Introduzione all'Informatica

Loriano Storchi

loriano@storchi.org

http://www.storchi.org/

SOFTWARE

Software

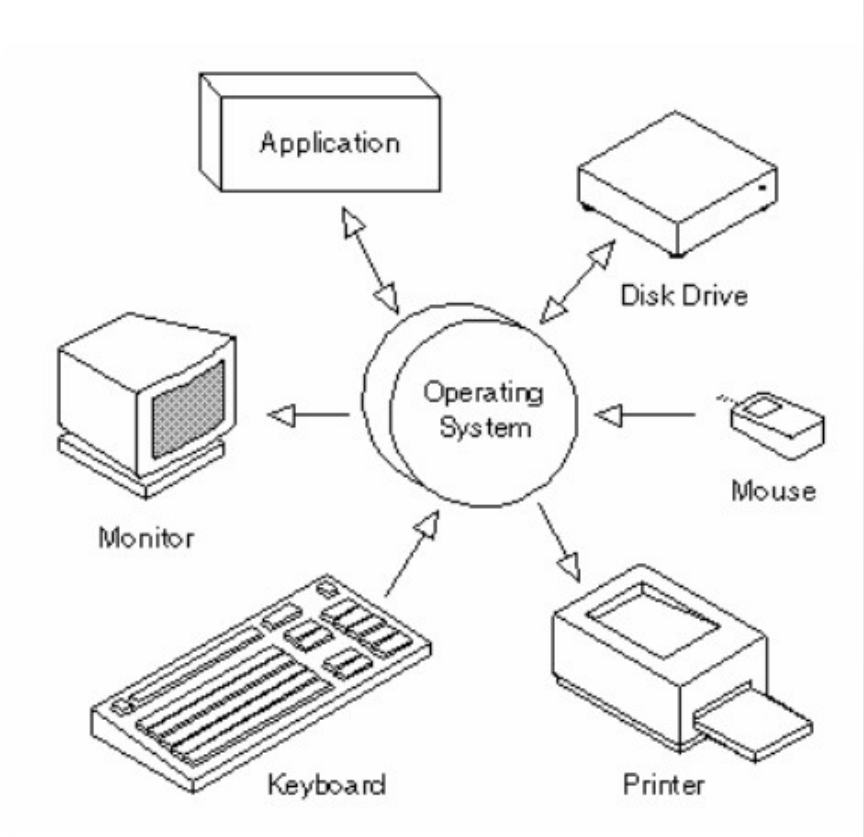
- **Software di base**: dedicato alla gestione dell'elaboratore stesso, ad esempio il Sistema Operativo

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

SISTEMI OPERATIVI

Sistema Operativo

- **E’ un software low-level che assiste l’utente e le applicazioni ad alto livello** ad interagire con l’hardware ed i dati che i programmi immagazzinano nel computer
- Un **OS esegue le operazioni base**, come ad esempio riconoscere l’input dalla tastiera, mandare l’output al display, tenere traccia delle directory e dei files nel disco, e controllare le periferiche come le stampanti

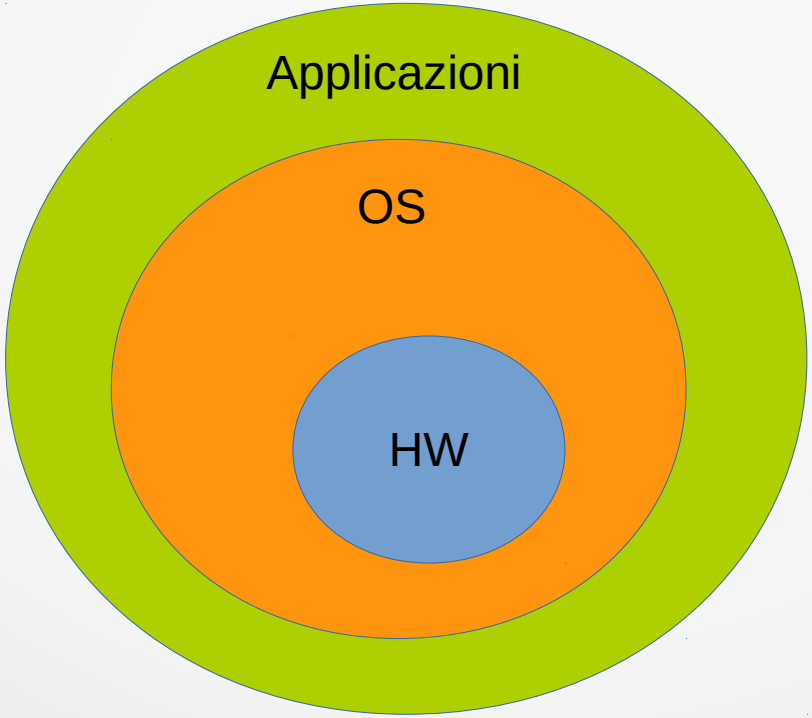


Sistema Operativo

- **Esecuzione dei programmi**: gli OS mettono a disposizione un ambiente dove l'utente puo' eseguire programmi applicativi senza doversi preoccupare dell'**allocazione della memoria o dello scheduling della CPU**
- **Operazioni di I/O**: Ogni applicazione richiede un qualche input e produce un qualche output. L'**OS nasconde i dettagli necessari a gestire questo tipo di operazioni** rispetto all'hardware sottostante
- **Comunicazione**: Ci sono situazioni in cui due **applicazioni devono comunicare l'una con l'altra, sia che si trovino sullo stesso computer (processi differenti), che su computer differenti**. L'OS si occupa di gestire questo tipo di **inter-process communication**

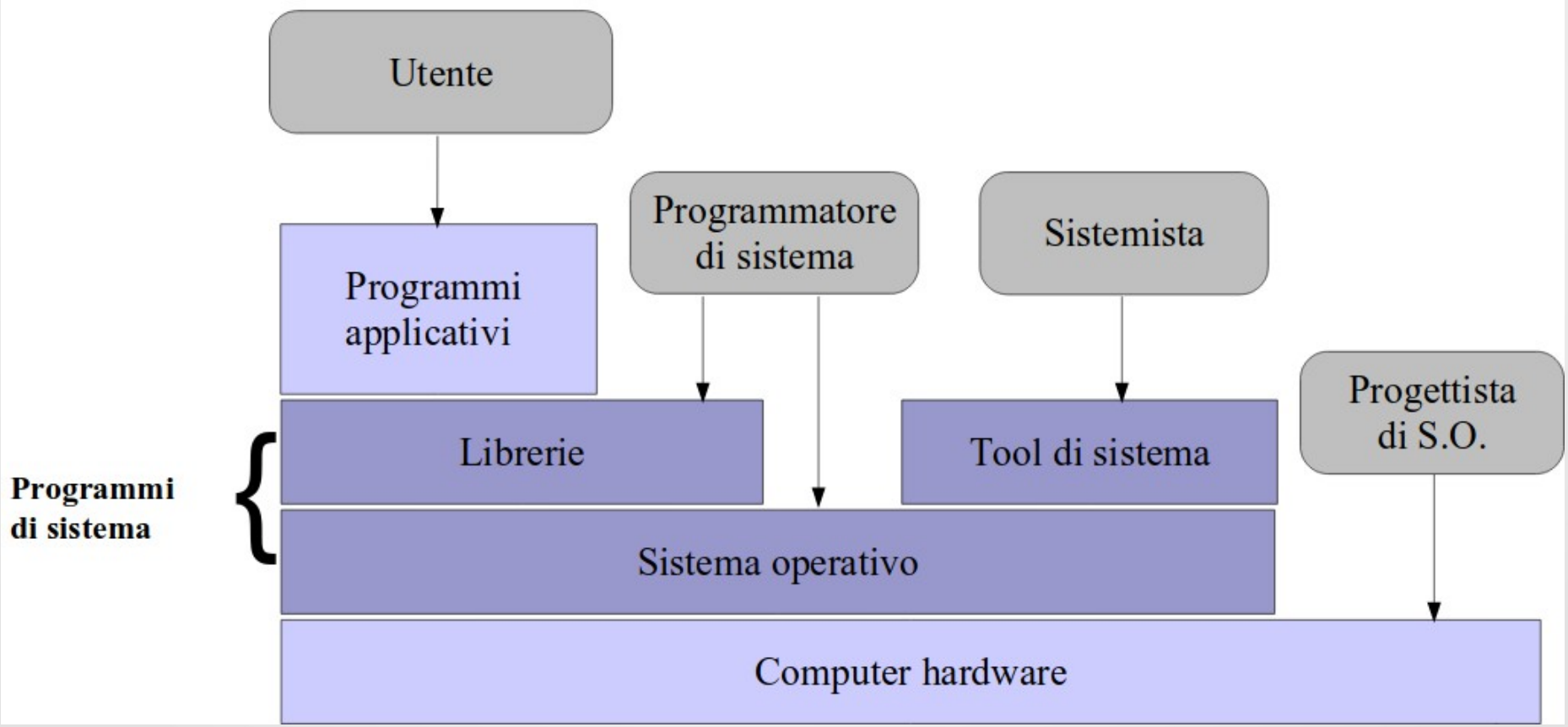
Sistemi Operativi

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



Sistemi Operativi

Visione a strati delle componenti hardware e software, Fornisce ad esempio al programmatore **un API facile da usare**, nasconde i dettagli dell'hardware



SISTEMI OPERATIVI STORIA E CARATTERISTICHE SALIENTI

Sistema Operativo: storia

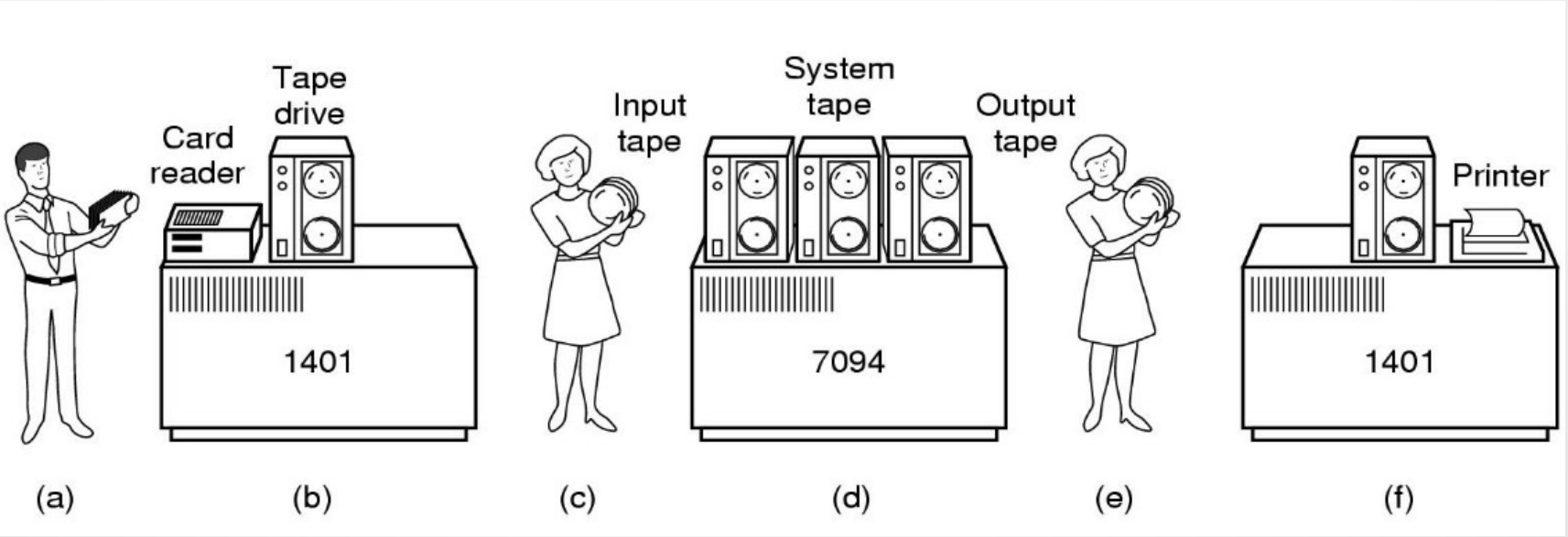
- **Babbage** (1792-1871) Cerca di costruire una **macchina meccanica** analitica e programmabile senza sistema operativo
  - **Prima programmatrice della storia Lady Ada Lovelace** (figlia di Byron)
- **Macchine a valvole** (1944-1955), sono **progettate , costruire e programmate da un singolo gruppo di persone**
  - **Programmate in linguaggio macchina**, usate per il solo calcolo numerico
  - **Non esiste OS**, non esiste distinzione fra progettista, programmatore ed utente
  - **Il singolo utente scrive i programma**, lo carica, carica i dati aspetta l'output e poi passa all'esecuzione del programma successivo
  - **Nessuno immagina che i calcolatori andranno mail oltre i laboratori di ricerca**

Sistema Operativo: storia

- **I transistor** (1955-1965) si possono costruire macchine piu' affidabili ed economiche
  - Si iniziano ad usare per **compiti diversi dal calcolo numerico** di base
  - **Chi costruisce la macchine e' diverso da chi la programma e quindi usa (utente == programmatore)**
  - Si introducono i **primi linguaggi ad "alto livello"** come Assembly e FORTRAN e si usano **schede perforate**
  - **Primi esempi di OS,** detti **Monitor residenti** :
    - **Controllo** sulla macchina e' dato al **monitor**
    - Il **controllo** e' **passato al job** che viene seguito
    - Una volta terminato il job, il **controllo torna al monitor**

Sistema Operativo: storia

- Per evitare i tempi morti fra l'esecuzione di un job e l'altro vengono introdotti i **nastri per la memorizzazione dei job**



Sistema Operativo: storia

- **I circuiti integrati** (1965 – 1980) **sparisce la figura dell'operatore come interfaccia verso il computer**:
  - La **Programmazione** avviene principalmente usando **linguaggi ad alto livello** come il C
  - Arrivano gli **editor di testi e** terminali che permettono di operare
  - **Compaiono i sistemi operativi** con caratteristiche moderne:
    - **Interattivi**
    - **Multi-programmazione**
    - **Time sharing**

Sistema Operativo: Multiprogrammazione

- **Utilizzare il processore mentre altri job stanno facendo i/O**
- Ci sono quindi **piu' job contemporaneamente** in memoria
- Lo **scheduler** (componente dell'OS) **gestisce l'uso della CPU**, mentre un job fa i/O l/altro usa la CPU , si **eliminano cosi' i tempi morti** (idle della CPU)
- Quindi **le routine di i/O devono essere fornite dall'OS**
- Il sistema operativo deve occuparsi di **allocare la memoria per i job multipli**
- Allo stesso modo l'OS deve essere in grado di **allocare le risorse di i/O fra processi diversi**

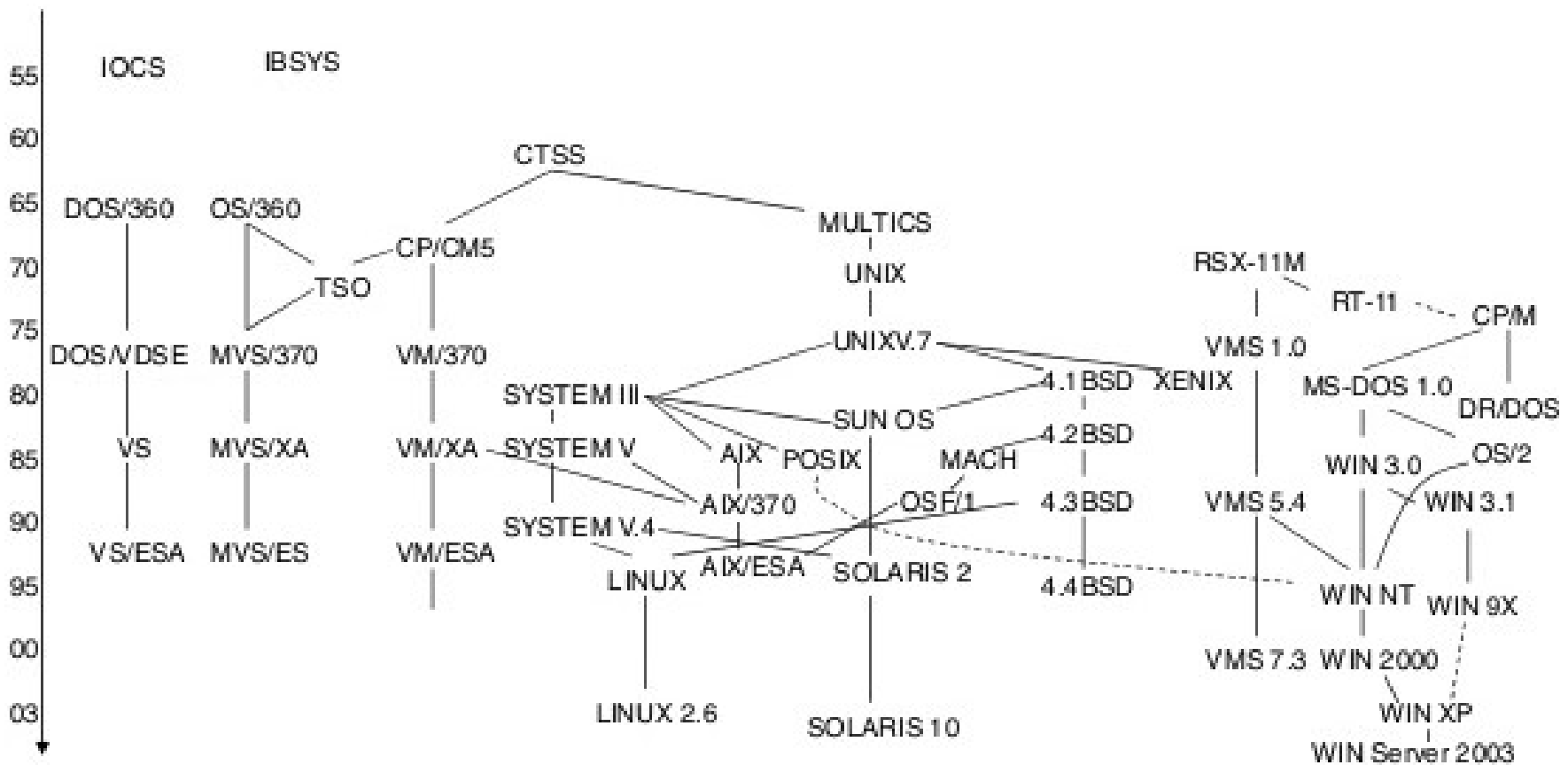
Sistema Operativo: Time-sharing

- E’ in buona sostanza l’**estensione del concetto di multiprogrammazione**
- Il tempo di esecuzione della CPU viene diviso in intervalli (**quanti**). Allo scadere di un quanto l’esecuzione del job corrente viene interrotta, anche se non deve fare i/O, e si passa ad un altro processo (job) . Processi in generale appartenenti ad utenti diversi
- Il **context-switch (passaggio)** e’ molto rapido e trasparente all;utente che ha l’**impressione che vengano eseguiti molti programmi in parallelo**
- La presenza di **piu’ utenti** rende necessario l'inserimento di meccanismi di protezione della memoria e del file-system
- Un job viene caricato dal disco alla memoria, e viceversa (**swapping**)

Sistema Operativo: Storia

- **CTSS (Compatible Time-Sharing System) 1965**: viene introdotto il concetto di multi programmazione ed il concetto di time-sharing
- **Multics 1965**: introduce il concetto di processo
- **Unix 1970**: derivato di Mutics e CTSS inzialmente sviluppato ai Bell-labs . Inizialmente sviluppato su due architetture specifiche, **poi sviluppato in C** (inizialmente tutto in assembly).

Sistemi Operativi: Storia



TERMINOLOGIA UN SUNTO

Sistema Operativo

- Tipicamente in un sistema informatico ci sono **(molti) processi concorrenti e “poche” risorse**, ad esempio la CPU. Un sistema operativo deve **gestire l'allocazione delle risorse**.
- Esiste un termine di sistemi operativi diffuso per tale allocazione noto come **Scheduling**
- A seconda del tipo di sistema operativo, gli obiettivi dello scheduling dei processi possono essere diversi.
- Indipendentemente dal tipo di sistema, al momento di **allocare le risorse deve essere prevista una certa equità**, la politica dichiarata deve essere applicata e l'utilizzo complessivo del sistema deve essere **bilanciato**.
- Gli obiettivi che devono essere raggiunti caratterizzano il tipo di sistema operativo.

Sistema Operativo: Multiprogrammato

- In un sistema operativo **multiprogrammazione** ci sono uno o **più programmi (processi) residenti nella memoria** principale del computer pronti per essere eseguiti. Solo un programma alla volta potra' usare la CPU, gli altri sono in attesa del loro turno.
- Se un dato programma (processo) chiede un'operazione di I/O mentre viene eseguita l''operazione in questione l'uso della CPU e' passato ad un altro processo
- **Il programma in esecuzione continua fino a quando non restituisce volontariamente la CPU o quando si blocca per svolgere funzioni di I/O**.
- Come potete vedere, l'obiettivo è molto chiaro: **i processi in attesa di IO non dovrebbero bloccare altri processi** che a loro volta sprecano il tempo della CPU. L'idea è di mantenere la CPU occupata finché ci sono processi pronti per essere eseguiti.

Sistema Operativo: Multiprogrammato

- Affinché tale sistema funzioni correttamente, **il sistema operativo deve essere in grado di caricare più programmi in partizioni separate della memoria** principale e **fornire la protezione** richiesta onde evitare che un processo acceda alla partizione di memoria dell'altro.
- Quando si hanno più programmi in memoria l'OS deve gestire la frammentazione, di fatto quando i programmi entrano o escono (swapping) dalla memoria principale.
- Un altro problema che deve essere gestito è che **programmi di grandi dimensioni potrebbero non poter essere caricati in una sola volta in memoria, questo aspetto e' generalmente gestito mediante il concetto di Virtual Memory**. Nei moderni sistemi operativi i programmi sono suddivisi in blocchi di dimensioni uguali chiamati pagine
- il termine multi-programmazione è un termine **vecchio** perché nei moderni sistemi operativi l'intero programma non è caricato completamente nella memoria principale

Sistema Operativo: Multitasking

- Il **multitasking** ha lo stesso significato della multiprogrammazione in senso generale poiché entrambi si riferiscono al fatto di **avere più (programmi, processi, attività, thread) in esecuzione allo stesso tempo**.
- Multitasking è il termine utilizzato nei moderni sistemi operativi quando più attività condividono una risorsa comune (CPU e memoria). In qualsiasi momento la CPU sta eseguendo un'attività solo mentre altre attività attendono il proprio turno.
- **L'illusione del parallelismo** si ottiene quando la CPU viene riassegnata a un'altra attività (context switch). Esistono poche differenze principali tra multitasking e multiprogrammazione.
- Un **task in un sistema operativo multitasking non è necessariamente un intero programma applicativo** (i programmi nei moderni sistemi operativi sono suddivisi in **pagine logiche**).
- **Il task può anche riferirsi a un thread** di esecuzione quando un processo è suddiviso in attività secondarie.
- Il task non usa la CPU fino a quando non termina come nel vecchio modello multiprogrammazione, **ma piuttosto ha una quantità precisa del tempo della CPU chiamato quantum**.
- **Per iper-semplificare il multitasking e la multiprogrammazione si riferiscono ad un concetto simile (condivisione del tempo di CPU) il primo viene utilizzato nei sistemi operativi moderni mentre lil secondo veniva utilizzato in sistemi operativi precedenti**.

Sistema Operativo: Time-Sharing

- Chiaramente in un sistema a processore singolo, l'esecuzione parallela è un'illusione. Solo una singola istruzione di un singolo processo (task) alla volta puo' essere eseguita dalla CPU
- **Il tempo della CPU viene condiviso tra i processi (tasks)**. I sistemi operativi multiprogrammazione e multitasking non sono altro che sistemi di condivisione del tempo.
- Nella multiprogrammazione, anche se la CPU è condivisa tra i programmi, ma non è l'esempio ottimale di time sharing, infatti un programma continua a funzionare finché non si blocca (termina o operazione di I/O)
- **In un sistema multitasking (sistema operativo moderno) la quantità totale del tempo della CPU viene divisa in quanti temporali, ed ogni processo o task usa un numero equo di quanti temporali**

Sistema Operativo: Multiutente

- Un sistema single user ma multitasking permette ad un singolo utente di accedere a computer ma il singolo utente potra’ avviare piu’ applicazioni (task)

- Multiutente: piu’ utenti potranno acceder contemporaneamente al sistema ed condividerne le risorse , ed il sistema si occupera’ di schedulare l’uso delle risorse fra i vari utenti e quindi processi

Sistema Operativo: Multiprocesso

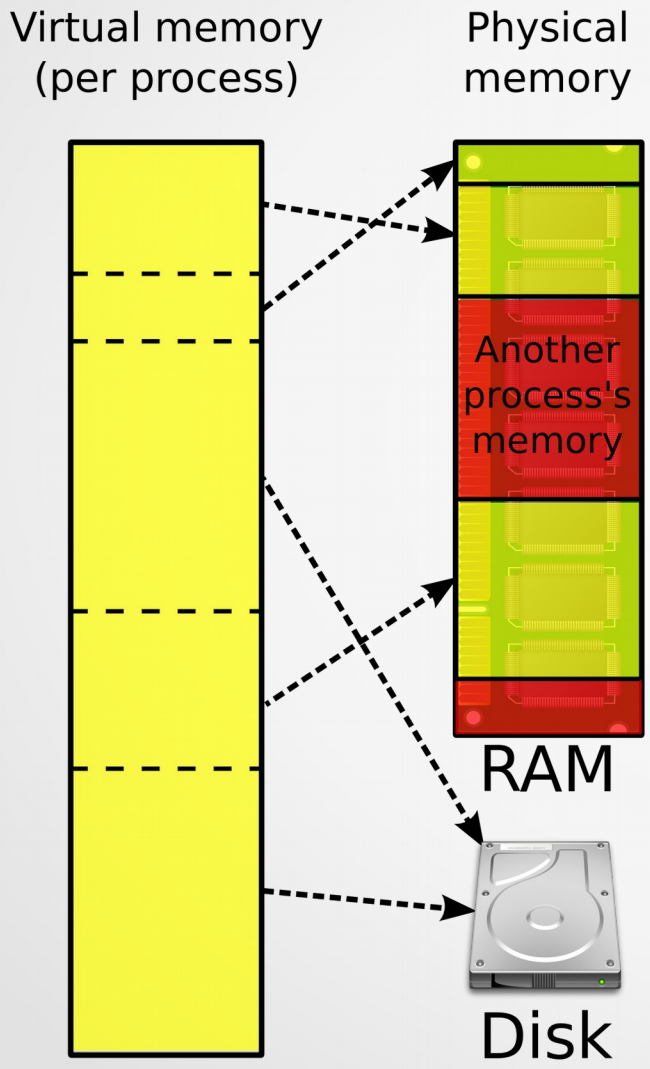
- Il **multiprocesso** a volte si riferisce all'esecuzione di più processi (programmi) allo stesso tempo. Questa dedizioni puo' creare confusione, avendo appena visto il concetto di multiprogrammato e multitasking
- **Il concetto di multiprocesso "generalmente" si riferisce effettivamente alle unità CPU anziché ai processi in esecuzione. Cioe' e' l'hardware sottostante che fornisce più di un processore.**
- Esistono molte varianti sullo schema di base, ad esempio avere più core su un singolo die o più die in un unico package o più package in un unico sistema.
- In sintesi, il multiprocesso si riferisce essenzialmente all'hardware sottostante (più CPU, core)

Sistema Operativo: Multithreading

- Il **multi threading** è un modello di esecuzione che consente a un **singolo processo di avere più segmenti di codice (thread) eseguiti** contemporaneamente nel contesto di quel processo.
- È possibile pensare ai thread come processi secondari che condividono le risorse del processo padre ma che vengono eseguite indipendentemente. **Più thread di un singolo processo possono condividere la CPU in un singolo sistema CPU o (puramente) eseguire in parallelo in un sistema multicore ad esempio**.
- Un sistema multitasking può avere processi multi-thread in cui diversi processi condividono la CPU e allo stesso tempo ognuno ha i propri thread.
- Ad esempio una **GUI in cui vuoi emettere un comando che richiede tempi lunghi di esecuzione (un calcolo matematico complesso). A meno che non si esegua questo comando in un thread di esecuzione separato, non sarà possibile interagire con la GUI dell'applicazione principale**

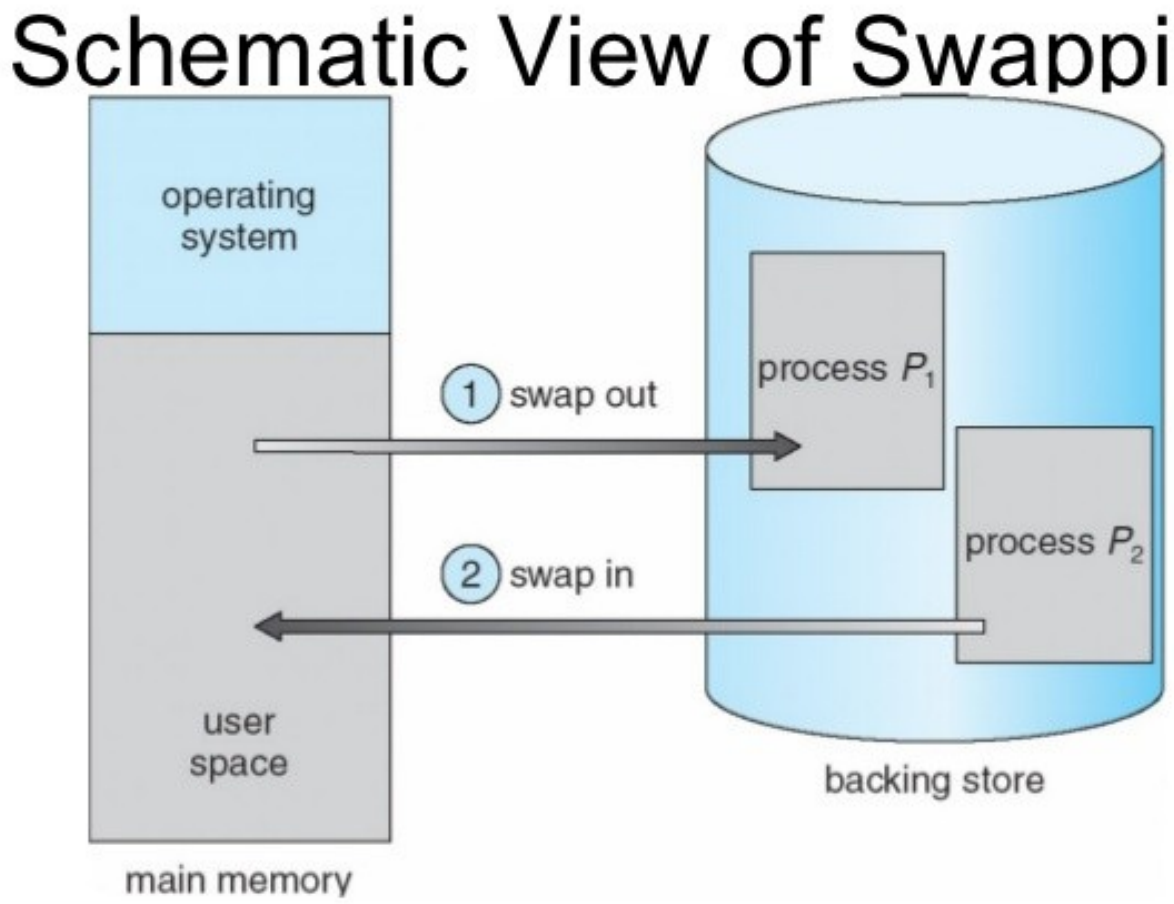
DUE PICCOLI APPROFODNIMENTI

Virtual Memeory



La memoria virtuale (Virtual Memory) è una tecnica di gestione della memoria che fornisce una "astrazione idealizzata delle risorse di archiviazione effettivamente disponibili su una data macchina" che quindi "crea l'illusione per gli utenti (processi, programmi, applicazioni)di un grande spazio di memoria. "

Swapping


Schematic View of Swapping
operating system
user space
main memory
1 swap out
2 swap in
process P_1
process P_2
backing store

IN BREVE COMPONENTI FONDAMENTALI DI UN SISTEMA OPERATIVO

Sistema Operativo: gestione dei processi

- Un processo e' in pratica **un programma in esecuzione , che quindi utilizza risorse**
- l'OS deve essere in grado di:
  - **Creare e terminare i processi** stessi
  - **Gestire la comunicazione fra i processi** stessi
  - **Sospendere e riattivare i processi**

Sistema Operativo: gestione della memoria principale

- La memoria principale in pratica e' un'array di byte indirizzabili singolarmente, che puo' essere condiviso fra CPU e dispositivi di I/O
- l'OS deve essere in grado di:
  - **Tenere traccia ad esempio di quale aree di memoria sono utilizzate e da chi** (da quale processo)
  - **Decidere ad esempio per quali processi allocare una data area di memoria** quando e' libera
  - In ultima analisi **allocare a liberare lo spazio in memoria**

Sistema Operativo: gestione della memoria secondaria

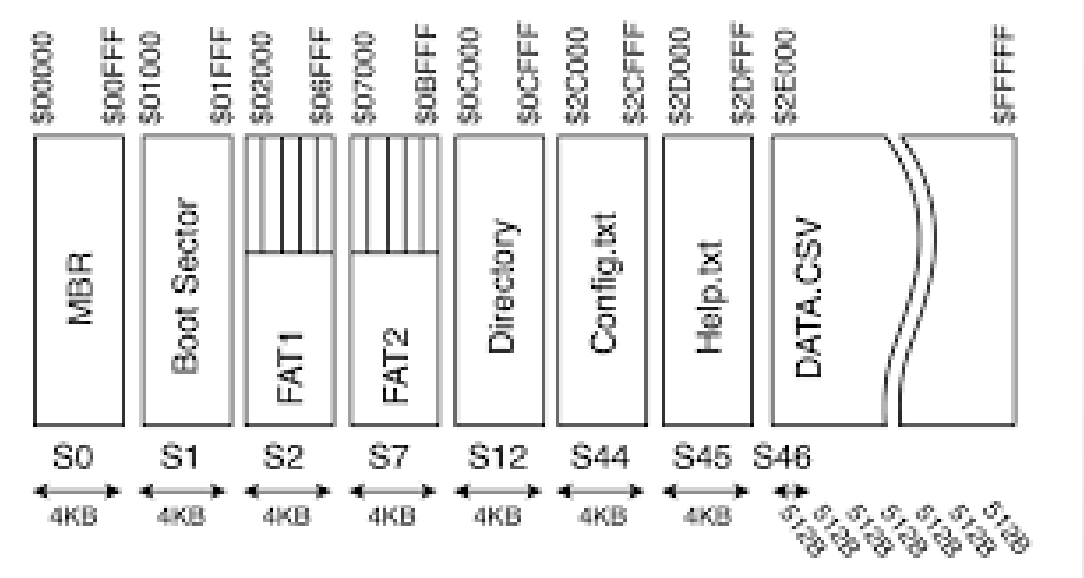
- I calcolatori sappiamo sono dotati di una **memoria secondaria** non volatile e di grandi dimensioni (gradi rispetto alla memoria principale che e' volatile)
- L'OS deve poter gestire le seguenti attivita':
  - **Allocare lo spazio richiesto e liberarlo** quando non piu' utilizzato
  - **Gestire lo scheduling di accesso ai dispositivi** (ad esempio hard-disk o CD/DVD o chiavette USB)

Sistema Operativo: gestione del filesystem

- Un **file** (archivio) e' l'astrazione informatica del concetto di contenitore di informazioni, indipendentemente dal dispositivo sul quale viene memorizzato
- Un **filesystem** e' sostituito da molti **files** (una **directory** contiene riferimenti ad altri files), L'OS deve poter gestire le seguenti attivita':
  - **Creare e cancellare files e directory,** manipolarli
  - **Codificare il filesystem** sulla memoria secondaria
  - Esempi, ext3, ext4, NTFS, FAT32 ...

Sistema Operativo: gestione del filesystem

- Il **filesystem** : deve gestire l'allocazione dello spazio sul disco, mantenere ad esempio un indice di dove sono memorizzati i dati relativi ad un dato file . Mantenere le caratteristiche di un file ad esempio i dati di accessi, i permessi, i nomi etc etc.



Sistema Operativo: gestione dell'I/O

- L'OS deve gestire l'I/O e quindi l'interazione con i vari dispositivi hardware:
  - Un interfaccia comune per la gestione dei vari d**evice driver**
  - Avere diversi **driver (kernel module)** per i diversi dispositivi , quindi componenti specifiche per le varie componenti hardware del sistema
  - Un sistema di **buffering e caching** delle informazioni da e verso i vari dispositivi

Sistema Operativo: protezione

- L’OS deve implementare un **meccanismo di protezione software**. Quindi gestire e controllare l’accesso dei vari programmi (processi) alle risorse condivise del sistema (ad esempio la memoria)
  - Distinguere fra uso autorizzato o meno
  - Fornire meccanismi di base per attuare la protezione

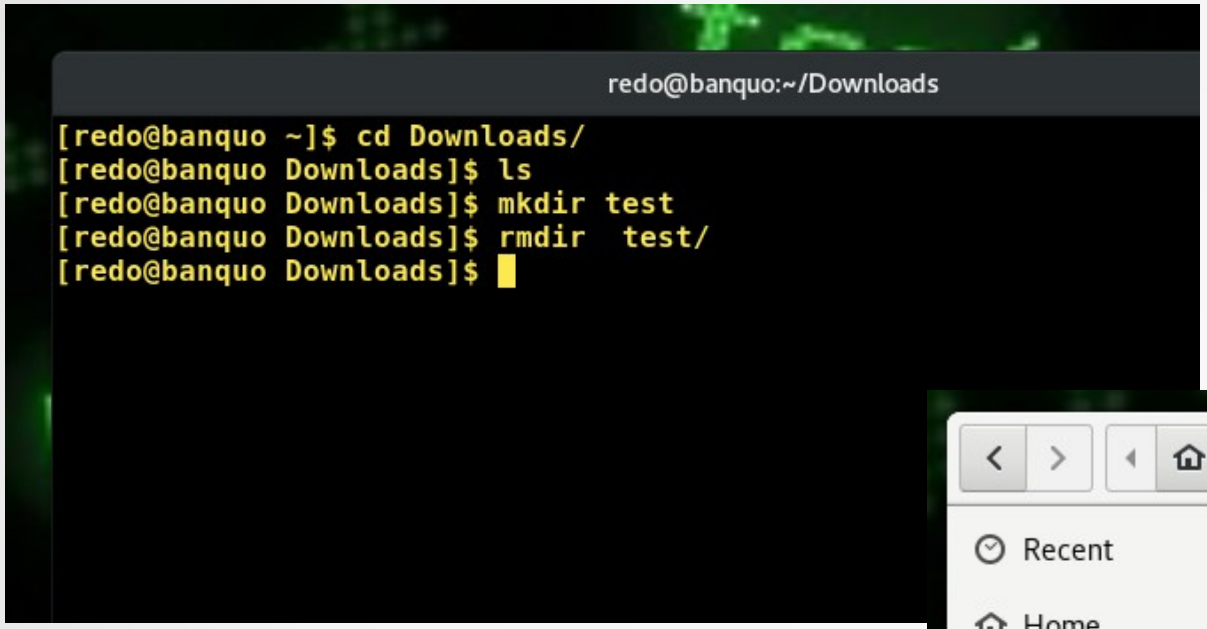
Sistema Operativo: networking

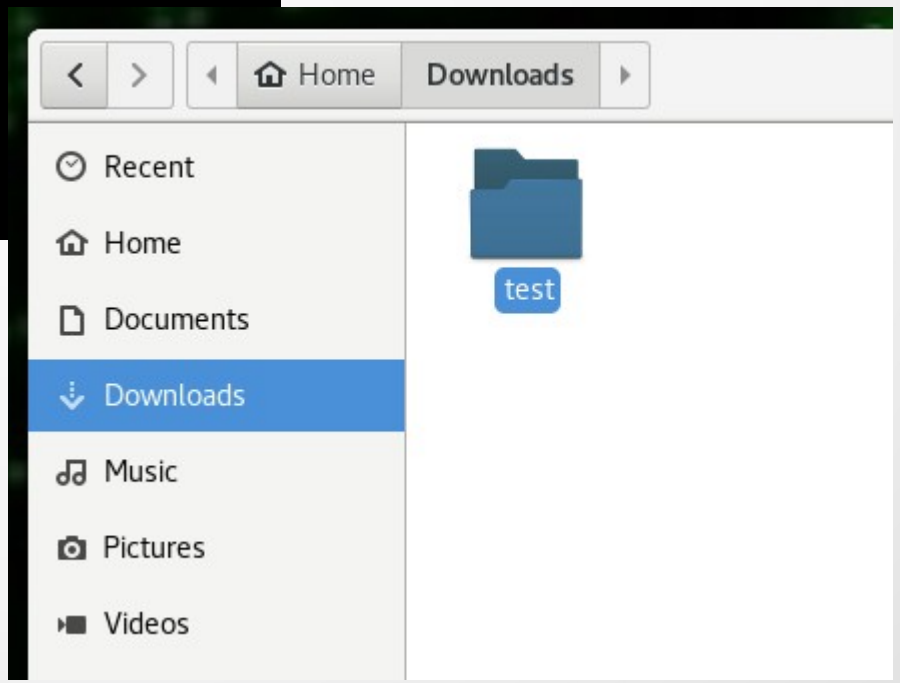
- Oramai una componente essenziale di un OS e’ appunto il networking, q uidni la capacita’ di far comunicare due o piyu’ elaboratori (processi) e di condividere risorse.
- Un’OS fornisce i protocolli di comunicazione di base come **TCP/IP , UDP**
- **Oltre che servizi di comunicazione ad alto livello** come ad esempio filesystem condivisi (SMB, NFS) e molti altri.

Sistema Operativo: interprete dei comandi

- I sistemi operativi forniscono un’interfaccia utente:
  - Avviare o terminare un programma
  - Interagire con le componenti base del sistema operativo , come il filesystem
- Possiamo avere essenzialmente due tipi:
  - Grafica, e quindi icone e finestre
  - Testuale, linea di comando

Sistema Operativo: interprete dei comandi





CAMBIO DI PROSPETTIVA

Sistema Operativo: prospettiva del programmatore

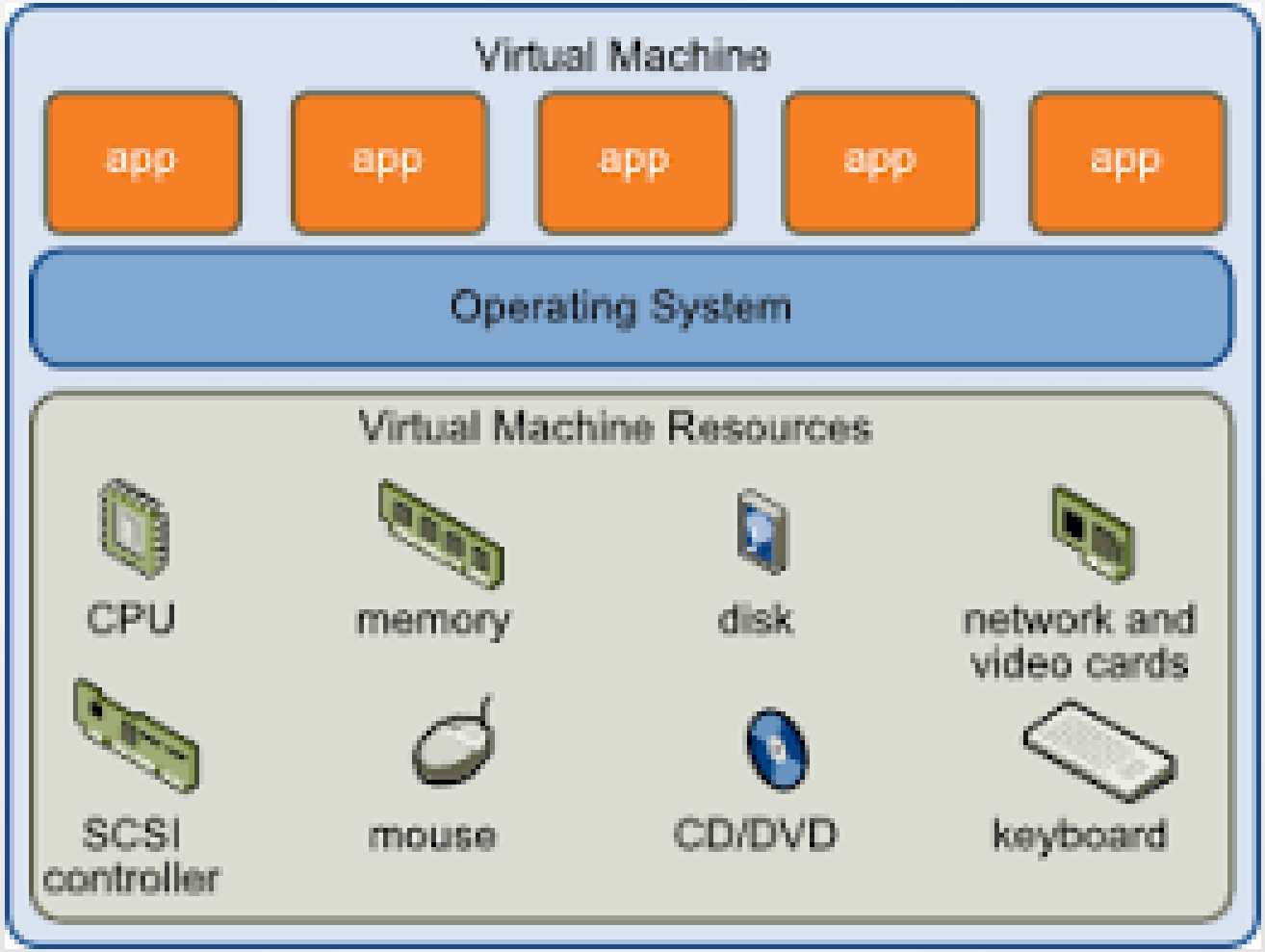
- La System Call (chiamate di sistema) ad esempio:

UNIX	WIN32	UNIX	WIN32
fork	CreateProcess	mkdir	CreateDirectory
waitpid	WaitForSingleObject	rmdir	RemoveDirectory
execve	-	link	-
exit	ExitProcess	unlink	DeleteFile
open	CreateFile	mount	-
close	CloseHandle	umount	-
read	ReadFile	chdir	SetCurrentDirectory
write	WriteFile	chmod	-
lseek	SetFilePointer	kill	-
stat	GetFileAttributesEx	time	GetLocalTime

VMs, DOCKER e Cloud Computing

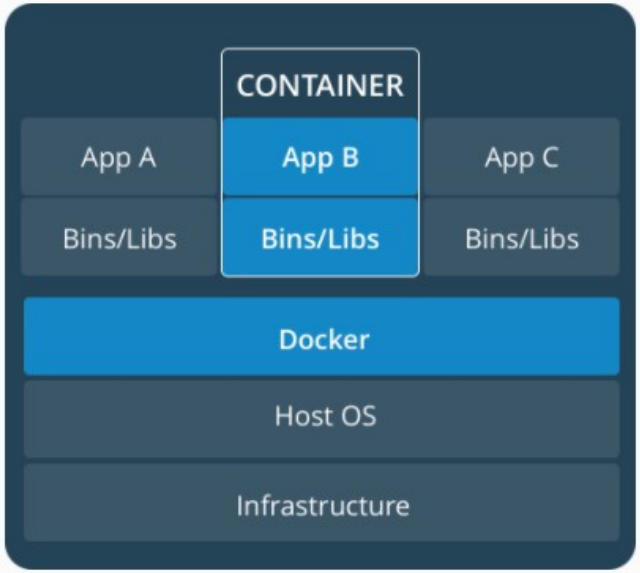
Cloud computing

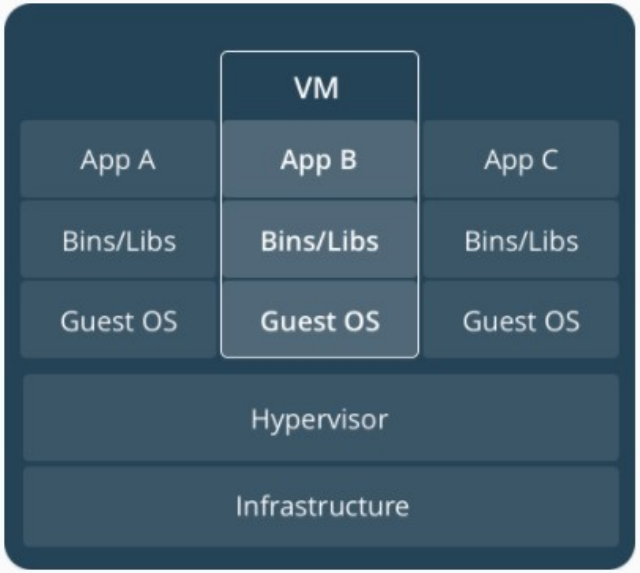
- Virtual Machine



Cloud computing

- Docker vs VM





CONTAINERS

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.

VIRTUAL MACHINES

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

Cloud computing

- Infrastruttura virtuale