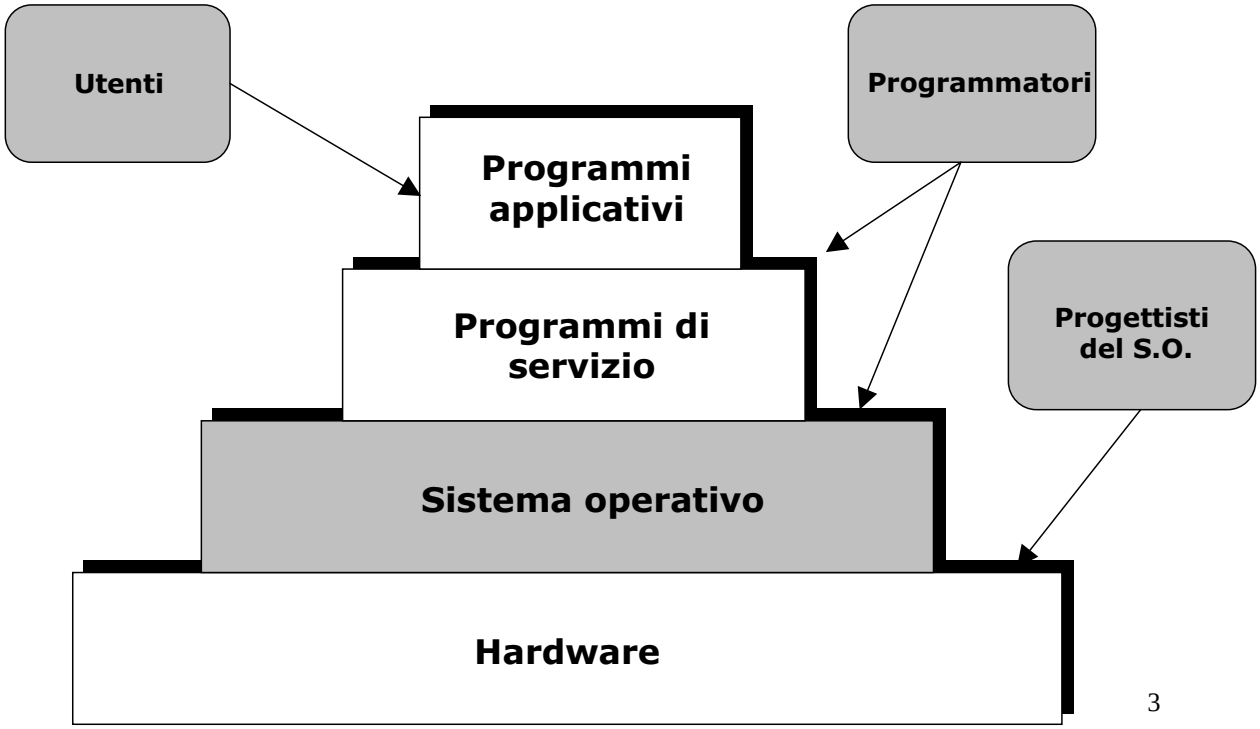
CALCOLATORI ELETTRONICI II

A.A 2001-2002

Maurelio Boari

Obiettivi e funzioni di un Sistema Operativo

Livelli di un S.O.



Funzioni di un Sistema Operativo

• Un Sistema Operativo (S.O.) è un insieme di programmi che operano sull'hardware di un calcolatore con l'obiettivo di

- rendere più semplice ed efficace lo sviluppo di programmi
- realizzare politiche di gestione delle risorse hardware

Rendere più semplice ed efficace lo sviluppo di programmi

• Disponibilità di apposite operazioni (system calls) per l'utilizzo delle risorse HW (es. dispositivi di I/O)

• Definizione di una macchina estesa (o **virtuale**)

• Indipendenza del software applicativo da modifiche apportate all'hardware (trasparenza)

Macchina virtuale

Es: controllore di un floppy disk

• numerosi comandi: *lettura, scrittura, movimento del braccio, formattazione delle tracce, etc.*

• ogni comando ha più parametri: *indirizzo del blocco, numero di settori per traccia, etc.*

• numerose condizioni di stato e di errore al completamento del comando.

Realizzare politiche di gestione delle risorse hardware

- Regolamentazione dell'impiego delle risorse evitando conflitti di accesso.

- Scelta dei criteri con cui assegnare una risorsa a fronte di più richieste contemporanee (limitazione delle risorse)

- Necessità di nascondere all'utente i dettagli hardware legati al particolare dispositivo
- Uso di apposite **system call**

Classificazione dei Sistemi Operativi

Organizzazione interna

- monoprogrammati
- multiprogrammati
- a divisione di tempo

Visibilità utente

- sistemi a lotti (batch)
- sistemi interattivi
- sistemi dedicati
- sistemi in tempo reale
- sistemi distribuiti

Evoluzione dei Sistemi Operativi

Evoluzione dei S.O.

- **Elaborazione seriale**
  - Assenza di sistema operativo
  - Macchine controllate da una console con display luminosi, interruttori a chiavetta, dispositivi di ingresso e una stampante
  - Prenotazione del tempo di macchina
  - Tempo di preparazione include il caricamento del compilatore,del programma sorgente, il salvataggio del programma compilato, il caricamento ed il linking

Evoluzione dei S.O.

- **Semplici sistemi batch**
  - Monitors
    - Software che controlla l'esecuzione dei programmi
    - Programmi raggruppati in "batch" dall'operatore prima della lettura
    - Al termine dell'esecuzione di un programma il controllo ritorna al monitor
    - Monitor residente in memoria centrale e disponibile per l'esecuzione

Job Control Language (JCL)

- Particolare tipo di linguaggio di programmazione
- Fornisce istruzioni al monitor
  - Quale compilatore usare
  - Quali dati usare

Esempio:

```
$JOB
$FTN

<istruzioni FORTRAN>

$LOAD
$RUN

<Dati>

$END
```

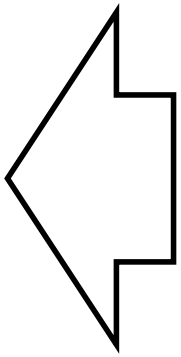
Hardware

- **Protezione della memoria**
  - L'area di memoria contenente il monitor non può essere alterata
- **Timer**
  - Impedisce ad un job di monopolizzare il sistema
- **Istruzioni privilegiate**
  - Eseguibili solo dal monitor. Es: Istruzioni di I/O
- **Interruzioni**

Classificazione dei Sistemi Operativi

Organizzazione interna

- monoprogrammati
- multiprogrammati
- a divisione di tempo

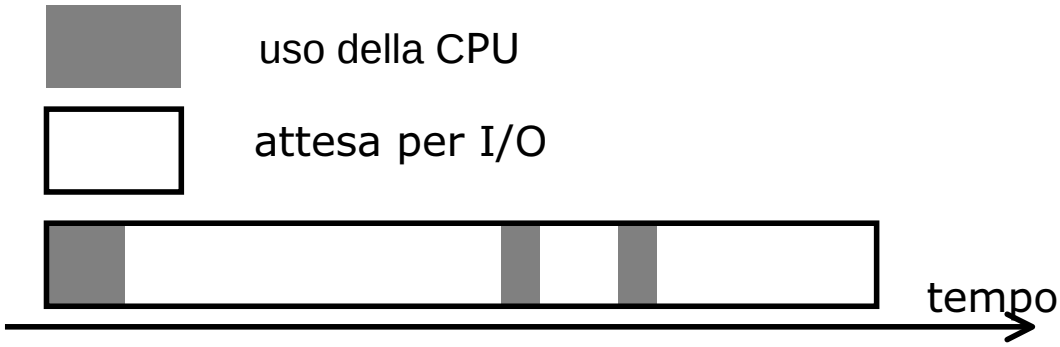


Sistemi Monoprogrammati

• In memoria centrale risiede, oltre al sistema operativo, al più un programma applicativo.

• Tutte le risorse hardware e software del sistema sono dedicate ad un solo programma (**sistema monoutente**).

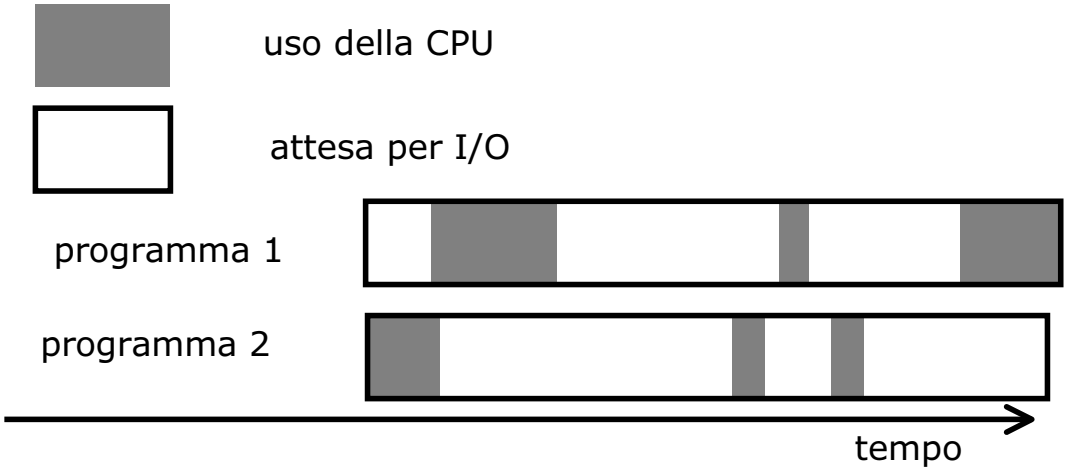
• **Bassa utilizzazione** della CPU



Sistemi Multiprogrammati

• Gestione **contemporanea** di più programmi indipendenti presenti nella memoria principale (**sistema multi- utente**)

• Migliore utilizzazione delle risorse (riduzione dei tempi morti)

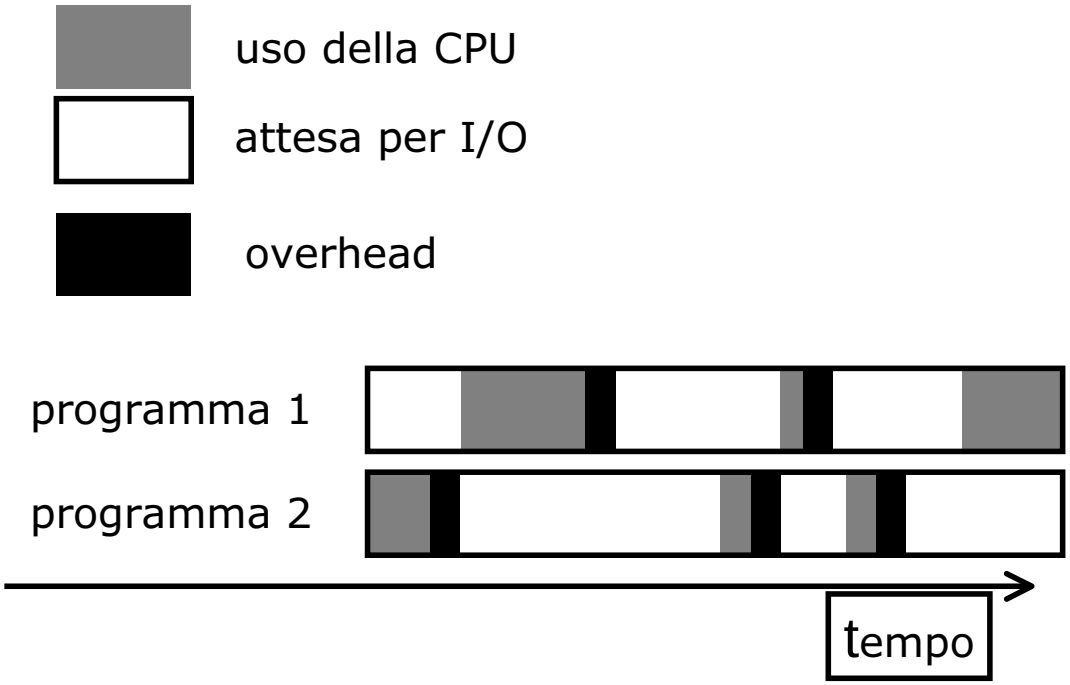


- maggiore complessità del Sistema Operativo**:**
- algoritmi per la gestione delle risorse (CPU, memoria gestione delle, I/O)
- protezione degli ambienti dei diversi programmi

Sistemi a partizione di tempo (time - sharing)

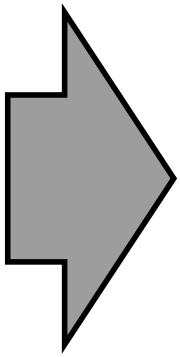
- Estensione della multiprogrammazione
- Ad ogni programma il S.O. assegna ciclicamente un intervallo (**quantum**) di tempo della CPU, fino al suo completamento.
- Al termine dell'intervallo (o durante, se il programma inizia un'operazione di I/O) la CPU viene assegnata ad un altro programma (**politica Round-Robin**).
- Tempo impiegato dal S.O. per trasferire il controllo da un programma ad un altro (**overhead**).

Overhead



Classificazione dei Sistemi Operativi

Visibilità utente



- sistemi a lotti (batch)
- sistemi interattivi
- sistemi dedicati
- sistemi in tempo reale
- sistemi distribuiti

Sistemi batch

• I programmi sono inseriti a lotti nella memoria di massa e successivamente elaborati in **multiprogrammazione**

• Obiettivo: migliore utilizzazione delle risorse (elevato t**hroughput**)

• Scelta dell'insieme di programmi (***job mix***) in memoria principale che ottimizzano l'utilizzo delle risorse

• Obiettivo: **tempi di risposta** accettabili per l'utente

• Utilizzo della tecnica **time - sharing**

Sistemi interattivi

• Più utenti, ognuno collegato ad un proprio terminale, utilizzano contemporaneamente il sistema di elaborazione.

• Obiettivo: **tempi di risposta** accettabili per l'utente.

• Utilizzo della tecnica **time sharing.**

Sistemi dedicati

• Le risorse del calcolatore sono dedicate ad una specifica applicazione (**embedded systems**). Es:elettronica di consumo (centrali elettroniche delle automobili, etc.)

• Sistema operativo semplice: interpreta i comandi, gestisce la memorizzazione dei dati sui file, attiva i programmi applicativi.

• Programmatore: conoscenza dell'hardware (non c'è virtualizzazione delle risorse da parte del S.O)

Sistemi in tempo reale

• Il sistema di calcolo deve rispondere entro un limite di tempo (**tempo di reazione**)

• *Tempo reale stretto* (**hard real-time**): il tempo risposta è molto stringente come vincolo e non può non essere rispettato (es: processi industriali)

• *Tempo reale debole* (**soft real-time**): il tempo di risposta deve essere ragionevole (es: sistema interattivo per la prenotazione voli)

• Caratteristica del S.O.: garantire il **rispetto dei vincoli temporali**

Sistemi transazionali

Sistemi di **tipo interattivo** usati in applicazioni di tipo gestionale caratterizzate da

- natura delle operazioni compiute: interrogazione e aggiornamento di archivi.
- criticità delle operazioni: necessità di evitare malfunzionamenti che pregiudichino la correttezza dei dati (es: transazioni bancarie)
- elevato numero di potenziali utenti e dislocazione geografica del sistema (es: prenotazioni aeree)

Transazione (sistemi transazionali)

insieme di operazioni che deve soddisfare a quattro proprietà:

ATOMICITÀ (Atomicity)

CONSISTENZA (Consistency)

ISOLAMENTO (Isolation)

DURATA (Durability)

ACID

• **atomicità**: azioni indivisibili anche in presenza di altre azioni eseguite contemporaneamente sugli stessi archivi

• **isolamento:** indipendenza fra le transazioni

• **la consistenza**: due soli modi per terminare transazione: in modo corretto, con la modifica persistente dei dati (commit), o con insuccesso, con il ripristino dello stato iniziale dei dati (abort)

• **durata**: una transazione andata a buon fine produce effetti permanenti: possono essere modificati da altre transazioni, ma non da malfunzionamenti

STRUTTURA DEI SISTEMI OPERATIVI

Componenti del Sistema Operativo

Chiamate di sistema

Organizzazione di un Sistema Operativo

Componenti di un sistema operativo

Gestore dei processi

Gestori della memoria principale e secondaria

Gestore dei dispositivi di I/O

Gestore dei file

Sistema di protezione e sicurezza

Gestione della comunicazione tra sistemi distribuiti

Interprete dei comandi

Processi

- Un programma in esecuzione
- Un' istanza di un programma in esecuzione
- L'entità che può essere assegnata a ed eseguita da un processore
- Un'unità di attività caratterizzata da un flusso esecuzione sequenziale, da uno stato corrente e con associato un insieme di risorse di di sistema

Difficoltà nella progettazione di software di sistema

- **Sincronizzazione impropria**
  - Processo in attesa di un segnale di I/O deve ricevere il segnale dal dispositivo.
- **Fallimento della mutua esclusione**
  - Operazioni del programma non determinate; un programma dovrebbe dipendere solo dai dati di ingresso e non dall'attività degli altri programmi
- **Stallo (deadlock)**

Gestione della memoria

- Isolamento dei processi
- Allocazione e gestione automatica della memoria
- Supporto per la programmazione modulare
- Protezione e controllo degli accessi
- Memorizzazione a lungo termine

Memoria virtuale

- Consente ai programmi di indirizzare la memoria da un punto di vista logico (senza preoccuparsi della quantità di memoria fisicamente disponibile).
- Quando un programma è in esecuzione solo una parte del programma e dei dati può risiedere nella memoria centrale: altre porzioni del programma e dei dati sono mantenute nella memoria di massa.

Impaginazione

• La memoria fisica è suddivisa in un numero fissato di blocchi di ampiezza fissata chiamate **pagine.**

• Un indirizzo virtuale è rappresentato da un numero di pagina e da un offset entro la pagina.

• Ogni pagina può essere allocata in un punto qualunque della memoria fisica

• Differenza tra indirizzi logici e indirizzi fisici

Protezione dell'informazione e sicurezza

• **Controllo dell'accesso**
Regola l'accesso degli utenti al sistema

• **Controllo del flusso di informazione**
Regola il flusso dei dati nel sistema e verso gli utenti

• **Certificazione**
Verifica che i meccanismi di controllo dell'accesso e del flusso vengano eseguiti in accordo alle specifiche

Schedulazione e gestione delle risorse

- **Equità**

Tutti i processi della stessa classe devono avere la stessa possibilità di accesso alla risorsa per la quale competano

- **Tempo di risposta differenziale**

Discriminare tra diverse classi di processi nell'assegnazione delle risorse

- **Efficienza**

Massimizzare il *throughput*, minimizzare il tempo di risposta e soddisfare il maggior numero di utenti contemporaneamente

Chiamate di sistema (System Calls)

• I programmi di utente comunicano con il Sistema Operativo e richiedono ad esso particolari servizi tramite **system calls**

• Esempi di system call:

lettura e scrittura dei file, operazioni di I/O,

allocazione della memoria, gestione del tempo,

invio di messaggi etc.

• Possono essere direttamente utilizzate in un **linguaggio** ad alto livello (Es. C in UNIX).

• Sopra il livello delle system call è collocata una collezione di **funzioni di libreria** (interfaccia C per le system call)

Applicazioni C in UNIX

• Ogni applicazione C può richiedere l'esecuzione di una system call attraverso una chiamata alla specifica funzione C di libreria che la rappresenta.

• La procedura di libreria:

  • Memorizza i parametri delle chiamate

  • Trasferisce il controllo al S.O. (INT) per l'esecuzione del servizio richiesto

  • Ritorna il controllo al programma chiamante, trasferendo il *codice di stato* (o altri parametri)

Esempio (Unix): READ System Call

programma in C

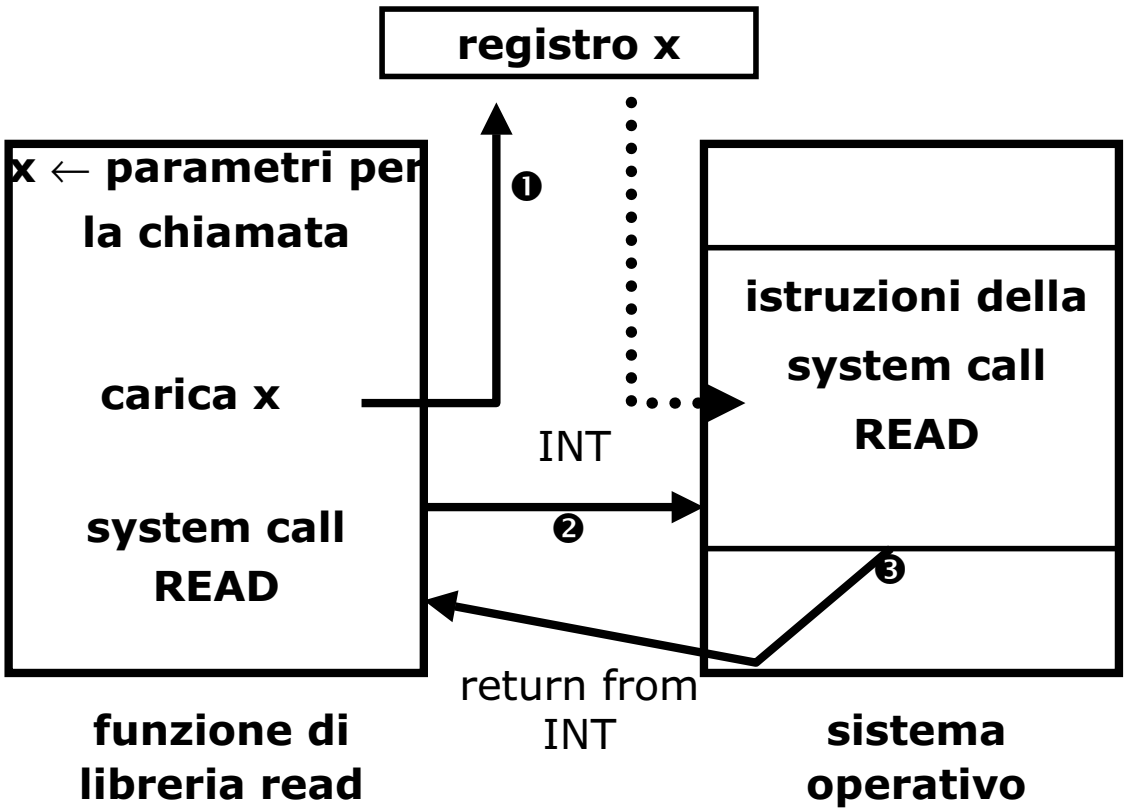
count = read (file, buffer, nbyte)

read = nome della procedura di libreria

file = nome del file

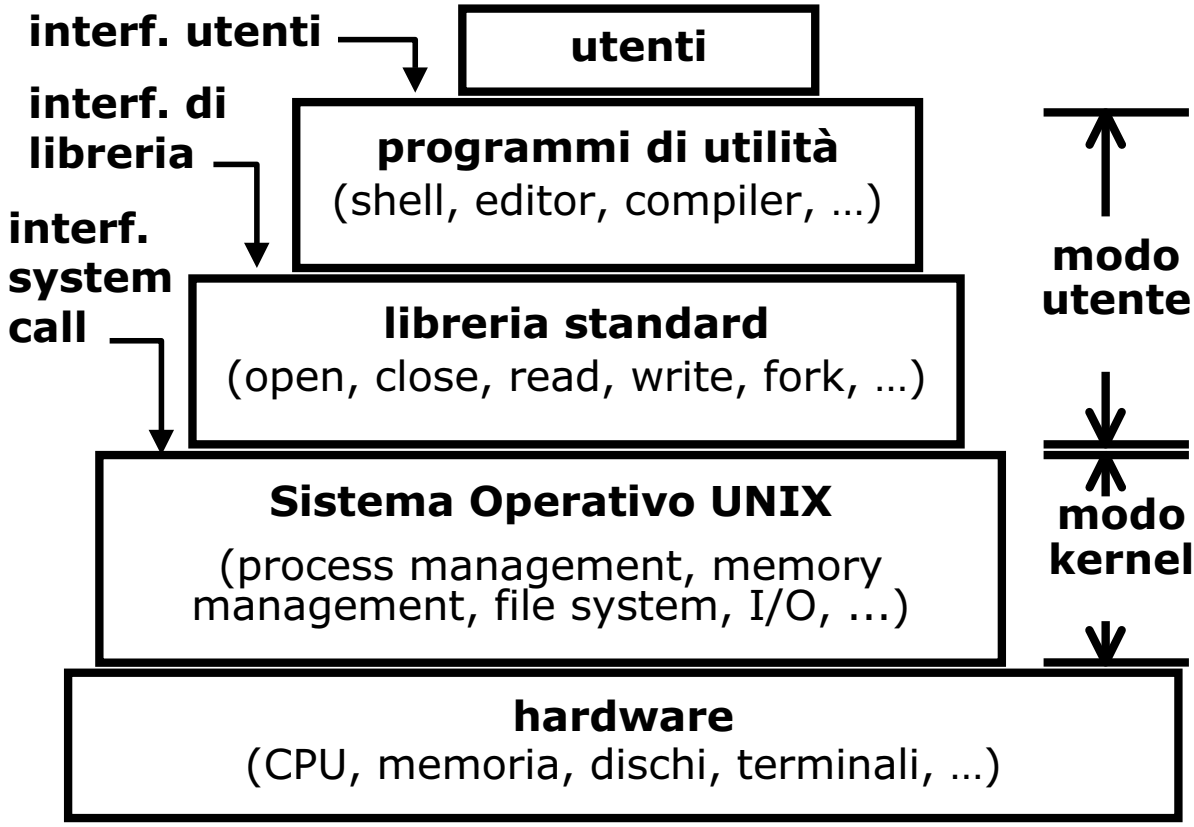
buffer = area di memoria per i dati

nbyte = numero di byte da leggere



Passaggio dei parametri alla system call

- direttamente nei registri
- in una tabella di memoria il cui indirizzo è passato in un registro
- tramite uno *stack* (operazioni di *push and pop*)



Modalità di funzionamento di un sistema di calcolo

modo utente (*user mode***)**

- usato per la normale esecuzione dei programmi
- non è possibile accedere liberamente a tutte le risorse del sistema (es. dispositivi di I/O)

modo kernel (*supervisor mode*)

- usato per lo svolgimento dei servizi richiesti al Sistema Operativo tramite le system call
- non esiste alcun limite alle operazioni effettuabili

Programmi di utilità

• UNIX, NT mettono a disposizione **programmi di sistema** che possono essere direttamente utilizzati dall'utente:

Compilatori, editor, elaborazione test, gestione dei file etc

• L'utilizzo di questi programmi può avvenire:
  • con interfacce a caratteri (keyboard oriented)
  • con interfacce di tipo grafico (mouse oriented)

Interfacce in Unix

A CARATTERI

processore dei comandi (shell) che interpreta ed esegue le richieste dell'utente espresse sotto forma di comandi (linguaggio dei comandi)

GRAFICHE

uso di finestre, icone, menu e di un dispositivo di puntamento (mouse)

CDE (Common Desktop Environment)

Struttura dei Sistemi Operativi

Organizzazione di un sistema operativo

- Come sono organizzate le varie componenti di un S.O.?
- Quali sono le modalità di interazione tra esse?

- Sistemi monolitici
- Sistemi a livelli
- Modello cliente-servitore

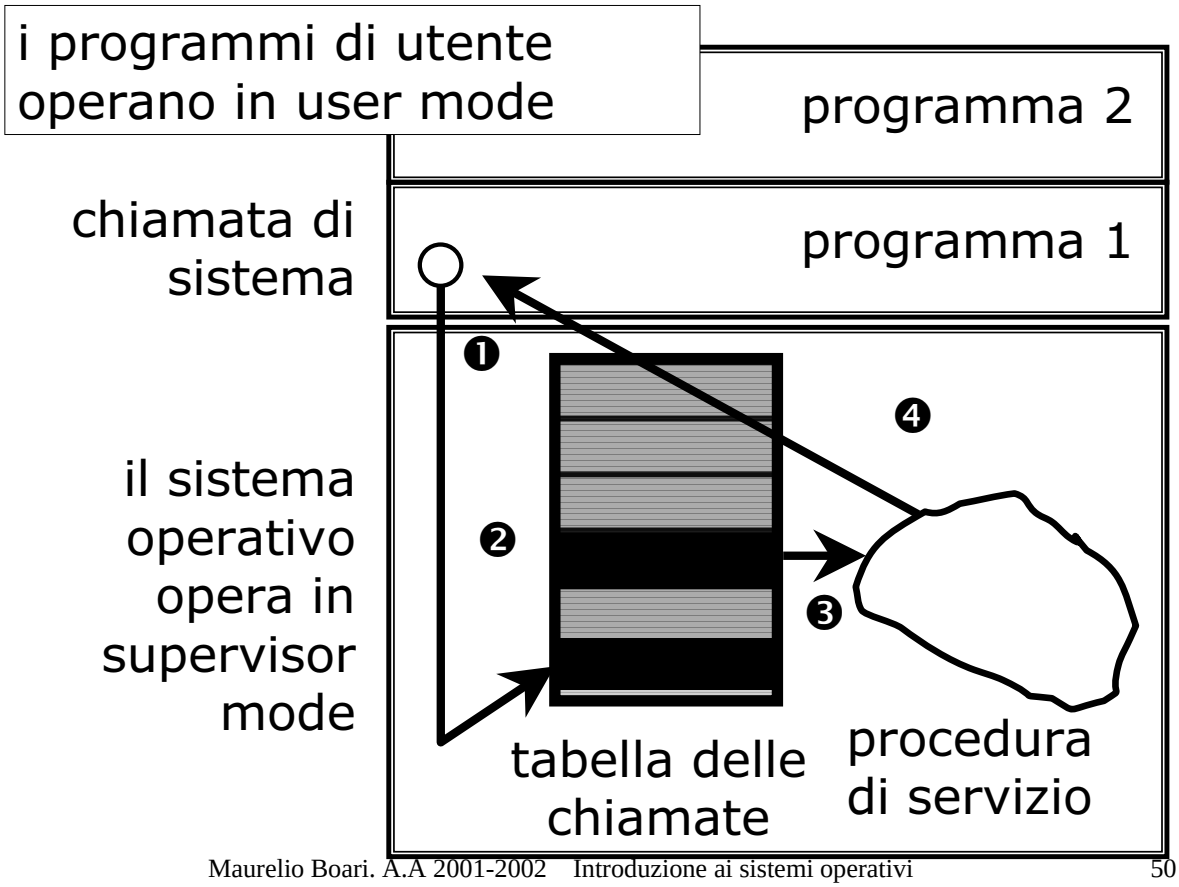
Sistemi monolitici

• Il Sistema Operativo è scritto come in insieme di **procedure di servizio**, tutte allo stesso livello di gerarchia.

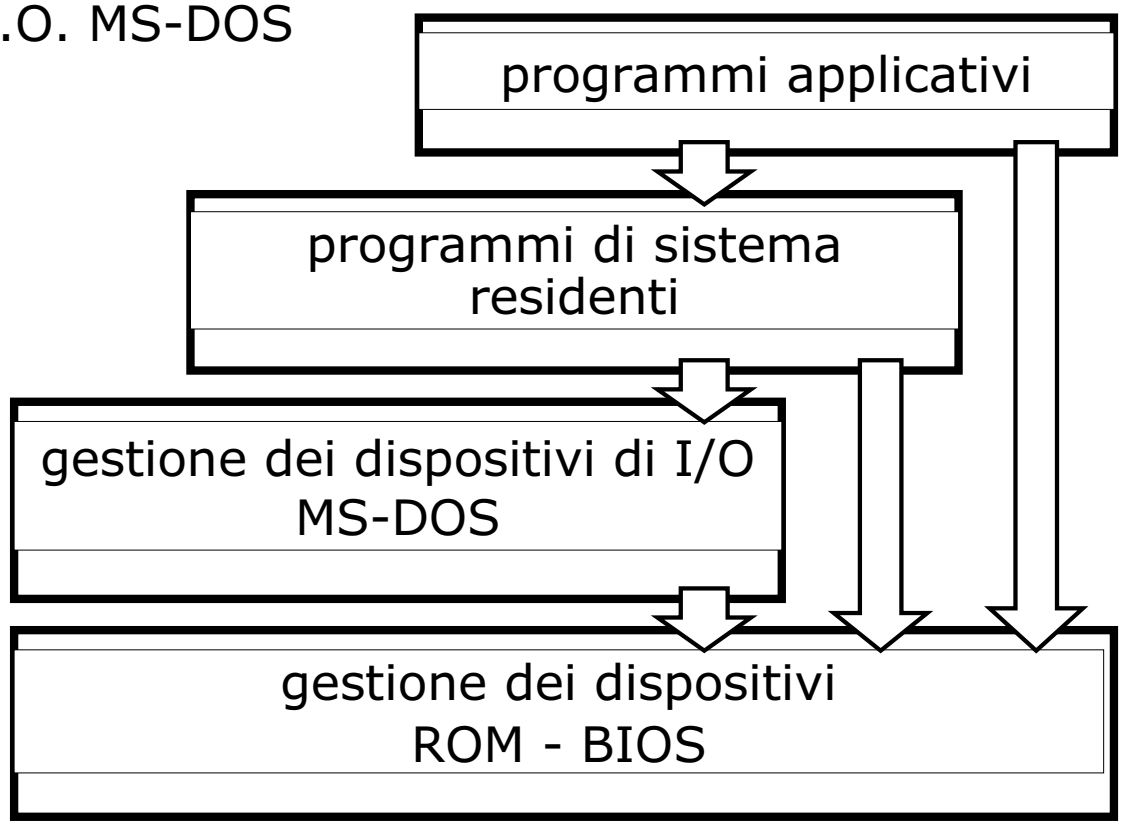
• Ciascuna procedura di servizio corrisponde ad una **chiamata di sistema.**

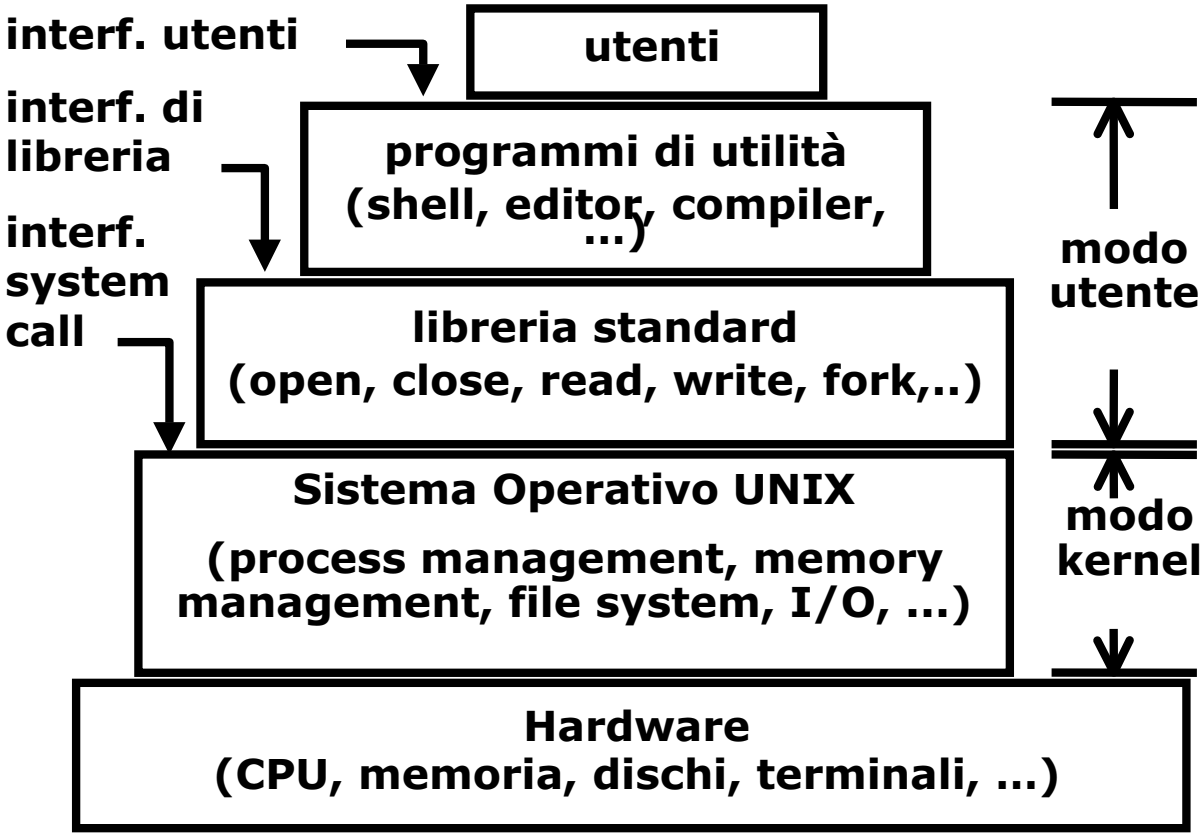
• Ciascuna può chiamarne un'altra secondo ben definite **interfacce**.

• **Procedure di utilità** utilizzate dalle procedure di servizio.



S.O. MS-DOS

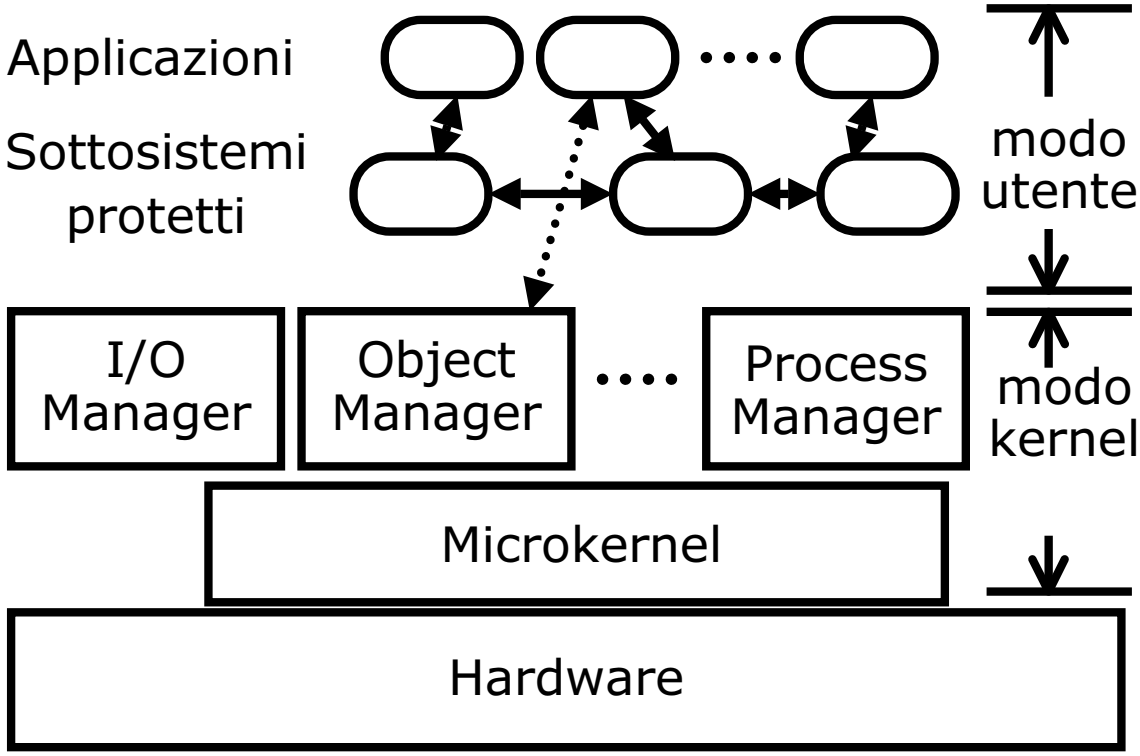

programmi applicativi
programmi di sistema residenti
gestione dei dispositivi di I/O MS-DOS
gestione dei dispositivi ROM - BIOS


interf. utenti
interf. di libreria
interf. system call
utenti
programmi di utilità (shell, editor, compiler, ...)
libreria standard (open, close, read, write, fork,..)
Sistema Operativo UNIX
(process management, memory management, file system, I/O, ...)
Hardware
(CPU, memoria, dischi, terminali, ...)
modo utente
modo kernel

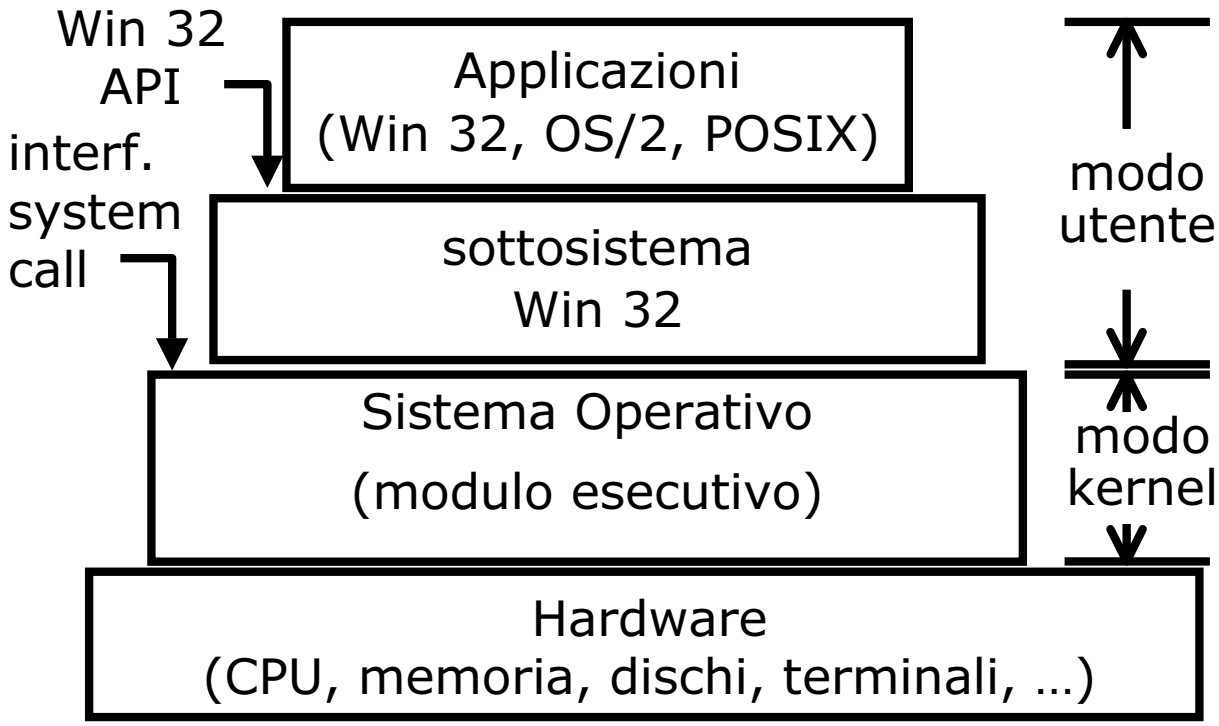
Sistemi a livelli

- Le funzioni del S.O. sono organizzate a **livelli gerarchici**
- Ogni livello definisce un tipo di servizio e le modalità per essere utilizzato dai livelli superiori
- **Macchine virtuali**-Ogni livello definisce una nuova macchina, aggiungendo nuove funzionalità alle procedure.
- Aiuto nella progettazione.
- Difficoltà pratiche nella realizzazione(gerarchia)

Windows NT



Windows NT



Sottosistema Di Ambiente

• Interfaccia di programmazione attraverso **API** (**A**pplication **P**rogram **I**nterface) **WIN 32** .

• **API** sono funzioni che un utente può utilizzare nel progetto di una applicazione per richiedere servizi al S.O.

• **WIN 32** è un insieme di funzioni molto esteso e documentato. Trasformano le richieste ricevute in chiamate ai servizi di sistema di NT.

• Le funzioni di WIN 32 API sono implementate da librerie ad aggancio dinamico (**DLL, D**ynami**c L**inked **L**ibrary**).**
Funzioni in codice eseguibile che possono essere caricate da un'applicazione dinamicamente a tempo di esecuzione

DDL

Principali DLL:

- Kernel 32. DDL : Gestione della memoria, dei threads e dei processi
- User 32. DDL : Gestione dell'interfaccia con l'utente e del sistema delle finestre.
- GDI. DDL : Gestione della grafica

• Win32 API rappresenta la specifica (in termini di nomi, di parametri passati e di comportamento) di tutte le funzioni che si possono usare per programmare in Windows.

• Stessa specifica per le diverse piattaforme Microsoft (NT, 95/98,CE)

Sottosistema di ambiente

• **Portabilità limitata.** Implementazione delleWin32 diversa per ognuna delle piattaforme. Ogni piattaforma implementa un sottoinsieme diverso delle funzioni specificate in WIN 32 API.

• E' possibile eseguire applicazioni conformi allo standard Posix, OS/2, MS-DOS e Windows 3.x.

• Due sottosistemi di ambiente, allo stesso livello di WIN 32, offrono le API di POSIX e OS/2

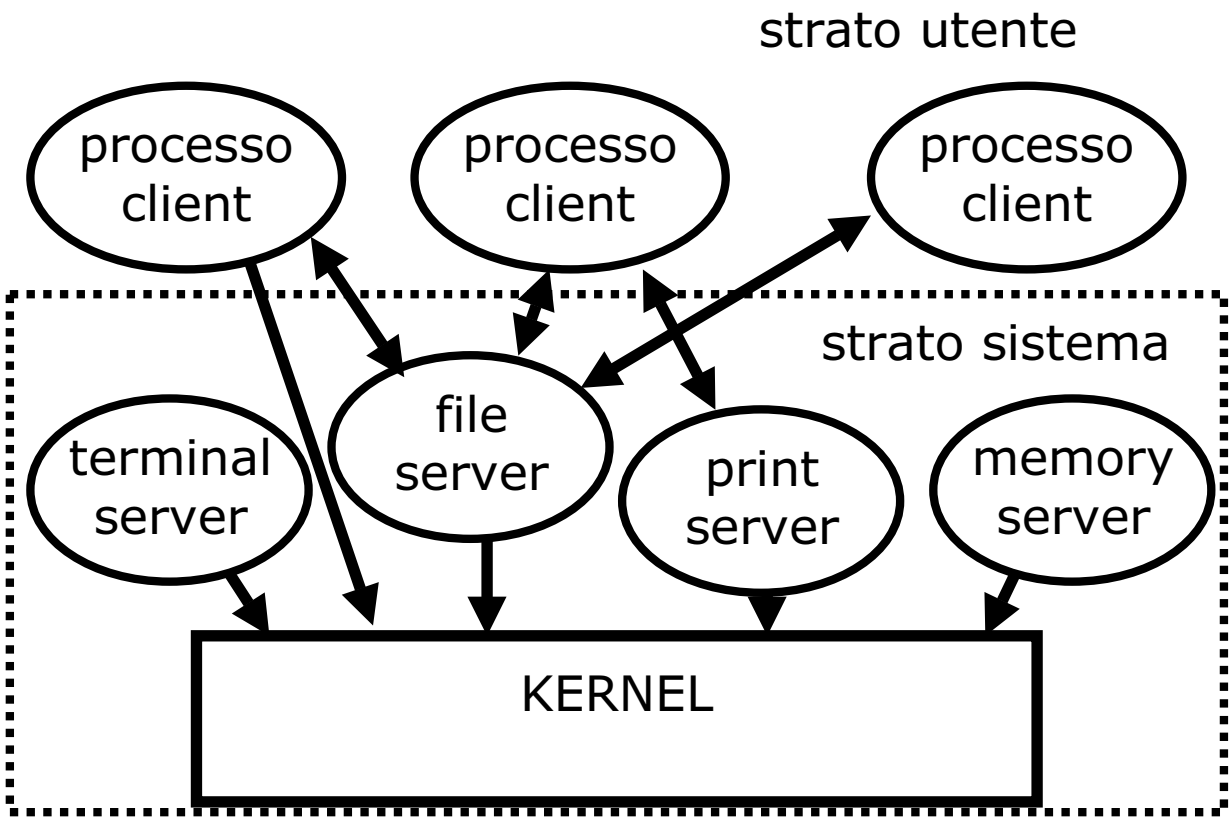
• Le applicazioni per MS-DOS e Windows 3.x usano un ambiente di lavoro confinato (macchina virtuale) all'interno di Win 32.

Modello cliente-servitore

• La maggior parte delle funzioni del S.O. (processi server) è realizzata a livello dei processi utente.

• Per richiedere un servizio (es. lettura di un file), un processo utente (processo client) invia una richiesta ad un processo server.

• Al termine del lavoro, il processo server rispedisce al client un messaggio di risposta.



- Il kernel include i dati che descrivono un processo e realizza i meccanismi di comunicazione tra i processi (microkernel)

- I processi server realizzano le politiche di gestione delle risorse

Vantaggi:

- **modularità** - le funzioni del S.O. sono suddivise in parti di dimensioni ridotte e ben specificate
- **portabilità** - separazione tra politiche e meccanismi

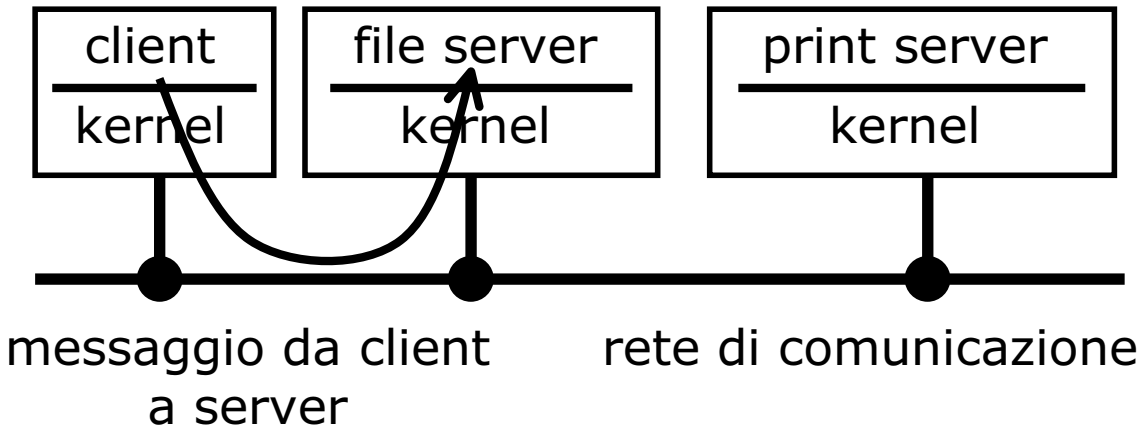
Svantaggi:

- **perdita di efficienza** - comunicazioni tra i sistemi client server

Estensione ai sistemi distribuiti

• Più calcolatori fra loro collegati, ciascuno contenente una copia del kernel ed un certo numero di processi server e/o client

• I processi client richiedono i servizi inviando le richieste tramite il kernel



• Non essendoci condivisione di dati tra processi, il modello client-server è particolarmente adatto ai sistemi distribuiti

CARATTERISTICHE DEI MODERNI SISTEMI OPERATIVI

- Architettura microkernel
- Multithreading
- Multiprocessing simmetrico
- Sistemi operativi distribuiti
- Architettura orientata agli oggetti

MICROKERNEL

• Concetto base: eliminare dal kernel tutte le componenti non essenziali realizzandole in ambiente utente tramite **processi server.**

• Vantaggi: maggiore **modificabilità** delle politiche di gestione delle risorse realizzate dai processi server. Maggiore **affidabilità**: se un servizio viene meno il resto del S.O. rimane operante.

• Problemi:minore **efficienza**. *Overhead* dovuto alla gestione dei processi

FUNZIONI PRESENTI NEL MICROKERNEL

- Gestione delle interruzioni
- Gestione dei processi
- Supporto alla comunicazione tra i processi
- Strumenti per la gestione della memoria

SUPPORTO ALLA COMUNICAZIONE TRA PROCESSI

Scambio di messaggi (message passing).
Es.: un processo utente (**client**) che intende accedere ad un file deve interagire con il file server.

Possibile realizzazione:
Nel server sono presenti delle **porte** (code di messaggi) per ogni processo.
Alla porta è associata una lista di accessibilità che indica quali processi possono comunicare con il processo proprietario della porta.

MULTITHREADING

• Tecnica per mezzo della quale un processo,eseguendo un'applicazione, viene suddiviso in **thread** (flussi di esecuzione elementare) eseguibili c**ontemporaneamente.**

• **Thread:** unità base di esecuzione. Viene eseguito sequenzialmente ed è interrompibile.
Dal punto di vista dell'allocazione e della gestione delle risorse il concetto di thread è analogo a quello di processo.

• **Processo:** collezione di più thread che ne condividano le variabili locali e le risorse di sistema associate (file aperti, dispositivi di I/O)

SYMMETRIC MULTIPROCESSING

• Il termine **SMP** si riferisce sia all'architettura hardware del computer sia al comportamento del S.O. al di sopra di tale architettura.

• **Multiprocessore simmetrico**:
  • più processori
  • condivisione della memoria principale, dei dispositivi di I/O, interconnessione tramite bus (o altro schema di connessione)
  • tutti i processori possono effettuare le stesse funzioni (da cui il termine simmetrico).

• Il S.O. di una macchina SMP attribuisce i processi o i thread a **tutti i processori**. La presenza di più processori è trasparente all'utente.

SISTEMA OPERATIVO DISTRIBUITO

• **Sistema multicomputer:** ogni computer ha la propria memoria principale e secondaria e propri dispositivi di I/O. I computer sono connessi tra loro secondo varie tipologie.

• **Sistema operativo di rete:** consente il trasferimento di file tra computer collegati tra loro.

• **Sistema operativo distribuito:** offre all'utente l'astrazione di un singolo spazio di memoria centrale e secondaria, **trasparenza** rispetto alle unità di elaborazione e rispetto ad altre risorse equivalenti.

ARCHITETTURA ORIENTATA AGLI OGGETTI

• **Progettazione orientata agli oggetti.**
Consente una maggiore strutturazione dei S.O.ed una più semplice **modificabilità** (senza distruggere l'integrità del sistema).

• Rende più semplice la scrittura del S.O. e la verifica della sua **correttezza.**

Panoramica
su
Windows NT e Unix

WINDOWS NT

• Progettato per sfruttare la potenza dei microprocesori a 32 bit (es. memoria virtuale.)

• **Multitasking** a singolo utente:

  • Sviluppo di applicazioni complesse suddivisibili in parti con scambio di informazioni (finestre e mouse)
  • Elaborazione client/server. Lato client: colloquio con l'utente e comunicazione (protocolli) con il server

NT-Architettura

NT Executive

- Struttura modulare: ciascuna funzione di sistema è gestita da una componente del sistema operativo (**modulo**)
- Utilizzo di interfacce standardizzate (**API**)
- Ogni modulo può essere rimosso, aggiornato o rimpiazzato.
- Non è una struttura a **microkernel**: per motivi di **efficienza** molte funzioni sono eseguite in modo **kernel**.

STORIA

1981. DOS 1.0. Sviluppato da Microsoft per il primo PC IBM; 1400 linee di codice in assembler; processore Intel 8086, 8KB di memoria.(MS-DOS)

1983. DOS 2.0. Supporta hard disk e directory gerarchiche. Sviluppato per PC XT IBM. Redirezione dell'I/O e stampa in background. Parte residente in memoria: 24KB.

1984. DOS 3.0. Indirizzamento esteso e capacità di protezione di memoria. Sviluppato per PC AT Intel 80286. Successive versioni per 80386 per PC IBM PS/2 (32 bit, non utilizzati). Parte residente in memoria 46KB.

STORIA

1990. Windows 3.0. Interfaccia simile a quella del Macintosh. Necessità di essere eseguita insieme al sottostante DOS. Successive versioni: Windows 95.

1993. Windows Nt (3.1). S.O completamente nuovo a 32 bit, multitasking in ambiente a singolo utente. Contemporaneo sviluppo da parte di IBM di OS/2 con le stesse caratteristiche.

1996. Windows NT 4.0. Stessa interfaccia di Windows 95. Implementazione nel kernel di diverse funzioni grafiche precedentemente realizzate in ambiente Win 32.

.

STORIA

1998. Windows Nt 5.0. Aggiunta di servizi e funzioni per l'elaborazione distribuita. NT workstation e NT server. Quest'ultimo comprende servizi necessari per server di rete.

2000. Windows 2000. Evoluzione di Windows NT 4.0 che offre un nuovo insieme di servizi distribuiti e di rete (Active Directory, infrastruttura distribuita per la sicurezza, infrastruttura centralizzata per l'amministrazione delle macchine Windows sulla stessa rete locale).

2001. Windows XP

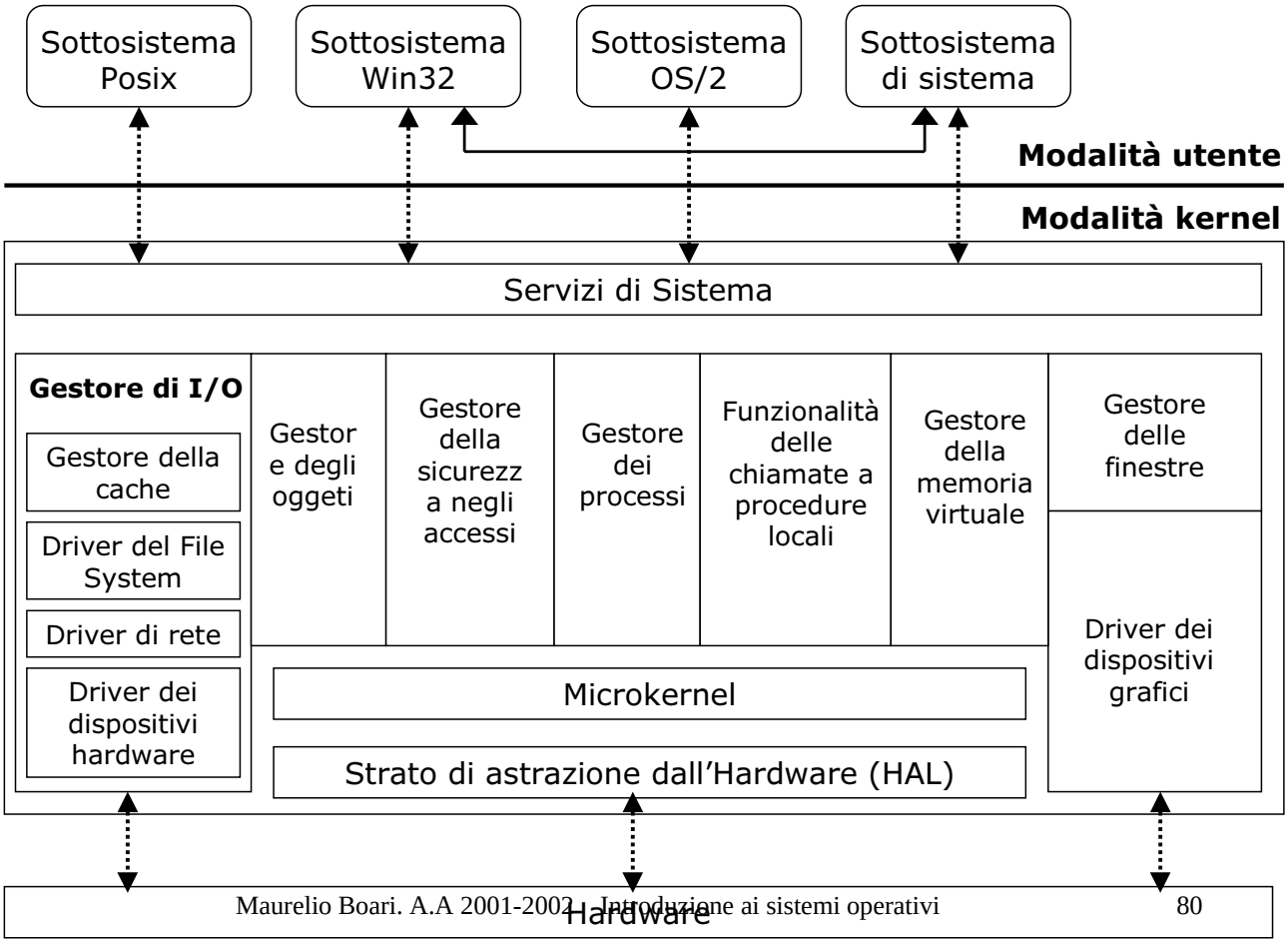
STRUTTURA DI NT

• **Strato di astrazione dall'hardware (HAL).** Fornisce un'astrazione software per i dispositivi hardware quali orologi, controllori della memoria cache e della memoria centrale, controllori di interrupt, bus di sistema, etc...

• **Microkernel.** Mette a disposizione i meccanismi per la schedulazione dei threads, dei processi, i cambi di contesto,gestione delle interruzioni ed eccezioni. Non è paginabile o interrompibile.

• **Servizi executive.** Moduli per funzioni specifiche di sistema. Ad eccezione del gestore di I/O, dei moduli grafici e delle finestre, tutti i servizi utilizzano HAL.

• **Servizi di sistema.** Forniscono un'interfaccia al software in modalità utente.



UNIX

- Comandi Unix e librerie
- Chiamate di sistema
- Kernel
- hardware

UNIX

- System V Release 4 (SVR4)
- Solaris 2.x
- 4.4BSD
- Linux

UNIX

- Hardware circondato dal S.O.
- S.O. chiamato **kernel**
- Dotato di un insieme di servizi per l'utente:
  - Shell
  - Compilatore C