

# Struttura dei sistemi operativi

# Struttura dei sistemi operativi

---

- Componenti
- Servizi di un sistema operativo
- System Call
- Programmi di sistema
- Struttura del sistema operativo
- Macchine virtuali
- Progettazione e Realizzazione



# Componenti del sistema

---



**Interprete dei comandi**

**Sistema di protezione**

**Accesso alla Rete**

**Gestione della memoria secondaria**

**Gestione dell'I/O**

**Gestione dei file**

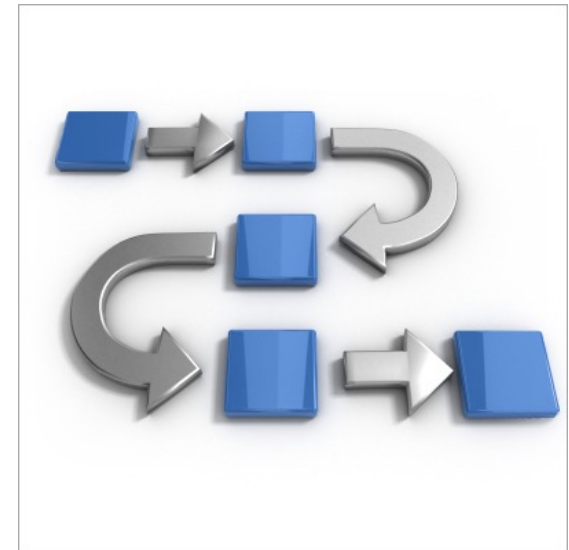
**Gestione della memoria principale**

**Gestione dei processi**

# Gestione dei Processi

---

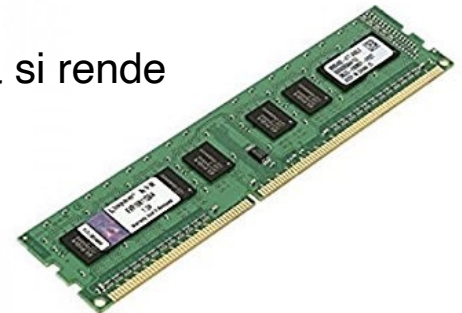
- Un **processo** di elaborazione è un programma in esecuzione.
- Un processo ha bisogno di un certo numero di risorse per svolgere il suo compito: CPU, memoria, file e dispositivi di I/O.
- Il sistema operativo è responsabile delle seguenti attività per la gestione dei processi:
  - creazione e terminazione dei processi.
  - sospensione e ripristino dei processi.
  - Meccanismi per:
    - ▶ Sincronizzazione di processi,
    - ▶ Comunicazione tra processi,
    - ▶ Gestione dello stallo.



# Gestione della memoria centrale

---

- La memoria centrale è costituita da una **sequenza di word** ognuna con un proprio indirizzo.
- E' uno spazio per conservare dati facilmente accessibile per la CPU e i dispositivi di I/O.
- Il sistema operativo che si occupa della gestione della memoria principale è responsabile di:
  - Tenere traccia di quali parti della memoria sono correntemente usate e da chi.
  - Decidere quali processi caricare quando la memoria si rende disponibile.
  - Allocare and deallocare lo spazio di memoria.



# Gestione dei file (File System)

---

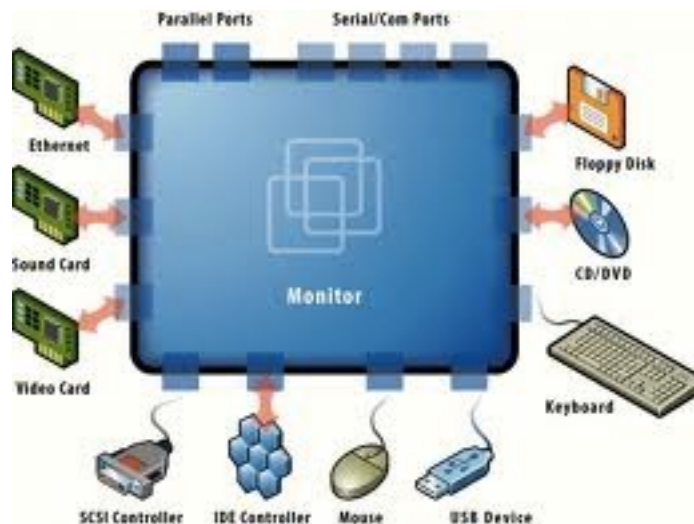
- Un file è un insieme di informazioni definite dal suo creatore. Generalmente i file contengono programmi (in formato sorgente o oggetto) e dati.
- Le directory sono file particolari.
- La componente della gestione dei file del sistema operativo è responsabile delle seguenti attività:
  - Creazione e cancellazione di file.
  - Creazione e cancellazione di directory.
  - Supporto di primitive per manipolare file e directory.
  - Mappare file in memoria secondaria.
  - Backup dei file su memorie non volatili.



# Gestione dell'I/O

---

- Il sistema di gestione dell'I/O consiste di:
  - Un sistema di buffer-caching
  - Una interfaccia generale per i driver dei dispositivi.
  - Driver per dispositivi hardware specifici.



# Gestione della memoria secondaria

---

- Poiché la memoria centrale è volatile e non riesce a contenere tutti i programmi e i dati in maniera permanente, i computer hanno bisogno di una memoria secondaria che contenga tutto quello che serve al computer.
- La maggior parte dei computer usano i dischi come dispositivi di memoria secondaria per contenere dati e programmi.
- Per la gestione della memoria secondaria il sistema operativo si occupa di:
  - allocazione della memoria,
  - gestione della memoria libera,
  - scheduling del disco.



Ultra ATA



SATA



SCSI



Solid State



External



# Accesso alla rete

---

- In un sistema distribuito (es: rete locale) ogni calcolatore ha la propria memoria locale.
- I calcolatori sono connessi attraverso una rete di comunicazione.
- Le comunicazioni sono basati su *protocolli*.
- L'accesso alla rete permette:
  - Condivisione di risorse,
  - Prestazioni più alte,
  - Maggiore disponibilità di dati,
  - Maggiore affidabilità.



# Sistema di protezione

---

- Questo componente dei sistemi operativi fornisce meccanismi per controllare l'accesso di processi e utenti alle risorse del sistema e/o degli utenti.
- I meccanismi di protezione devono:
  - distinguere tra usi autorizzati e usi non autorizzati.
  - specificare i controlli che devono essere imposti,
  - fornire i mezzi per l'attuazione della protezione



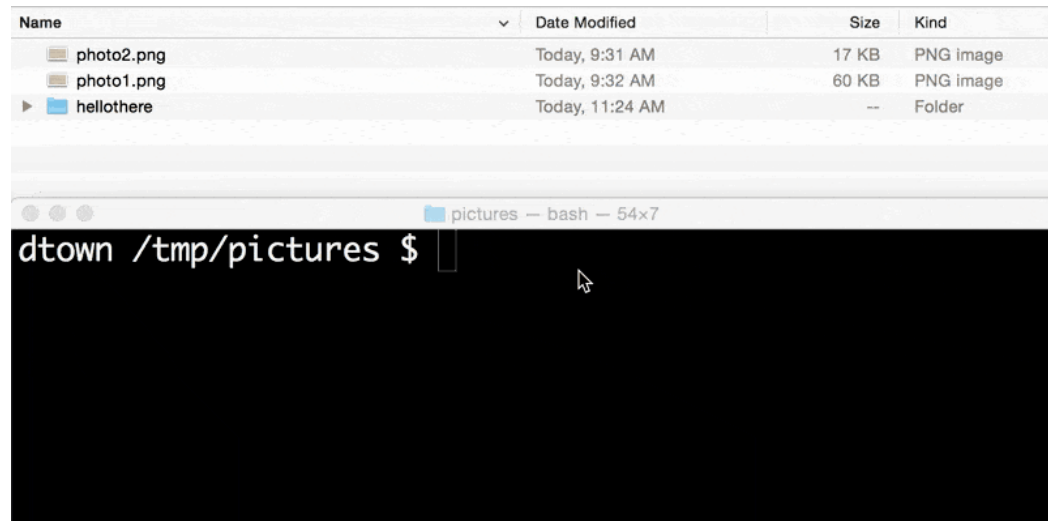
# Interprete dei comandi

---

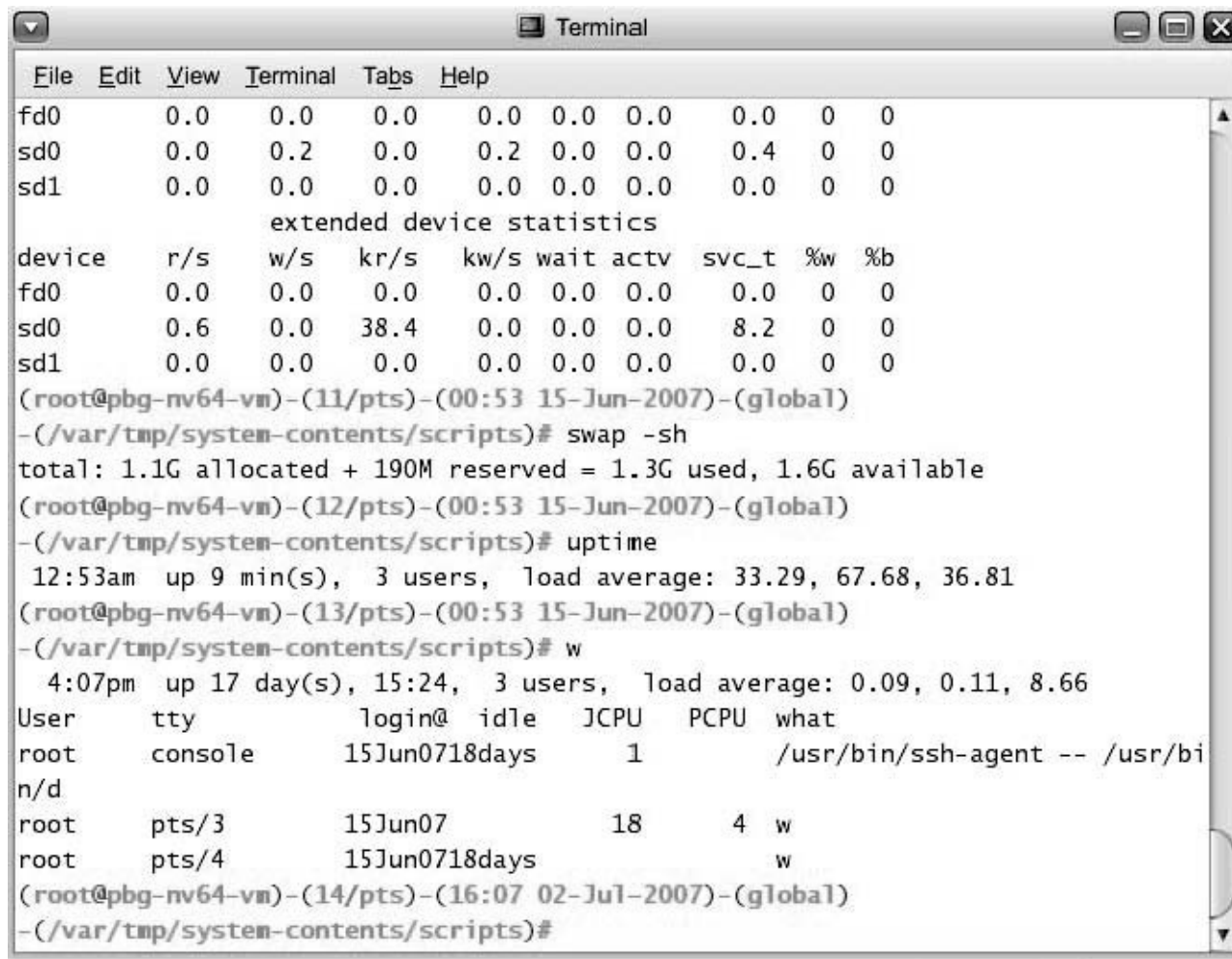
- Il sistema operativo riceve continuamente comandi tramite istruzioni di controllo che riguardano:
  - creazione e gestione di processi
  - gestione dell'I/O
  - gestione della memoria secondaria
  - gestione della memoria centrale
  - accesso al file system
  - protezione
  - networking
  
- E' necessario per il sistema interpretare i comandi ricevuti.

# Interprete dei comandi

- Il programma che legge e interpreta i comandi viene chiamato:
  - Interprete dei comandi
    -
  - shell (in UNIX)
- La sua funzione è di ricevere il comando, controllarne la correttezza, gestire la sua esecuzione da parte dei diversi livelli del sistema e mostrare i risultati.

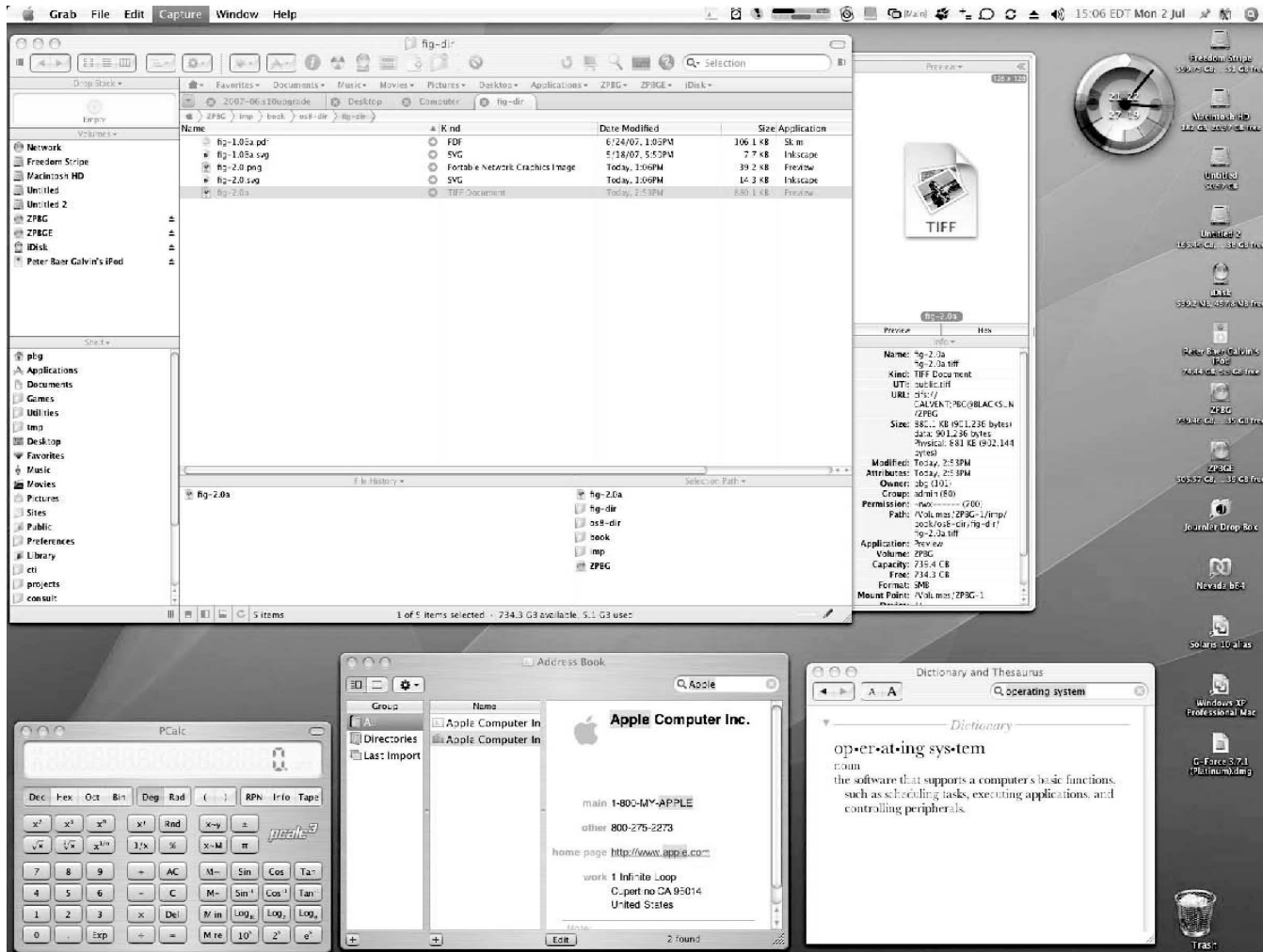


# Shell Bourne (Solaris 10)



```
File Edit View Terminal Tabs Help
fd0      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
sd0      0.0    0.2    0.0    0.2  0.0  0.0    0.4  0  0
sd1      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
          extended device statistics
device   r/s    w/s    kr/s    kw/s wait actv  svc_t  %w  %b
fd0      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
sd0      0.6    0.0   38.4    0.0  0.0  0.0    8.2  0  0
sd1      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
(root@pbg-nv64-vn)-(11/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vn)-(12/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# uptime
12:53am up 9 min(s), 3 users, load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vn)-(13/pts)-(00:53 15-Jun-2007)-(global)
-(/var/tmp/system-contents/scripts)# w
4:07pm up 17 day(s), 15:24, 3 users, load average: 0.09, 0.11, 8.66
User      tty          login@ idle   JCPU   PCPU   what
root      console      15Jun07 18days 1      /usr/bin/ssh-agent -- /usr/bi
n/d
root      pts/3        15Jun07 18      4      w
root      pts/4        15Jun07 18days      w
(root@pbg-nv64-vn)-(14/pts)-(16:07 02-Jul-2007)-(global)
-(/var/tmp/system-contents/scripts)#
```

# Interfaccia grafica di Mac OS X

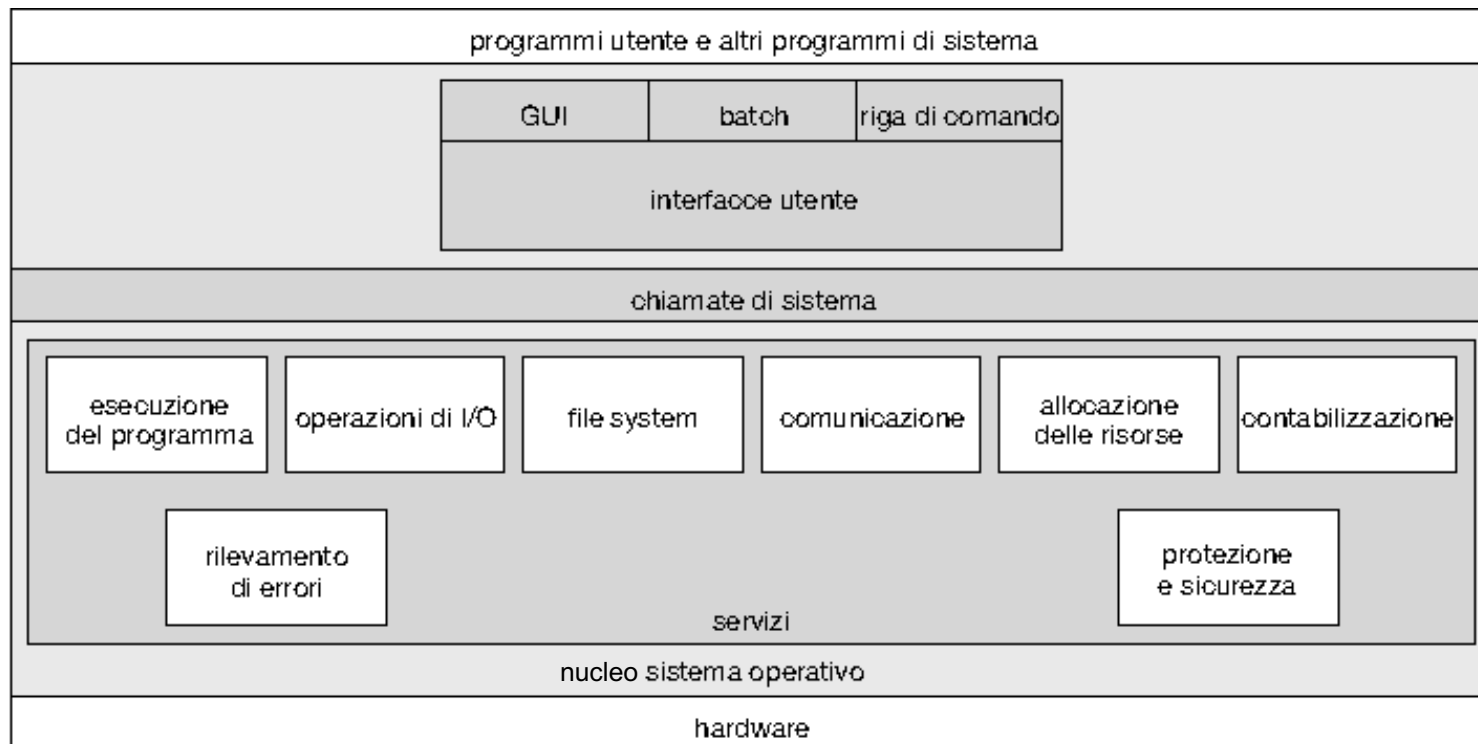


# Servizi di un sistema operativo

---

- **Esecuzione di programmi** – capacità di caricare in memoria un programma ed eseguirlo.
- **Operazioni di I/O** – poiché i programmi non possono eseguire direttamente le operazioni di I/O, il sistema operativo deve fornire meccanismi per le operazioni di I/O.
- **Gestione del File system** – fornire ai programmi modalità per leggere, scrivere, creare e cancellare file.
- **Comunicazioni** – scambio di informazioni tra processi in esecuzione sullo stesso computer o su computer connessi in rete.
- **Scoperta di errori** – rilevamento di errori nella CPU, nella memoria, nell'I/O o nei programmi utente.

# Servizi di un sistema operativo

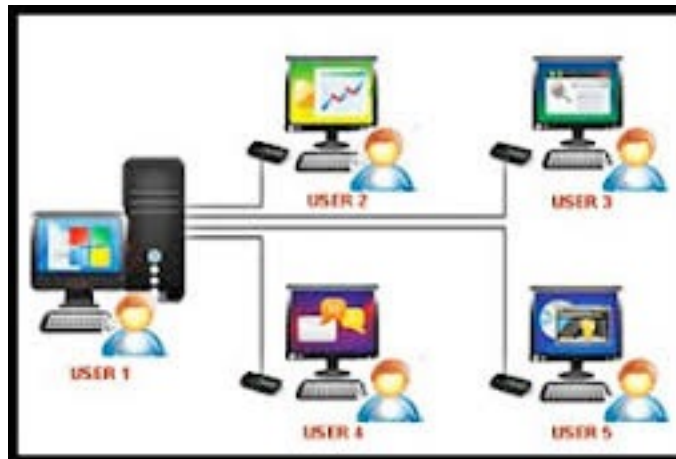




# Funzioni aggiuntive

---

- Esistono funzioni del sistema operativo che servono non tanto per l'esecuzione dei programmi utente, ma per assicurare l'esecuzione efficiente delle operazioni del sistema:
  - **Allocazione di risorse** – allocazione di risorse per i programmi eseguiti contemporaneamente.
  - **Accounting** – memorizzazione dell'uso delle risorse da parte dei vari utenti per il calcolo di costi e per la generazione di statistiche.
  - **Protezione** – controllo degli accessi a tutte le risorse del sistema.

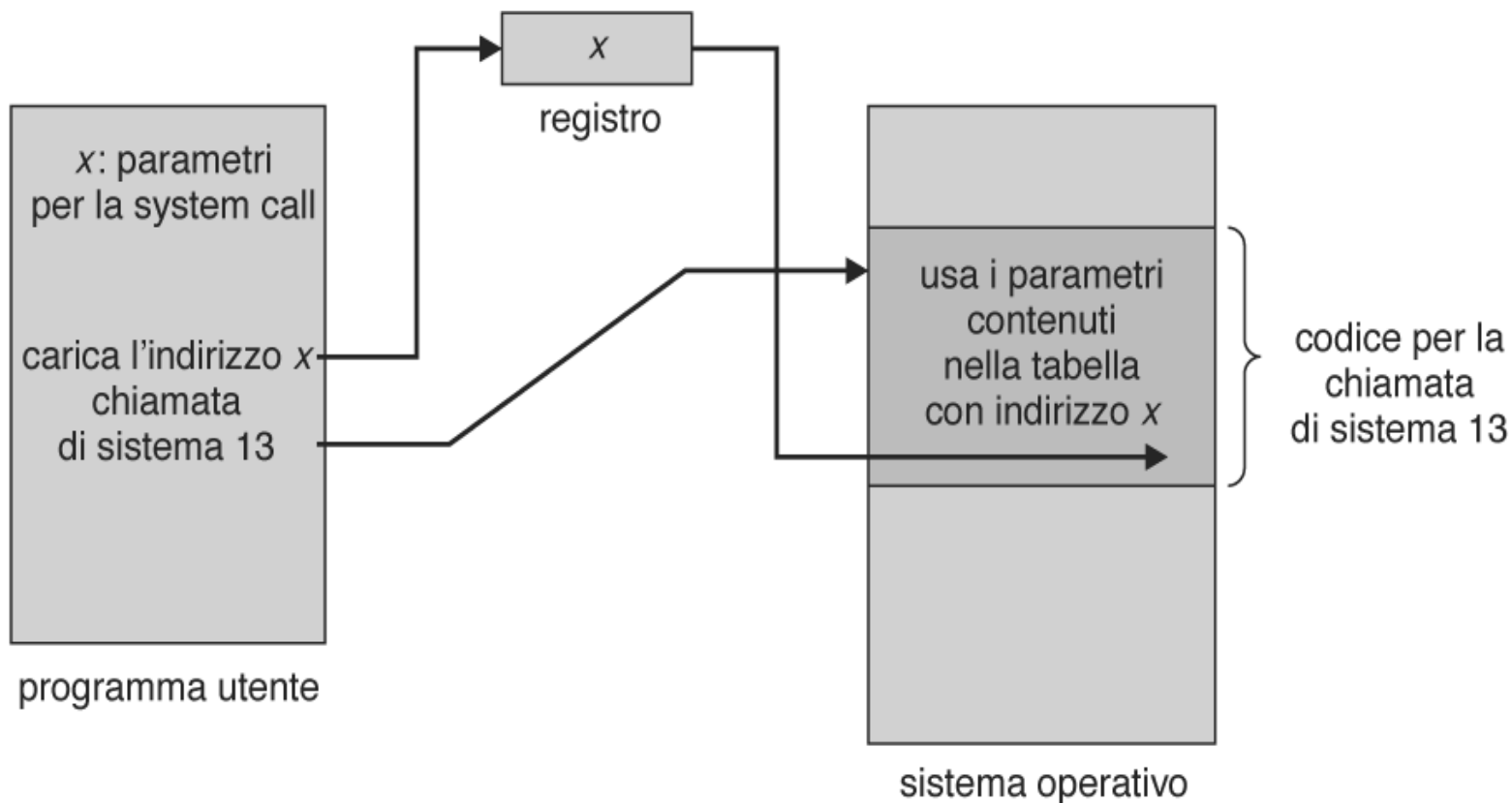


# System call

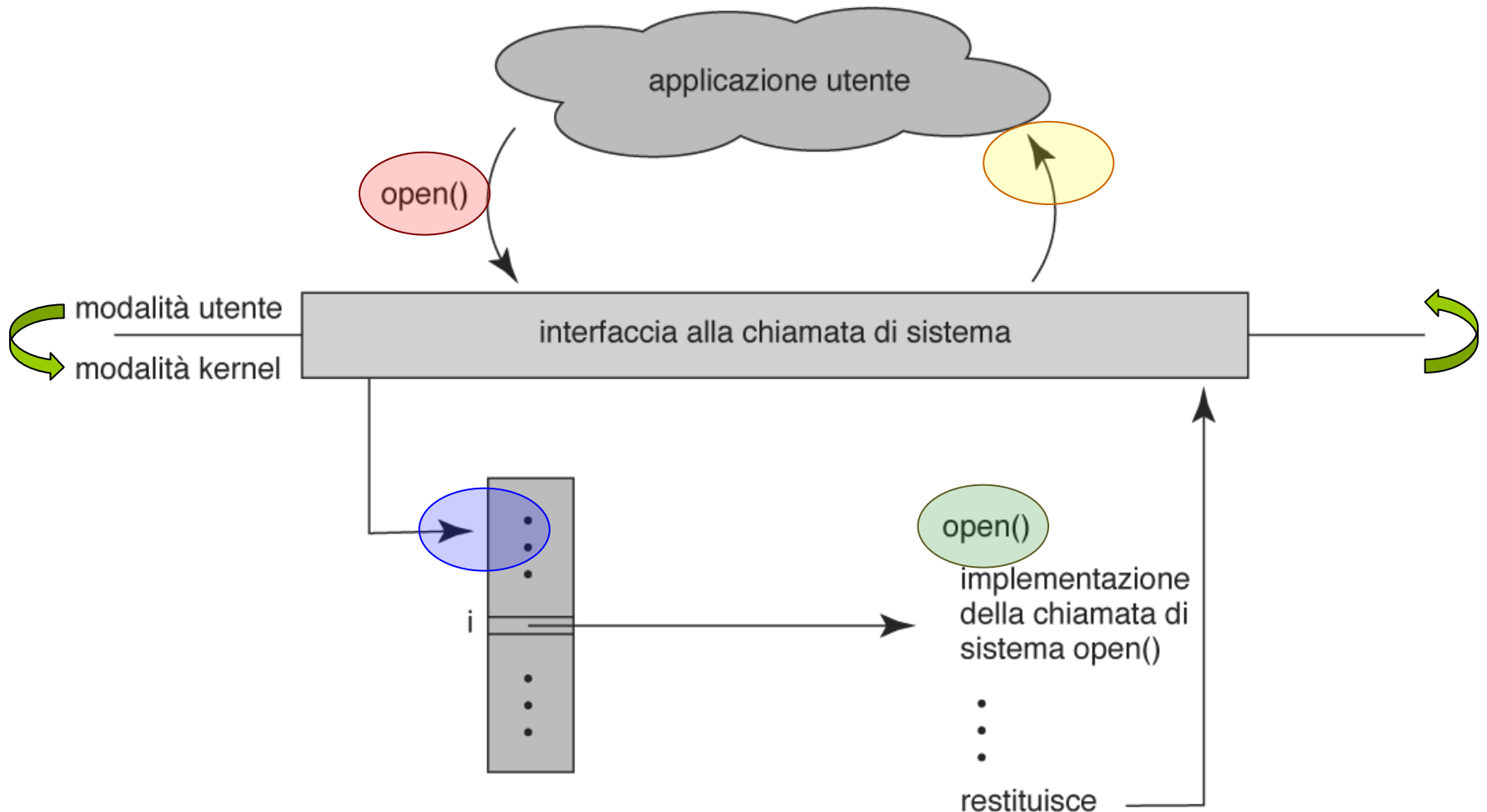
---

- **Le System call forniscono l'interfaccia di programmazione tra un programma e il sistema operativo.** Similitudine con le API (Application Programming Interface), ma di livello più basso.
  - Generalmente a basso livello, in alcuni casi con sintassi simile ai linguaggi di alto livello.
  - Alcuni sistemi hanno system call con sintassi simile a quella di linguaggi come C, e C++.
  
- Generalmente vengono usati tre metodi per il passaggio dei parametri tra un programma in esecuzione e il sistema operativo:
  - 1 Passaggio dei parametri in *registri*.
  - 2 Memorizzazione dei parametri in una *tabella in memoria* centrale e passaggio dell'indirizzo della tabella, come un parametro, in un registro.
  - 3 *Push* (inserimento) dei parametri in uno *stack* e *pop* (prelievo) dallo stesso dal sistema operativo.

# Passaggio di parametri mediante tabella

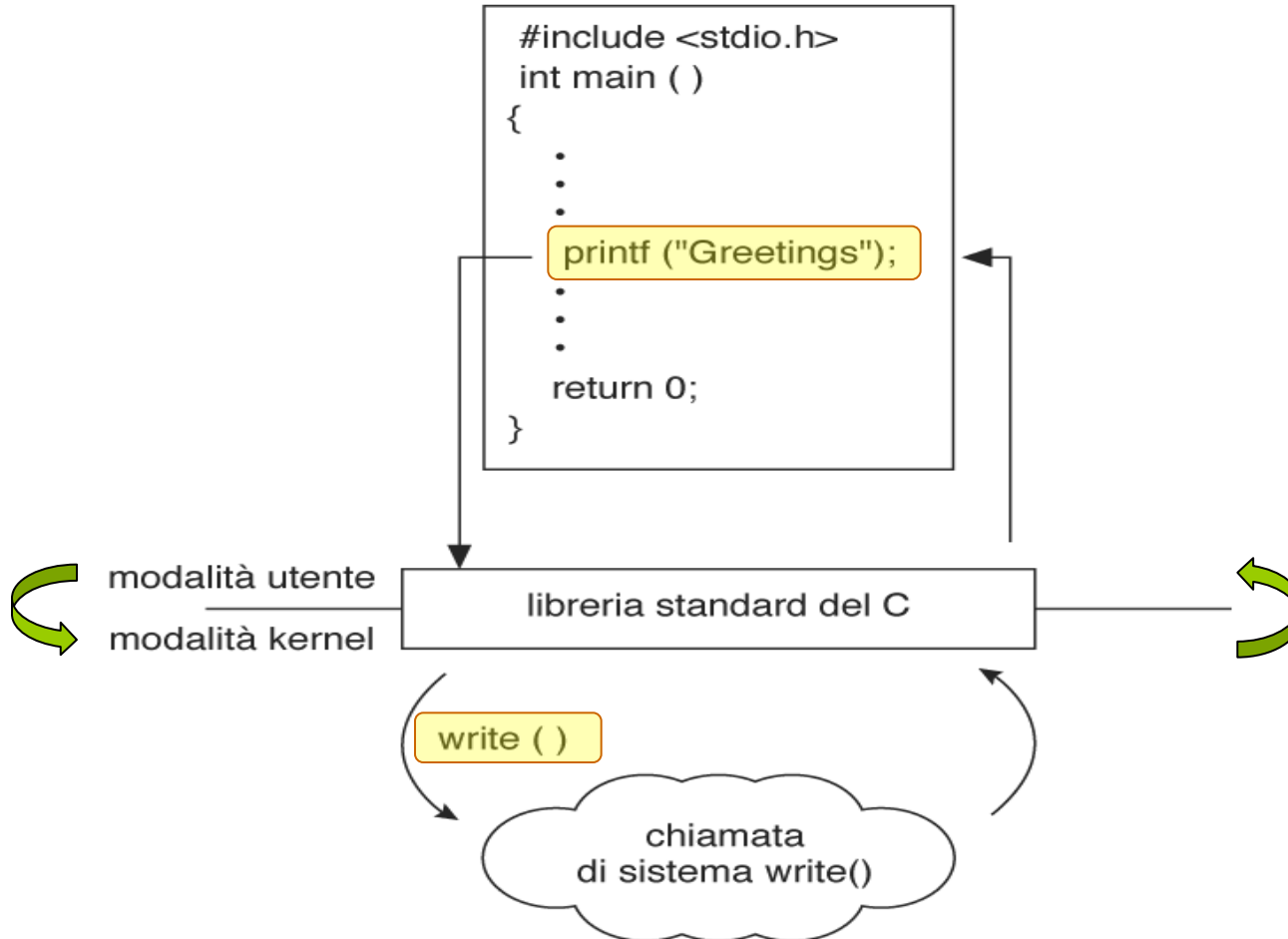


# Chiamata di sistema open() invocata da un'applicazione



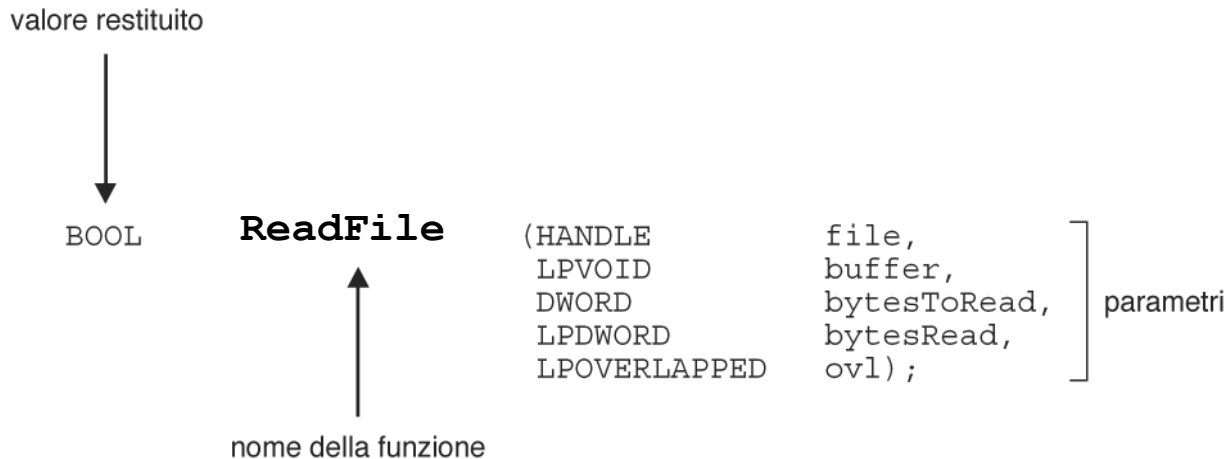
# Esempio di libreria standard del linguaggio C

- Programma C che invoca la chiamata di libreria ***printf()***, la quale a sua volta chiama la system ***call write()***



# Esempio di API di Windows

- Consideriamo la funzione ***ReadFile()*** nelle API di Win32: una funzione per leggere da un file

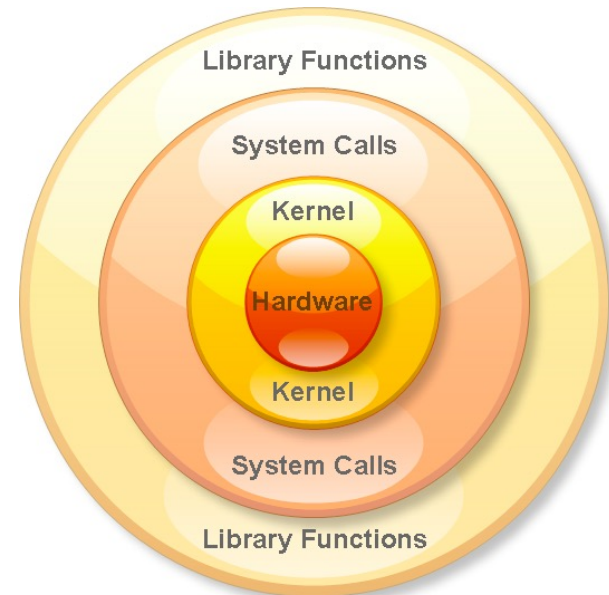


- Alcuni parametri passati a ***ReadFile()***
  - HANDLE file— il file da leggere
  - LPVOID buffer— il buffer in cui verranno scritti i dati letti
  - DWORD bytesToRead—il numero di byte da leggere
  - LPDWORD bytesRead— indirizzo della variabile con il numero di byte letti
  - ...

# Tipi di system call

- Le system call offerte da un sistema operativo riguardano le diverse funzionalità del sistema, come:

- Controllo dei processi,
- Gestione dei file,
- Gestione dei dispositivi di I/O,
- Informazioni sul sistema,
- Comunicazioni.



- Le API generalmente sono librerie di più alto livello rispetto alle system call. Esempi: Le API di Java, le API di POSIX, le API di SQL Server, le API di Google Maps.

# Programmi di sistema

---

- I programmi di sistema forniscono un ambiente conveniente per lo sviluppo e l'esecuzione di applicazioni. Essi si occupano di:
  - Gestione dei file
  - Informazioni di stato
  - Supporto ai linguaggi di programmazione
  - Caricamento ed esecuzione dei programmi
  - Comunicazioni
- La maggior parte degli utenti conosce il sistema operativo tramite i programmi di sistema e non tramite le system call.



# Obiettivi di progetto

---

## ■ Obiettivi utente

- il sistema operativo dovrebbe essere conveniente da usare, facile da apprendere, affidabile, sicuro, e veloce.

## ■ Obiettivi del sistema

- il sistema operativo dovrebbe essere facile da progettare e mantenere, flessibile, affidabile, libero da errori ed efficiente.

# Meccanismi e politiche

---

- I ***meccanismi*** determinano come fare qualcosa, le ***politiche*** determinano cosa fare.
- La separazione tra politiche e meccanismi è un principio molto importante perché permette la massima flessibilità se le politiche verranno cambiate.

# Implementazione del sistema

---

- I sistemi operativi sono stati sviluppati tradizionalmente in assembler per ragioni di efficienza e di accesso alle risorse hardware.
- Oggi vengono implementati tramite linguaggi di alto livello. Il codice scritto in questi linguaggi è:
  - più veloce da sviluppare.
  - più compatto.
  - è facile da comprendere e da correggere.
- Un sistema operativo è facile da *portare* su altro hardware se scritto in un linguaggio di alto livello.

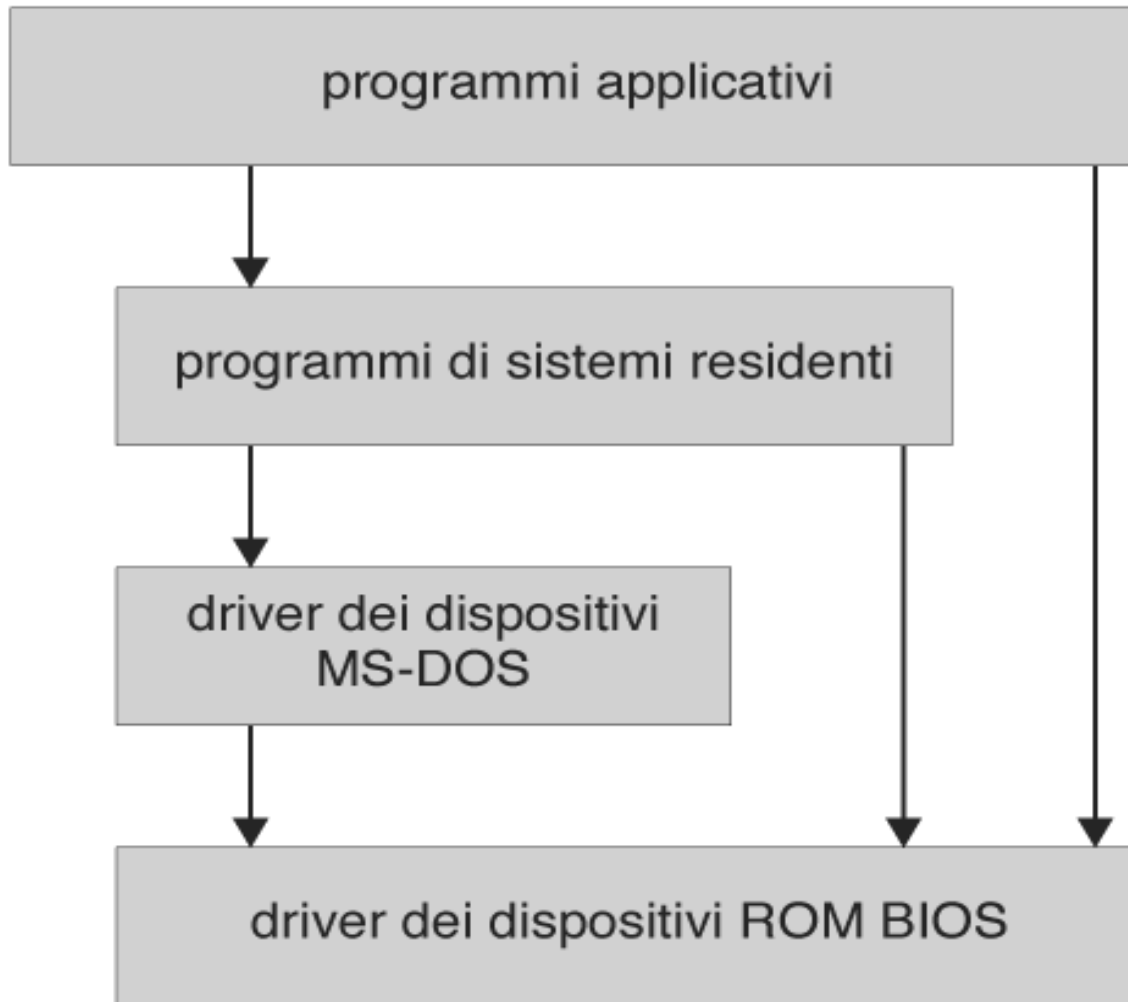
# Debugging dei sistemi operativi

---

- *Debugging* è l'attività di trovare e risolvere gli errori, o *bug*
- I sistemi operativi generano file di log che contengono informazioni sugli errori
- Il fallimento di un'applicazione può generare file di *core dump* che catturano la memoria del processo
- I fallimenti del sistema operativo possono generare file di *crash dump* contenenti la memoria del kernel

# Struttura di MS-DOS

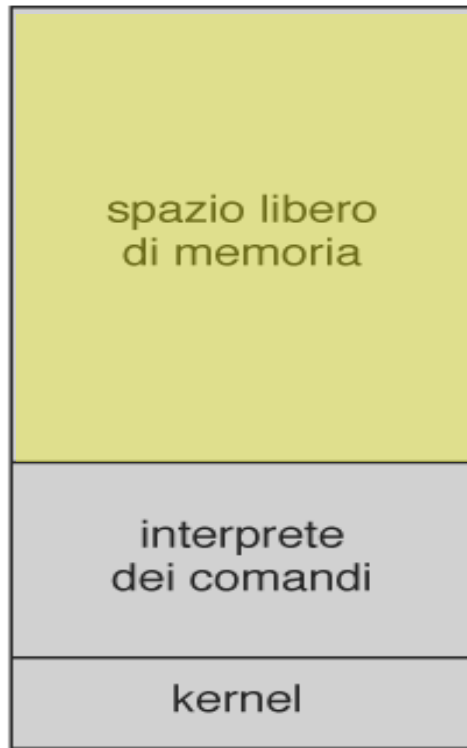
---



I tre programmi principali di MS-DOS sono:

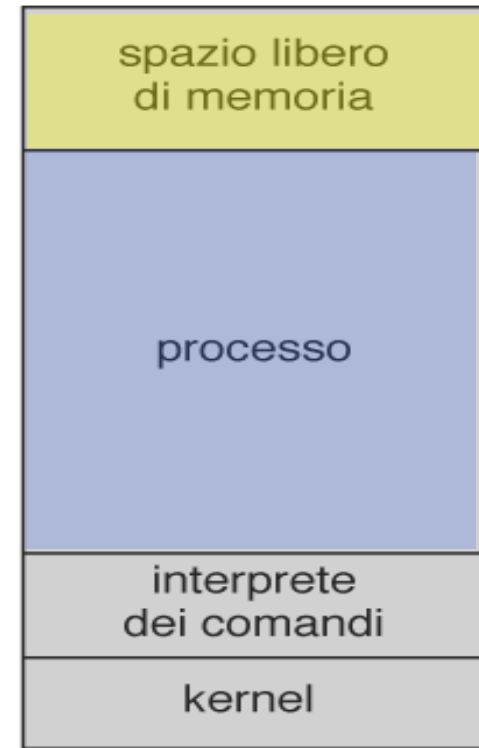
- IO.SYS
- MSDOS.SYS
- COMMAND.COM

# Esecuzione in MS-DOS



(a)

(a) All' avviamento del sistema



(b)

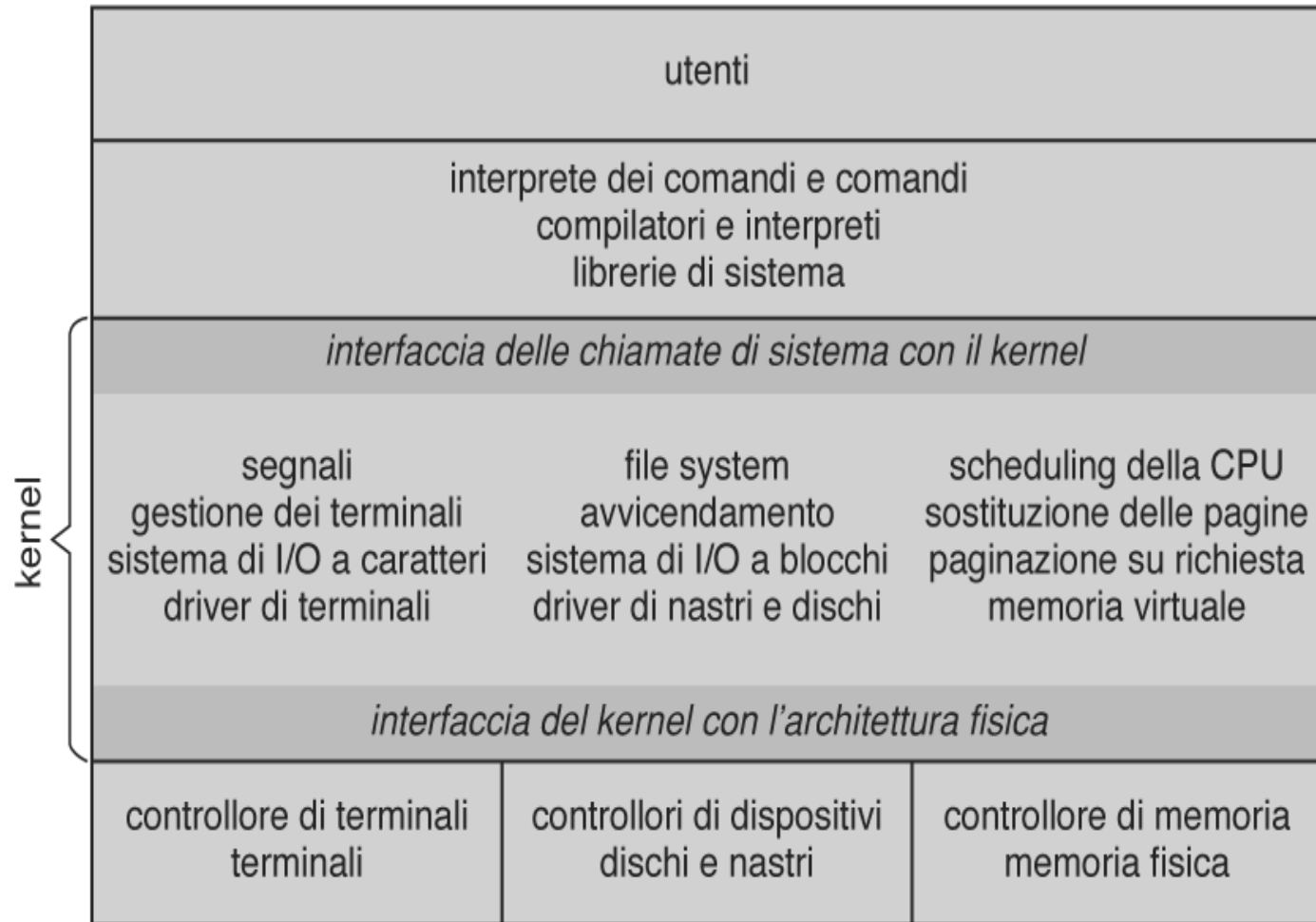
(b) Durante l' esecuzione di un programma

# Struttura del sistema UNIX

---

- UNIX – le versioni iniziali avevano una strutturazione limitata.
- UNIX consiste di due componenti principali.
  - I programmi di sistema
  - Il nucleo (kernel)
    - ▶ Tra il livello delle system call e l'hardware,
    - ▶ Fornisce il file system, lo scheduling della CPU, gestione della memoria, ed altre funzioni di base.

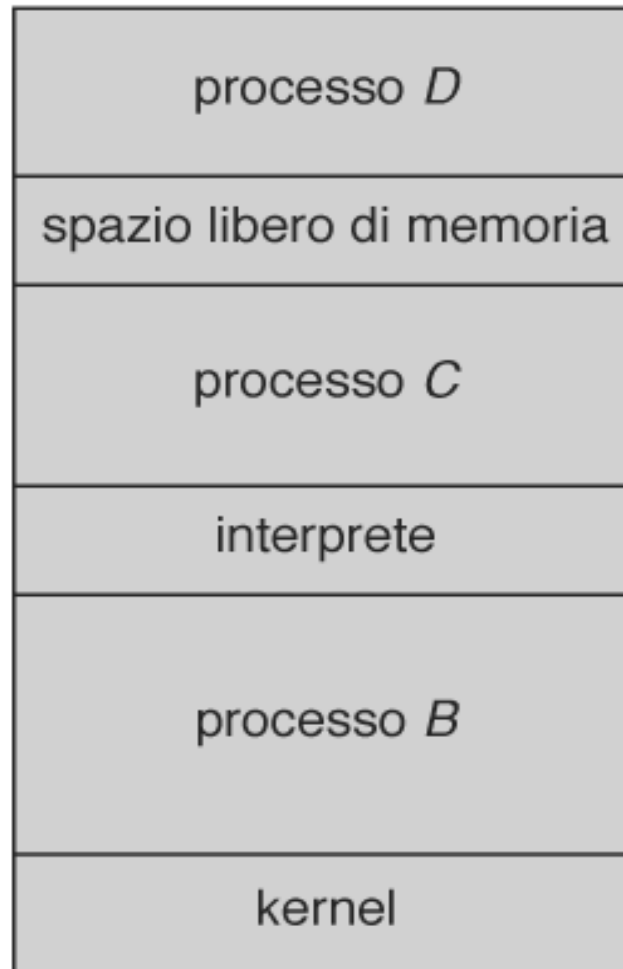
# Struttura di UNIX





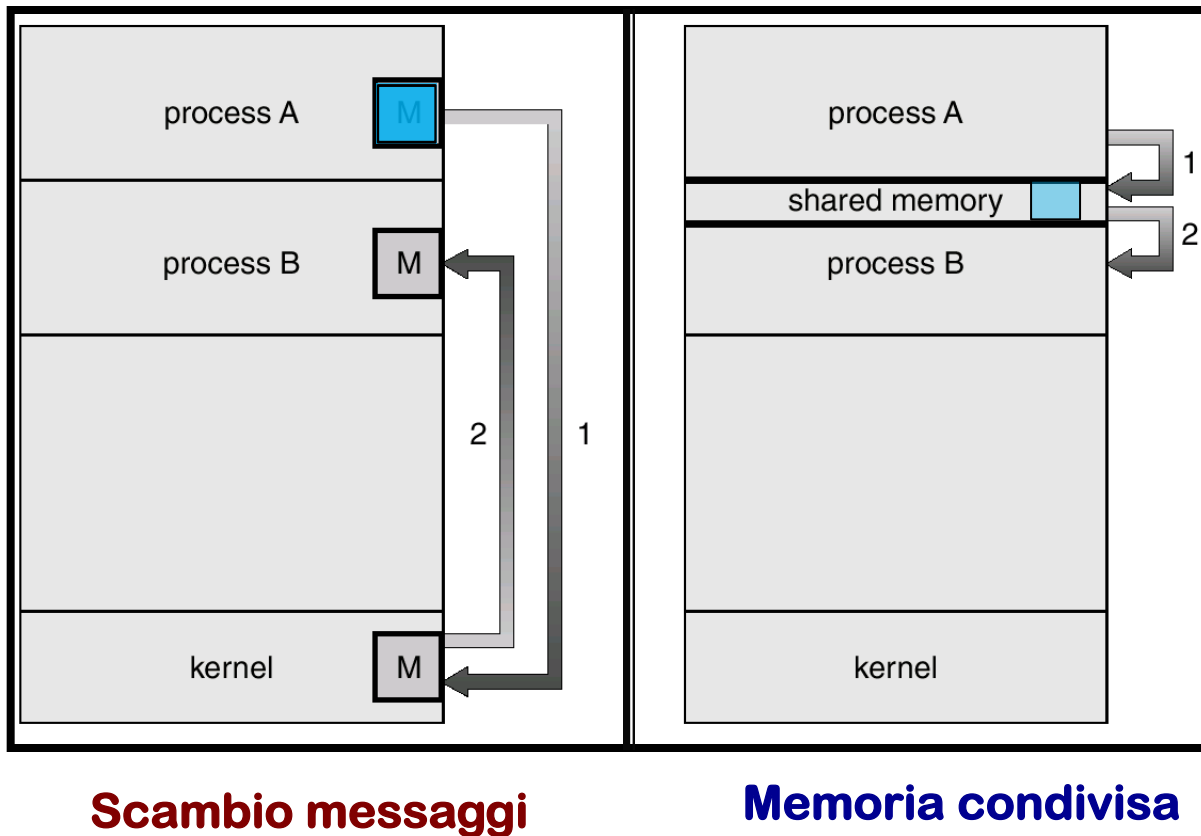
# UNIX: Esecuzione di più programmi

---



# Modelli di comunicazione

- Le comunicazioni tra i processi possono avvenire usando lo scambio di messaggi o la memoria condivisa.



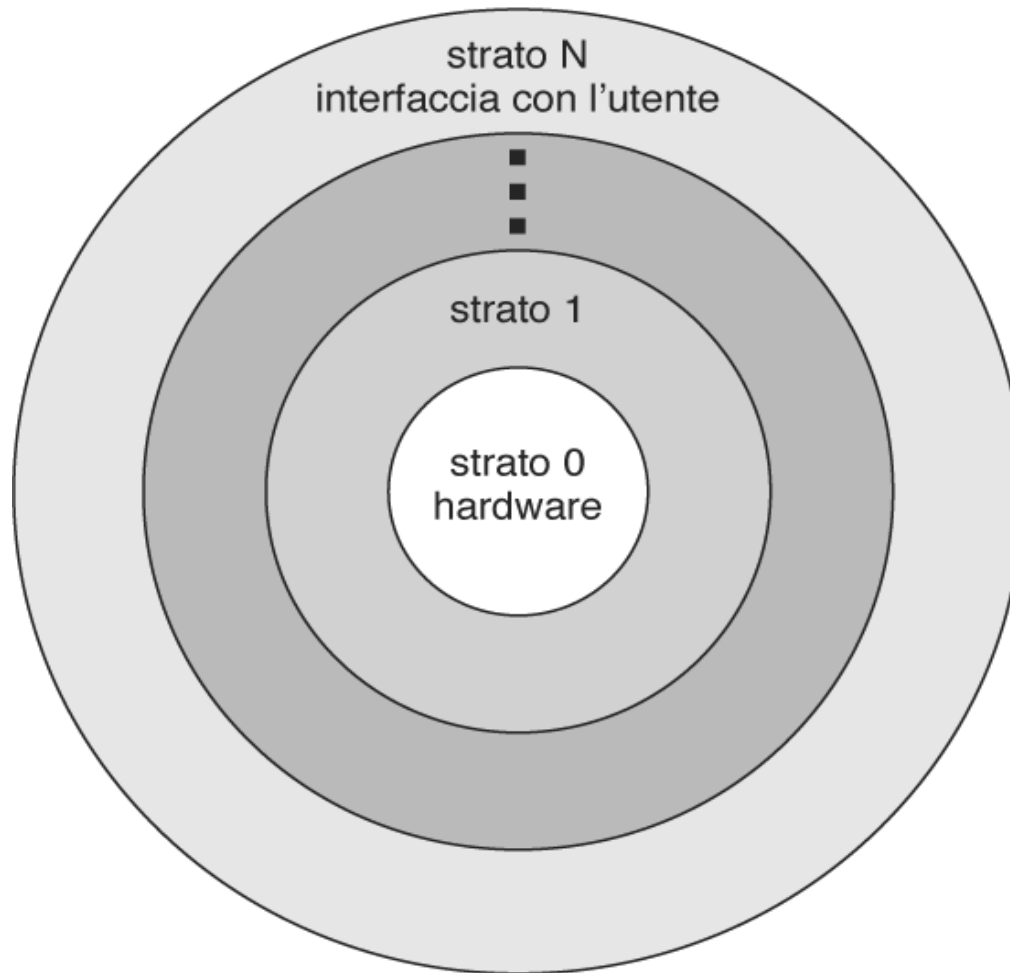
# Approccio a livelli (o strati)

---

- Il sistema operativo è composto da un certo numero di livelli (strati) ognuno costruito a partire dai livelli sottostanti.
- Il livello più basso (*layer 0*), è l'hardware; il più alto (*layer N*) è l'interfaccia utente (interprete comandi).
- In ogni livello sono usate le funzionalità dei livelli sottostanti (modularità).

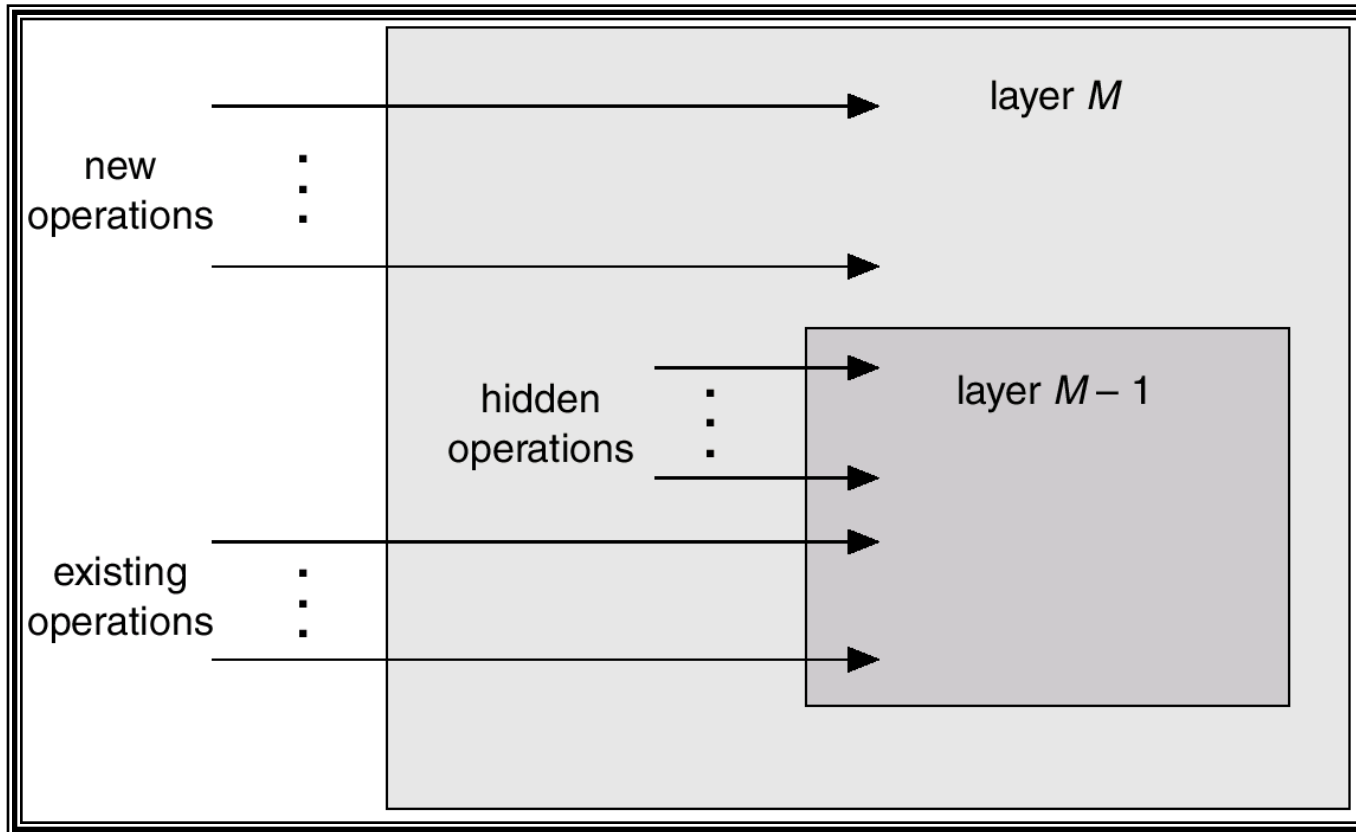
# Struttura a strati di un sistema operativo

---



# Un livello del S.O.

---



# Android



# Struttura del sistema a microkernel

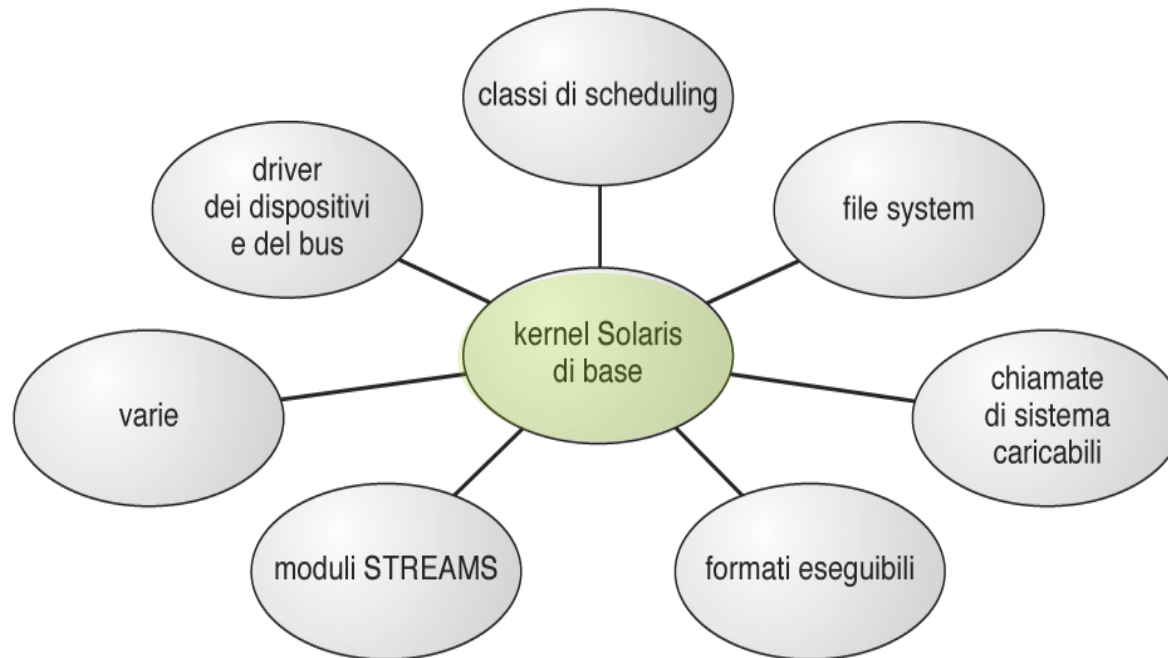
---

- La maggior parte delle funzioni sono spostate fuori dal nucleo nello “user space”.
- Le comunicazioni tra i moduli utente avvengono tramite lo scambio di messaggi .
- Benefici:
  - maggiore facilità di estensione del microkernel
  - maggiore facilità di portare il S.O. su nuove architetture
  - maggiore affidabilità
  - maggiore sicurezza.

# Struttura modulare di Solaris

---

- Il sistema Solaris prevede una serie di componenti caricabili dinamicamente costruiti intorno al kernel.
- Ogni modulo può chiamare gli altri secondo un modello object-oriented.





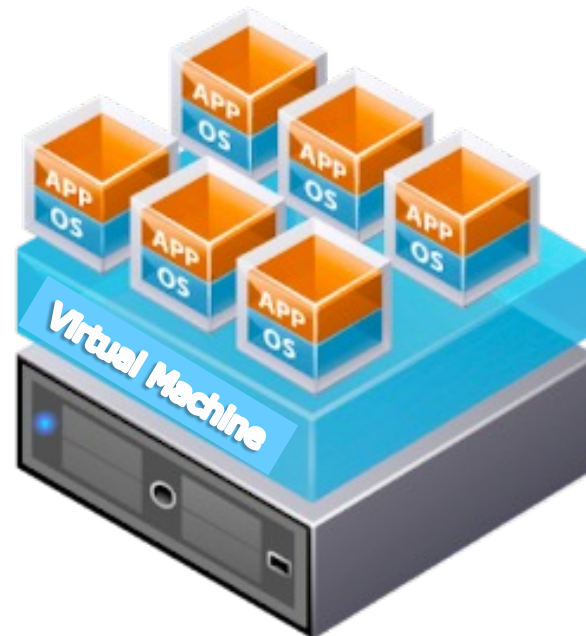
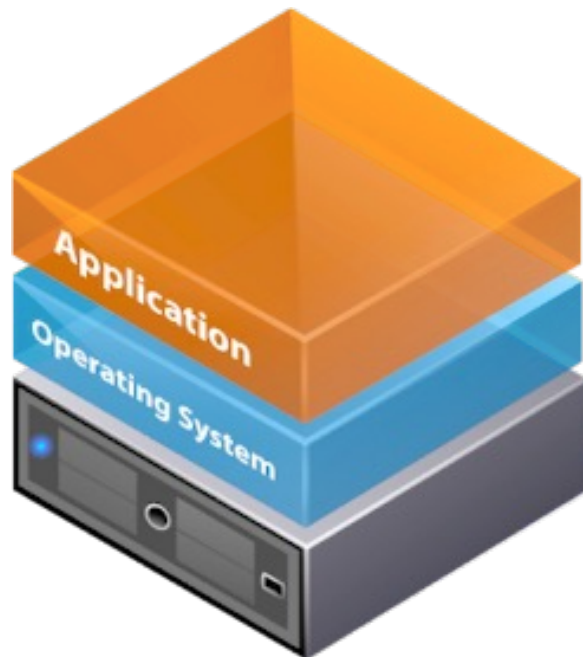
# Macchine Virtuali

---

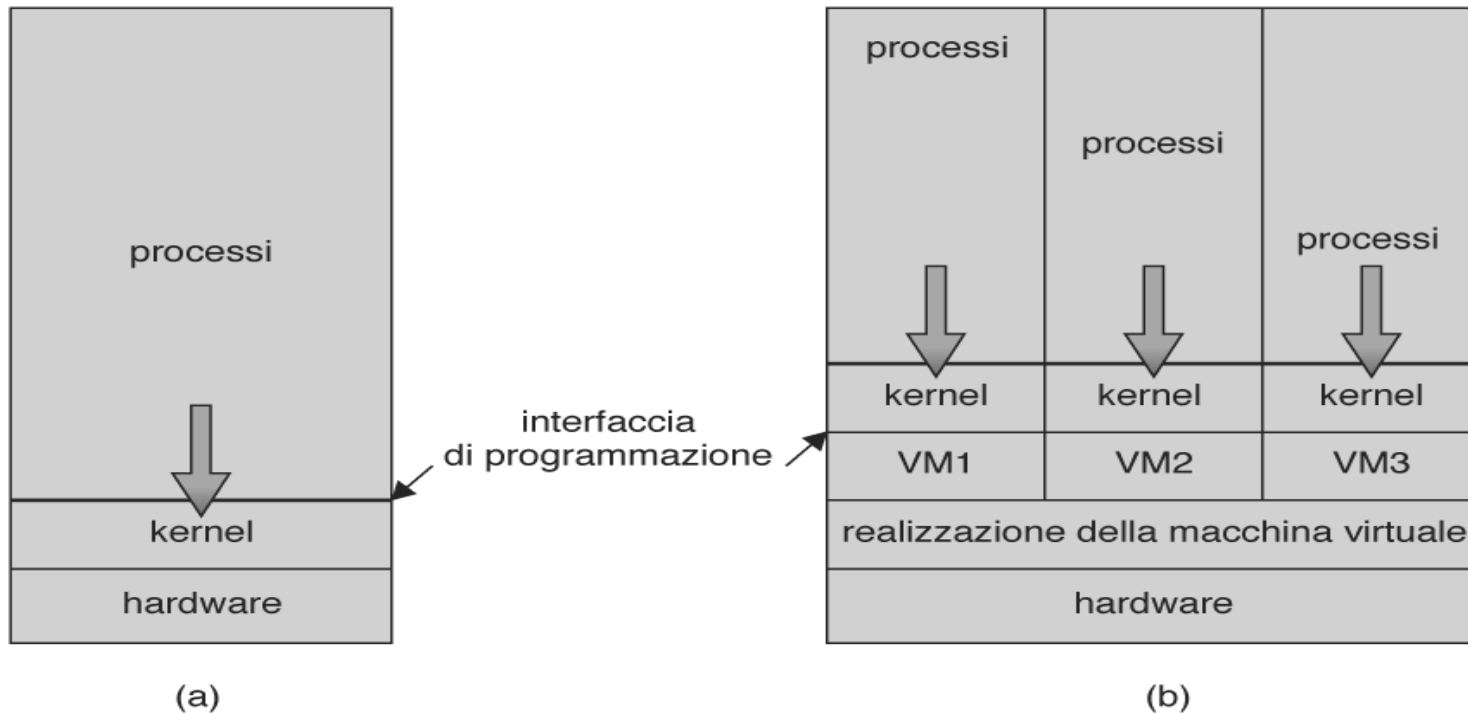
- Una **macchina virtuale** porta l'approccio a livelli alla sua massima esplicitazione.
- L'hardware e il nucleo sono considerati come se fossero un unico elemento.
- Una macchina virtuale fornisce una *identica interfaccia* a diverse architetture hardware e più macchine virtuali possono essere installate su uno stesso computer.
- Il sistema operativo crea l'impressione di diversi processi ognuno in esecuzione su un proprio processore usando la propria memoria.

# Macchine Virtuali

- Le risorse fisiche di un computer sono condivise per creare le macchine virtuali.
- Lo scheduling della CPU crea l'impressione che ogni utente abbia il proprio processore.
- Si possono avere virtualizzazioni di dispositivi di I/O.



# Modelli di sistema



(a) Senza macchina virtuale (b) Macchina virtuale

# Vantaggi/Svantaggi delle Macchine Virtuali

---

- Il concetto di macchina virtuale fornisce una protezione completa delle risorse del sistema poiché ogni macchina virtuale è isolata dalle altre. L'isolamento tuttavia non permette condivisione di risorse.
- Una macchina virtuale è un ambiente ideale per la ricerca e lo sviluppo che vengono svolti sulla macchina senza perturbare il normale funzionamento del sistema operativo.
- Il concetto di macchina virtuale è complesso da implementare a causa della necessità di fornire un duplicato esatto della macchina sottostante.

# Java Virtual Machine

---

- Il programmi Java dopo la compilazione sono ***architecture-neutral*** e vengono eseguiti dalla Java Virtual Machine (JVM).
- La JVM consiste di
  - un class loader
  - un class verifier
  - un runtime interpreter

# Java Virtual Machine

---

