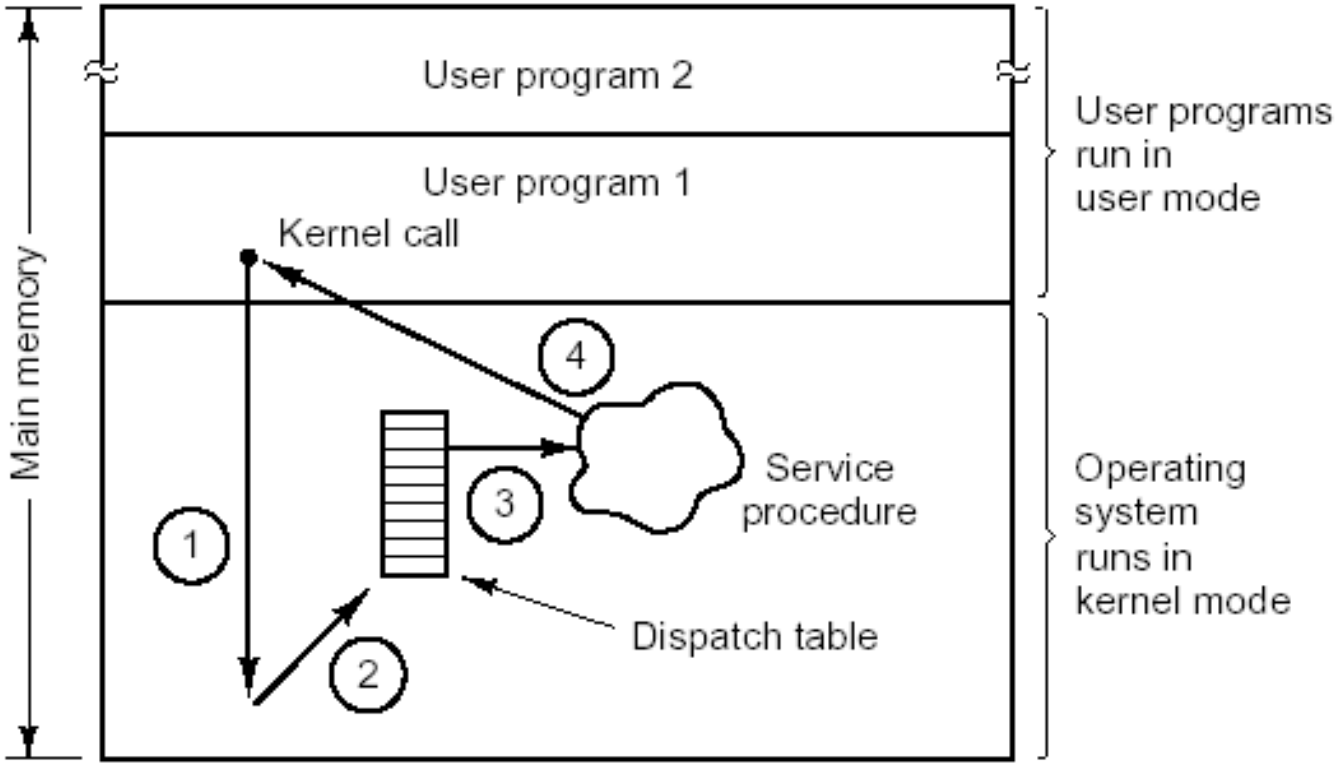
O.S. ARCHITECTURE

L'architettura del S.O. può essere

- ✓ **monolitica**, quando esso è composto da un unico modulo che serve le richieste dei programmi-utente una alla volta;
- ✓ **a macchina virtuale**, se esso offre a ciascun programma-utente visibilità di un particolare hardware
- ✓ **client-server**, se esso prevede un nucleo minimo di funzioni comuni (*microkernel*) a tutte le stazioni di un sistema distribuito, a cui alcune stazioni (*server*) aggiungono funzioni specifiche per offrire servizi ad altre stazioni (*client*).
- ✓ **a livelli**, se esso è articolato in diversi moduli, ciascuno dei quali svolge specifiche funzioni, ed ogni modulo può servire le richieste di più programmi-utente

MONOLITIC ARCHITECTURE OF AN O.S.

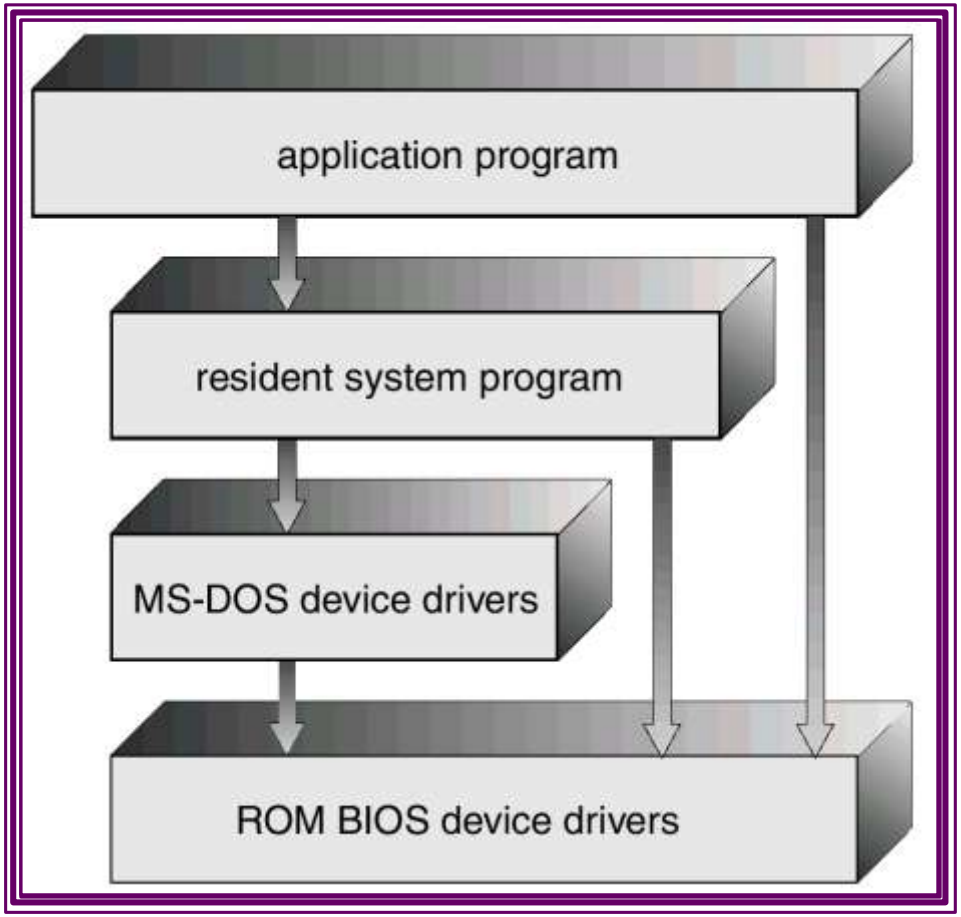


How a system call can be made:

(1) User program traps to the kernel.
(2) Operating system determines service number required.
(3) Operating system calls service procedure.
(4) Control is returned to user program.

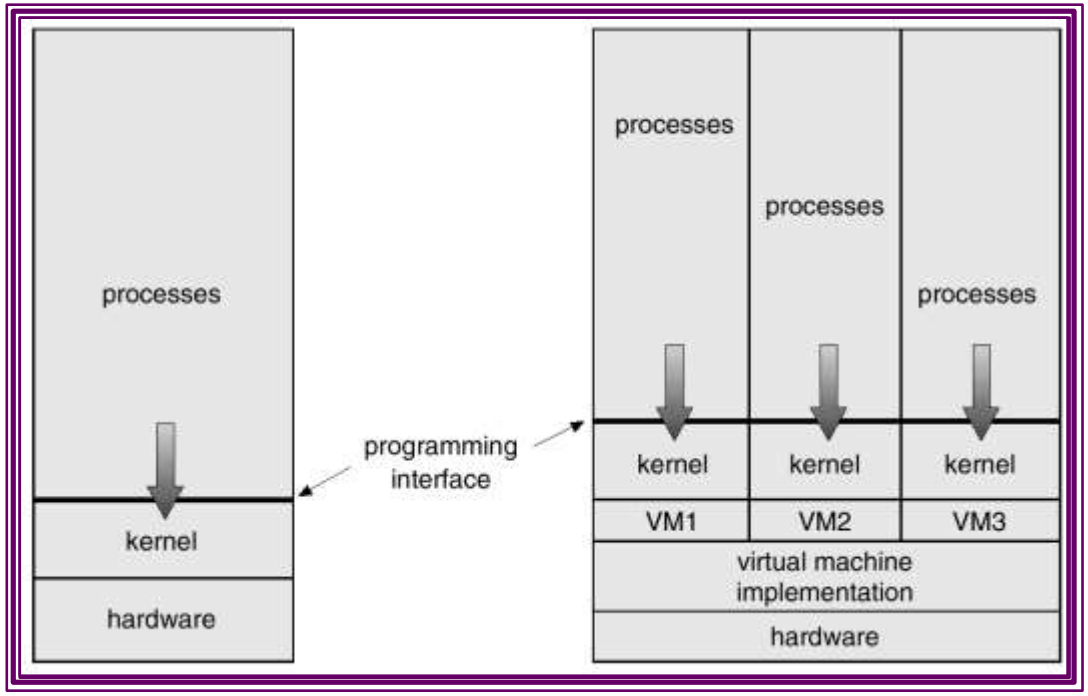
MONOLITIC ARCHITECTURE OF AN O.S.

- ✓ MS-DOS – written to provide the most functionality in the least space
  - Φ not divided into modules
  - Φ although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.



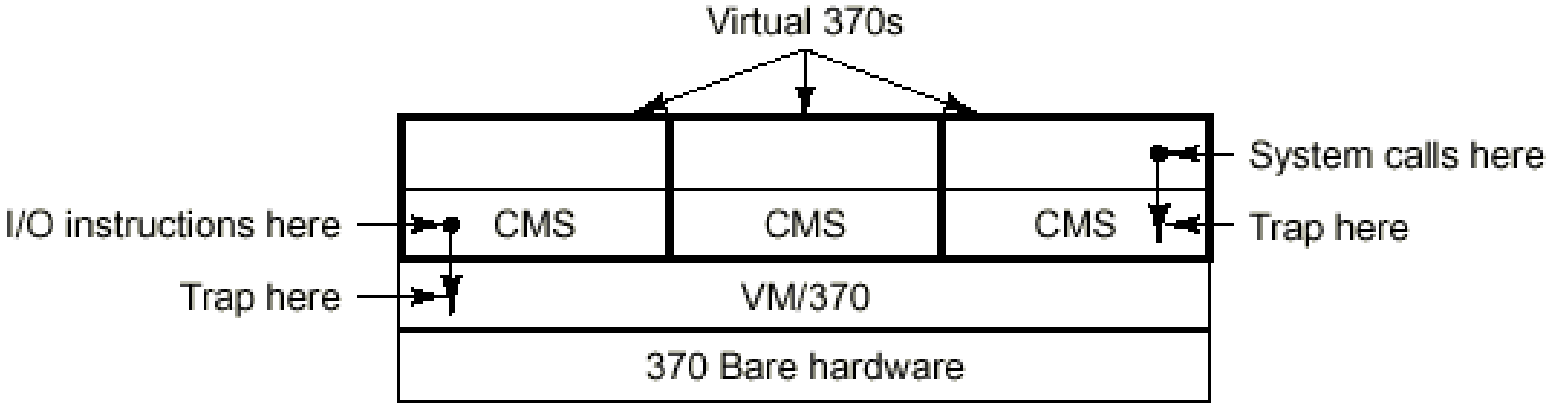
VIRTUAL MACHINE ARCHITECTURE OF AN O.S.

- ✓ *A virtual machine* takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.
- ✓ A virtual machine provides an interface *identical* to the underlying bare hardware.
- ✓ The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory.
- ✓ Quando un processo effettua una system call, questa viene tradotta in una trap al kernel della particolare VMi, come se la trap debba essere eseguita sulla macchina reale. Le istruzioni della trap vengono tradotte in istruzioni di I/O verso il kernel operante effettivamente (virtual machine implementation) sull'hardware sottostante.



VIRTUAL MACHINE PRO&CONTRO

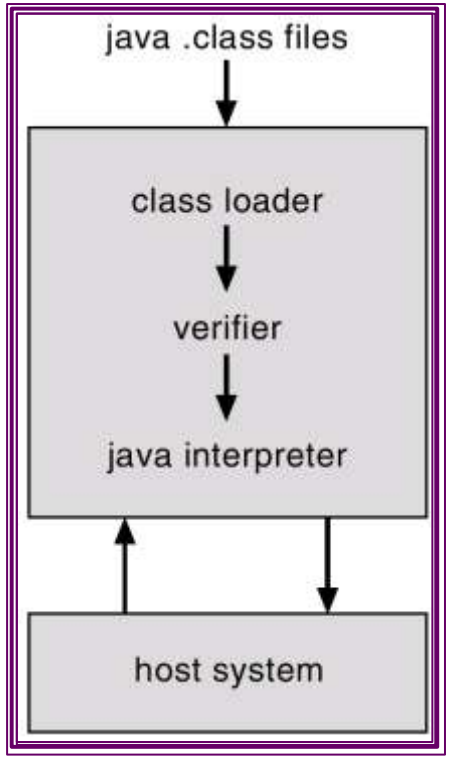
- ✓ The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- ✓ A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
- ✓ The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine. It may be necessary to emulate each single machine instruction.



The structure of VM/370 with CMS.

Java Virtual Machine (JVM)

- ✓ Compiled Java programs are platform-neutral bytecodes executed by a Java Virtual Machine (JVM).
- ✓ JVM consists of
  - Φ class loader
  - Φ class verifier
  - Φ runtime interpreter
- ✓ Just-In-Time (JIT) compilers increase performance



EXOKERNEL

- Estensione dell'idea di macchina virtuale
- Ogni macchina virtuale di livello utente vede solo un sottoinsieme delle risorse dell'intera macchina
- Ogni macchina virtuale può eseguire il proprio sistema operativo
- Le risorse vengono richieste all'exokernel, che tiene traccia di quali risorse sono usate e da chi
- Semplifica l'uso delle risorse allocate: l'exokernel deve solo tenere separati i domini di allocazione delle risorse

MECCANISMI E POLITICHE

I kernel tradizionali (monolitici) sono poco flessibili

Distinguere tra meccanismi e politiche:
- i ***meccanismi*** determinano ***come fare*** qualcosa;
- le ***politiche*** determinano ***cosa deve essere fatto***.

Ad esempio:
- ***assegnare*** l'esecuzione ad un processo è un ***meccanismo***;
- ***scegliere*** quale processo attivare è una ***politica.***

Questa separazione è un principio molto importante: permette la massima flessibilità, nel caso in cui le politiche debbano essere cambiate.

Estremizzazione:
il ***kernel fornisce solo i meccanismi***, mentre le ***politiche vengono implementate in user space***.

SISTEMI CON MICROKERNEL

Microkernel: il ***kernel è ridotto all'osso***, ***fornisce soltanto i meccanismi***:

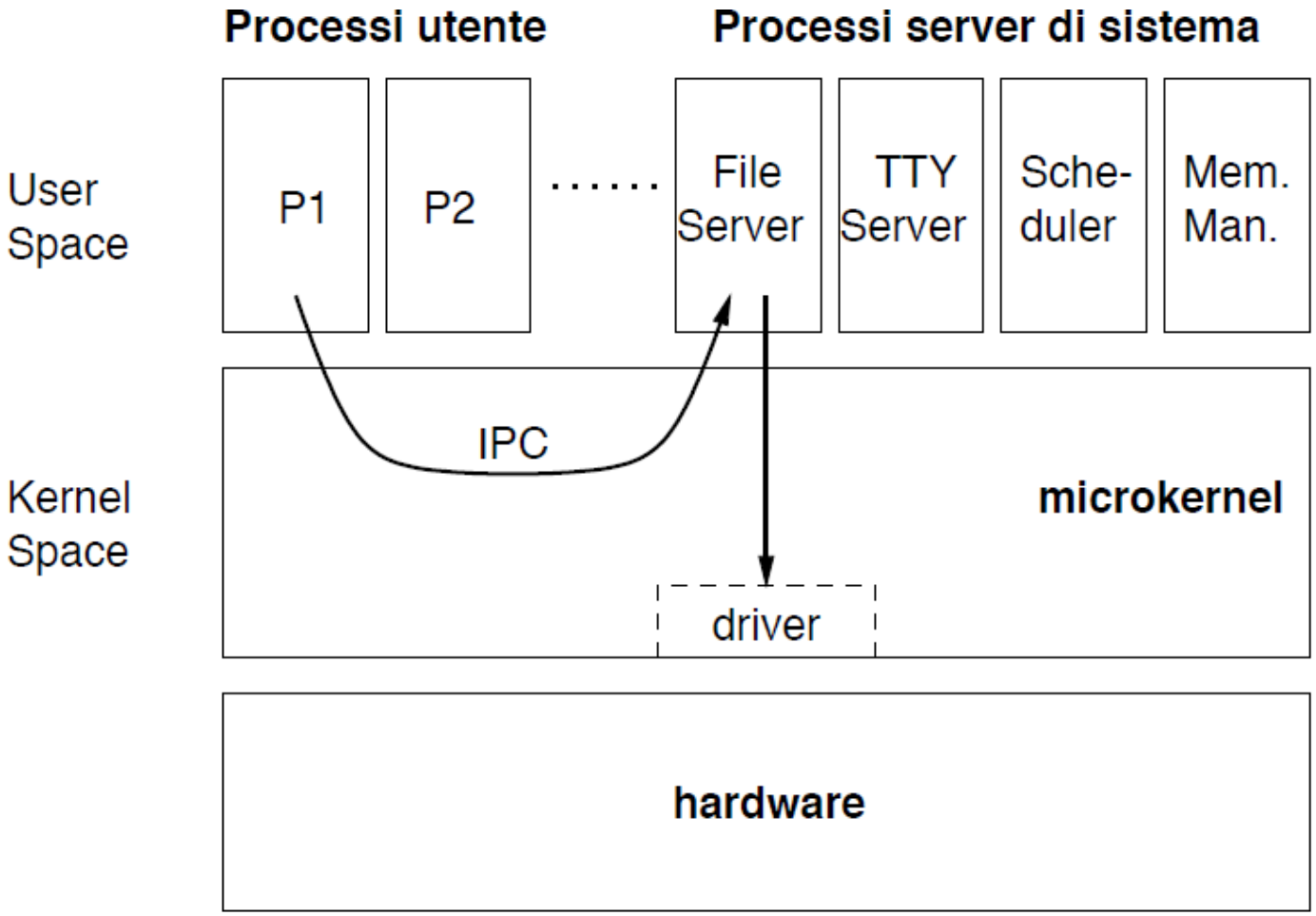
- Un *meccanismo di comunicazione* tra processi
- Una *minima gestione* della memoria e dei processi
- Gestione dell'hardware di basso livello (*driver*)

Tutto il resto viene gestito da processi in spazio utente: ad esempio, tutte le politiche di gestione del file system, dello scheduling, della memoria sono implementate come processi.

Meno efficiente del kernel monolitico
Grande flessibilità; immediata scalabilità in ambiente di rete

I sistemi operativi recenti sono basati, in diverse misure, su microkernel (AIX4, BeOS, GNU HURD, MacOS X, QNX, Tru64, Windows NT . . .)

MICROKERNEL: FUNZIONAMENTO DI BASE



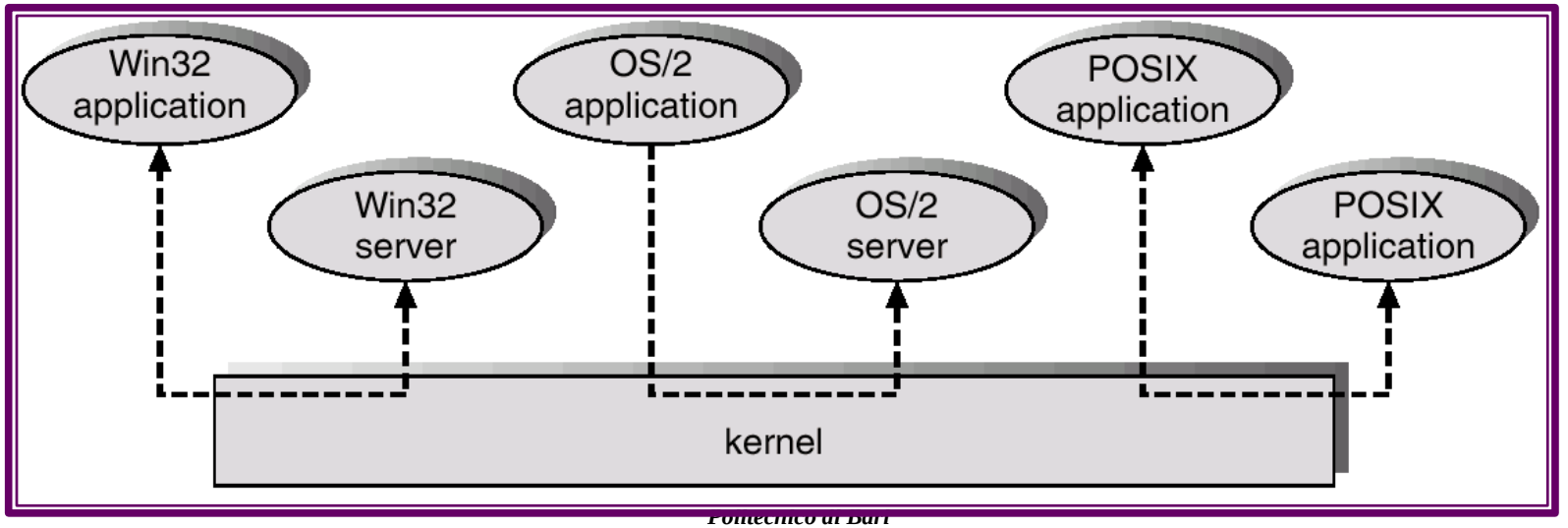
IPC → Inter Process Communication facility

CLIENT-SERVER ARCHITECTURE OF AN O.S.

The microkernel

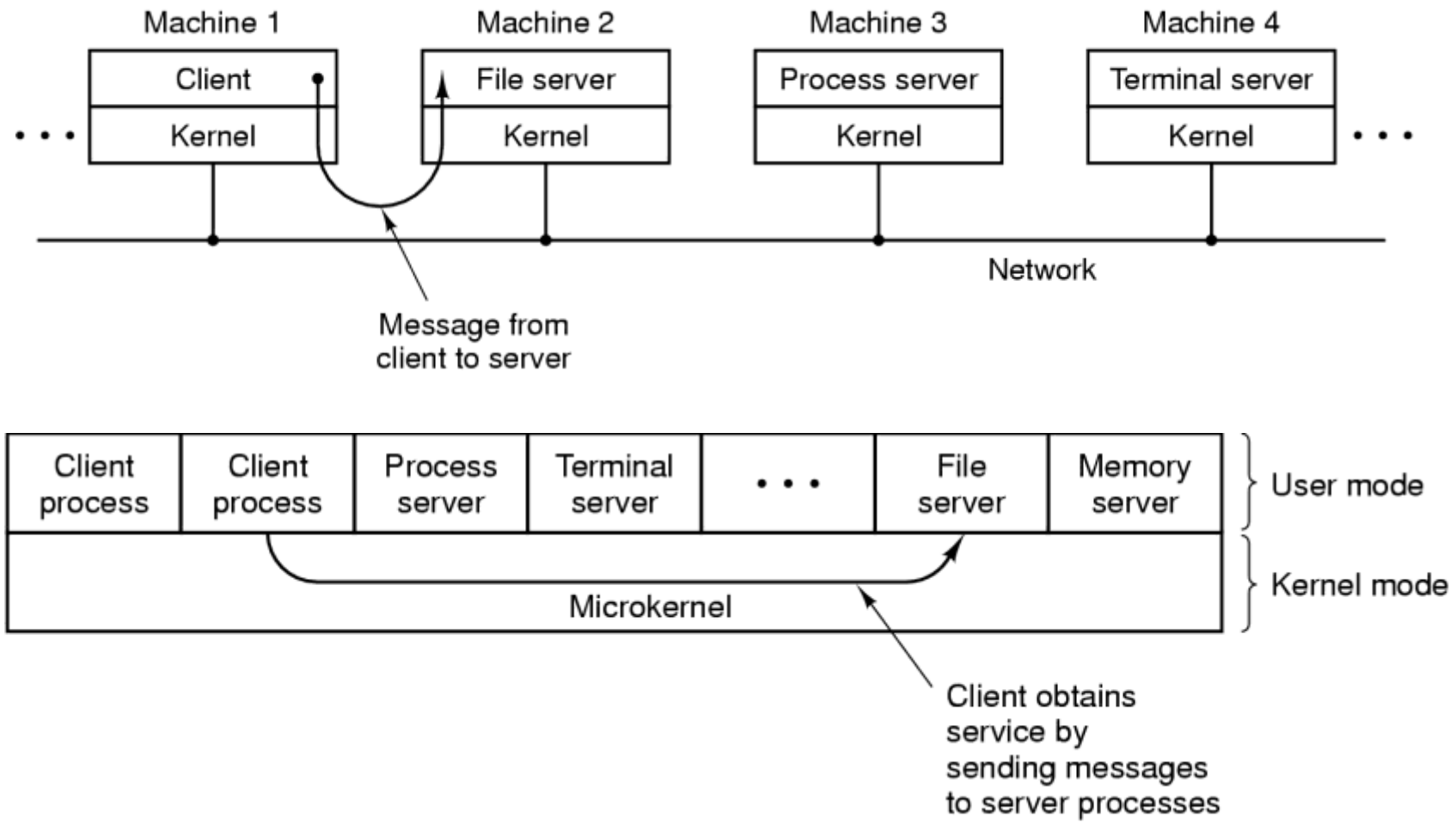
- ✓ Moves as much from the kernel into "*user*" space.
- ✓ Communication takes place between user modules using message passing.
- ✓ Benefits:
  - easier to extend a microkernel
  - easier to port the operating system to new architectures
  - more reliable (less code is running in kernel mode)
  - more secure

Windows NT client-server architecture



CLIENT-SERVER ARCHITECTURE OF AN O.S.

General model



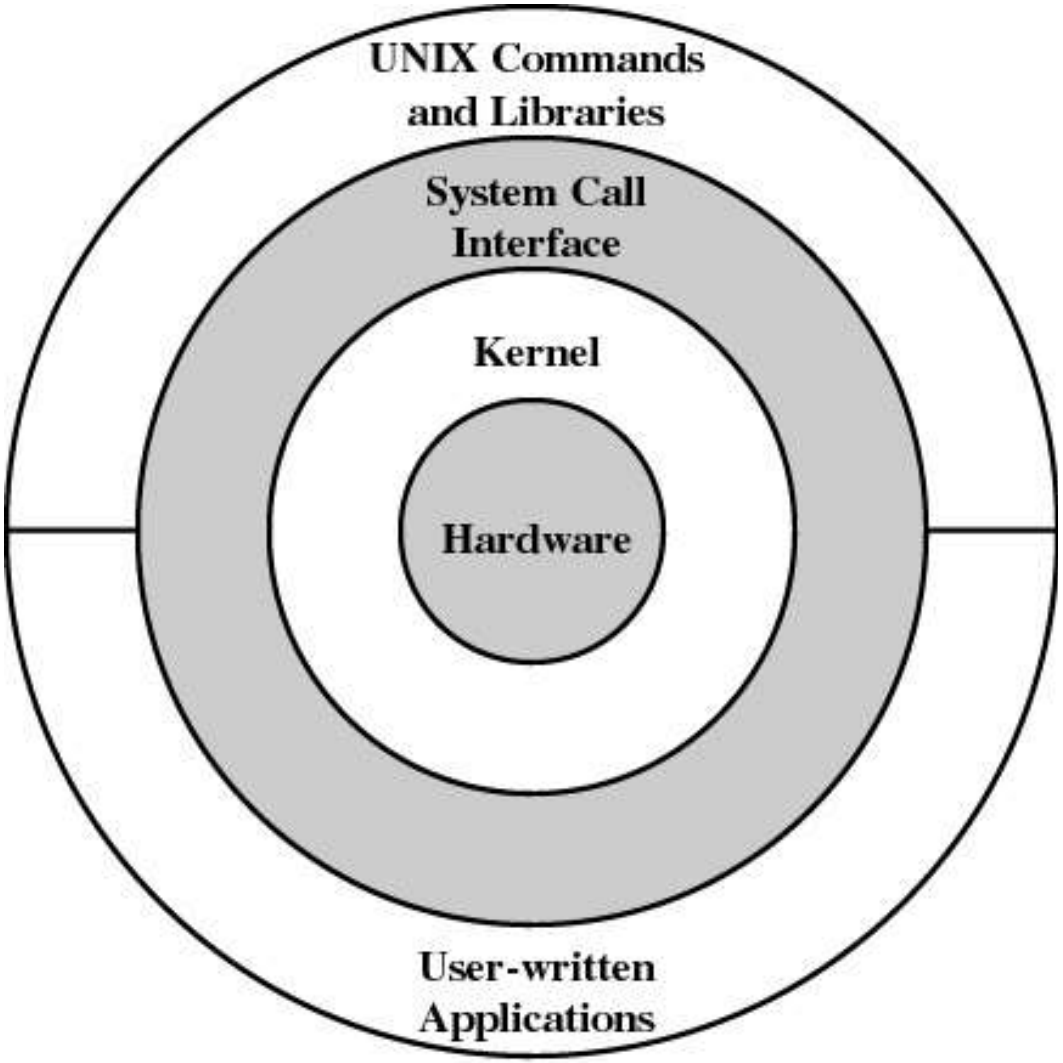
LAYERED ARCHITECTURE OF AN O.S.

- ✓ The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- ✓ With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.

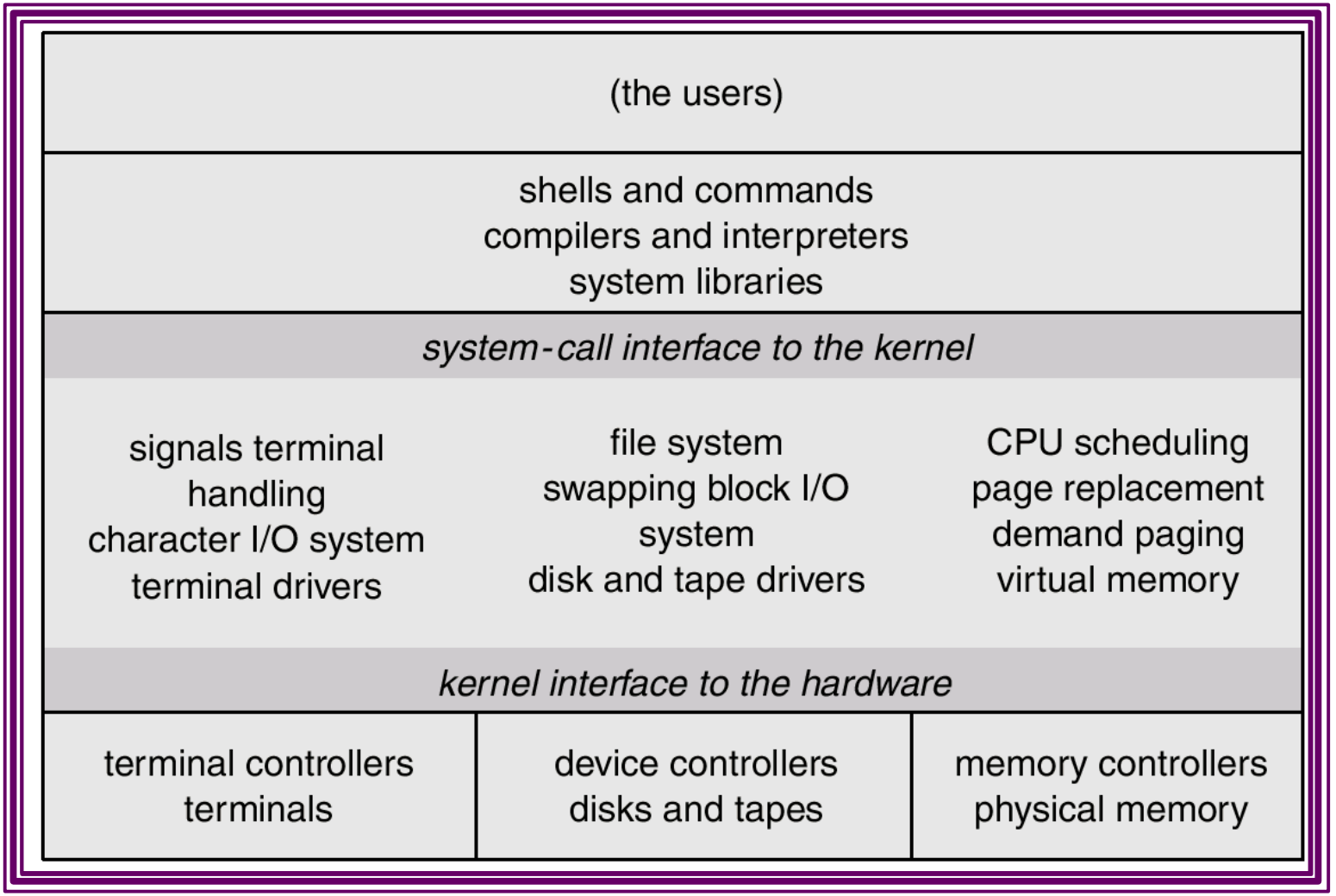
The original UNIX operating system had limited structuring. It consists of two separable parts.

- Φ Systems programs
- Φ The ***kernel***
  - consists of everything below the system-call interface and above the physical hardware
  - provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

LAYERED ARCHITECTURE OF UNIX

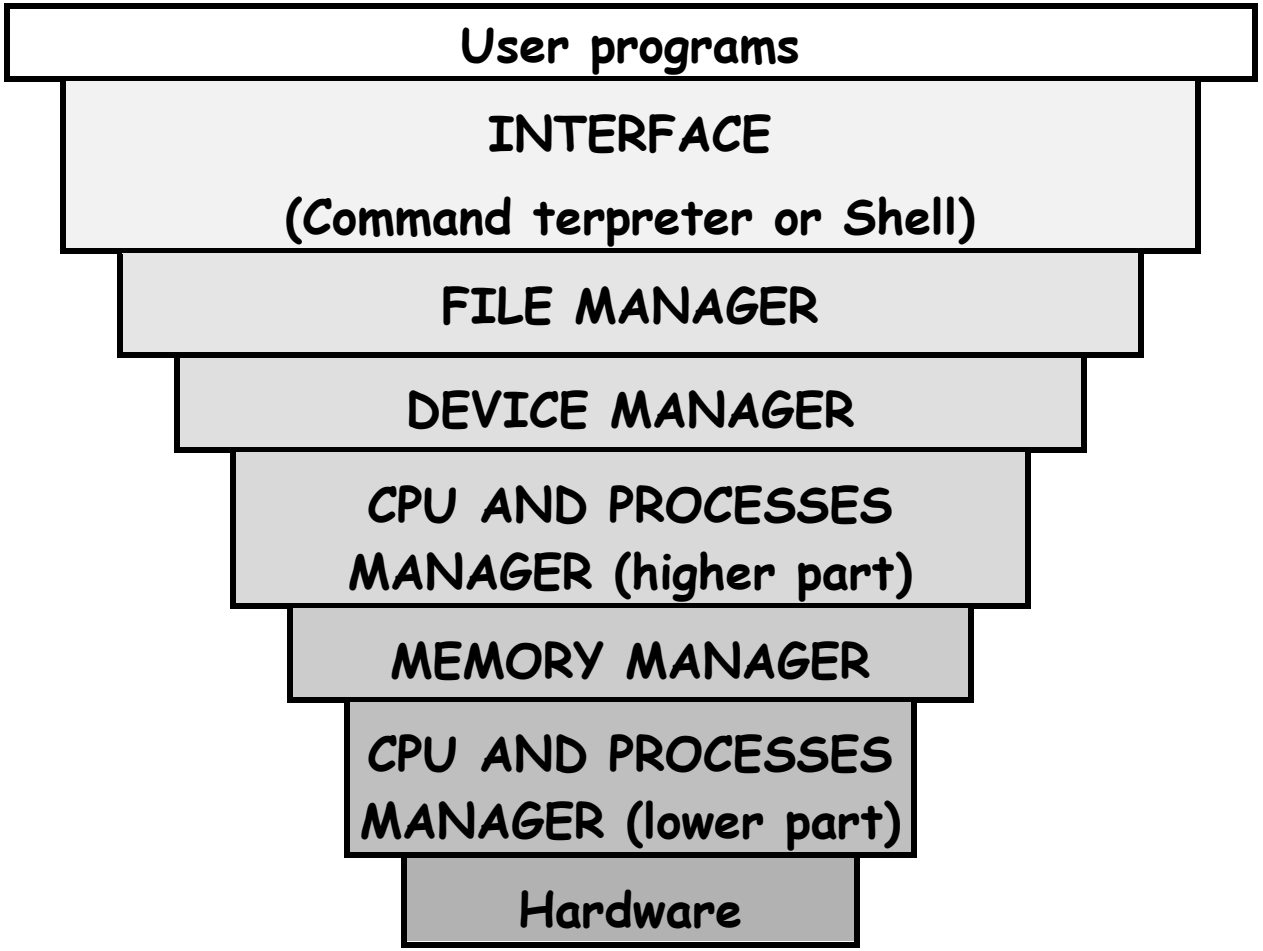


LAYERED ARCHITECTURE OF UNIX



GENERAL MODEL OF AN O.S. LAYERED ARCHITECTURE

The general model



GENERAL MODEL OF AN O.S. LAYERED ARCHITECTURE

General model

L'Interfaccia comprende il Job Scheduler.

I livelli tra l'Interfaccia e l'Hardware costituiscono il **Nucleo** o **kernel**.

I moduli di un livello possono utilizzare moduli dei livelli sottostanti mediante chiamate al Supervisore o ***primitive*** sincrone (***trap*** o ***interrupt interni)***.

Ogni livello garantisce la gestione (*management*) di una risorsa e comprende tutti i moduli che contribuiscono a svolgere tale compito (*resource manager*).

I *resource manager* sono:

- il gestore dei ***file***
- il gestore dei ***dispositivi***
- il gestore della ***CPU*** e dei ***processi***
- il gestore della ***memoria centrale***

Il Manager della CPU e dei processi (o Process Scheduler) è suddiviso in due parti:

- la parte alta comprende i moduli per la ***creazione/distruzione di processi*** e per la ***comunicazione tra processi***;
- la parte bassa comprende i moduli per la ***sincronizzazione di processi***.

CPU AND PROCESS MANAGEMENT

The operating system is responsible for the following activities connected with CPU management:

- ✓ Keep track of which process is currently using CPU.
- ✓ Decide to which process (among those in the ready status) to assign CPU.
- ✓ Allocate and deallocate CPU according to the process priorities.

The operating system is responsible for the following activities in connection with process management.

- Φ Process creation and deletion.
- Φ Process suspension and resumption.
- Φ Provision of mechanisms for:
  - ☞ process synchronization
  - ☞ process communication
  - ☞ avoid, prevent and solve deadlocks

MAIN MEMORY MANAGEMENT

- ✓ Main memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- ✓ RAM main memory is a volatile storage device. It looses its contents in the case of system failure.
- ✓ The operating system is responsible for the following activities in connections with memory management:
  - Φ Keep track of which parts of memory are currently being used and by whom.
  - Φ Decide which processes to load when memory space becomes available.
  - Φ Allocate and deallocate memory space as needed.

SECONDARY STORAGE DEVICES

- ✓ Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
- ✓ Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- ✓ The operating system is responsible for the following activities in connection with ***disk management***:
  - Free space management
  - Storage allocation
  - Disk scheduling

FILE MANAGEMENT

- ✓ A **file** is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- ✓ The operating system is responsible for the following activities for ***file management***:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

DEVICE MANAGEMENT

- ✓ The ***I/O system*** consists of:
  - Φ *A buffer-caching system*
  - Φ *A general device-driver interface*
  - Φ *Drivers for specific hardware devices* (secondary storage, printers, tablets, ecc.)

SISTEMI DI PROTEZIONE

Per **Protezione** si intende un ***meccanismo per controllare l'accesso*** da programmi, processi e utenti sia al sistema, sia alle risorse degli utenti.

Il meccanismo di protezione deve:

- distinguere tra *uso autorizzato e non autorizzato*
- fornire un *modo per specificare i controlli* da imporre
- forzare gli utenti e i processi a *sottostare ai controlli richiesti*

NETWORKING (SISTEMI DISTRIBUITI)

Un **sistema distribuito** è una *collezione di processori* che non condividono memoria o clock. Ogni processore ha una memoria propria.

I processori del sistema sono *connessi attraverso una rete di comunicazione*.

Un sistema distribuito fornisce agli utenti l'*accesso a diverse risorse di sistema*.

L'*accesso ad una risorsa condivisa* permette:

- Aumento delle prestazioni computazionali
- Incremento della quantità di dati disponibili
- Aumento dell'affidabilità

INTERPRETE DEI COMANDI

Molti comandi sono dati al sistema operativo attraverso ***control statement*** che servono per:

- creare e gestire i processi
- gestione dell'I/O
- gestione della memoria secondaria
- gestione della memoria principale
- accesso al file system
- protezione
- networking

INTERPRETE DEI COMANDI (Cont.)

Il programma che legge e interpreta i comandi di controllo ha diversi nomi:

- ***interprete delle schede di controllo*** (sistemi batch)
- ***interprete della linea di comando*** (DOS, Windows)
- ***shell*** (in UNIX)
- ***interfaccia grafica***: Finder in MacOS, Explorer in Windows, gnome-session in Unix. . .

La sua funzione è di ricevere un comando, eseguirlo, e ripetere.

SERVIZI DEI SISTEMI OPERATIVI

- ***Esecuzione dei programmi***: caricamento dei programmi in memoria ed esecuzione.
- ***Operazioni di I/O***: il sistema operativo deve fornire un modo per condurre le operazioni di I/O, dato che gli utenti non possono eseguirle direttamente,
- Manipolazione del ***file system***: capacità di creare, cancellare, leggere, scrivere file e directory.
- ***Comunicazioni***: scambio di informazioni tra processi in esecuzione sullo stesso computer o su sistemi diversi collegati da una rete. Implementati attraverso memoria condivisa o passaggio di messaggi.
- ***Individuazione di errori***: garantire una computazione corretta individuando errori nell'hardware della CPU o della memoria, nei dispositivi di I/O, o nei programmi degli utenti.

FUNZIONALITÀ ADDIZIONALI DEI SISTEMI OPERATIVI

Le funzionalità addizionali esistono per assicurare l'efficienza del sistema, piuttosto che per aiutare l'utente

- ***Allocazione delle risorse***: allocare risorse a più utenti o processi, allo stesso momento
- ***Accounting***: tener traccia di chi usa cosa, a scopi statistici o di rendicontazione
- ***Protezione***: assicurare che tutti gli accessi alle risorse di sistema siano controllate

CHIAMATE DI SISTEMA (System Calls)

Le chiamate di sistema formano l'*interfaccia tra un programma in esecuzione e il sistema operativo*.

Generalmente, sono disponibili come speciali istruzioni assembler

Linguaggi pensati per *programmazione di sistema* permettono di eseguire system call direttamente (e.g., C, Bliss, PL/360).

Tipi di chiamate di sistema

Controllo dei processi: creazione/terminazione processi, esecuzione programmi, (de)allocazione memoria, attesa di eventi, impostazione degli attributi,.

. . . . .

Gestione dei file: creazione/cancellazione, apertura/chiusura, lettura/scrittura, impostazione degli attributi,. . .

Gestione dei dispositivi: allocazione/rilascio dispositivi, lettura/scrittura, collegamento logico dei dispositivi (e.g. mounting). . .

Informazioni di sistema: leggere/scrivere data e ora del sistema, informazioni sull'hardware/software installato,. . .

Comunicazioni: creare/cancellare connessioni, spedire/ricevere messaggi,. . .