Sistemi operativi

Created	@March 5, 2026 9:36 PM
Class	sistemi operativi
Lia&Simo	Ⓢ SIMONA LI CALZIⒾ lia🦋

Info importanti

Esame: teoria(65%) e lab(35%). la teoria e la pratica si possono fare in appelli diversi. la teoria è sia test che orale. la prova di teoria se superata scade a maggio dell'anno successivo. , Il test è a risposta aperta ci sono anche esercizi.

link:https://diraimondo.dmi.unict.it/teaching/sistemi-operativi/

le slide (4/5 unità vengono caricate quando si finisce l'argomento) : https://git.crypto.dmi.unict.it/teaching/operating-systems.2024-2025 (ci sono quelle dell'anno scorso).

Lezione 5 Marzo

Cos'è un sistema operativo?

Un moderno calcolatore è formato da: uno o più processori, memoria centrale, dischi, stampanti e altre periferiche I/O.

Per gestire tutte le componenti serve uno strato intermedio software: il Sistema Operativo.(Sistema operativo software che gestisce l'hardware)

I processi sono dei programmi che una volta mandati in esecuzione diventano un'entità attiva. Nei sistemi multiprogrammati troviamo più processi.

-Programma: entità statica

-Processo: entità dinamica

-Multiprogrammazione: capacità del sistema operativo di gestire più processi.

Qualunque elaboratore che ha delle risorse da gestire ha un sistema operativo, anche se non interagiamo direttamente con esso ma con delle applicazioni.

Tra i processi e l'hardware c'è questo strato software che possiamo indicare come sistema operativo, che può essere visto da due punti di vista. Punto di vista basso (hardware) quindi il SO è un gestore di risorse; punto di vista alto(utente) chi utilizza un dispositivo che in realtà interagisce con le applicazioni.

Modalità di esecuzione:

1. Kernel Mode (Supervisor Mode): È una modalità privilegiata. Il codice in esecuzione nella CPU può eseguire tutte le azioni previste dalla CPU.

Può eseguire:

Manipolazione diretta della memoria (allocazione, protezione, mappatura degli indirizzi

fisici)

Accesso diretto all'hardware e ai registri di controllo

Gestione delle interruzioni e delle eccezioni

Configurazione della MMU (Memory Management Unit)

I processi eseguiti in modalità kernel vengono definiti **processi multiprogrammati:** possono avviare autonomamente altri programmi e/o sottoprocessi secondo quanto stabilito dalla loro logica interna.

I processi lavorano in spazi separati tra di loro, il Sistema Operativo agisce come un'entità separata e superiore che arbitra l'accesso alle risorse.

Il **Kernel** è il nucleo del Sistema Operativo. Non è hardware, è software. A differenza dei normali programmi, il kernel è una componente **residente in memoria** (viene caricato nella RAM all'avvio del sistema e da quel momento in poi ha il controllo diretto dell'hardware)
Il kernel, in particolare, è progettato per operare in maniera **parallela e compartimentata**.

Questa caratteristica è fondamentale per due ragioni principali:

1. **Efficienza del sistema**: consente di gestire più richieste contemporaneamente senza colli
di bottiglia.

2. **Isolamento dei conflitti**: processi che competono per le stesse risorse vengono arbitrati
dal kernel, evitando stati inconsistenti o deadlock.

Il codice in esecuzione nella CPU ha dei limiti, ci sono delle istruzioni privilegiate e non privilegiate.

2. User Mode: È una modalità **non privilegiata**. (ES. riprogrammare il controller della memoria o accedere direttamente al disco.)

Quando un'applicazione ha bisogno di una risorsa hardware (es. leggere un file), non può farlo da sola. Deve invocare una **System Call** (chiamata di sistema). In quel momento, la CPU effettua un "trap" (un'interruzione software), passa dalla modalità utente alla modalità kernel, esegue il compito richiesto in modo sicuro e poi torna alla modalità utente.

È importante non cadere nell'errore di credere che *tutto* il codice eseguito in modalità utente sia completamente estraneo al kernel.

In pratica, gli utenti interagiscono con il sistema tramite interfacce grafiche (**GUI — Graphical User Interface**), le quali consentono di compiere operazioni tipicamente riservate al sistema operativo in modo accessibile e sicuro, senza necessità di interagire direttamente con il kernel.

Chiamate di Sistema (System Call)

Una system call (chiamata di sistema) **è il meccanismo fondamentale tramite cui un programma in esecuzione (user mode) richiede un servizio al kernel del sistema operativo, che opera con privilegi elevati (kernel mode)**.

Esempio: Apertura di un file

Quando un utente (o un programma) richiede l'apertura di un file, si innesca la seguente:

1. **L'applicazione** esegue una serie di operazioni predefinite (es. chiamata a open() in C).
2. **La CPU passa in modalità kernel**.
3. **Il kernel** esegue una serie di controlli e operazioni:
   Verifica l'esistenza del file nel filesystem.

   Controlla i permessi di accesso dell'utente richiedente
   Alloca le strutture dati necessarie
   Restituisce il controllo all'applicazione con il risultato

Il passaggio da modalità utente a modalità kernel e ritorno avviene in maniera trasparente per l'utente finale, ma è un meccanismo fondamentale per garantire la **sicurezza** e l'**integrità** del
sistema.

Astrazione

Il sistema operativo astrae l’interfaccia, usa l’hardware e offre ai programmi applicativi una propria interfaccia che sia ad alto livello, più comoda da gestire.

Uno dei concetti chiave nell'architettura dei sistemi operativi è quello di **astrazione**. Il kernel opera a un livello di programmazione molto basso, a stretto contatto con l'hardware. Operazioni che sembrano banali nascondono in realtà una complessità considerevole.

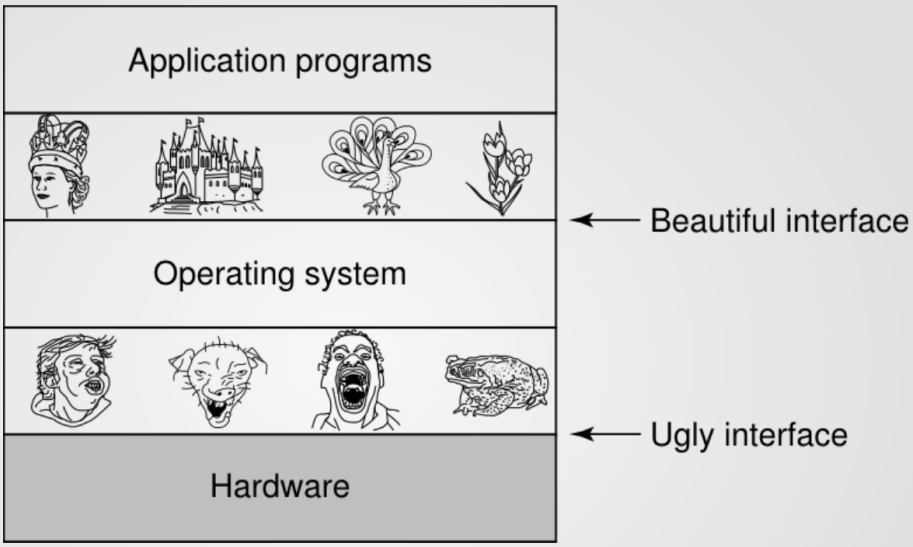
L'obiettivo dell'astrazione non è elencare queste complessità, ma **nasconderle**. Il kernel espone all'utente (e ai programmi applicativi) un'interfaccia semplificata — un insieme di istruzioni ad alto livello — che permette di eseguire operazioni complesse senza dover conoscere i dettagli implementativi dell'hardware sottostante. Questo rende il sistema più usabile e portabile.

Cpu e memoria virtuali

Su un protocollo posso avere una o più CPU, nel caso più semplice ne abbiamo una sola, sappiamo che si possono eseguire più processi contemporaneamente e questo può essere visto in tanti modi , ma anche come un’astrazione in cui il Sistema Operativo crea delle **CPU VIRTUALI**. Fa credere ad ogni processo di avere una propria CPU ma in realtà sta gestendo le n CPU virtuali con l’unica fisica che ha. Questo viene chiamato **time sharing.**

La **Memoria Virtuale** è un'astrazione fondamentale fornita dal Sistema Operativo che permette a ogni processo di operare in un ambiente di lavoro isolato e apparentemente illimitato.

Il sistema operativo come macchina estesa



- concetto di **astrazione**

L’astrazione non è solo una semplificazione ma viene utilizzata dal SO per rendere l’hardware utilizzabile.

- **Approccio Bottom-Up:** Il SO è un **Resource Manager**. Gestisce la complessità dei segnali elettrici e dei registri hardware, offrendo in cambio un'interfaccia pulita (File, Processi).
- Gestire una risorsa significa arbitrare i conflitti(più processi potrebbero voler accedere ad uno stesso file), fornirne un utilizzo ordinato e coordinato.

Multiplexing

Un sistema operativo moderno deve essere in grado di gestire:

Più programmi in esecuzione contemporanea
Più utenti connessi simultaneamente

Per farlo in maniera ordinata e controllata, si ricorre al concetto di **Multiplexing**.

Strategie di gestione

1. MULTIPLEXING NEL TEMPO: Si usa per risorse che non possono essere divise fisicamente (CPU). Passare da un processo P1 , all'altro P2, significa che il SO deve fare sia un lavoro di salvataggio (i dati dell'unica CPU fisica che ho) e di ripristino(ricaricare gli stessi valori lasciati nei registri della CPU, sostituire P2 con P1).

Il multiplexing nel tempo permette di creare CPU virtuali.

2. MULTIPLEXING NELLO SPAZIO: Si usa per risorse che possono essere spezzettate (RAM). Rientra in questo caso tutto quello che riguarda degli spazi di memoria da gestire. Diversi processi occupano contemporaneamente porzioni differenti della risorsa, il Sistema Operativo divide la RAM in blocchi e li assegna ai processi dove c'è spazio, garantendo l'**isolamento**.

Il multiplexing nello spazio permette di creare l'astrazione della memoria virtuale.

> 💡 **Perché è fondamentale? Senza multiplexing, ogni programma dovrebbe attendere che tutti gli altri terminassero prima di poter ottenere l'uso della CPU o della memoria. I sistemi moderni sarebbero di fatto inutilizzabili.**

- **Thread:** Sono flussi di esecuzione multipli **dentro lo stesso processo**. Condividono lo stesso spazio di memoria (stesse variabili, stessi file aperti) ma hanno registri e stack separati.

In un programma multi-thread : il primo thread esegue una procedura , il secondo un' altra ecc.. ma tutto questo avviene nello stesso ambiente.

Flusso di Esecuzione (Execution Flow)

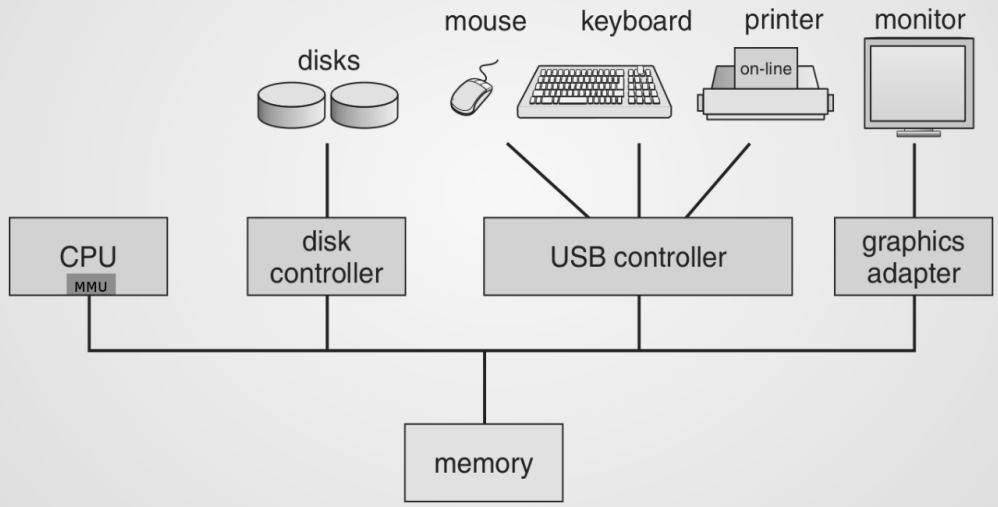
Il **flusso di esecuzione** descrive la sequenza ordinata di istruzioni che la CPU elabora per
portare a termine un programma. In un sistema multiprogrammato, questo flusso non è lineare:
il sistema operativo può interrompere l'esecuzione di un processo (tramite un'**interruzione** o allo scadere del suo *time slice*), salvarne lo stato nel **PCB** (Process Control Block), e riprendere l'esecuzione di un altro processo dallo stesso punto in cui era stato sospeso.

Questo meccanismo, noto come **context switch**, è ciò che rende possibile l'esecuzione parallela apparente su un singolo processore.

Il Calcolatore

Uno sguardo all'hardware

- Architettura (semplificata) di un calcolatore



Un calcolatore (in maniera generica o astratta)è caratterizzato dalla presenza di una o più CPU, ogni CPU è collegata alle periferiche tramite BUS, solitamente specializzati, che ottimizzano la comunicazione tra CPU e le varie componenti periferiche dell'hardware.

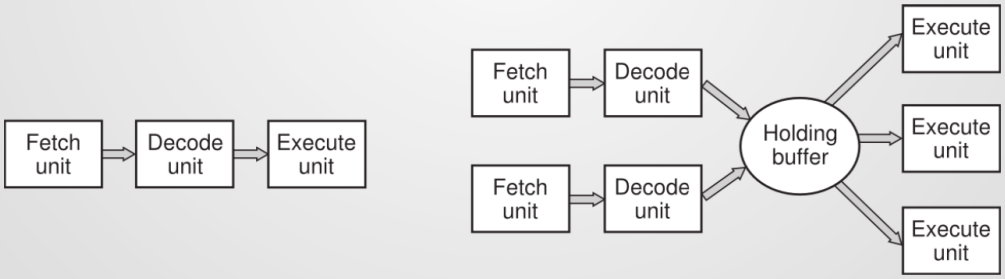
Le componenti da attenzionare sono:

RAM, memoria volatile(limitata ma veloce e direttamente accessibile dalla CPU).

Processore

Il Processore

- **Ciclo di base**: prelevamento (fetch), decodifica, esecuzione
- Registri particolari:
  - **Program Counter** (PC);
  - **Stack Pointer** (SP);
  - **Program Status Word** (PSW).
- Progettazioni avanzate: **pipeline, cpu superscalare**
  - non del tutto trasparenti al SO



Il processore è un'unità capace di unire codice, rilevato dal classico ciclo di fetch, decode, execute:

preleva dalla RAM la prossima istruzione da eseguire;

la decodifica, capendo qual è la sua semantica e gli operandi necessari;

nel momento in cui viene decodificata poi viene eseguita con operazioni logiche, aritmetica o altro...

La **RAM** funge da memoria primaria ad alta velocità per i dati e le istruzioni in uso. Per accellerare ulteriormente l'accesso, utilizza anche la **cache**, posizionata vicinissima al processore.

Dei registri generici della CPU ce ne occorrono un numero limitato, di appoggio per i dati provenienti dalle varie operazioni svolte nella CPU; ma ci sono anche dei "registri specifici" per la gestione del ciclo fetch, decode ed execute si chiamano UNIT.

Il **Program Counter** è un registro che indica alla CPU qual è l'indirizzo della prossima istruzione da eseguire. o prende l'indirizzo successivo, o se siamo in caso di SALTO va a reimpostare l'indirizzo successivo. Il ciclo inizia proprio con l'istruzione localizzata esattamente nel PC.

Le esecuzioni effettuate nella CPU vengono affiancate da una struttura esterna, lo stack.

Lo stack di esecuzione di un programma è gestito come se fosse una pila, push e pop sempre sulla testa. In questa memoria di appoggio alloco variabili(chiesto di istanziare variabile di nome A nella memoria) e viene inserito nello stack tramite un push, questo permette che quando la procedura si chiude(quando la routine dove avevo dichiarato A si chiude) vengono fatti dei pop che deallocano quelle locazioni di memoria allocate nella funzione. A potrebbe anche essere una procedura che ha argomenti(all'interno dello stack vengono memorizzati anche l'indirizzo di ritorno e gli argomenti x e y e inseriti nel frame di attivazione che sto creando in locazioni prefissate)

alla fine della procedura, sa quanti elementi deve deallocare e quali sono i valori di ritorno. Ciò accade anche quando una procedura A chiama una procedura B. se A chiama B allora creo frame di attivazione per A e per B e anche per altri successivi che siano C,D ecc.. ma attenzione! il valore di ritorno di C sarà B, di B sarà A e cosi via...

tutto questo funziona anche per la ricorsione, infatti si utilizza uno stack per darle vita. Per crearlo, devo avere spazio in memoria e un puntatore alla testa(memorizzato nello **Stack Pointer**).

Lo **Stack Pointer** è un registro della Cpu che si occupa di portare la CPU nella testa dello stack, registro importante che deve essere presente e lavora a stretto contatto con CALL, RETURN...

(allocazione con new non va nello stack ma nell'heap)

Altro registro importante è il Program Status Word **(PSW)**, o registro di stato, **è un registro hardware fondamentale della CPU che memorizza l'esito dell'ultimo compare attuale di un programma in esecuzione**. Contiene bit di flag che indicano i risultati delle operazioni (es. zero, riporto, overflow) e bit di controllo per la gestione degli interrupt e i privilegi.

La PSW è un'area di memoria o un registro hardware che racchiude lo stato istantaneo del processore. Quando si verifica un evento inaspettato o una richiesta esterna, il contenuto della PSW viene salvato, permettendo al processo interrotto di riprendere l'esecuzione esattamente da dove si era fermato.

w

Componenti tipici:

- **Condition Codes (CC):** Risultati di operazioni (es. zero, overflow, riporto).
- **Modalità di Privilegio:** Indica se la CPU opera in modalità "Kernel/Supervisore" o "Utente".
- **Abilitazione Interrupt:** Bit che attivano o disattivano la ricezione di interrupt esterni.
- **Program Counter (PC):** Spesso la PSW include o lavora in tandem con il PC, che contiene l'indirizzo della prossima istruzione da eseguire

- **Kernel Mode (Supervisore):** La PSW ha un bit impostato che permette l'esecuzione di istruzioni privilegiate (accesso diretto all'hardware, gestione della memoria).

- **User Mode (Utente):** La PSW indica una modalità a privilegi limitati. Se un programma utente cerca di eseguire istruzioni critiche (es. I/O), la CPU verifica la PSW e genera un'eccezione (trap).

PSW è il registro che permette il **Context Switch** (cambio di contesto):

- **Flag di Condizione:** Fondamentali per le decisioni logiche (se il flag `Zero` è attivo dopo un `compare`, allora le due variabili erano uguali).
- **Protezione (Kernel vs User Mode):** Questo è il pilastro della sicurezza informatica moderna. Impedisce a un normale programma (User Mode) di "distruggere" il sistema operativo, limitando l'accesso diretto all'hardware