



UNIVERSITÀ
di **VERONA**

Dipartimento
di **INFORMATICA**

Architettura di un sistema operativo

Anno Accademico 2020-2021

Graziano Pravadelli

Struttura di un S.O.

Sistemi
monolitici

Sistemi “a
struttura
semplice”

Sistemi a livelli

Virtual
machine

Sistemi basati
su kernel

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

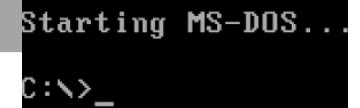
Codice dipendente dall'architettura HW era distribuito su tutto il S.O.
Test e debug difficile

Sistemi a struttura semplice

Minima organizzazione gerarchica

- Definizione dei livelli della gerarchia molto flessibile
- Strutturazione mira a ridurre costi di sviluppo/manutenzione

Esempi
MS-DOS, UNIX originale

A black rectangular box representing a terminal window. It contains the text "Starting MS-DOS..." in a monospaced font, followed by a new line showing the command prompt "C:\>_" in the same font.

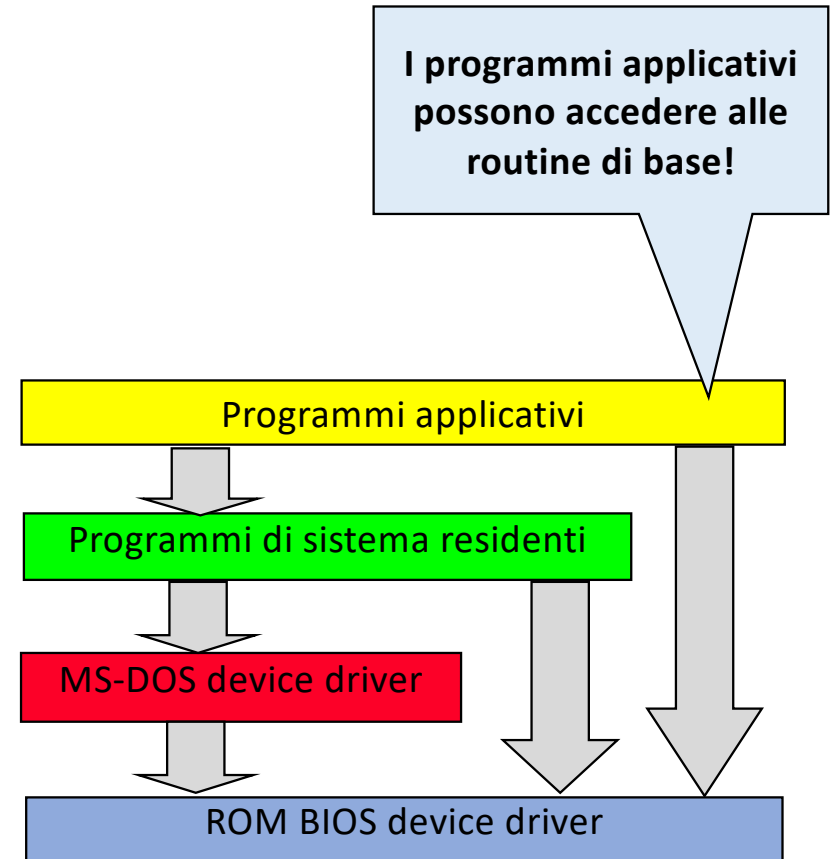
MS-DOS

Pensato per fornire maggior numero di funzionalità nel minimo spazio

Non suddiviso in moduli

Struttura minima, ma le interfacce e i livelli di funzionalità non sono ben definiti

Non prevede *dual mode* (perché Intel 8088 non lo forniva)



UNIX (originale)

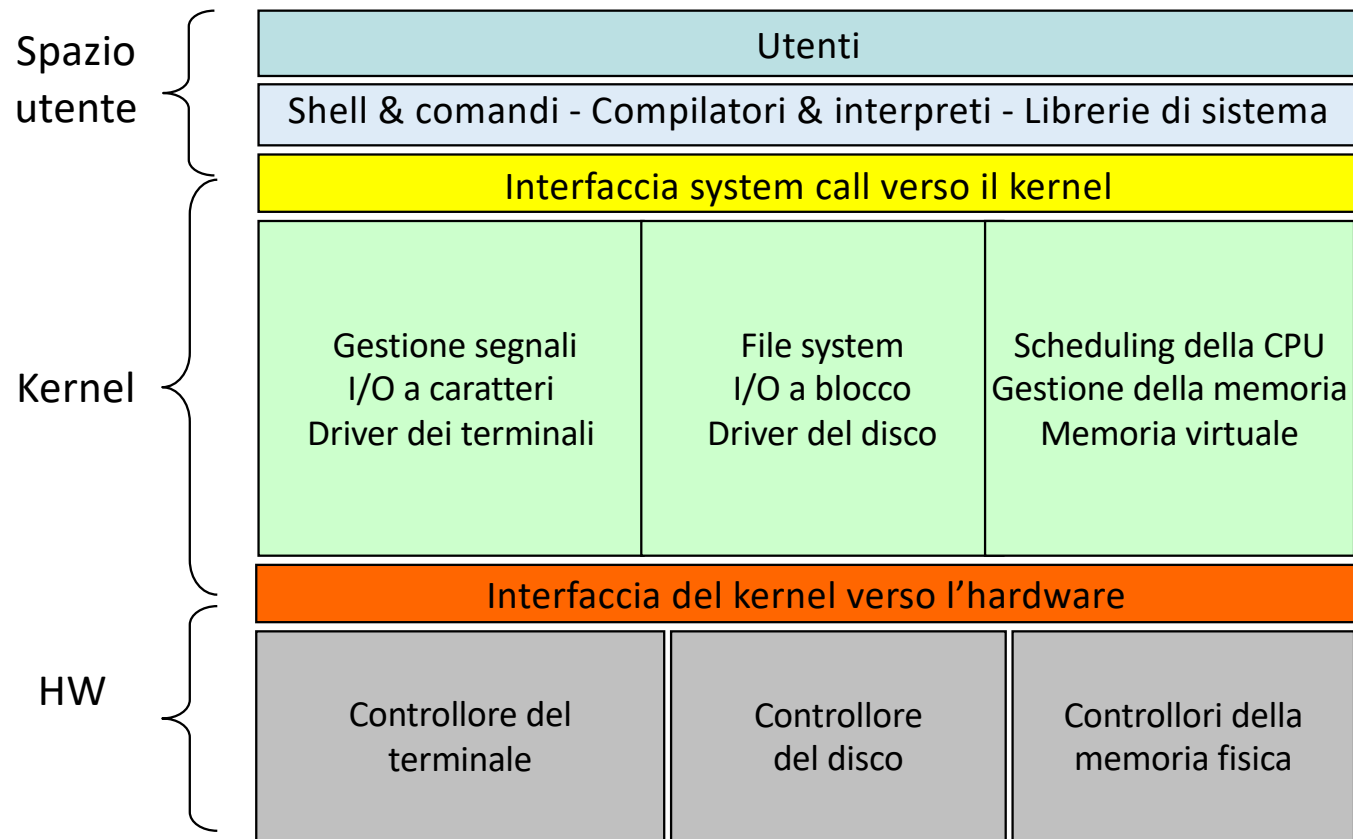
Struttura base
limitata a causa
delle limitate
funzionalità HW

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)



Sistemi a livelli

Servizi organizzati per livelli gerarchici

- Interfaccia utente (livello più alto) → ... → HW (livello 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

Es.: THE, MULTICS, OS/2

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

THE (Dijkstra 1968)

Sistema operativo
accademico per sistemi
batch

Primo esempio di sistema
a livelli

- Insieme di processi cooperanti
sincronizzati tramite semafori

Livello 5: Programmi utente

Livello 4: Gestione I/O

(astrazione dispositivi fisici)

Livello 3: Device driver della console

(comunicazione utente-console)

Livello 2: Gestione della memoria

(memoria virtuale senza supporto HW)

Livello 1: Scheduling della CPU

(con priorità, permette multiprogrammazione)

Livello 0: Hardware

Virtual machine

Estremizzazione dell'approccio a livelli (IBM VM 1972)

- Pensato per offrire un sistema timesharing “multiplo”

HW e S.O. trattati come hardware

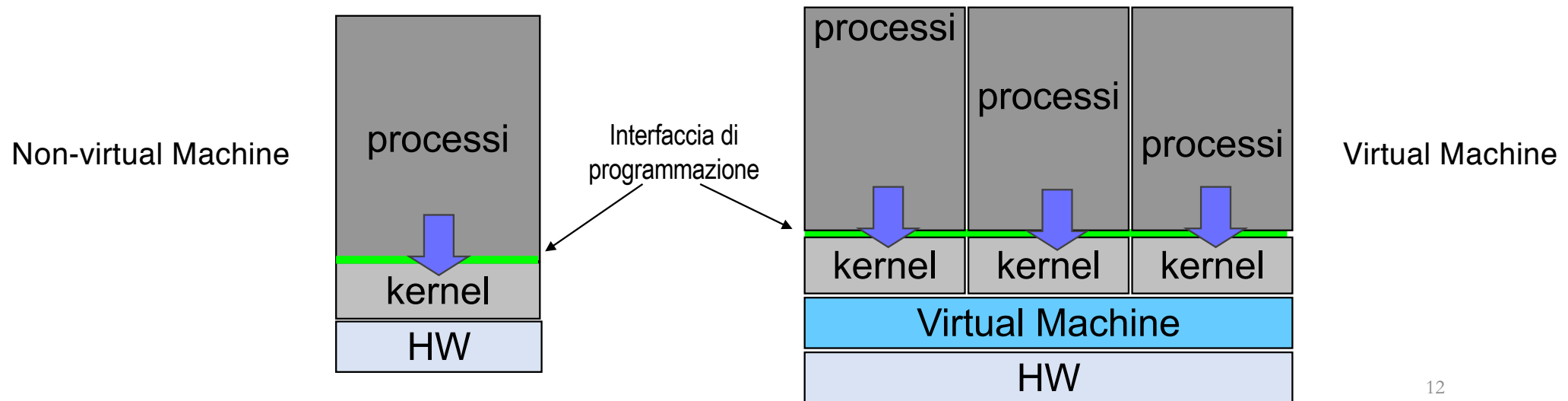
- Una VM fornisce un'interfaccia identica all'HW sottostante
- Il S.O. esegue sopra la VM
- La VM dà l'illusione di processi multipli, ciascuno in esecuzione sul proprio HW

Possibilità di presenza di più S.O.

Virtual machine

Concetto chiave → separazione di

- multiprogrammazione (Virtual Machine)
- presentazione (S.O.)



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
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:

condividere 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)
- Linux Docker

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

Sistemi client-server

Tutti i servizi del S.O. sono realizzati come processi utente (client)

Il client chiama un processo servitore (server) per usufruire di un servizio

Il server, dopo l'esecuzione, restituisce il risultato al client

Il kernel si occupa solo della gestione della comunicazione tra client e server

Il modello si presta bene per S.O. distribuiti

S.O. moderni realizzano tipicamente alcuni servizi in questo modo

Implementazione di un S.O.

