

# Sistemi Operativi

**C.d.L. in Informatica (laurea triennale)**  
Anno Accademico 2022-2023

Canale A-L

Dipartimento di Matematica e Informatica – Catania

## Introduzione

Prof. Mario Di Raimondo

# Sistemi Operativi

- 9 CFU (72 ore)
- Struttura:
  - teoria
  - laboratorio
- Propedeuticità:
  - Architettura degli Elaboratori
  - Programmazione I
- Esame:
  - prova teoria (65%):
    - scritto + orale
  - prova pratica in laboratorio (35%)

# Risorse e contatti

- Prof. Mario Di Raimondo
  - Email: [diraimondo@dmf.unict.it](mailto:diraimondo@dmf.unict.it)
  - Home Page: <https://diraimondo.dmf.unict.it>
  - Ufficio: stanza 355 – Blocco I
- Pagina del corso:  
<https://diraimondo.dmf.unict.it/teaching/sistemi-operativi/>
- Calendario
- Gruppo Telegram
- FAQ

# Libri e appunti

- **Testo principale:**

- *Andrew S. Tanenbaum, Herbert Bos*  
**I moderni sistemi operativi**  
4ª edizione (2016) o 5ª edizione (2023)  
Pearson

- **Testo secondario:**

- *Abraham Silberschatz, Peter Baer Galvin, Greg Gagne*  
**Sistemi operativi - Concetti ed esempi**  
9ª edizione (2014) o 10ª edizione (2019)  
Pearson

- **Slide**

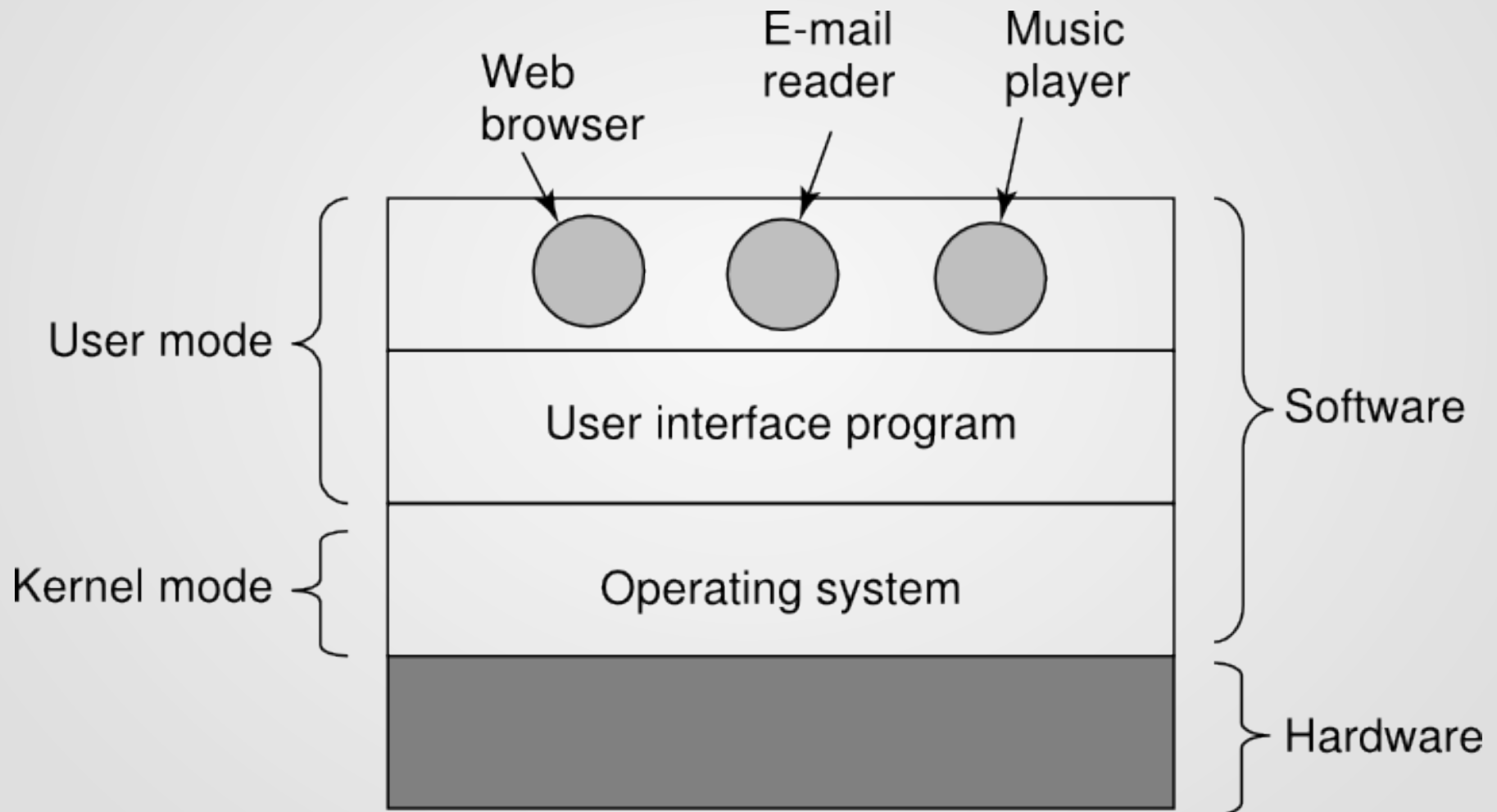


# Cos'è un Sistema Operativo?

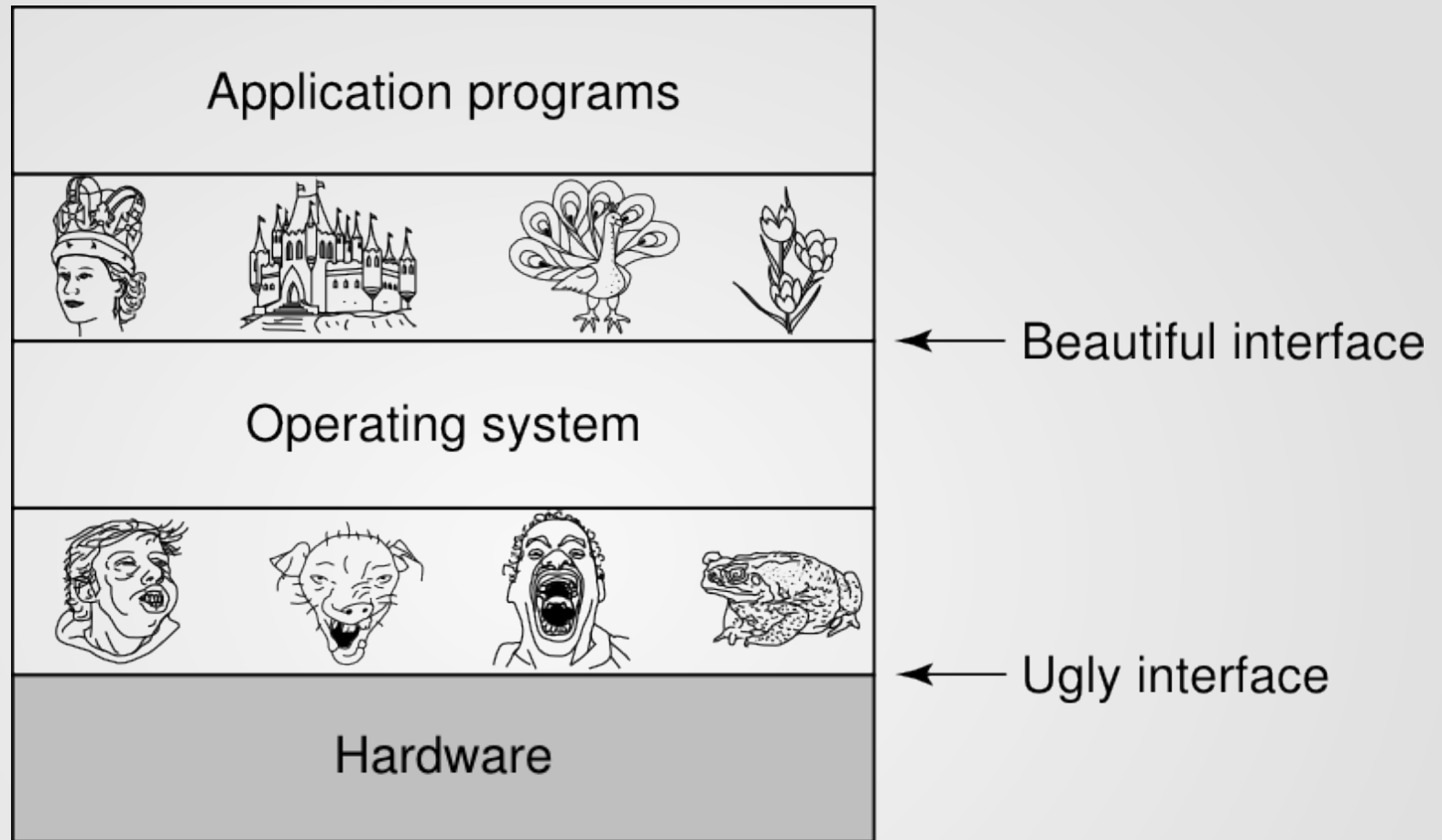
- Un **moderno calcolatore** è tipicamente formato da:
  - uno o più processori;
  - memoria centrale;
  - dischi;
  - stampanti e altre periferiche di I/O.
- I dettagli di basso livello sono **molto complessi**.
- Gestire tutte queste componenti richiede uno strato intermedio software: il **Sistema Operativo**.

# Cos'è un Sistema Operativo?

- Doppia modalità supportate dall'hardware:
  - **modalità kernel** (o supervisor);
  - **modalità utente**.



# Il sistema operativo come macchina estesa



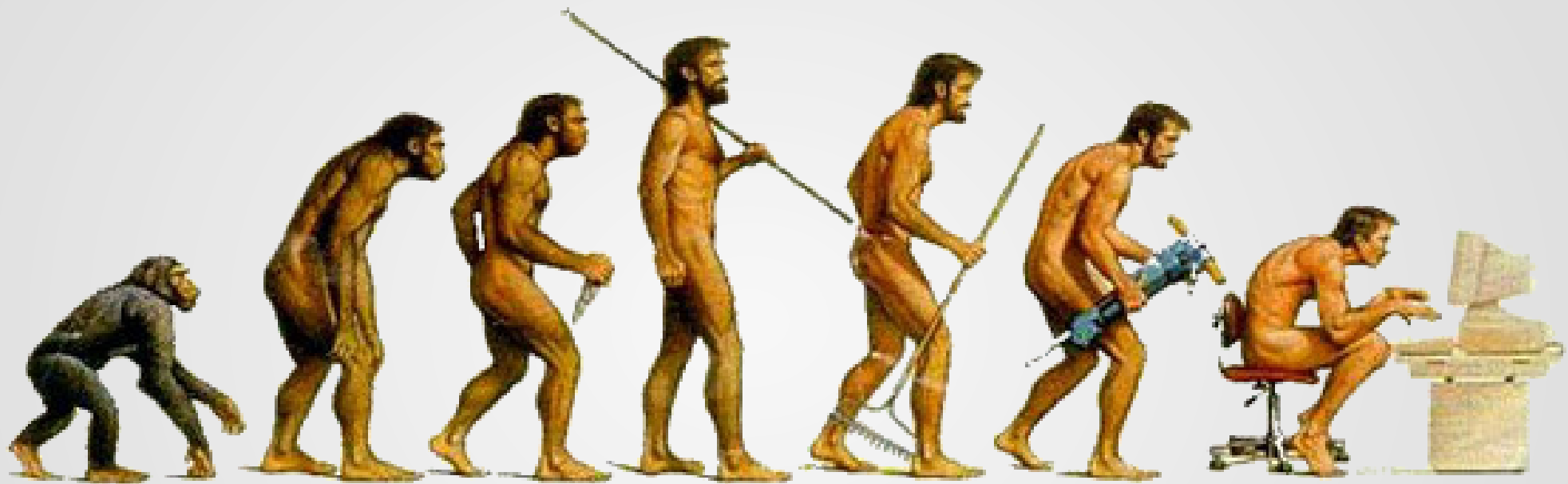
- concetto di **astrazione**

# Il sistema operativo come gestore delle risorse

- Da un moderno sistema operativo ci aspettiamo che gestisca:
  - **più programmi** in esecuzione;
  - **più utenti.**
- Necessita **allocazione ordinata e controllata** di risorse quali: processori, memoria, unità di I/O,...
- Non solo hardware: file, database,...
- **Multiplexing:**
  - **nel tempo:** CPU, stampante,...
  - **nello spazio:** memoria centrale, disco,...

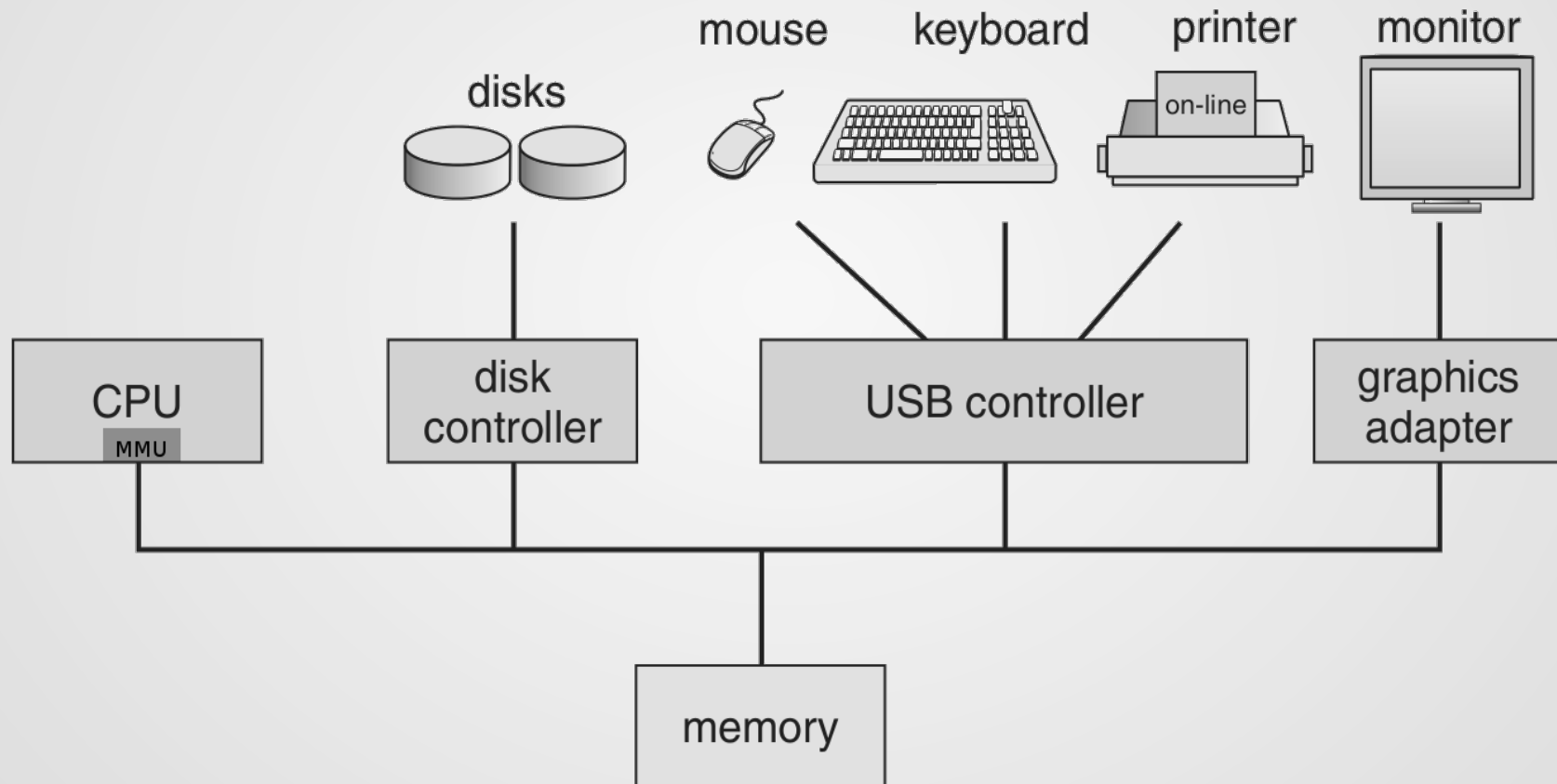


# L'evoluzione dei sistemi operativi



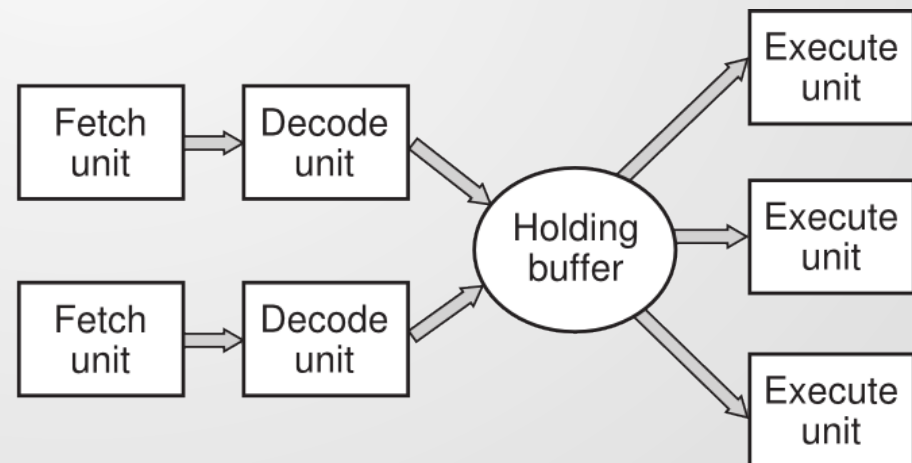
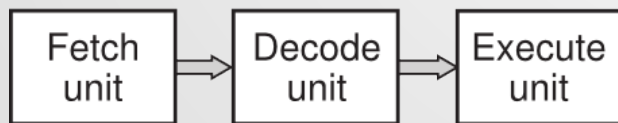
# Uno sguardo all'hardware

- Architettura (semplificata) di un calcolatore



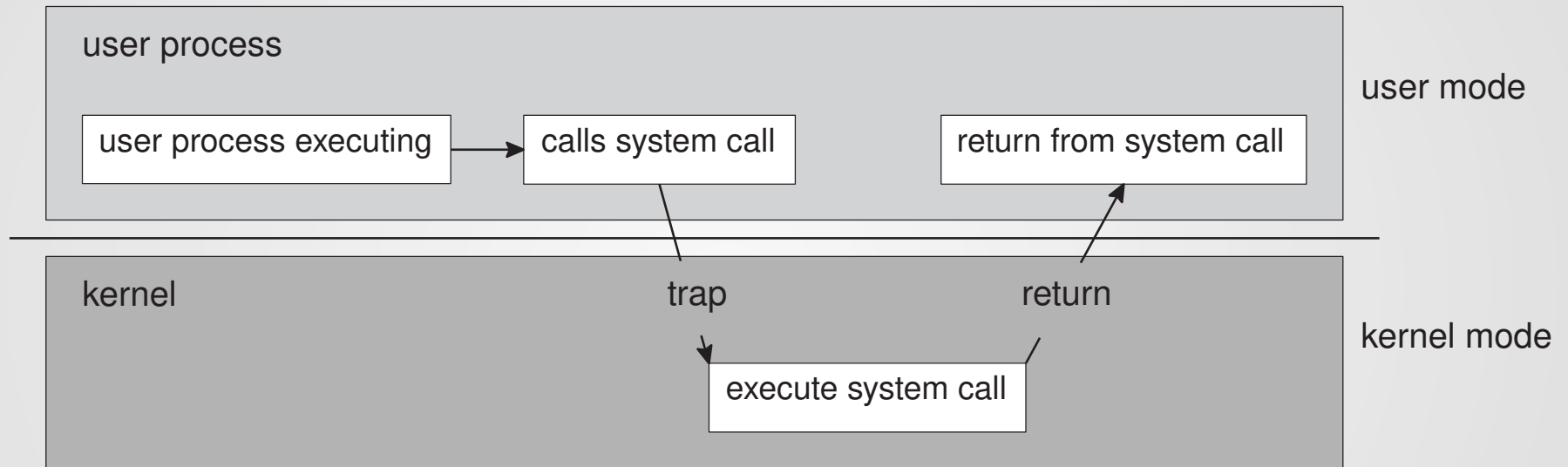
# Il Processore

- **Ciclo di base:** prelevamento (fetch), decodifica, esecuzione
- Registri particolari:
  - **Program Counter (PC);**
  - **Stack Pointer (SP);**
  - **Program Status Word (PSW).**
- Progettazioni avanzate: **pipeline, cpu superscalare**
  - non del tutto trasparenti al SO



# Modalità di esecuzione

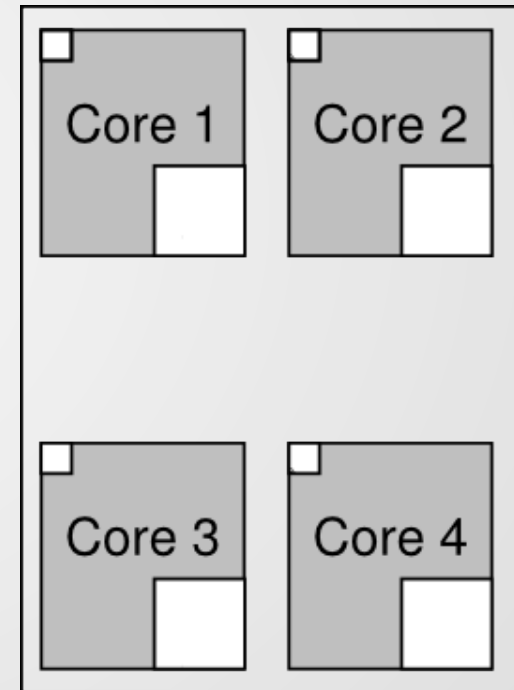
- Doppia modalità di esecuzione;
- **chiamate di sistema (TRAP);**



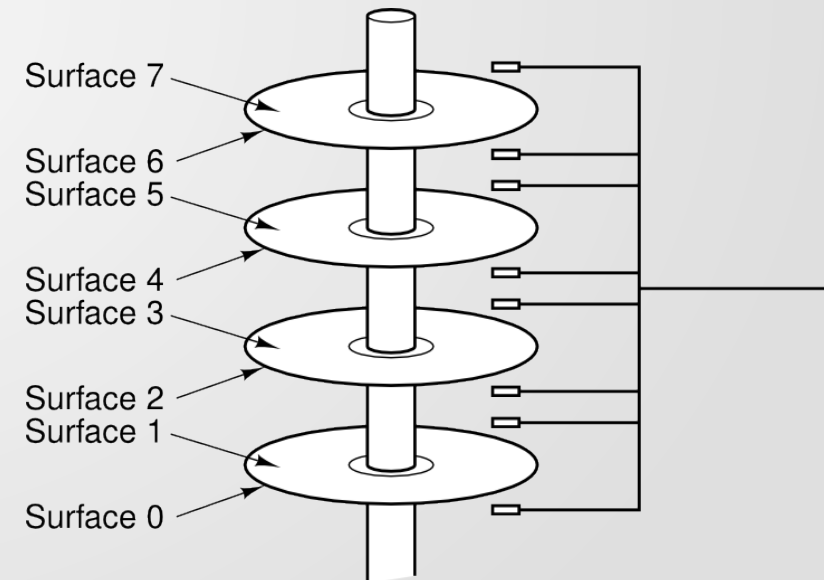
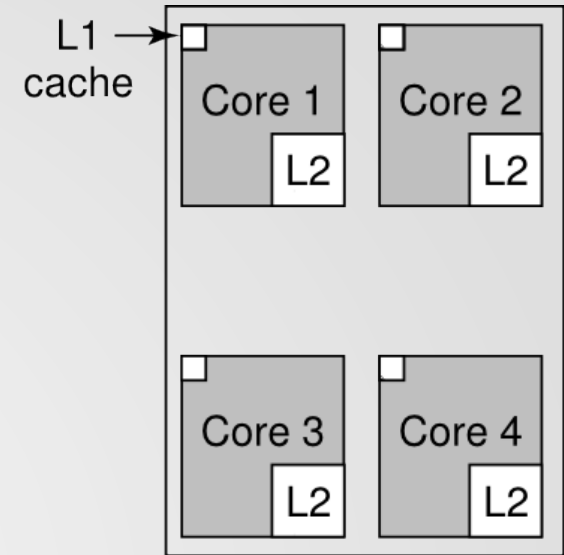
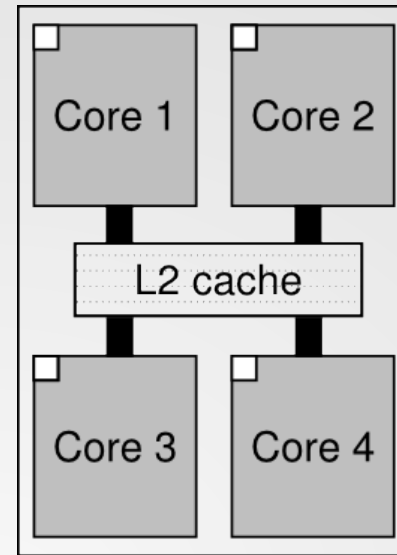
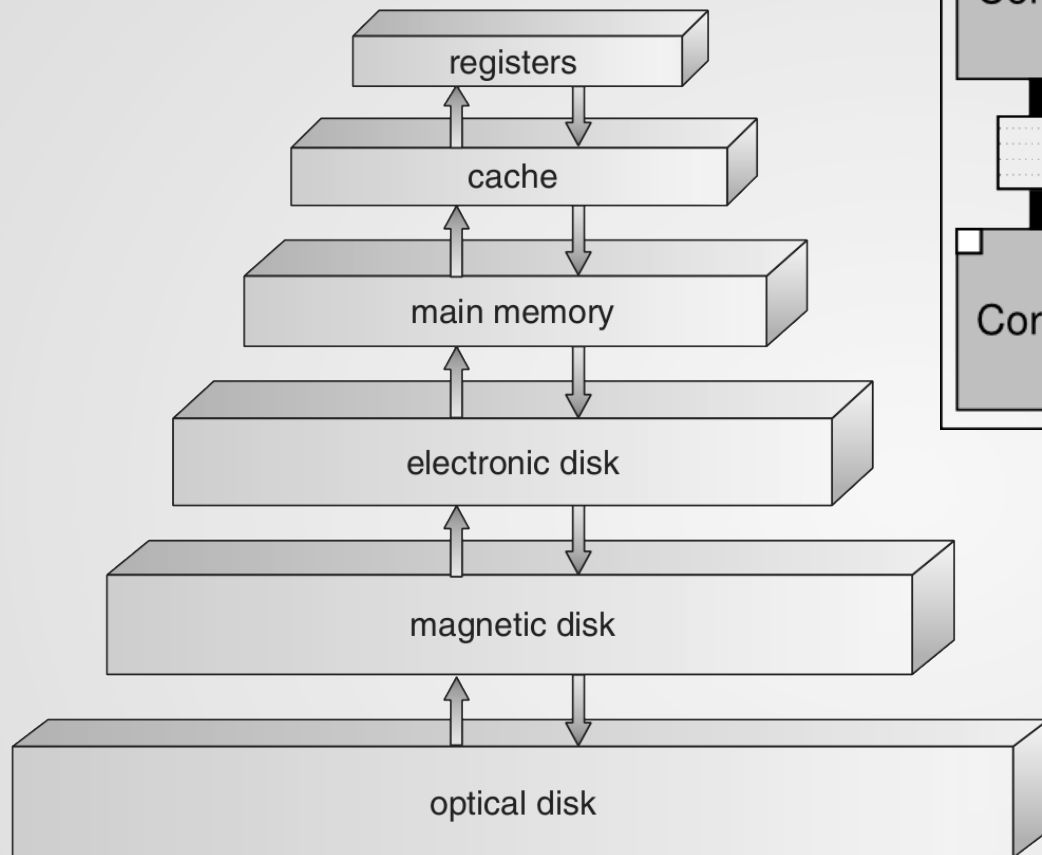
- **interrupt hardware.**

# Più processori

- **Multithreading** (o hyperthreading):
  - tiene all'interno della CPU lo stato di due thread;
  - non c'è una esecuzione parallela vera e propria;
  - il S.O. deve tenerne conto.
- **Multiprocessori**, vantaggi:
  - **throughput**;
  - **economia di scala**;
  - **affidabilità**;
- **Multicore**;
- **GPU**.



# Memorie



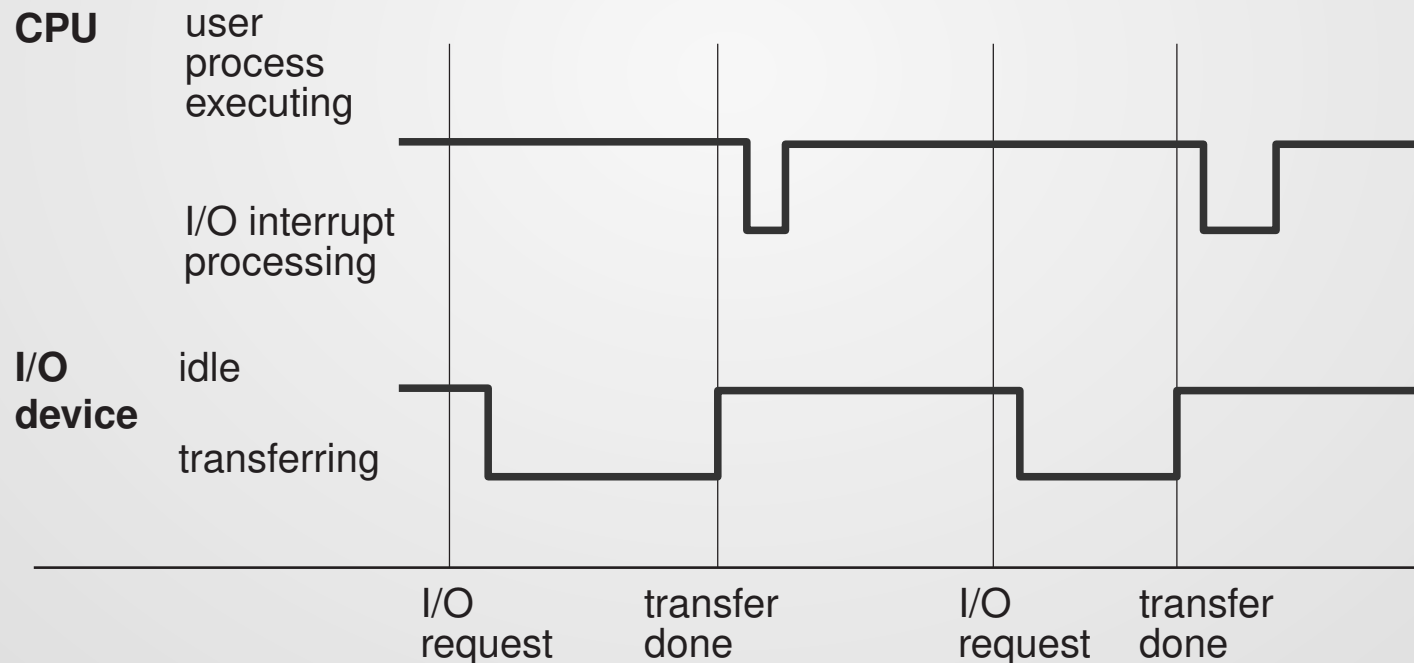
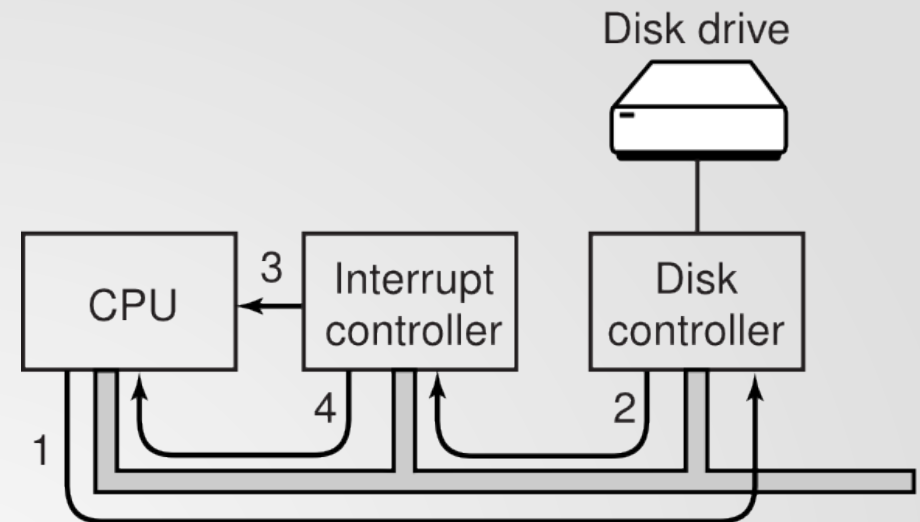
- Unità di misura: kB/MB/GB vs. KiB/MiB/GiB

# Dispositivi di I/O

- Si individuano due componenti:
  - il **controller**: più semplice da usare per il S.O.;
  - il **dispositivo** in sé: interfaccia elementare ma complicata da pilotare.
  - esempio: dischi SATA.
- Ogni controller ha bisogno di un **driver** per il S.O.
- Il driver interagisce con il controller attraverso le **porte di I/O**:
  - istruzioni tipo IN / OUT;
  - mappatura in memoria.

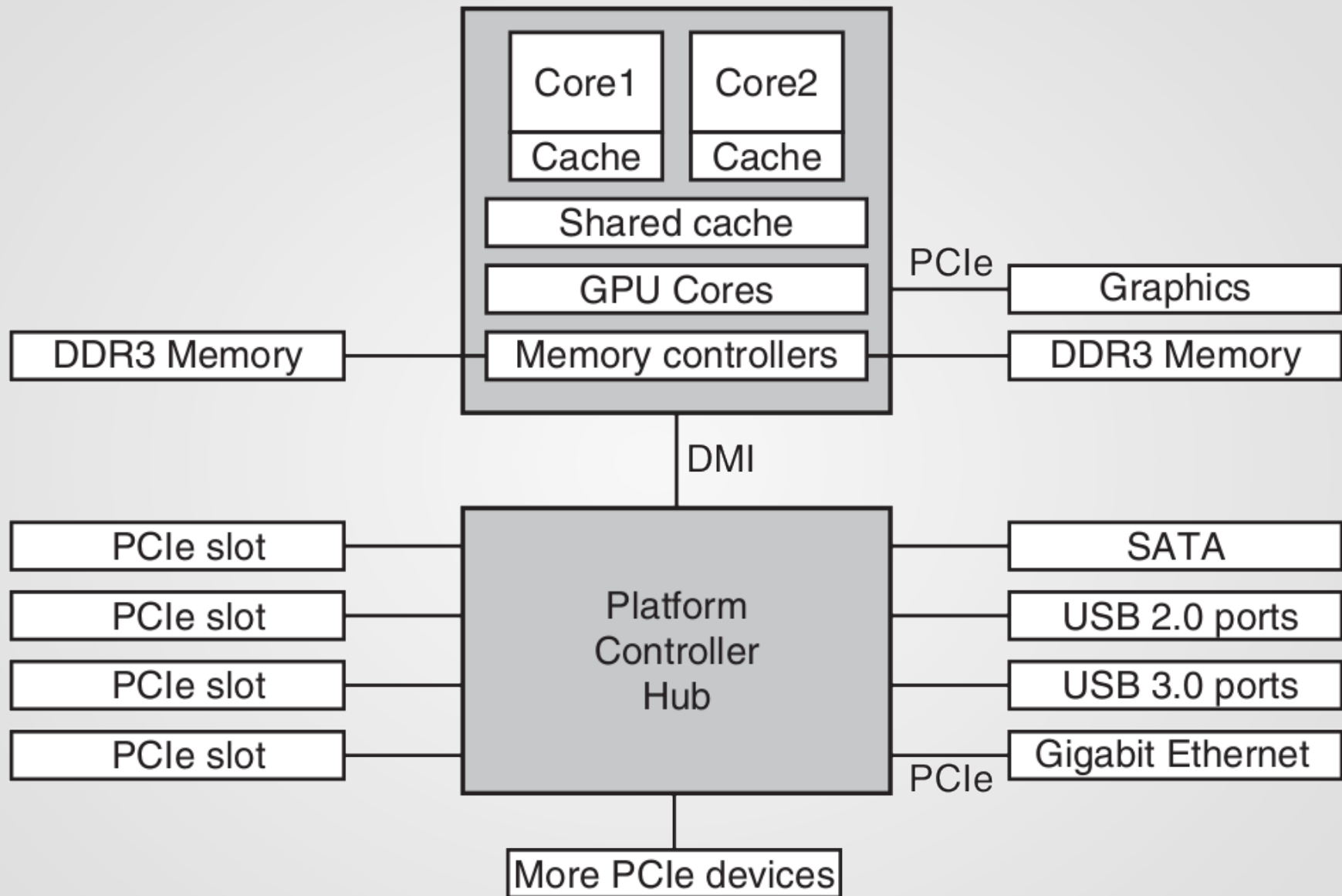
# Dispositivi di I/O

- Modalità di I/O:
  - **busy waiting;**
  - con programmazione di **interrupt**;
  - con uso del **DMA**.





# Bus



# Lo zoo dei sistemi operativi

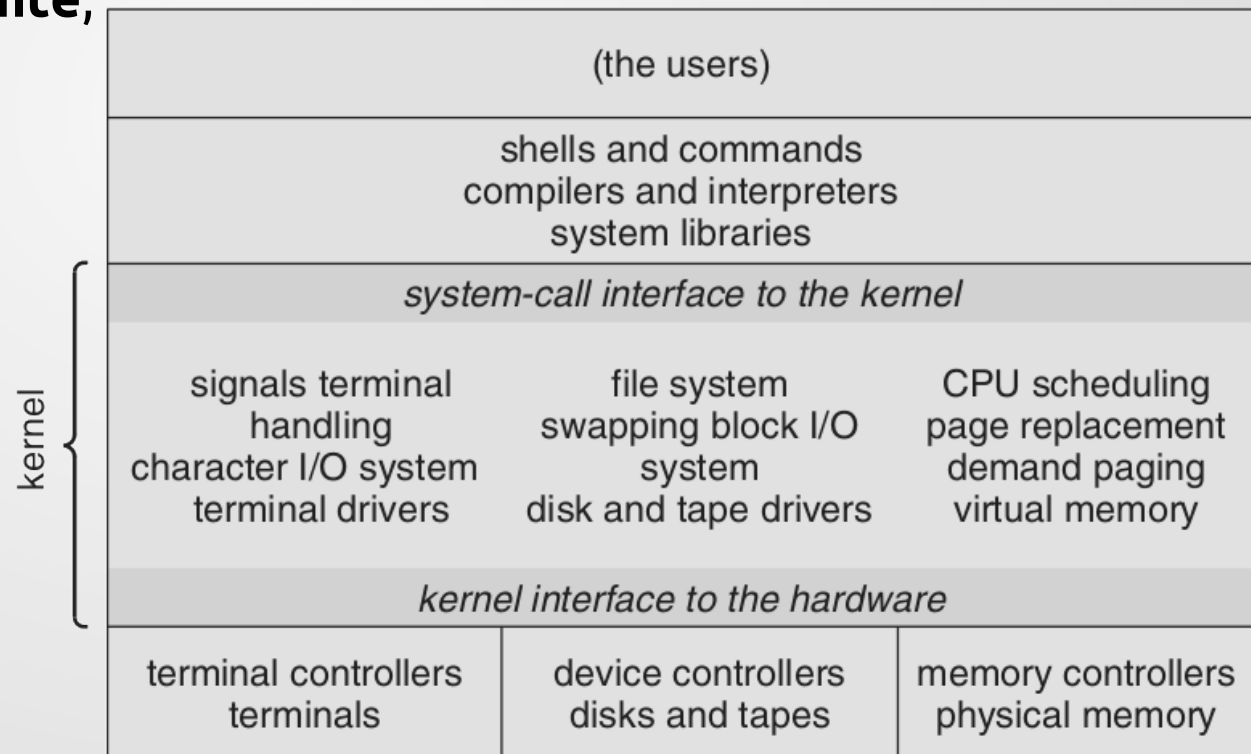
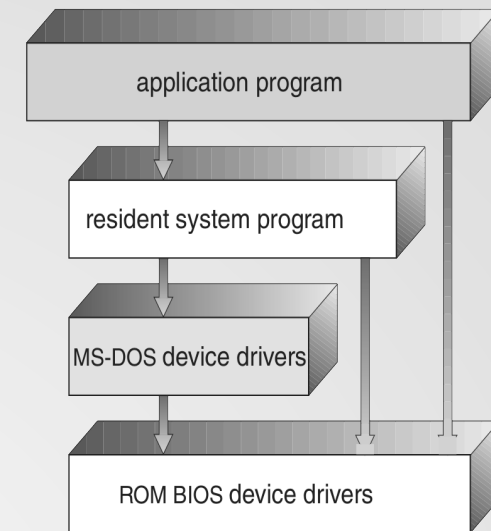
- Sistemi operativi per mainframe/server
- Sistemi operativi per personal computer
- Sistemi operativi per palmari/smart-phone
- Sistemi operativi per sistemi integrati (embedded)
- Sistemi operativi real-time

# Struttura di un sistema operativo

- Alcune **possibili strutture** per un SO:
  - Monolitici
  - A livelli (o a strati)
  - Microkernel
  - A Moduli
  - Macchine virtuali
- categorie **con intersezioni** (sistemi ibridi);
- **tassonomia** non per forza completa o condivisa.

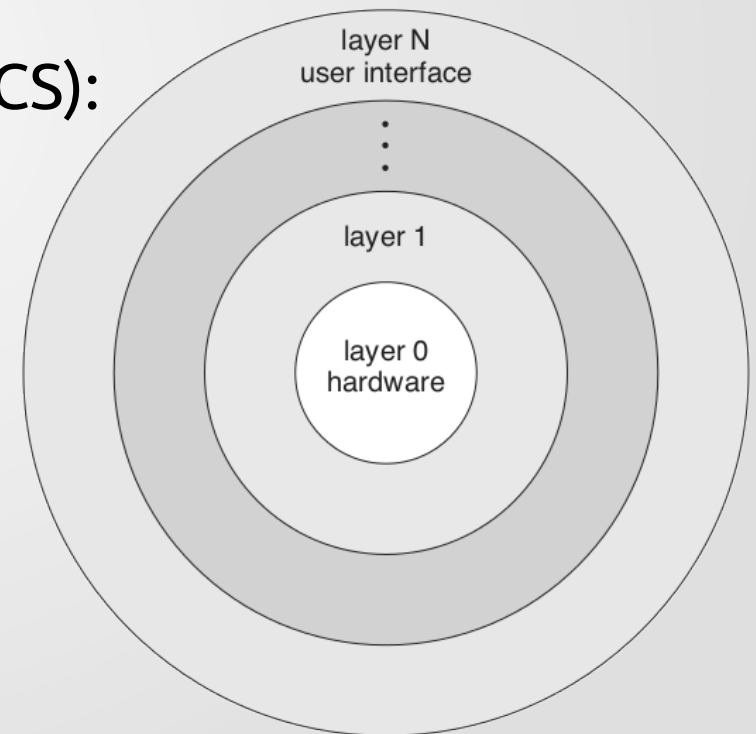
# Struttura monolitica

- **Monolitici:**
  - **nessun supporto hardware;**
    - problemi...;
    - esempi: MS-DOS, UNIX;
  - arrivò il supporto hardware alla **modalità kernel/utente;**
    - **unico kernel** con tutto dentro;
    - ogni componente può richiamare tutti gli altri
    - poco gestibile nel tempo.



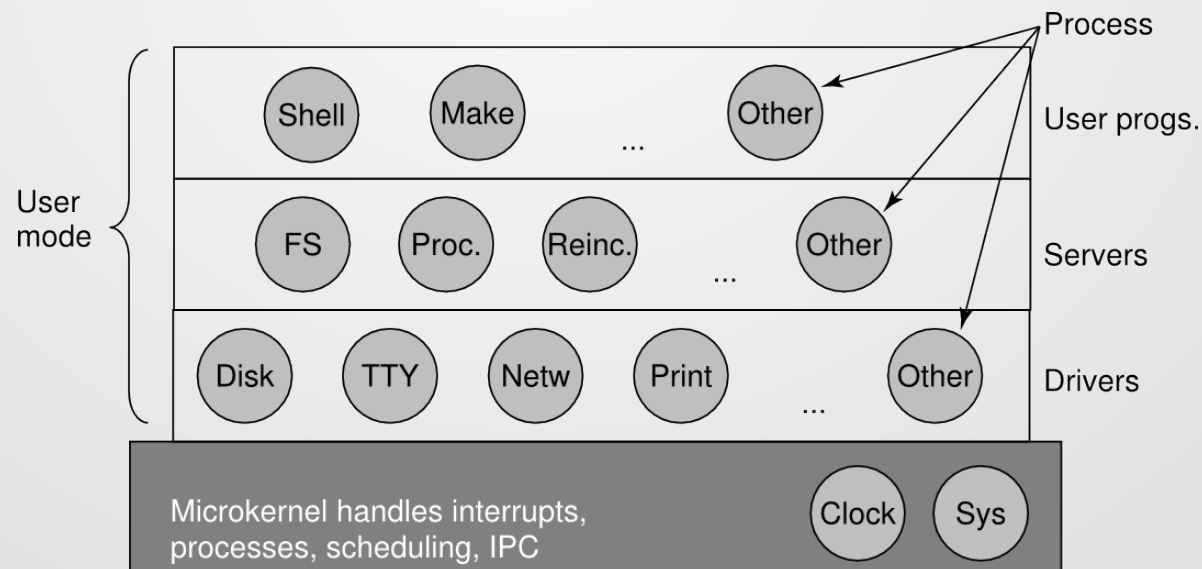
# Struttura a livelli (o a strati)

- Si utilizza una **gerarchia di livelli**;
- ogni livello implementa delle **funzionalità** impiegando quelle fornite da quello inferiore;
- **migliore progettazione**, più semplice da sviluppare e controllare (incapsulamento tipo OOP); suddivisione a livello progettuale;
- variante ad **anelli concentrici** (MULTICS):
  - **separazione forzata** dall'hardware;
- **problemi di prestazioni** dovuti alle chiamate nidificate e al relativo overhead.



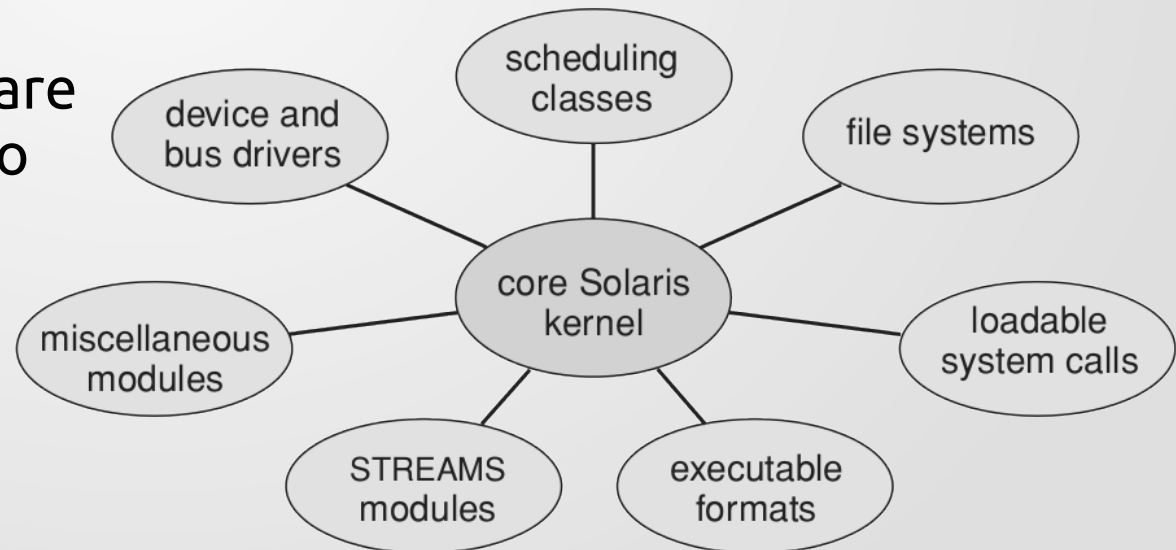
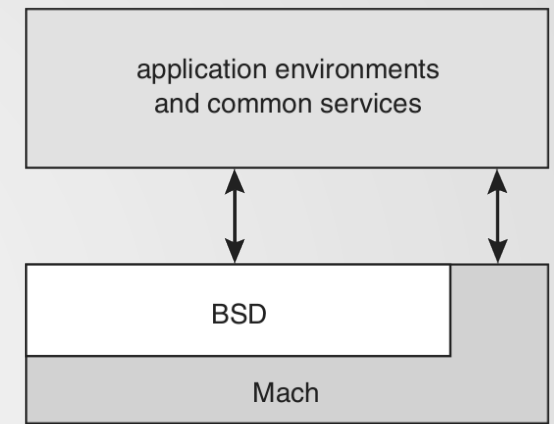
# Microkernel

- Uso di un **microkernel minimale** che si occupa dello scheduling, memoria e IPC;
- tutto il resto gestito da **moduli** (livello utente): filesystem, driver di dispositivi;
- comunicazione attraverso **messaggi**;
- miglior design (componenti piccoli) e migliore stabilità;
- esempi: MINIX 3, Mach, QNX, Mac OS (Darwin), Windows NT



# Struttura a Moduli

- Idea della **programmazione OO** applicata al kernel;
- **moduli** che implementano un qualche aspetto specifico;
  - filesystem, driver,...
- **kernel principale** a funzionalità ridotte;
- moduli **caricabili dinamicamente**;
- design pulito (ad oggetti);
- **efficiente**:
  - ogni modulo può invocare qualunque altro modulo direttamente;
  - niente messaggi;
- esempi: Solaris, Linux, Mac OS (ibrido).



# Macchine virtuali

- L'estremizzazione del concetto di astrazione porta alla **virtualizzazione**;
- **Perchè?** Uso di più SO, VPS, isolamento dei servizi,...
- **Simulazione, paravirtualizzazione.**
- Viene tutto gestito dallo **Hypervisor**:
  - **Hypervisor di tipo 1:**
    - gira direttamente sull'hardware;
    - esempi: VMware ESX/ESXi, Microsoft Hyper-V hypervisor;
  - **Hypervisor di tipo 2:**
    - è un processo in un SO Host
    - esempi: VMware Workstation, VirtualBox.
- **Supporto hardware** per efficienza.
- **Java Virtual Machine.**

