01a - Introduzione

Sistema di Elaborazione

Per sistema di elaborazione si intende l'insieme di risorse hardware e software per l'elaborazione automatica delle informazioni.

Architettura di un Sistema di Elaborazione

1 - **Hardware**: le risorse di calcolo fondamentali (CPU, memoria, dispositivi di I/O).

2 - **Sistema Operativo**: controlla e coordina l'utilizzo dell'hardware da parte dei vari programmi applicativi di utenti diversi.

3 - **Programmi Applicativi**: definiscono il modo in cui le risorse del sistema devono essere usate per risolvere i problemi di elaborazione dei diversi utenti (es. compilatori, basi di dati, video game, videoscrittura, programmi scientifici, programmi contabili).

4 - **Utenti**: coloro che utilizzano i programmi applicativi (persone, macchine, altri computer).

Hardware

- **CPU o Processore**: dispositivo che esegue le istruzioni di un programma per l'elaborazione delle informazioni.
- **Memoria**: insieme di dispositivi per la memorizzazione temporanea (memoria centrale) o permanente (memoria di massa) delle informazioni.
- **Dispositivi di I/O**: insieme di dispositivi per l'acquisizione o la restituzione di informazioni.

Classificazione delle Architetture

SISD:
- Singolo flusso di Istruzioni
- Singolo flusso di Dati
→ tradizionali calcolatori monoprocessore

Single Instruction Single Data

È un'architettura in cui una singola unità esegue un singolo flusso di dati. Corrisponde alla classica architettura di von Neumann utilizzata in quasi tutti i personal computer, anche se è ormai obsoleta.
Ha un solo processore che lavora con un singolo flusso di istruzioni e un solo flusso di dati alla volta

SIMD:
- Singolo flusso di Istruzioni
- Molteplici flussi di Dati
→ array processor

Single Instruction Multiple Data

Alla classe SIMD appartengono le architetture composte da molte unità di elaborazione che eseguono contemporaneamente la stessa istruzione ma lavorano su insiemi di dati diversi. Generalmente, il modo di implementare le architetture SIMD è quello di avere un processore principale che invia le istruzioni da eseguire contemporaneamente ad un insieme di elementi di elaborazione che provvedono ad eseguirle

MISD:
- Molteplici flussi di Istruzioni
- Singolo flusso di Dati
→ nessun calcolatore noto rientra in questa categoria

Multiple Instruction Single Data

Più flussi di istruzioni (processi) lavorano contemporaneamente su un unico flusso di dati. Nel modello di calcolo MISD esistono più processori, ognuno con una propria memoria (registri), la quale a sua volta avrà un proprio flusso di istruzioni. Queste istruzioni verranno eseguite sullo stesso flusso di dati.

MIMD:

- Molteplici flussi di Istruzioni
- Molteplici flussi di Dati
→ multiprocessori
→ multicalcolatori strettamente accoppiati
→ multicalcolatori lascamente accoppiati

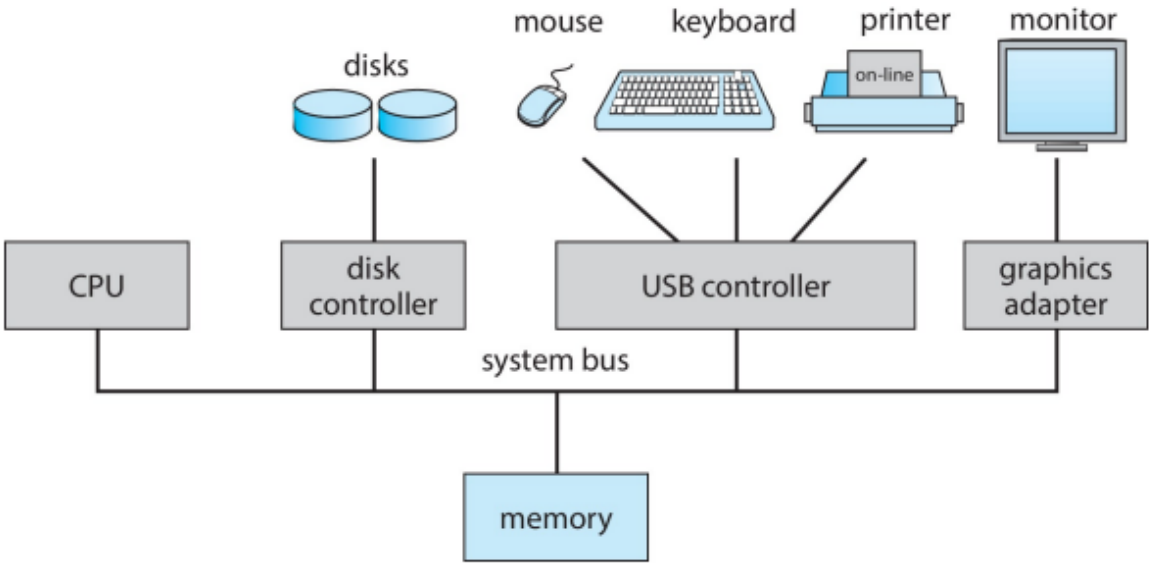
Multiple Instruction Multiple Data

È un'architettura parallela in cui diverse unità effettuano diverse elaborazioni su dati diversi. In configurazione multiprocessore le CPU possono condividere dati allocati in memoria comune accessibile via bus di sistema.

SISD - Monoprocessori

È dotato di una sola CPU in grado di eseguire un insieme di istruzioni di natura generale. Inoltre quasi tutti i sistemi possiedono altri processori specializzati, deputati a compiti particolari. Tutti questi processori di tipo specifico sono dotati di un insieme ristretto di istruzioni, e non eseguono processi utenti. Talvolta sono guidati dal sistema operativo, che può inviare loro informazioni sul compito da espletare successivamente, e controllarne lo status.
Ogni dispositivo è collegato al bus, una rete di connessione tra i dispositivi con i quali si scambiano informazioni. Viene utilizzata un tipo di architettura detta Master-Slave: la CPU (master) decide cosa fare. Questo evita i conflitti.
In realtà il bus non è un “filo” unico, ma è composto da più linee di comunicazione, in ognuna ci sarà un master che comunica con gli altri dispositivi.
La memoria è un dispositivo particolare, per leggere e scrivere si usa il controllore della memoria, che si interfaccia al bus, riceve richieste e trasmette

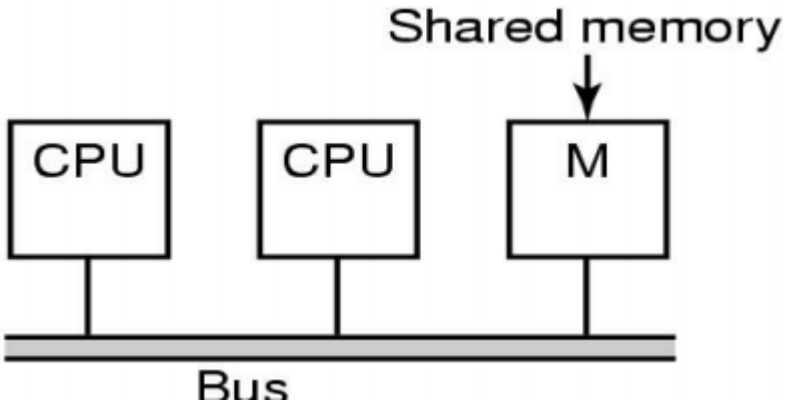
risposte interagendo con la memoria. Ogni dispositivo ha bisogno del suo controller.



MIMD - Multiprocessori

Questo tipo di sistemi dispone di più unità d'elaborazione in stretta comunicazione, che condividono i canali di comunicazione all'interno del calcolatore (bus), i timer dei cicli di macchina (clock) e talvolta i dispositivi di memorizzazione e periferici.

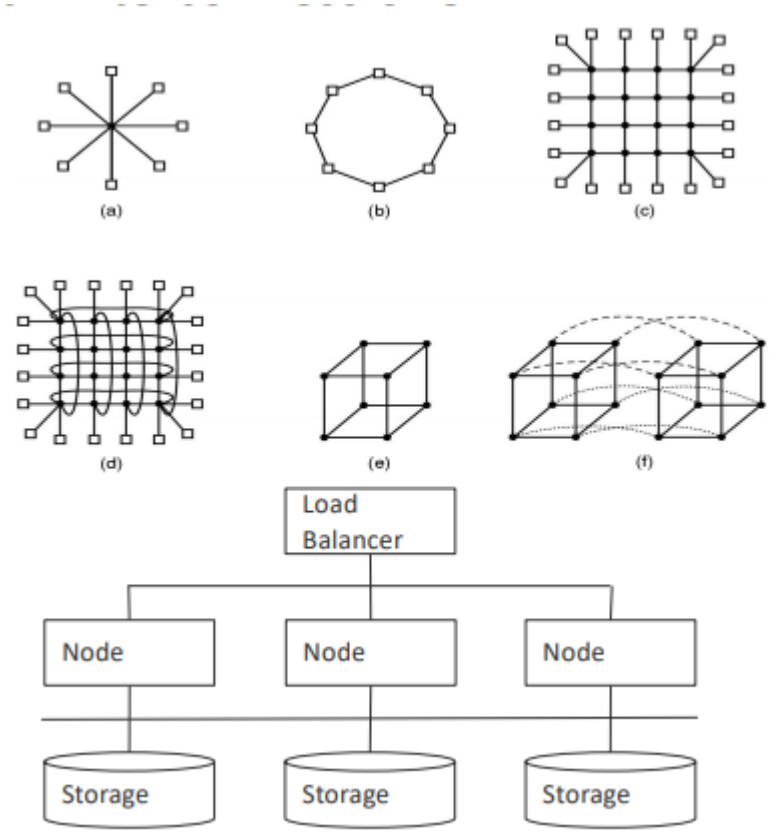
Aumentando il numero di unità di elaborazione è possibile svolgere un lavoro maggiore in meno tempo. Sono dotati di una sola memoria centrale condivisa tra tutti i processori, questo crea dei conflitti sulla gestione degli ingressi alla memoria. Gestisce diversi flussi di istruzioni contemporaneamente.



Multiprocessore

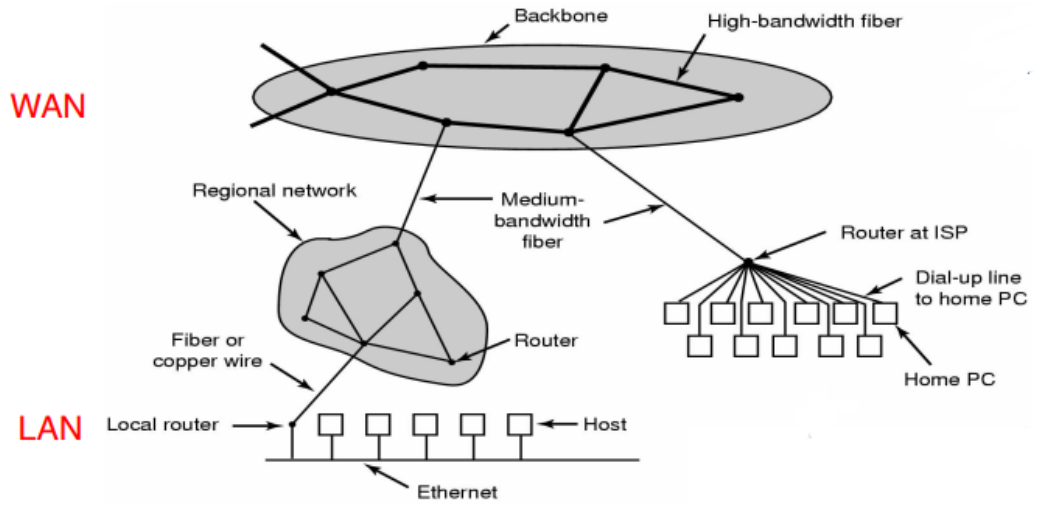
MIMD – Multicalcolatori strettamente accoppiati

Sono basati sull'uso congiunto di più unità d'elaborazione riunite per lo svolgimento di attività d'elaborazione comuni. In grado di seguire più flussi di istruzioni contemporaneamente. Ogni nodo di un'architettura di questo tipo ha un processore con la sua RAM e la sua memoria. (Usano la rete cablata, LAN)



MIMD – Multicalcolatori lascamente accoppiati

Ogni nodo è completamente autonomo e indipendente dagli altri. (Usano modem, rete distribuita)



Sistema Operativo

Un sistema operativo è un insieme di programmi (software) che gestisce gli elementi fisici di un calcolatore (hardware), fornisce una piattaforma ai programmi applicativi e agisce da intermediario tra l'utente e la struttura fisica del calcolatore.

Interfaccia - intermediario tra un utente e l'hardware di un computer.

Allocatore di Risorse – gestisce ed alloca in modo efficiente le risorse.

Programma di Controllo – controlla l'esecuzione dei programmi utente e le operazioni dei dispositivi di I/O.

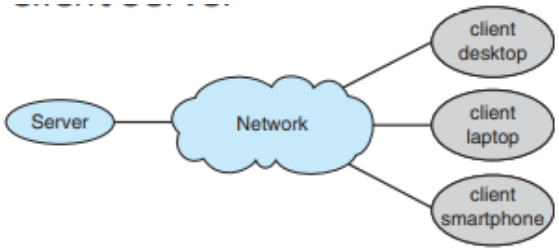
Kernel – l'unico programma che è sempre in esecuzione (tutti gli altri sono considerati programmi applicativi).

Obiettivi di un SO:

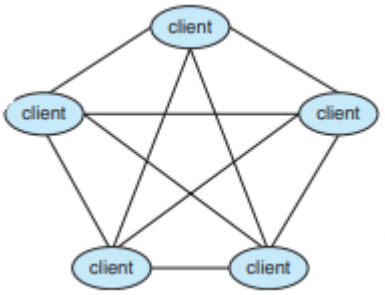
– Eseguire i programmi utente
– Facilitare la soluzione dei problemi degli utenti.
– Rendere il computer conveniente.

Classificazione dei Sistemi Operativi

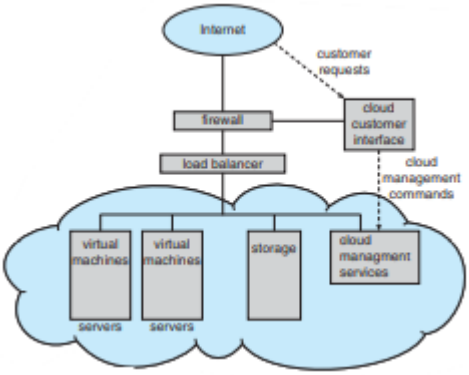
- Traditional
  – Macchine di tipo stand-alone general-purpose
  – Interconnected via wired or wireless networks
- Mobile
  – Smartphones, tablets, etc.
  – Interconnected via wireless networks
  – Caratteristiche extra: OS (GPS, giroscopio), App. (realtà aumentata)
- Client Server



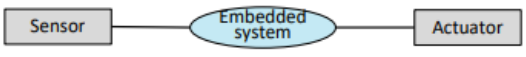
- Peer-to-Peer



- Cloud computing



- Real-time Embedded



BATCH - primi sistemi operativi ad essere stati sviluppati. In memoria troviamo un solo task (programma) e sistema operativo. Tutta la CPU è dedicata al task (chiamato job). Terminata l'esecuzione del programma l'area viene eliminata e

viene poi occupata dal nuovo programma. Lavorare su più programmi contemporaneamente è un modo per ottimizzare le risorse evitando i tempi morti.

MULTITASKING - manda in esecuzione più programmi contemporaneamente. Ci sono diversi job con il sistema operativo che si contendono la CPU tra di loro. Questa contesa della CPU viene regolata in base a tre criteri:
-
Multiprogramming: la CPU è occupata ad eseguire l'istruzione del primo programma. Quel programma non dovrà occupare tutto il tempo la CPU, e in quel momento lascia la CPU e viene assegnata al secondo programma.
-
Time-Sharing: se abbiamo però programmi che utilizzano solo la CPU si lascia al programma un tempo limite, finito quello la CPU lavora con un altro programma e così via.
-
Real-time: è stato introdotto perché alcuni programmi hanno vincoli temporali precisi da rispettare.

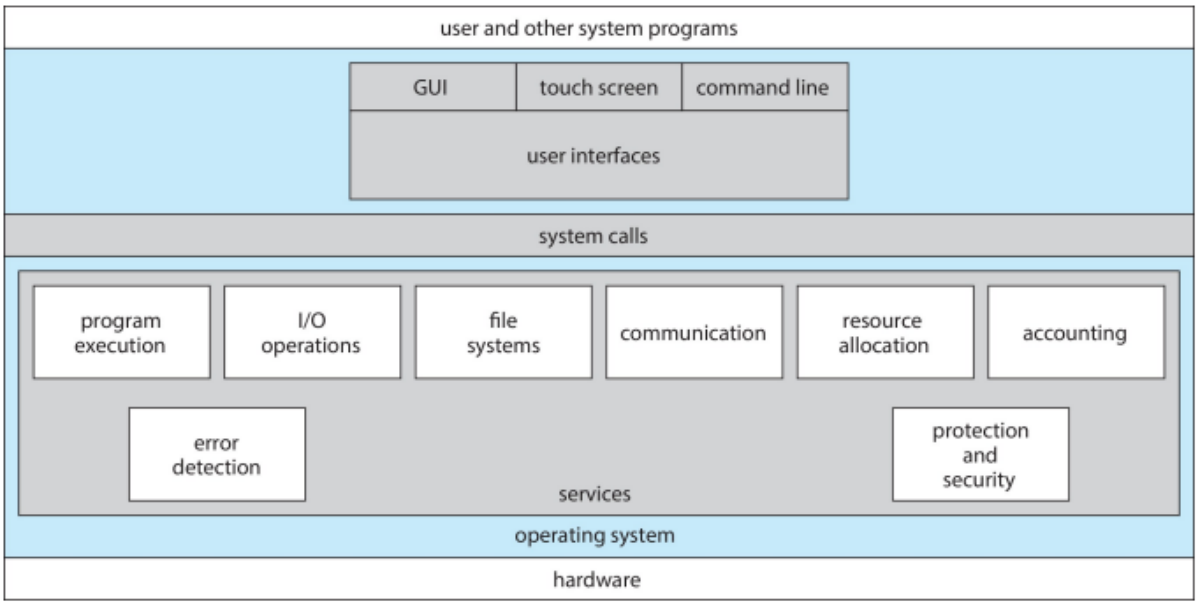
- Soft RT: massima priorità per quel programma, ma non garantisce che i tempi vengano rispettati.
- Hard RT: massima priorità nel rispettare i tempi.

Struttura dei Sistemi Operativi

Tutte le funzioni sono gestite dal kernel.

Struttura in base alla funzione:
– Gestione dei processi
– Gestione della memoria centrale
– Gestione dei dispositivi di I/O
– Gestione della memoria secondaria
– Gestione del file system
– Gestione delle reti
– Gestione della Sicurezza
– Interprete di comandi

Struttura in base al livello:
– Servizi di sistema
– Programmi di sistema



Struttura in basa alla funzione

Funzione: Gestione dei Processi

Un **processo** è un **programma in esecuzione**.

Ogni processo ha bisogno di determinate risorse per terminare il suo compito: CPU, memoria, file e dispositivi di I/O.

Il SO è responsabile della gestione dei processi:
– Creazione e cancellazione dei processi
– Sospensione e ripristino dei processi
– Meccanismi per la sincronizzazione e per la comunicazione dei processi

Funzione: Gestione della Memoria

La **memoria** centrale è un **array di word o byte** molto grande, ognuno con un suo indirizzo.

Permette l' accesso veloce a dati condivisi dalla CPU e dai dispositivi di I/O.

La memoria centrale è un dispositivo di memorizzazione volatile.

Il SO è responsabile delle seguenti attività relative alla gestione della memoria:
– Tenere traccia di quali parti della memoria sono utilizzate e da chi
– Decidere quali processi caricare in memoria
– Allocare e deallocare lo spazio di memoria

Funzione: Gestione dei File

Un **file** è una collezione di informazioni collegate tra loro e definite dal suo creatore.

Il SO è responsabile delle seguenti attività relative alla gestione dei file:
– Creazione e cancellazione dei file.
– Creazione e cancellazione di directory (cartelle).
– Mettere a disposizione primitive per la manipolazione di file e directory.
– “Mappare” i file su dispositivi di memorizzazione secondaria e/o altri dispositivi di I/O.
– Gestire il backup dei file.

Funzione: Gestione dell'I/O

Il sistema di gestione dell’I/O consiste in:
– Un sistema di buffer-caching
– Una interfaccia generale per i driver dei dispositivi
– Driver per specifici dispositivi hardware

Funzione: Gestione della memoria secondaria

La maggior parte dei moderni computer utilizza i dischi come principale mezzo di memorizzazione sia per programmi sia per i dati.

Il SO è responsabile delle seguenti attività relative alla gestione dei dischi:
– Gestione dello spazio libero
– Allocazione dellospazio
– Scheduling del disco

Funzione: Gestione delle Reti

Un sistema distribuito è una collezione di processori che non condividono ne la memoria ne il clock.

Il SO è responsabile delle seguenti attività relative alla gestione delle reti:
– Invio e ricezione di messaggi
– Instradamento e gestione delle connessioni
– Gestione dei conflitti
– Gestione della sicurezza

Funzione: Gestione della Sicurezza

La protezione si riferisce al meccanismo per controllare l’ acceso di processi e utenti alle risorse di un utente e a quelle di sistema.

Il sistema di protezione deve:
– distinguere tra un utilizzo autorizzato ed uno non autorizzato
– specificare quali controlli devono essere eseguiti ed i mezzi pereffettuarli

Funzione: Interprete dei Comandi

Command Line Interpreter (CLI)
Gran parte dei comandi sono dati al SO per mezzo di istruzioni. Tali istruzioni permettono:
– la creazione e la gestione dei processi
– la gestione dell'I/O
– la gestione della memoria secondaria
– secondary-storage management
– la gestione della memoria centrale
– l'accesso al file system
– la protezione delle risorse
– il networking

Il programma che legge ed interpreta tali istruzioni è chiamato in diversi modi:
– interprete comandi
– shell (in UNIX)
– prompt di MS-DOS (in MS Windows)

Esistono anche interfacce utente grafiche – Finder (MacOS)
– Explorer (MS Windows)
– X Window System (Unix)

Struttura in base al livello

Livello: Servizi di un SO

Caricamento ed esecuzione dei programmi.

Operazioni di I/O – un utente non può eseguirle direttamente.

Manipolazione del File-system – la possibilità per un processo o un utente di leggere, scrivere, creare e cancellare file.

Comunicazione – la possibilità per due processi (sulla stessa macchina o su due macchine differenti collegate tra loro in rete) di scambiarsi informazioni.

Rilevamento di errori.

Allocazione delle risorse – a più utenti/processi in esecuzione nello stesso momento.

Contabilizzazione dell'uso delle risorse – tenere traccia di chi e per quanto tempo ha utilizzato che cosa. Questo serve sia per raccogliere statistiche sul funzionamento del sistema sia nei sistemi in cui
l'accesso alle risorse è a pagamento.

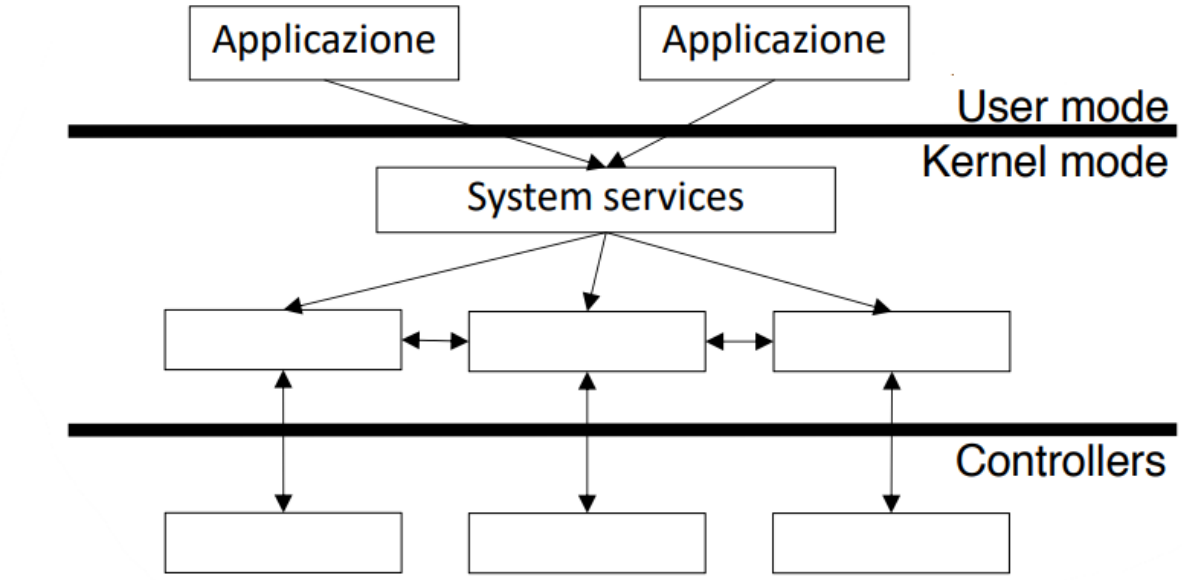
Protezione – assicurare che l'accesso alle risorse del sistema sia controllato.

Livello: Programmi di sistema

Costituiscono l'ambiente di sviluppo ed esecuzione dei programmi:
– Manipolazione e modifica file
– Stato del sistema
– Strumenti di ausilio alla programmazione
– Caricamento ed esecuzione di programmi
– Comunicazione
– Programmi di utilità

Per la maggior parte degli utenti, il SO è caratterizzato dai programmi di sistema che mette a disposizione piuttosto che dalle system call (esse costituiscono l'interfaccia tra un processo e il sistema operativo).

Architettura Monolitica



Idealmente si ha tutto in un unico componente.
Il sistema consiste di due parti separate: il kernel e i programmi di sistema
. A sua volta, il kernel è diviso in una serie di interfacce e driver dei dispositivi.
Tutto ciò che sta al di sotto dell'interfaccia alle chiamate di sistema e al di sopra dell'hardware costituisce il **kernel**. Esso comprende il file system, lo scheduling della CPU, la gestione della memoria, e le altre funzionalità del sistema rese disponibili tramite chiamate di sistema.

Vantaggi:

- Se si mette tutto in un unico componente si risparmia memoria
- Maggiore velocità (non si perde tempo per gestire le connessioni tra un componente e l'altro)

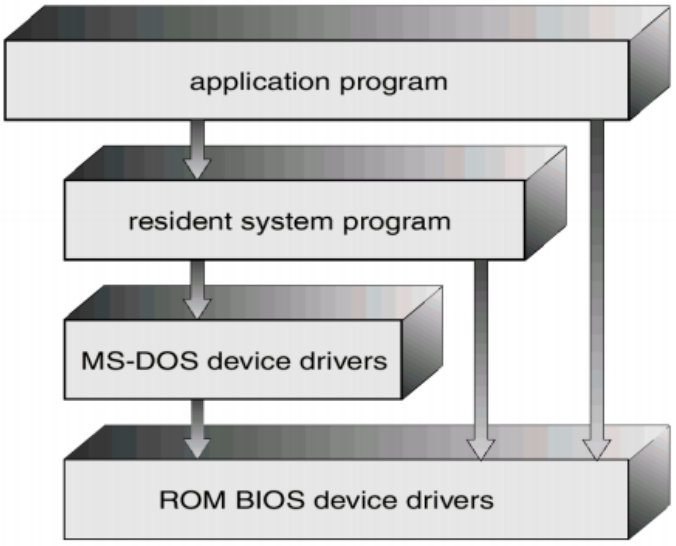
Svantaggi:

- Manutenzione più complicata perché bisogna modificare tutto ogni volta

Si contrappone all'ARCHITETTURA MODULARE, che ha la possibilità di fare manutenzione solo su uno dei moduli senza dover modificare gli altri non di interesse.

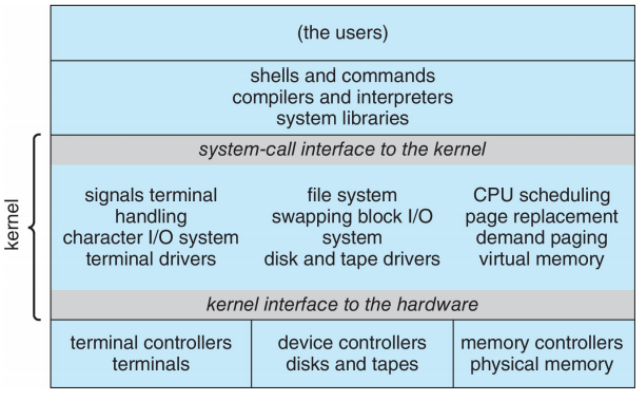
MS-DOS

MS-DOS è stato scritto (da Tim Paterson) tenendo conto delle limitazioni dell'hardware e per offrire il massimo delle funzionalità occupando il minimo spazio. Infatti MS-DOS, anche se ha un minimo di strutturazione:
– non è diviso in moduli
– le interfacce e i livelli di funzionalità non sono ben separati

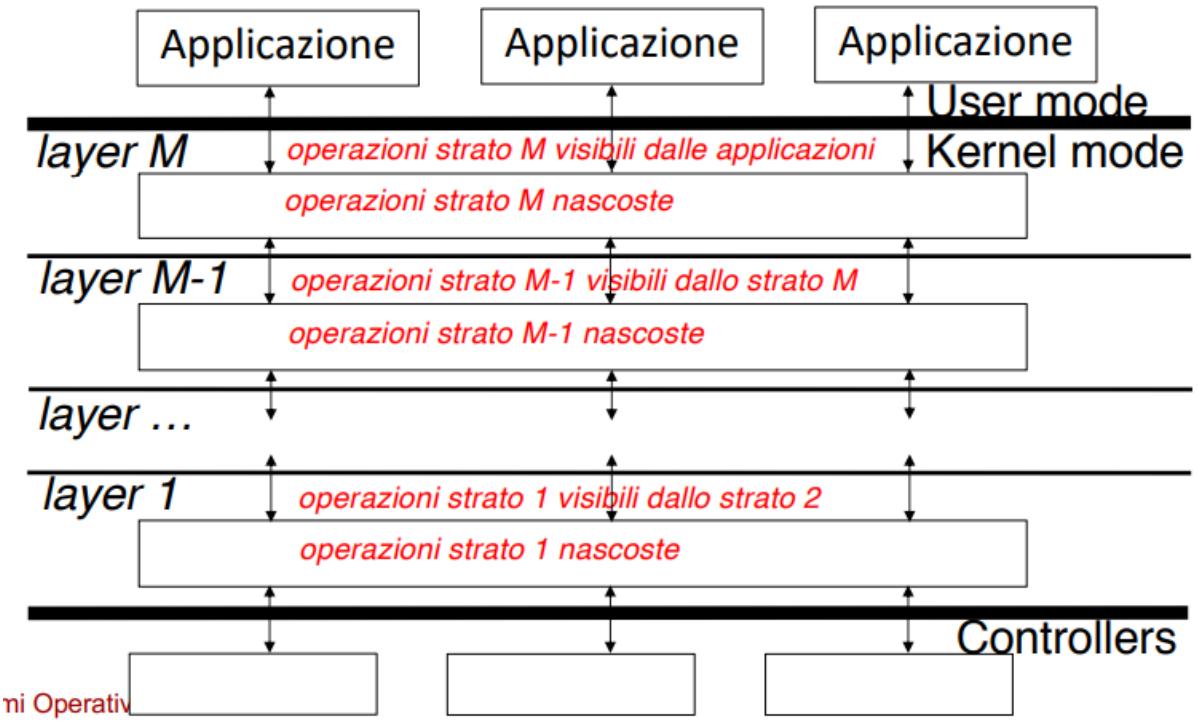


UNIX

UNIX è stato scritto (da Ken Thompson e Dennis Ritchie) tenendo conto delle limitazioni dell'hardware.
Quindi lo UNIX originale ha una strutturazione limitata. Il sistema consiste di due parti:
– programmi di sistema
– il kernel:
• tutto ciò che è compreso tra l'interfaccia con le system call e l'hardware del sistema
• tutte le funzioni di gestione della CPU, della memoria ed altre ancora sono riunite in un unico livello.



Architettura Stratificata



In presenza di hardware appropriato, i sistemi operativi possono essere suddivisi in moduli più piccoli e gestibili di quanto non fosse possibile nelle prime versioni di MS-DOS e UNIX. Ciò permette al sistema operativo di mantenere un controllo molto più stretto delle applicazioni che girano sulla macchina. Inoltre, gli sviluppatori del sistema godono di maggiore libertà nel modificare i meccanismi interni del sistema e nel suddividere il sistema in moduli.

Il sistema è suddiviso in un certo numero di livelli o strati: il più basso corrisponde all'hardware (strato 0), il più alto all'interfaccia utente (strato M).

Il vantaggio principale offerto da questo metodo è dato dalla semplicità di progettazione e dalle funzionalità di debug. Gli strati sono composti in modo che ciascuno usi solo funzioni (o operazioni) e servizi che appartengono a strati di livello inferiore. Questo metodo semplifica il debugging e la verifica del sistema. Il primo strato si può mettere a punto senza intaccare il resto del sistema, poiché per realizzare le proprie funzioni usa, per definizione, solo lo strato fisico, che si presuppone sia corretto. Passando alla lavorazione del secondo strato si presume, dopo la messa a punto, la correttezza del primo. Il procedimento si ripete per ogni strato. Se si riscontra un errore, questo deve trovarsi in quello strato, poiché gli strati inferiori sono già stati corretti; quindi la

suddivisione in strati semplifica la progettazione e la realizzazione di un sistema. Un componente che si trova in un certo strato necessita di un altro componente ad un livello più basso, gli strati più bassi sono quelli più stabili perché più vicini all'hardware.

Uno svantaggio è che il codice occupa più spazio e i tempi si allungano.
Il Sistema operativo è organizzato in diversi strati o livelli. Ogni strato è costruito sullo strato precedente.

L'hardware si trova a livello 0 e l'interfaccia utente a livello M.

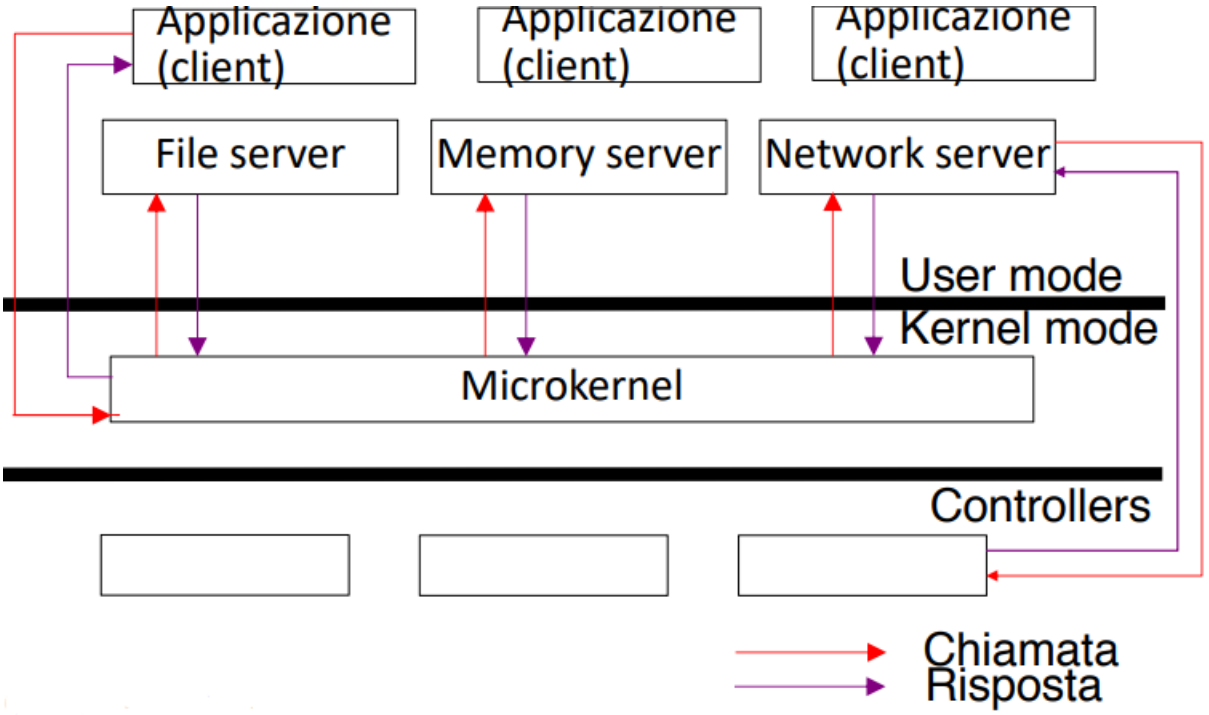
Lo strato M può solamente richiamare operazioni interne ed operazioni pubbliche dello strato M-1, e non può mai usare quelle dello strato M+1.

- Il SO è suddiviso in un certo numero di strati (chiamati livelli).
- Ogni strato è costruito sopra gli strati precedenti. Il livello più basso (livello 0) è l'hardware ed il livello più alto (livello N) è l'interfaccia utente.
- Per rendere il sistema modulare, i livelli sono definiti in modo tale che ogni livello utilizza soltanto funzioni (operazioni) e servizi che appartengono ai livelli inferiori, mentre a sua volta mette a disposizione funzioni ai componenti che si trovano negli strati superiori.

Vantaggi: rende il software modulare, abbastanza facile da comprendere, minore possibilità di commettere errori di progettazione...

Svantaggi: se si stratifica eccessivamente il software, aumenta il numero di interfacce, aumenta il numero di strati che devono essere attraversati da una richiesta, aumentano i tempi...

Architettura a Microkernel (Client/Server)

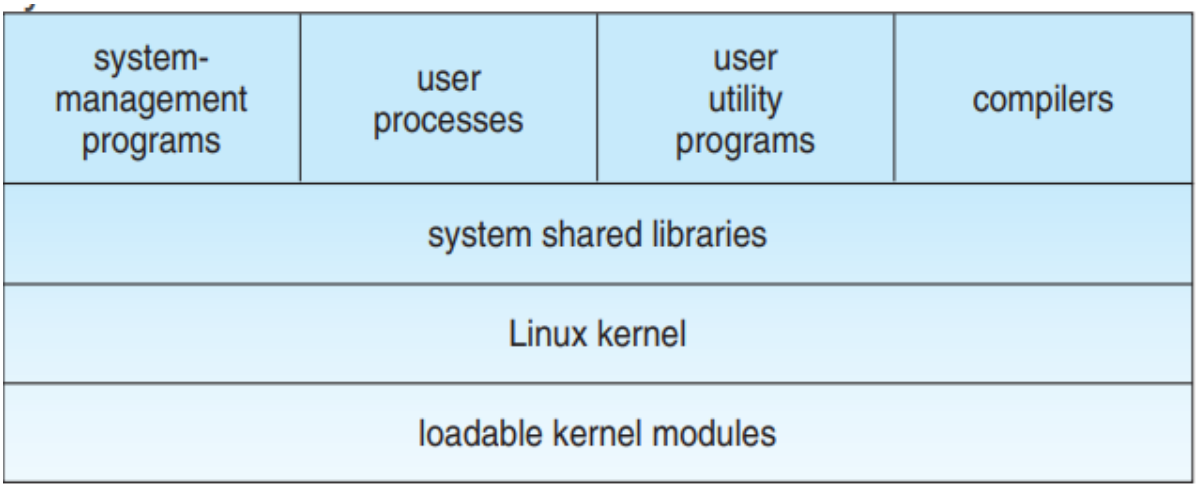


Si progetta il sistema operativo rimuovendo dal kernel tutti i componenti non essenziali, realizzandoli come programmi di livello utente e di sistema. Ne risulta un kernel di dimensioni assai inferiori. In generale, un microkernel offre i servizi minimi di gestione dei processi, della memoria e di comunicazione. Lo scopo principale del microkernel è fornire funzioni di comunicazione tra i programmi client e i vari servizi, anch'essi in esecuzione nello spazio utente. La comunicazione tra i moduli viene gestita dal microkernel.

Benefici:

- è più facile estendere un microkernel
- più sicuro
- si adatta più facilmente ai sistemi distribuiti
- più affidabile (meno codice gira in modalità kernel)
- è più facile fare il porting del SO su nuove architetture

Architettura Ibrida



Le architettura a microkernel sono state ritenute inutilmente troppo complicate.

Mischia i vantaggi dell'architettura stratificata e dell'architettura monolitica. Monolitica ma con moduli. È possibile aggiungere nuovi moduli linkandoli dinamicamente. Alcune parti del Sistema Operativo vengono divise in moduli. I moduli permettono di non dover ricompilare l'intero kernel ogni volta. Al caricamento il modulo viene linkato dinamicamente al kernel e viene eseguito in kernel mode; se contiene degli errori può danneggiare il kernel (il sistema si può bloccare).

Linux ha tre componenti principali: Kernel, System Libraries e SystemUtilities.
- Il
kernel

- Realizza tutte le funzioni del sistema operativo e serve per isolare il resto del sistema dalle caratteristiche dell'hardware
- Viene eseguito in kernel mode ed ha accesso a tutte le risorse fisiche del sistema
- Tutto il codice del kernel e le strutture dati sono mantenuti nello stesso spazio degli indirizzi

- Le **librerie di sistema**

- Definiscono un insieme standard di funzioni attraverso le quali le applicazioni interagiscono con il kernel
- Vengono eseguite in user mode

- Le **system utilities**

- Eseguono dei compiti di gestione specializzati
- Vengono eseguite in user mode

I moduli del kernel
– Sono sezioni del kernel che possono essere compilate, caricate e scaricate in modo indipendente dal resto del kernel.
– Un modulo del kernel tipicamente serve per realizzare un driver di un dispositivo, un file system o un protocollo di rete.
– Questo permette a terze parti di distribuire driver o file system che non possono essere distribuiti con licenza GPL.
– Questo permette a linux di essere configurato con un kernel minimo, senza moduli esterni.

Architettura a Macchine Virtuali

Le macchine virtuali sono nate come conseguenza della base dell'idea delle architetture stratificate.

L'idea alla base delle macchine virtuali è di astrarre dalle unità hardware del singolo computer progettando per ciascuna unità un ambiente esecutivo software diverso, così da dare l'impressione che ognuno di loro giri sulla propria macchina.

Il sistema operativo può dare l'impressione che ogni processo sia dotato del proprio processore e del proprio spazio di memoria (virtuale).
Ogni processo ospite può usufruire di una copia (virtuale) del calcolatore sottostante. Solitamente il processo ospite è un sistema operativo. In questo modo una singola macchina fisica può far girare più sistemi operativi concorrentemente, ciascuno sulla sua macchina virtuale

- Vantaggi: Protezione del dispositivo e delle sue risorse, una macchina virtuale è il veicolo ideale per la ricerca e lo sviluppo di nuovi SO. Infatti, lo sviluppo viene fatto sulla macchina virtuale e questo non danneggia il normale funzionamento della macchina.
-
Svantaggi: Difficile realizzazione, utile nelle applicazioni di cloud-computing (JVM)