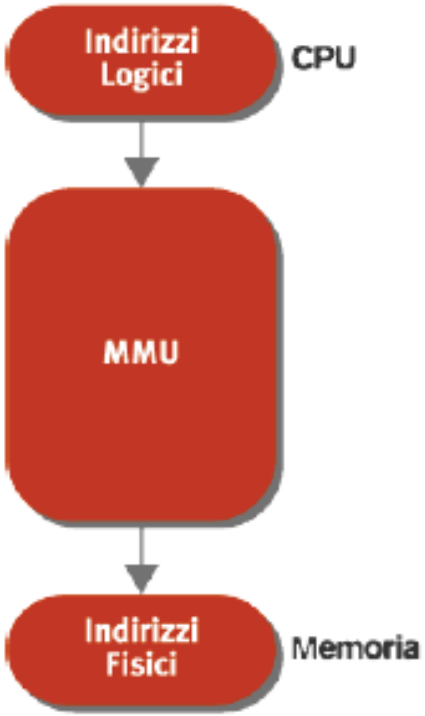
Gestore della memoria

Definizione (Gestore della memoria)

Il **gestore della memoria** *è il componente del sistema operativo che si occupa di gestire la memoria per conto dei processi.*

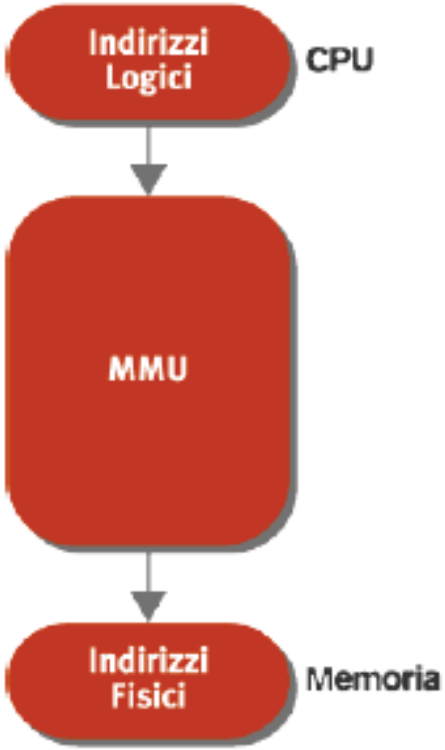
- Compiti del gestore della memoria:
  1. Tenere traccia di quali processi occupano quali porzioni di memoria
     - Uso di **tabelle di allocazione**
  2. Assegnare memoria ai processi che ne facciano richiesta
     - **Allocazione** di **pagine** di memoria
  3. Rilasciare la memoria quando non è più richiesta
     - **Deallocazione** di pagine di memoria

Memoria fisica e memoria logica



- **Spazio di indirizzamento logico**
  - Ogni processo è associato ad uno spazio di indirizzamento logico
  - Gli **indirizzi usati in un processo** sono indirizzi logici
- **Spazio di indirizzamento fisico**
  - Ad ogni indirizzo logico corrisponde un indirizzo fisico della memoria
  - Serve una **funzione di traduzione** da indirizzi logici a indirizzi fisici
  - La Memory Management Unit (MMU) è un dispositivo che esegue tale funzione

Memoria fisica e memoria logica



- Vantaggi di questa traduzione:
  - È meglio **nascondere** ai processi la reale organizzazione della memoria
    - Per **semplificare** il codice
    - → ad es: il codice usa sempre gli indirizzi da 0 a 1000
    - → indipendentemente dal gestore
    - → anche se in RAM non esistono tutti gli indirizzi da 0 a 1000
    - Per **proteggere** la memoria
    - → evitare che un processo interagisca su zone su cui non ha i diritti
  - La MMU può **ottimizzare** l'utilizzo della memoria
    - Il singolo processo non può sapere

File System

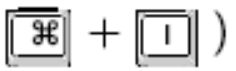
Il file system: un'astrazione

- I computer possono utilizzare **diversi media** per registrare in modo permanente le informazioni
  - esempi: dischi rigidi, floppy, nastri, dischi ottici
  - ognuno di questi media ha **caratteristiche fisiche diverse**
- Un **file system** nasconde la complessità dei diversi media proponendo una astrazione:
  - indipendente dal supporto di memorizzazione
  - efficiente
  - conveniente da usare
- **Per l'utente**, un file system è composto da **due elementi**:
  - **file**: l'**unità logica di memorizzazione**;
  - **directory** (**folder**): un insieme di informazioni per **organizzare i file** che compongono un file system

File

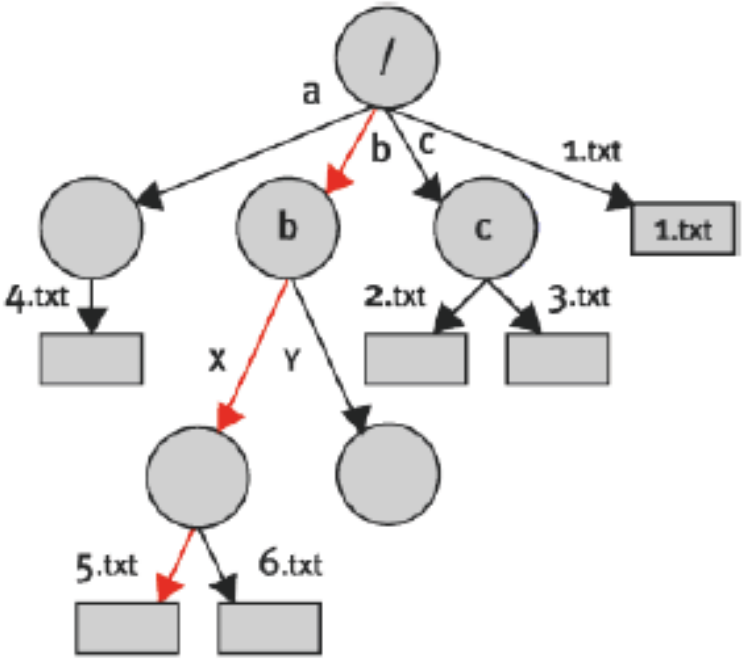
- Ogni file è caratterizzato da un insieme di **attributi**:
  - **nome** del file
  - informazioni su **locazione** e **dimensione**
    - difficile memorizzare su blocchi contigui ⇒ frammentazione
    - utilizzo di **indici** per tener traccia dei blocchi occupati
  - informazioni **temporali**: creazione, ultima modifica, ultimo accesso
  - **proprietà**: per associare un file a un proprietario (**owner**)
    - quali sono i **privilegi** dell'owner sul file
    - chi ha il permesso di **leggere**, **modificare**, **eseguire** un file
  - **tipo**: file, immagini, ... Per indicare il tipo:
    - uso di **estensioni** del nome, es. `Lucidi_07.pdf`
    - informazioni **nel contenuto**, es. primi caratteri: `%PDF...`

Come operare sui file

- Funzioni del SO per operare con i file:
  - **Creazione di un file**: `create()`
  - **Apertura/chiusura di un file**: `open()`/`close()`
  - **Lettura/modifica**: `read()`/`write()`
  - **Posizionamento**: `lseek()`
  - **Cancellazione**: `unlink()`
  - **Modifica degli attributi**: `chmod()`, `chown()`
- L'**apertura** serve per caricare tutte le informazioni sulla gestione di un file in alcune strutture dati del SO
- La **lettura** e **scrittura** avvengono tramite l'astrazione di una **testina di lettura**
- Operazioni comuni per un utente:
  - esecuzione di un programma (es. *double click*)
  - spostamento di un file (es. *drag & drop*)
  - visualizzazione di informazioni sul file (es. )

Directory

- Concetto fondamentale per **organizzazione gerarchica**
  - Struttura **ad albero**, parte da un direttorio **radice**
  - Directory ⇒ **nodi**, file ⇒ **foglie**
  - Percorso (**path**) per raggiungere un file:
    - Relativo: a partire dal direttorio corrente (.)
    - Assoluto: a partire dal direttorio radice (/)



Come operare sulle directory

- Funzioni del SO per operare con i file:
  - **Creazione** di una directory: `mkdir`
  - **Modifica del direttorio corrente**: `cd`
  - **Visualizzazione del direttorio corrente**: `pwd`
  - **Modifica** di una directory $\Leftrightarrow$ **creazione/cancellazione di un file**: `create()`, `unlink()`
  - **Modifica degli attributi di una directory** `chmod()`, `chown()`
  - **"Esecuzione" di una directory $\Leftrightarrow$ transito** `cd`
  - **Linking di un file**: `ln`
- Operazioni comuni per un utente:
  - modifica del direttorio corrente (es. *double click*)
  - rinominazione di un direttorio (es. *click*)
  - visualizzazione di informazioni sulla directory (es. ⌘ + I)

Sicurezza nei Sistemi Operativi

Sicurezza

- Problemi di sicurezza visti per Internet:
  - **Disponibilità.** Ciò che inviamo viene ricevuto dal destinatario?
  - **Confidenzialità.** Ciò che inviamo viene letto solo dal destinatario?
  - **Autenticità.** Sappiamo con certezza chi è il mittente?
  - **Integrità.** Sappiamo che il documento non è stato modificato nel tragitto?
- In un Sistema Operativo, il problema della sicurezza:
  - è l'insieme di meccanismi che vengono utilizzati per il **controllo di accesso alle risorse**
  - in generale, coinvolge **non solo il sistema di calcolo**, ma anche aspetti amministrativi e legali

Problemi fondamentali relativi alla sicurezza

- **Autenticazione**: associare ad un utente l'identità
- **Autorizzazione**: verificare se un utente ha il diritto di compiere un'operazione
- **Protezione**: evitare che un'operazione venga compiuta da chi non ne ha i diritti

Rendere sicuro un sistema

- Quali tipi di attacchi attesi?
  - **Attacchi passivi**: accedere a contenuto senza modificarlo
  - **Attacchi attivi**: forzare l'integrità o disponibilità del sistema.
- **Politica di sicurezza** deve essere adeguata al **valore** del sistema da proteggere.
- Tipi di attacchi attesi variano da sistema a sistema:
  - Banche: attacchi mirati a sottrarre denaro
  - Siti Web: denial of service o modifica del contenuto
- Valutare anche il **costo** di una particolare politica di sicurezza

Autenticazione

- Meccanismi che devono essere basati sull'utente
- Possono concentrarsi su tre tipi di elementi:
  - Qualcosa che l'utente **sa**: un **dato segreto concordato** in precedenza (password o PIN)
  - Qualcosa che l'utente **possiede**: un **oggetto riconoscibile** da parte della macchina (scheda magnetica o smart card)
  - Qualcosa che l'utente **è**: conformità di una **caratteristica fisiologica o comportamentale** con un dato "biometrico" di riferimento.
- Tecniche con **sicurezza** e **costo** crescenti.
  - Password: tecnica più largamente utilizzata
  - Bancomat: tessera magnetica + PIN
  - Ingresso in banca: a volte, riconoscimento della retina

Debolezze dei sistemi basati su password

- Molti problemi causati da una scarsa cultura della sicurezza.
  - Dove conservare la password?
  - Password facili da ricordare e/o difficili da indovinare?
  - Attacchi brute-force e basati su dizionario
  - Password "sbirciata" di nascosto
  - Login spoofing
  - Sniffing (per password trasmesse in chiaro)
  - Keystroke logging, metodi acustici, elettromagnetici, telecamere
    - Anche per le normali chiavi! (key bitting), `http://vision.ucsd.edu/~blaxton/sneakey.html`
  - Parecchie idee su: `www.schneier.com`

Autorizzazione

- Due tipi di entità nei Sistemi Operativi:
  - **entità attive**: processi
  - **entità passive**: risorse, quali file, aree di memoria, ...
- Entità attive compiono operazioni sulle entità passive **per conto di utenti**.
- Compiti del meccanismo di autorizzazione:
  1. permettere agli utenti di **specificare le azioni** che un'entità attiva può compiere o meno su un'entità passiva
  2. a ogni **richiesta** di svolgere un'azione svolte da partedi un'entità attiva, **verificare** se è ammissibile
- Tale meccanismo **sovrintende ogni chiamata di sistema**
- Principio del **privilegio minimo**: consentire a ciascun processo il minimo indispensabile dei privilegi per svolgere un compito
  - Esempio: `passwd`

Protezione

- Per rappresentare l'**insieme di regole di autorizzazione**, si utilizza il concetto di **dominio di protezione**
  - Descrizione di cosa si può fare e cosa no su una certa entità
  - Insieme di coppie <*Entità passiva*, *operazione consentita*>

Definizione (dominio di protezione)

Un dominio di protezione è un insieme di associazioni (coppie) che descrivono un insieme di entità passive e i tipi di operazioni che possono essere effettuati su ognuna di esse.

- Ogni processo opera all'interno di un **dominio di protezione**
- In ciascun dominio di protezione possono trovarsi 0, 1 o più utenti e/o processi.

Realizzazione dei domini di protezione

- Varia a seconda del SO
- Windows: **Access Control List**
  - Ogni entità passiva è associata ad una lista, che contiene delle coppie <*dominio di protezione*, *operazione consentita*>
- UNIX/Linux: UGO +/- RWX
  - Ogni entità passiva è associata a due identificatori: **utente proprietario** (UID) e **gruppo di utenti** (GID)
  - Si divide il mondo dei processi in tre insiemi:
    - User (U)
    - Group (G)
    - Others (O)
  - A ciascuno di questi insiemi vengono garantiti/negati diritti di
    - Lettura (R)
    - Scrittura (W)
    - Esecuzione (X)

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**
- **Backdoor**
- **Worm**
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
  - Scopo: garantire accesso da remoto
  - Atttività più frequenti: spedire SPAM, rubare, modificare e/o distruggere dati (tra cui: password), caricare e scaricare file, spiare l'attività dell'utente
  - Portata del potenziale danno dipende dai privilegi della vittima
- **Backdoor**
- **Worm**
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
  - Tipicamente una debolezza già presente nel SO (senza installare programmi aggiuntivi)
- **Worm**
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
  - usa la rete per inviare copie di se stesso ad altri computer
    - es: usando rubrica e-mail per selezionare future vittime
  - non è attaccato a un programma esistente
  - tipico effetto: malfunzionamenti dovuti a consumo di risorse
- **Virus**
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
  - non viene eseguito autonomamente
  - worm e virus possono rimanere in letargo per periodi di tempo
- **Rootkit**
- **Grayware (spyware e adware)**
- **Dialer**

Malware

- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
- **Rootkit**: un sistema software di uno o più programmi con lo scopo di mascherare una avvenuta presa di controllo
- **Grayware (spyware e adware)**: alcune applicazioni fastidiose o indesiderate ma non particolarmente nocive
  - **Spyware**: registra consuetudini di navigazione in rete (**profiling**)
  - rischio di registrare informazioni private
  - **Adware**: software che mostra banner nei browser Web
- **Dialer**

Malware

- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
- **Rootkit**: un sistema software di uno o più programmi con lo scopo di mascherare una avvenuta presa di controllo
- **Grayware (spyware e adware)**: alcune applicazioni fastidiose o indesiderate ma non particolarmente nocive
- **Dialer**: programmi che stabiliscono una connessione a pagamento senza che la vittima se ne accorga
  - tipico scopo: ricavo da una connessione a pagamento provocata verso un numero costoso

Malware

- Molte categorie di software potenzialmente nocivo
- **Trojan**: programmi che simulano le funzionalità di programmi innocui, ma che contengono codice nocivo.
- **Backdoor**: un metodo per ad aggirare le normali procedure di autenticazione.
- **Worm**: un programma autoreplicante, che si diffonde autonomamente senza intervento degli utenti del sistema
- **Virus**: un programma autoreplicante che si installa all'interno di un programma legittino
- **Rootkit**: un sistema software di uno o più programmi con lo scopo di mascherare una avvenuta presa di controllo
  - attaccano il SO, e si installano come driver o moduli di kernel
  - spesso capaci di eludere anti-virus e anti-spyware
  - uso di backdoor
- **Grayware (spyware e adware)**
- **Dialer**

CLASSIFICAZIONE dei S.O.

In base al numero di utenti:

- **Mono-utente (*mono-user*):** un solo utente alla volta può utilizzare il sistema
- **Multi-utente (*multi-user*):** più utenti possono interagire contemporaneamente con la macchina.

Nel caso di più utenti contemporanei, **il Sistema Operativo deve fornire a ciascuno l'astrazione di *un sistema "dedicato".***

CLASSIFICAZIONE dei S.O.

In base al numero di programmi in esecuzione:

- **Mono-programmato (*mono-tasking*):** si può eseguire *un solo programma* per volta
- **Multi-programmato (*multi-tasking*):** il S.O. è in grado di portare avanti contemporaneamente l'esecuzione di più programmi (pur usando una sola CPU).

Nel caso di multi-programmazione **il S.O. deve gestire la suddivisione del tempo** della CPU fra i vari programmi.

CLASSIFICAZIONE dei S.O.

Esempi:

- **MS-DOS**: monoutente, monoprogrammato
- **Windows95/98**: monoutente, multiprogrammato
- **Windows XP, Vista**: mono/multiutente,multiprogrammato
- **UNIX e Linux**: multiutente, multiprogrammato

PROGRAMMI APPLICATIVI

Risolvono problemi specifici degli utenti:

- *word processor*: elaborazione di testi
- *fogli elettronici*: gestione di tabelle, calcoli e grafici
- *database*: gestione di archivi
- *suite* (integrati): collezione di applicativi capaci di funzionare in modo integrato come un'applicazione unica.

- Sono scritti in **linguaggi di programmazione** di alto livello
- Risentono in misura ridotta delle caratteristiche della architettura dell'ambiente sottostante (*portabilità*)

AMBIENTI DI PROGRAMMAZIONE

È l'insieme dei programmi che consentono la scrittura, la verifica e l'esecuzione di nuovi programmi (***fasi di sviluppo***).

Sviluppo di un programma:

- Affinché un programma scritto in un qualsiasi linguaggio di programmazione sia comprensibile (e quindi eseguibile) da un calcolatore, *occorre* ***tradurlo*** dal linguaggio originario al linguaggio della macchina.
- Questa operazione viene normalmente svolta da speciali programmi, detti ***traduttori***.

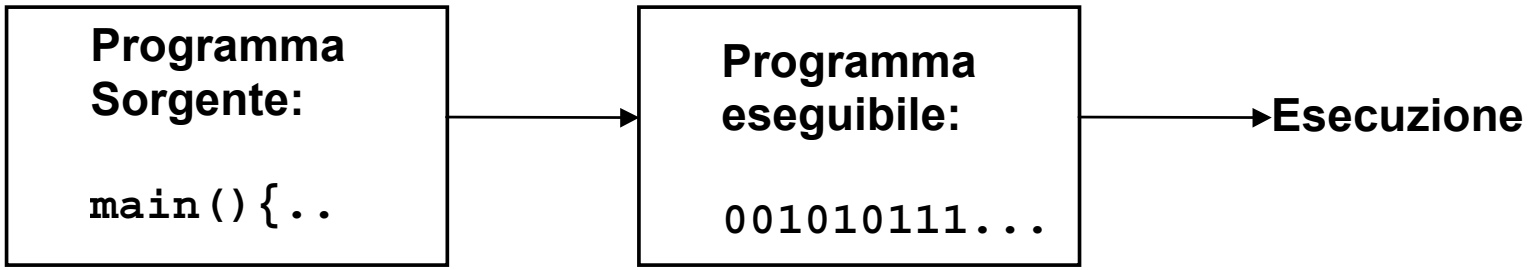
TRADUZIONE DI UN PROGRAMMA

PROGRAMMA	TRADUZIONE
`main()`	
`{ int A;`	`00100101`
`...`	
`A=A+1;`	`11001..`
`if....`	`1011100..`

Il traduttore converte

- ***il testo*** di un algoritmo scritto in un particolare linguaggio di programmazione (***sorgenti***)
- nella corrispondente ***rappresentazione in linguaggio macchina*** (programma ***eseguibile***).

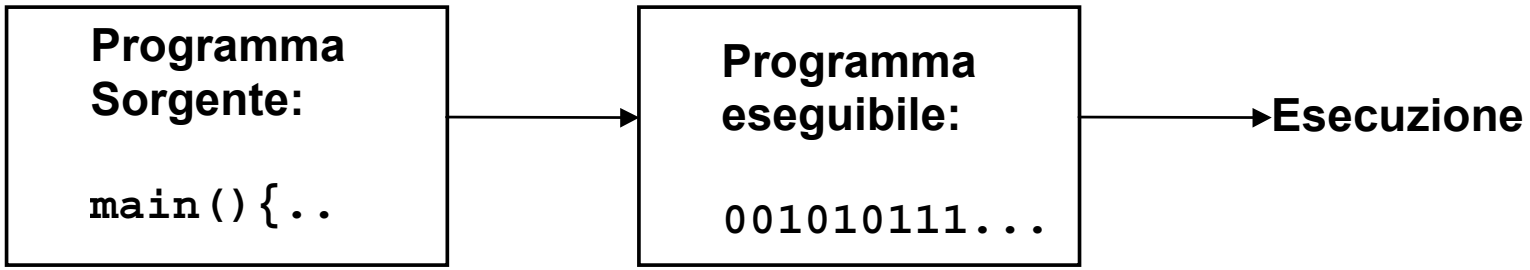
SVILUPPO DI PROGRAMMI



Due categorie di traduttori:

- i ***Compilatori*** traducono l'intero programma (senza eseguirlo!) e producono in uscita il programma convertito in linguaggio macchina
- gli ***Interpreti*** traducono ed eseguono immediatamente **ogni singola istruzione** del *programma sorgente.*

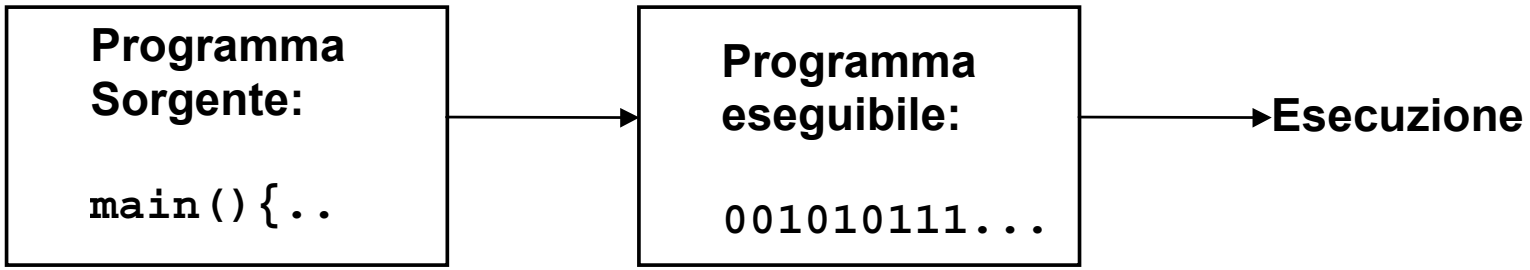
SVILUPPO DI PROGRAMMI (segue)



Quindi:

- **nel caso del compilatore**, lo schema precedente viene percorso ***una volta sola*** prima dell'esecuzione
- **nel caso dell'interprete**, lo schema viene invece attraversato ***tante volte quante sono le istruzioni*** che compongono il programma.

SVILUPPO DI PROGRAMMI (segue)



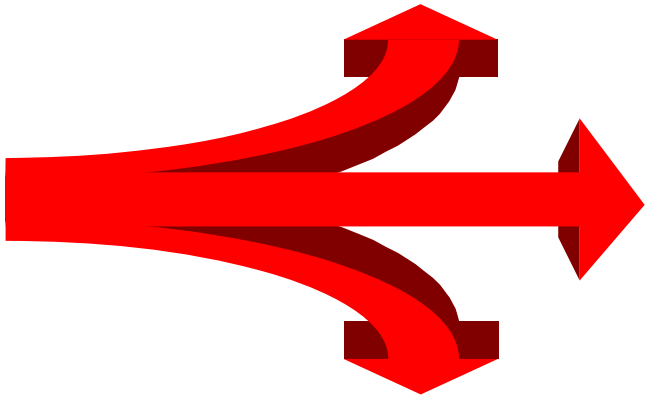
L'esecuzione di un programma ***compilato*** è **più veloce** dell'esecuzione di un programma ***interpretato***

AMBIENTI DI PROGRAMMAZIONE

COMPONENTI

- **Editor**: serve per creare file che contengono **testi** (cioè sequenze di caratteri).
  In particolare, l'editor **consente di scrivere il *programma sorgente*.**

E poi….



AMBIENTI DI PROGRAMMAZIONE

I° CASO: COMPILAZIONE

- **Compilatore**: opera la **traduzione di un programma *sorgente*** (scritto in un linguaggio ad alto livello) **in un *programma oggetto*** direttamente eseguibile dal calcolatore.

PRIMA si traduce ***tutto il programma***
POI si esegue **la versione tradotta**.

AMBIENTI DI PROGRAMMAZIONE (2)

I° CASO: COMPILAZIONE (segue)

- **Linker**: (*collegatore*) nel caso in cui la costruzione del programma oggetto richieda l'unione di ***più moduli*** (compilati separatamente), il linker provvede a **collegarli** formando un unico *programma eseguibile.*
- **Debugger**: (*"spulciatore"*) consente di **eseguire passo-passo** un programma, **controllando via via quel che succede**, al fine di ***scoprire ed eliminare errori*** non rilevati in fase di compilazione.

AMBIENTI DI PROGRAMMAZIONE (3)

II° CASO: INTERPRETAZIONE

- **Interprete**: ***traduce ed esegue*** direttamente *ciascuna istruzione* del *programma sorgente*, **istruzione per istruzione**.
  È alternativo al compilatore (raramente sono presenti entrambi).

Traduzione ed esecuzione sono *inter-calate*, e avvengono *istruzione per istruzione.*

PERSONAL COMPUTER

PC (ex "IBM-COMPATIBILI")

Usano processori della famiglia *Intel 80x86*:

- 8086
- 80286
- …
- Pentium
- Pentium MMX
- Pentium II
- Pentium III
- Pentium IV

Le prestazioni dipendono da:

- frequenza dell'orologio di sistema (*clock*)
- dimensione della RAM
- velocità/parallelismo delle linee dati/comandi (bus)

ALTRI SISTEMI DI CALCOLO

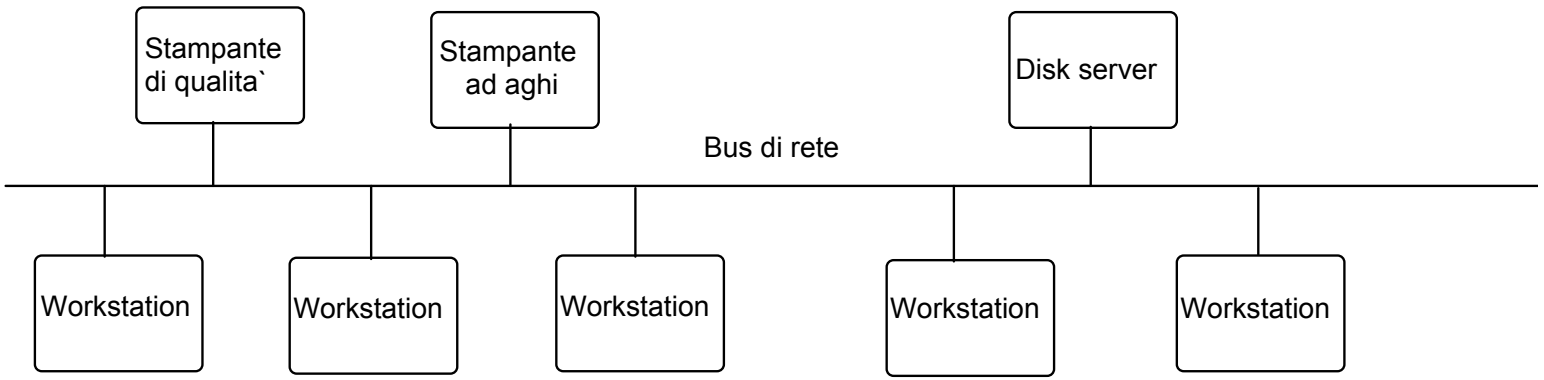
Workstation
sistemi con capacità di supportare più attività contemporanee, spesso dedicati a più utenti. Prestazioni normalmente superiori a quello di un tipico Personal Computer.

Mainframe
Macchine capaci di servire decine di utenti contemporaneamente, collegati tramite terminali

Super-calcolatori
Hanno molti processori, grandi memorie di massa e servono tipicamente centinaia o migliaia di terminali.

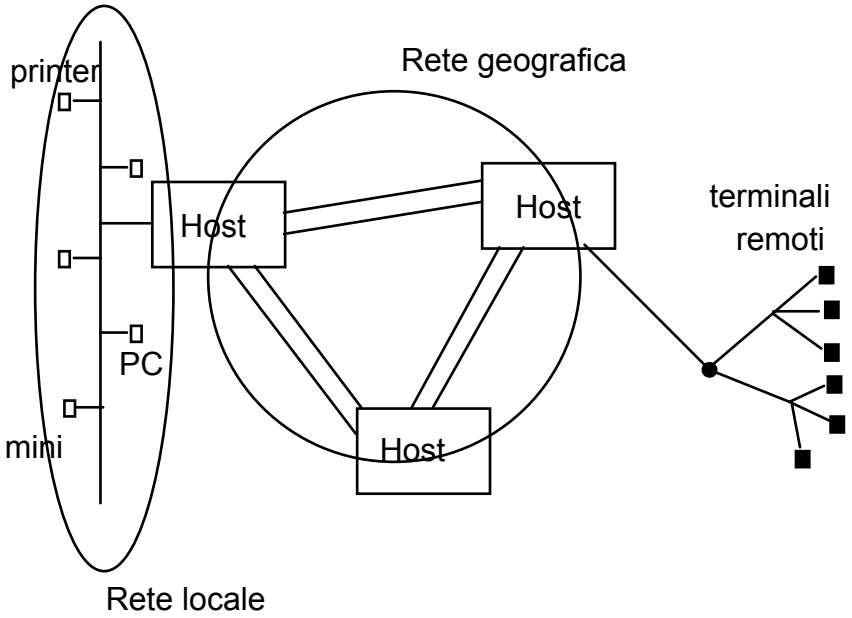
RETI DI CALCOLATORI

- **Reti Locali:**
  connettono elaboratori ***fisicamente vicini***
  (nello stesso ufficio o stabilimento).
- **LAN (Local Area Network)**



RETI DI CALCOLATORI (segue)

- **Reti *geografiche*:** collegano elaboratori medio-grandi situati anche ***a grande distanza***.
- **WAN (Wide Area Network)**



INTERNET: la rete delle reti

- **Internet:** la rete risultante dalla interconnessione mondiale di tutte le reti.
- Milioni di elaboratori ("**siti**") collegati **a ragnatela**
- **World-Wide Web (WWW)**