Sistemi Operativi 1

AA 2021/2022

Architettura di un Sistema Operativo

Principi di progettazione

- Principio molto importante è la separazione tra

  Policy: Cosa deve essere fatto?
  Meccanismi: Come farlo?

- I meccanismi determinano come fare qualcosa, policy decidono cosa deve essere fatto
  - La separazione tra policy e meccanismi e molto importante perché permette la maggior flessibilitá se le policy devono essere modificate in un secondo tempo.
    - Es. tutti i processi devono poter accedere alla CPU in un tempo finito…….

Principi di progettazione

- KISS: Keep it small and simple

- Altro principio è il *POLA:* Principle of the Least Privileges
  - Che dice che ogni componente deve avere solo i privilegi che sono necessari ad eseguire la sua funzione e nulla piú
  - Fondamentale per garantire l'affidabilita e la sicurezza di un S.O.

Struttura di un S.O.

- Sistemi monolitici
- Sistemi “a struttura semplice”
- Sistemi a livelli
- Sistemi basati su kernel
- Sistemi micro-kernel
- Virtual machine
- Sistemi client-server

Sistemi monolitici

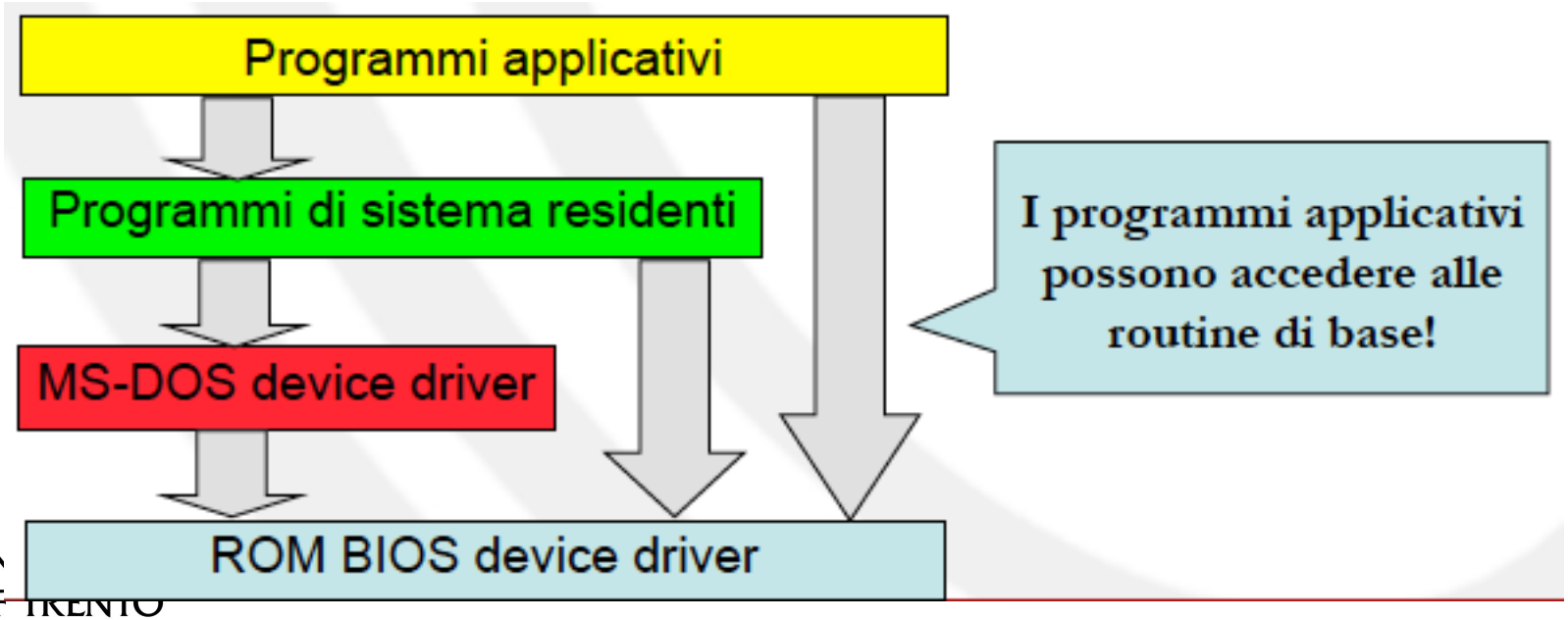
- No gerarchia
  - Unico strato SW tra utente e HW
  - Componenti tutti allo stesso livello
  - Insieme di procedure che possono chiamarsi vicendevolmente
- Svantaggi
  - Codice dipendente dall'architettura HW era distribuito su tutto il S.O.
  - Test e debug difficile

Sistemi a struttura semplice

- Minima organizzazione gerarchica
  - Definizione dei livelli della gerarchia molto flessibile
  - Strutturazione mira a ridurre costi di sviluppo e manutenzione
- Es.: MS-DOS, UNIX originale

MS-DOS

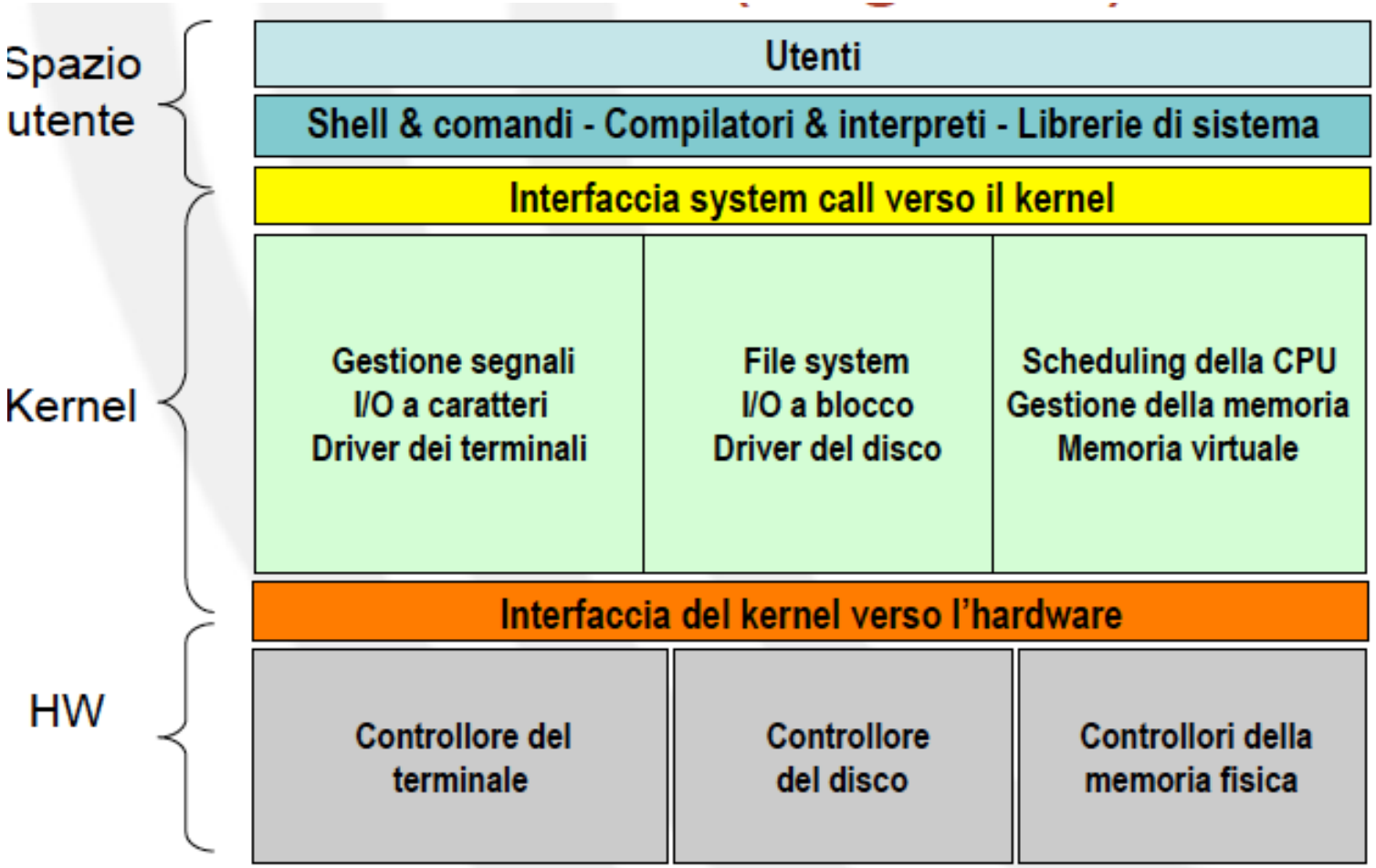
- Pensato per fornire il maggior numero di funzionalità nel minimo spazio
  - Non suddiviso in moduli
  - Possiede un minimo di struttura, ma le interfacce e i livelli di funzionalità non sono ben definiti
  - Non prevede dual mode (perché Intel 8088 non lo forniva)



UNIX (originale)

- Struttura base limitata a causa delle limitate funzionalità HW
  - Programmi di sistema
  - Kernel
    - Tutto ciò che sta tra il livello dell'interfaccia delle system call e l'HW
    - Fornisce
      - File system
      - Scheduling della CPU
      - Gestione della memoria
      - Altre funzioni

UNIX (originale)



Sistema a livelli

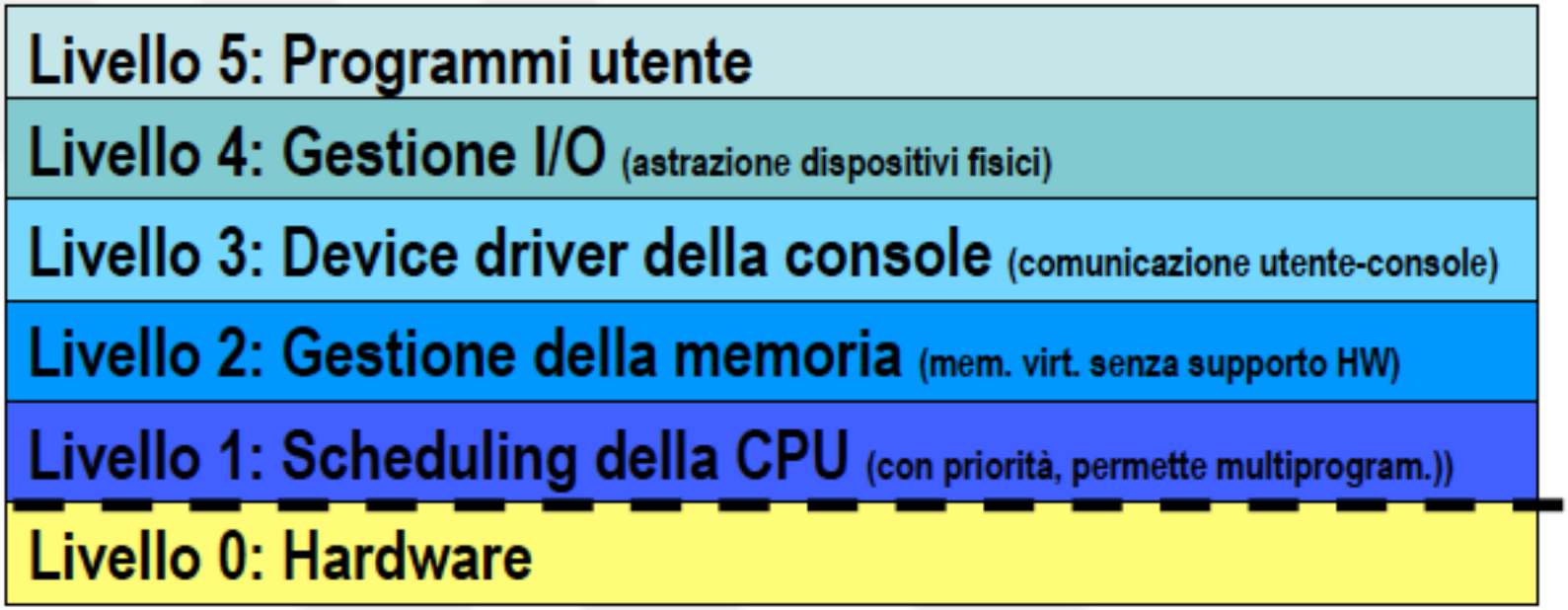
- Servizi organizzati per livelli gerarchici
  - Interfaccia utente (livello più alto) → … → HW (più basso)
  - Ogni livello:
    - può usare solo funzioni fornite dai livelli inferiori
    - definisce precisamente il tipo di servizio e l'interfaccia verso il livello superiore nascondendone l'implementazione

Sistemi a livelli

- Vantaggi:
  - Modularità: facilita messa a punto e manutenzione del sistema
- Svantaggi:
  - Difficile definire appropriatamente gli strati
  - Minor efficienza: ogni strato aggiunge overhead alle system call
  - Minor portabilità: funzionalità dipendenti dall'architettura sono sparse sui vari livelli
- Es.: THE, MULTICS, OS/2

THE (Dijkstra 1968)

- Sistema operativo accademico per sistemi batch
- Primo esempio di sistema a livelli
  - Insieme di processi cooperanti sincronizzati tramite semafori

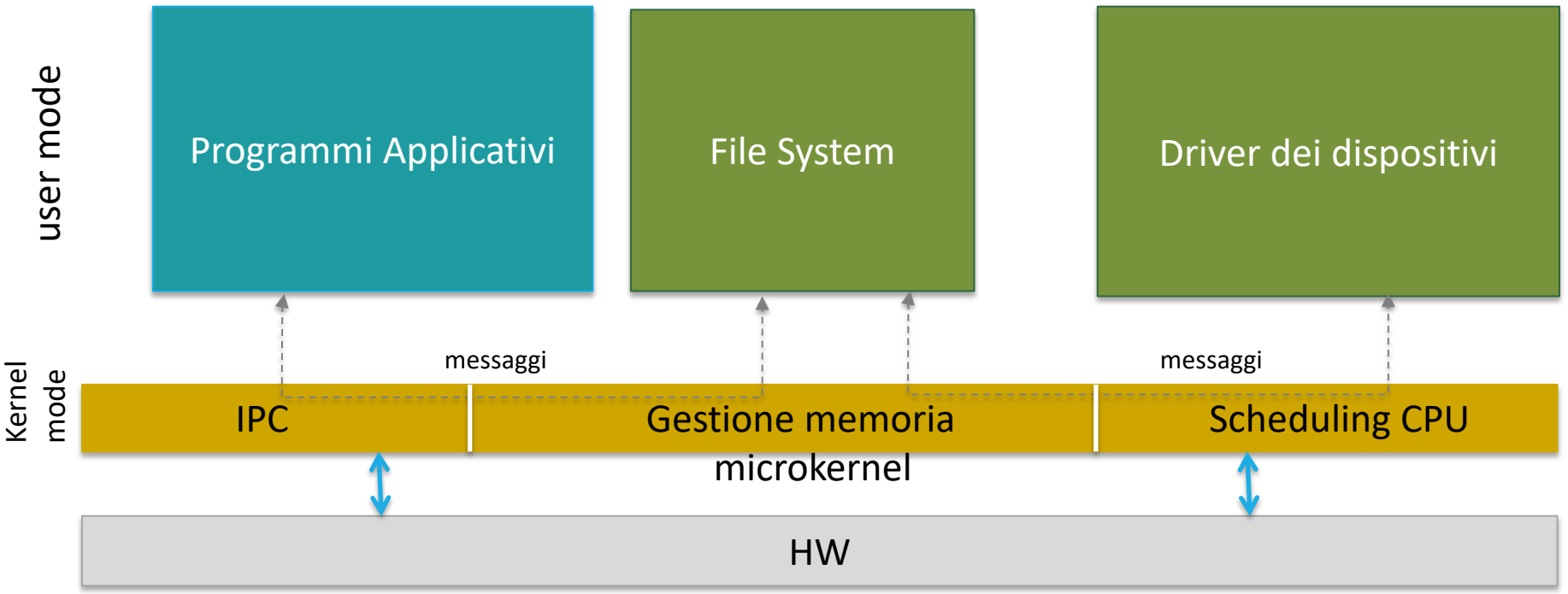


Sistemi basati su kernel

- Due soli livelli: Servizi kernel e servizi non-kernel
  - Alcune funzionalità fuori dal kernel (es.: File system)
  - Es.: Implementazioni “moderne” di UNIX
- Vantaggi
  - Vantaggi dei sistemi a livelli, ma senza averne troppi
- Svantaggi
  - Non così generale come un sistema a livelli
  - Nessuna regola organizzativa per parti del S.O. fuori dal kernel
  - Kernel complesso tende a diventare monolitico

Micro-kernel

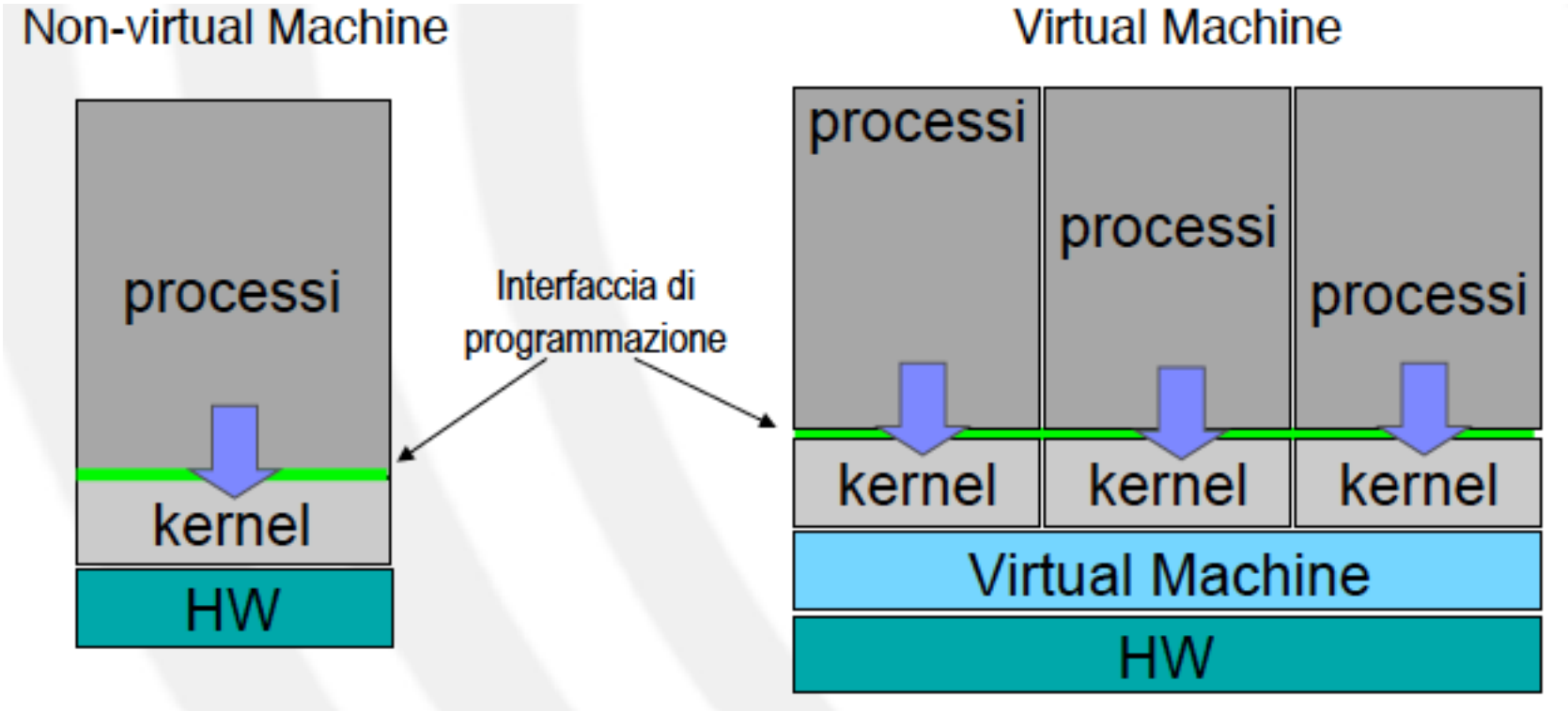
- Mettere nel kernel solo cio che è strettamente necessario



Virtual Machine

- Estremizzazione dell'approccio a livelli (IBM VM 1972)
  - Pensato per offrire un sistema timesharing "multiplo"
- HW e VM. trattati come hardware
  - Il S.O. esegue sopra la VM
  - La VM dà l'illusione di processi multipli, ciascuno in esecuzione sul proprio HW
- Possibilità di presenza di più S.O.

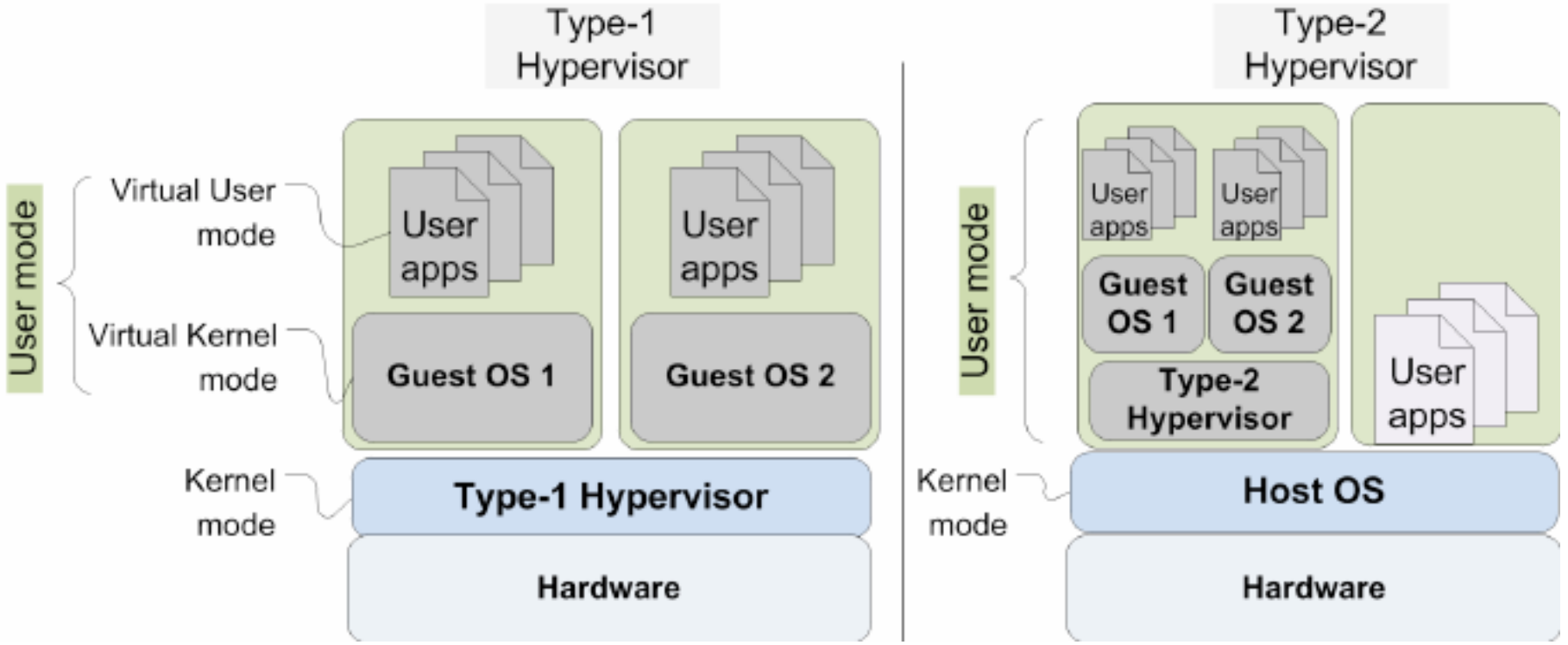
Virtual Machine



- Concetto chiave: separazione di
  - Multiprogrammazione (Virtual Machine)
  - Presentazione (S.O.)

Type1 and Type 2 hypervisor

VM= VM monitor (hypervisor) + VM runtime environment

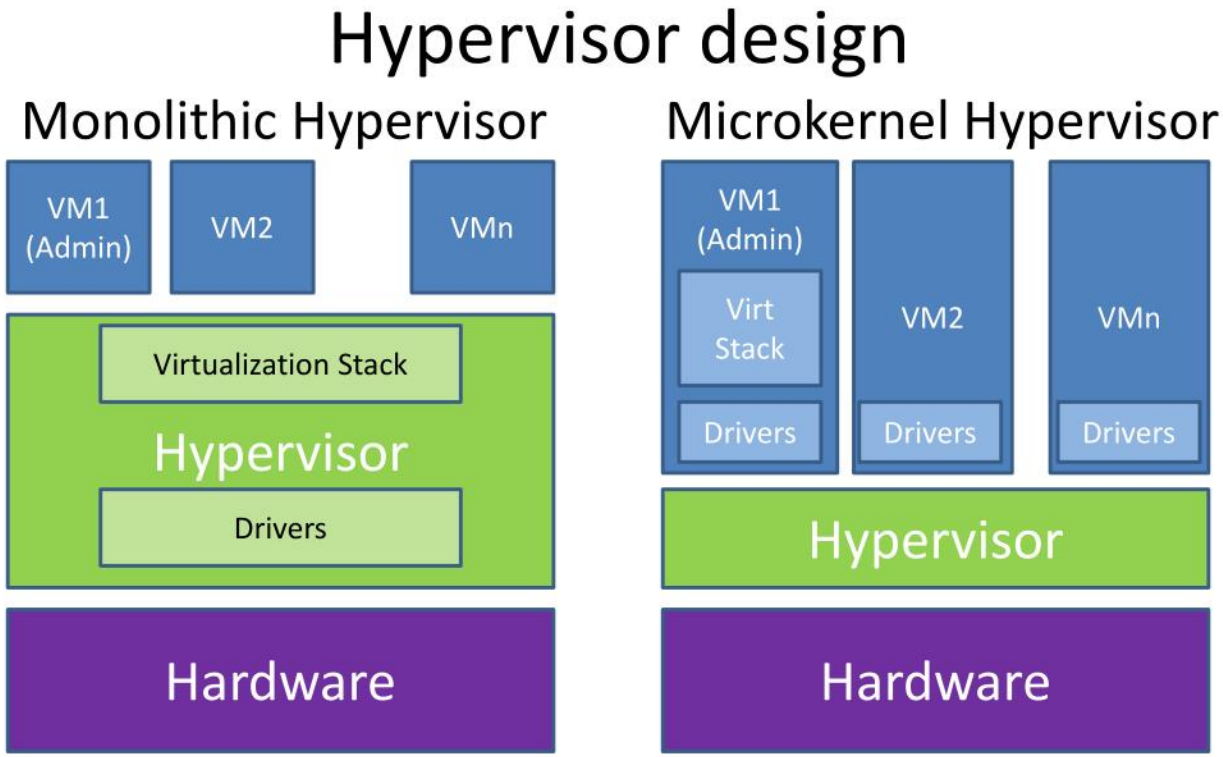


Es. VMware ESXi, Microsoft Hyper-V, and Apple Boot Camp, Xen, etc.

Es. VMware Workstation, Oracle VirtualBox, and Parallels Desktop for Mac, etc.

Monolithic vs Microkernel VM

Es. VMware ESXi



es. Hyper-V

- Both true Type 1 hypervisors – no host OS
- The hardware is the physical machine; OSs are all virtual

Virtual machine

- Vantaggi
  - Protezione completa del sistema: ogni VM è isolata dalle altre
  - Più di un S.O. sulla stessa macchina host
  - Ottimizzazione delle risorse
    - La stessa macchina può ospitare quello che senza VM doveva essere eseguito su macchine separate
  - Ottime per lo sviluppo di S.O.
  - Buona portabilità

Virtual machine

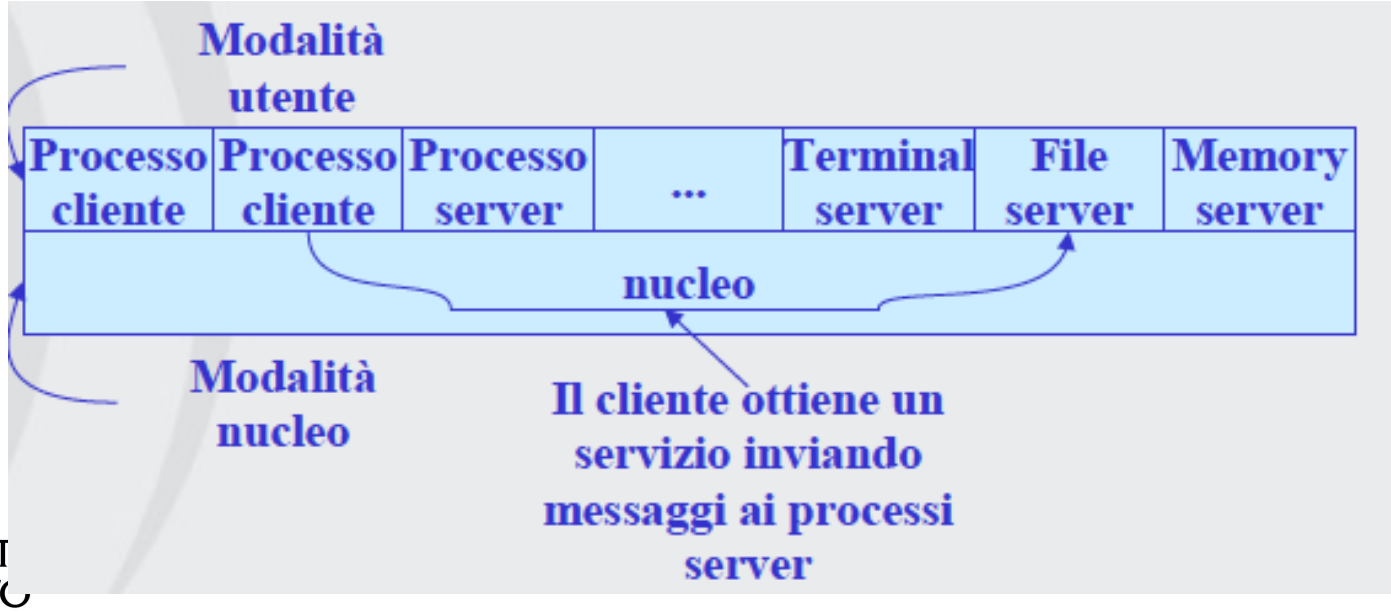
- Svantaggi
  - Problemi di prestazioni
  - Necessità di gestire dual mode virtuale
    - Il sistema di gestione delle VM esegue in kernel mode, ma la VM esegue in user mode
  - No condivisione: ogni VM è isolata dalle altre
    - Soluzione:
      - condivisione un volume del file system
      - Definire una rete virtuale tra VM via SW

Virtual machine

- Concetto usato ancora oggi, anche se in contesti diversi e con certi vincoli
- Esempi:
  - Esecuzione di programmi MS-DOS in Windows
    - Emulazione di 8086 (1MB memoria)
  - Esecuzione “contemporanea” di Linux e Windows (vMware, VirtualBox, …)
  - Java Virtual Machine (JVM)

Sistemi client-server

- Si basa sull'idea di portare il codice ai livelli superiori, lasciando un kernel più piccolo
- L'approccio è quello di implementare la maggior parte delle funzioni di sistema operativo nei processi utente
- Il kernel si occupa solo della gestione della comunicazione tra client e server
- Il modello si presta bene per S.O. distribuiti



Implementazione di un S.O.

- Tradizionalmente scritti in linguaggio assembler
- S.O. moderni scritti in linguaggi ad alto livello (C/C++)
- Vantaggi:
  - Implementazione più rapida
  - Più compatto
  - Più facile da capire e da mantenere
  - Portabilità