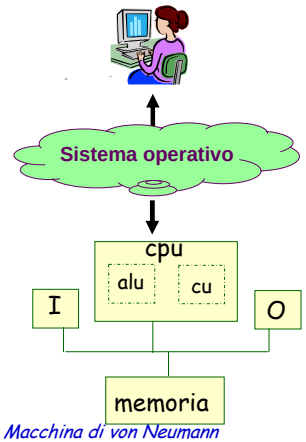
Lezione 2:

La struttura e funzioni dei Sistemi Operativi

- componenti di un sistema operativo
- servizi dei sistemi operativi
- modelli di architetture dei sistemi operativi
- Processo di compilazione e linking dei programmi

Compiti del S.O.



Macchina di von Neumann

Un S.O. ha il compito di rendere semplice (all'utente), l'utilizzo del calcolatore

Cosa deve fare un S.O. ?

Interazioni con:
- CPU
- Memoria
- Dispositivi di I/O

Componenti di un sistema operativo

Data la complessita' di un sistema di calcolo, i moderni sistemi operativi si possono progettare e gestire solo se si individuano al loro interno dei sottosistemi

Sottosistemi di un sistema operativo

- Gestione dei processi (gestione della CPU)
- Gestione della memoria centrale
- Gestione del file system
- Gestione del sistema di I/O
- Gestione della memoria secondaria
- Gestione del networking
- Gestione della protezione
- Interprete dei comandi

Gestione dei processi

Un processo è un programma in esecuzione.

necessita di alcune risorse per assolvere il proprio compito: tempo di CPU, memoria, file e dispositivi di I/O.

- Il sistema operativo è responsabile delle seguenti attività relative alla gestione dei processi:
  - Creazione e cancellazione di processi (utente e di sistema).
  - Sospensione e riattivazione di processi.
  - Fornire meccanismi per:
    - sincronizzazione di processi;
    - comunicazione fra processi;
    - gestione dei *deadlock*.

Gestione della memoria centrale

La memoria è una sequenza di parole o byte, ciascuna con un proprio indirizzo.

Fornisce un supporto rapidamente accessibili per la memorizzazione dei dati (ma volatile) condiviso dalla CPU e dai dispositivi di I/O.

- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione della memoria centrale:
  - Tener traccia di quali parti della memoria sono attualmente usate e da chi.
  - Decidere quali processi caricare in memoria quando vi è spazio disponibile.
  - Allocare e deallocare lo spazio di memoria in base alle necessità.

Gestione del file system

Un file è una collezione di informazioni correlate

- Permette una visione logica uniforme del processo di memorizzazione
- e' l'unità di memorizzazione logica

.

- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione di file:
  - Creazione e cancellazione di file e directory.
  - Supporto alle funzioni elementari per la manipolazione di file e directory.
  - Associazione dei file ai dispositivi di memoria secondaria.
  - Backup di file su dispositivi di memorizzazione non volatili

Gestione del I/O

La periferiche di I/O interfacciano l'utente con il computer

- Sono gestiti dai dispositivi di I/O
- Consentono di leggere, scrivere dati in maniera "friendly".

.

- Il sistema operativo è responsabile delle seguenti attività connesse alle operazioni di I/O:
  - gestione della memoria comprendente il buffering, il caching e lo spooling.
  - nascondere le caratteristiche di specifici dispositivi hardware

Gestione della memoria secondaria

La memoria secondaria e' un supporto per salvare i dati contenuti della memoria centrale.

- i dischi sono il principale mezzo di memorizzazione secondaria , sia per i programmi che per i dati
- è un dispositivo di memorizzazione non volatile .

- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione della memoria secondaria:
  - Gestione dello spazio libero.
  - Allocazione dello spazio.
  - Scheduling del disco.

Gestione del networking

Una rete e' un dispositivo per connettere un insieme di processori che non condividono né la memoria né il clock.

- La comunicazione avviene secondo un insieme di regole (protocollo)
- Consente la realizzazione di sistemi distribuiti
- Aumenta le funzionalita' dell'ambiente di calcolo

- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione del networking:
  - Uniformare l'accesso alle risorse.
  - Convertire dati tra differenti formati.
  - Proteggere il sistema da accessi indesiderati

Gestione della protezione

L'hardware di protezione e' l'insieme dei dispositivi per la protezione delle componenti del calcolatore (registri, bit mode,…)

Consente di controllare l'accesso da parte di processi o utenti a risorse del sistema di calcolo (di sistema e di altri utenti).

- Il sistema operativo è responsabile delle seguenti attività connesse alla protezione:
  - Distinzione fra uso autorizzato e non autorizzato di una risorsa.
  - Specifica dei controlli da imporre.
  - Fornire una modalità di imposizione

Interprete dei comandi

- È il programma che legge ed interpreta le istruzioni di controllo relative a:
  - creazione e gestione dei processi;
  - gestione di I/O;
  - gestione della memoria secondaria;
  - gestione della memoria centrale;
  - accesso al file system;
  - Comunicazione su rete;
  - protezione.
- Esistono interpreti "amichevoli" con finestre, o interpreti più potenti (ma più complessi) basati su interfaccia a carattere.
- viene chiamato in vari modi:
  - interprete del linguaggio di comando
  - *shell* (in UNIX)

Servizi del sistema operativo

Un programma, per essere mandato in esecuzione, ha bisogno che il sistema operativo gli permetta di accedere alle risorse del sistema

Esempi:
- usare la cpu
- accedere ai dispositivi di I/O
- accedere alla memoria secondaria
- usare la memoria centrale
- …

I programmi richiedono servizi ai sistemi operativi

Servizi del sistema operativo

- Per l'esecuzione di programmi (capacità di caricare un programma in memoria e mandarlo in esecuzione.)
- Per le operazioni di I/O (capacita' di accedere ai dispositivi di I/O)
- Per la manipolazione del file system (capacità dei programmi di leggere, scrivere e cancellare file).
- Per la comunicazioni tra processi (permettere scambio di informazioni fra processi in esecuzione sullo stesso elaboratore o su sistemi diversi, connessi per mezzo di una rete.)
- Per il rilevamento di errori (assicurare una corretta elaborazione rilevando errori nella CPU e nella memoria, in dispositivi I/O o in programmi utente.)

Altri servizi del SO

Esistono servizi addizionali atti ad assicurare l'efficienza delle operazioni di sistema piuttosto che orientate all'utente.

- Per l'allocazione di risorse (a più utenti o a job multipli in esecuzione contemporanea.)
- Per la contabilizzazione dell'uso delle risorse (tener traccia di quali utenti usano quali e quante risorse del sistema.)
- Per la protezione (assicurare che tutti gli accessi alle risorse di sistema siano controllati. La sicurezza di un sistema comincia con l'obbligo di identificazione tramite *password* e si estende alla difesa dei dispositivi di I/O esterni quali modem e adattori di rete, da accessi illegali.)

Come accedere ai servizi di un S.O.?

Le Chiamate di Sistema (o System Call) rappresentano lo strumento per accedere ai servizi del sistema operativo

- Sono generalmente disponibili come istruzioni in linguaggio Assembler.
- Alcuni linguaggi, definiti al fine di sostituire il linguaggio Assembler per la programmazione dei SO, permettono di effettuare le chiamate di sistema come funzioni di libreria (ad es., C, C++).
- forniscono l'interfaccia fra un programma in esecuzione e il sistema operativo.

Tipi di chiamate di sistema

- **Controllo di processo**: *end, abort, load, execute, create/terminate process, get/set process attributes, wait/signal event, allocate/free mem.*
- **Manipolazione dei file**: *create/delete file, open, close, read, write, reposition, get/set file attributes.*
- **Gestione dei dispositivi**: *request/release device, read, write, reposition, get/set device attributes, attach/detach devices.*
- **Gestione delle informazioni**: *get/set time/date, get/set system data, get/set file/device attributes.*
- **Comunicazione**: *create/delete communication connection, send/receive messages, transfer status information.*

Esempio: copia di un file

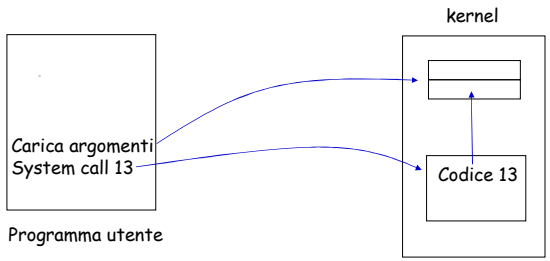
- stampa un messaggio che chiede i nomi dei file
- leggi i nomi dei file
- apri il file sorgente
- crea il file di destinazione
- leggi dal file sorgente
- scrivi sul file destinazione
- chiudi il file sorgente
- chiudi il file destinazione
- stampa un messaggio di successo

Anche un semplice programma puo' fare uso massiccio di chiamate di sistema

Passaggio parametri

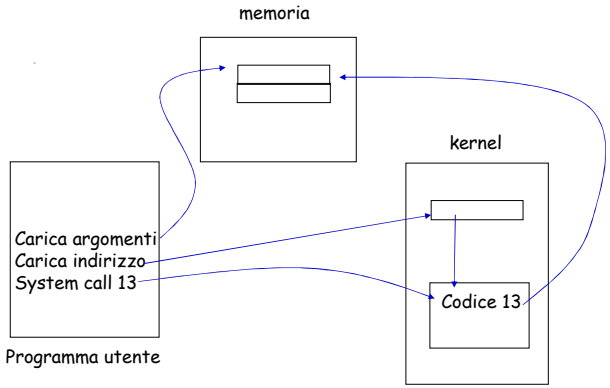
- Tre metodi generali sono impiegati per passare i parametri tra un programma in esecuzione e il sistema operativo attraverso chiamate di sistema
  - Impiego di *registri* del kernel (passaggio di parametri tramite registri).
  - Memorizzazione dei parametri in una tabella in memoria, e passaggio dell'indirizzo della tabella come parametro in un registro.
  - *Push* dei parametri nello stack da parte del programma. Il SO recupera i parametri con un *pop*.

1: impiego dei registri



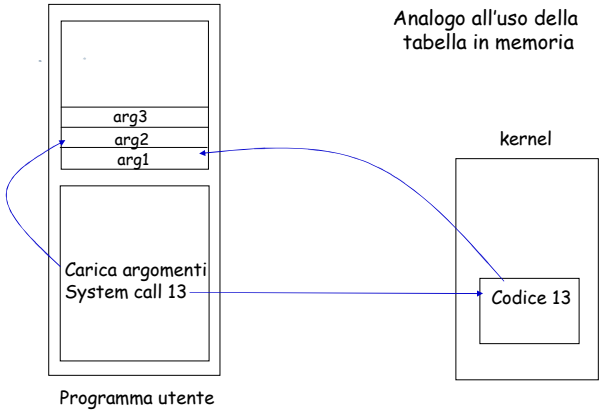
Vantaggio:
Meccanismo semplice e rapido
Svantaggio:
E se gli argomenti sono piu' dei registri?

2: impiego di tabella in memoria



Meccanismo usato da Linux

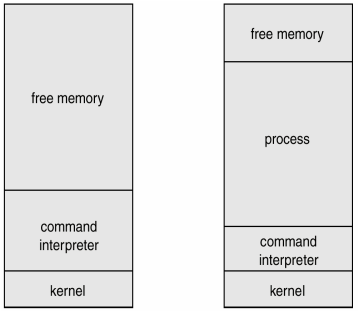
3: impiego dello stack



Esempio: controllo dei processi in MS-DOS

1. Allo startup in memoria c'e' solo l'interprete
2. Viene caricato un programma e viene sovrascritto parte dell'interprete
3. Al termine del programma si conserva lo stato di uscita
4. L'interprete riprende l'esecuzione e si ripristina

Sempre un solo processo alla volta



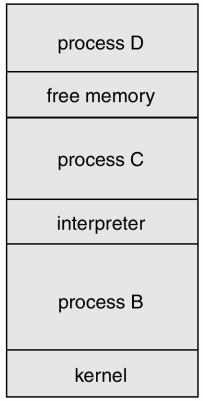
Esempio: controllo dei processi in UNIX

Unix e' un sistema multiprogrammato

L'interprete crea un nuovo processo per ogni programma

In processi possono essere in

- foreground (il controllo ritorna all' interprete dopo la fine del processo)
- background (il controllo torna subito all'interprete)



Programmi di sistema

- Da non confondere con le chiamate di sistema
- L'aspetto del SO per la maggioranza degli utenti è definito dai programmi di sistema, non dalle chiamate di sistema vere e proprie.
- I programmi di sistema forniscono un ambiente conveniente per lo sviluppo e l'esecuzione di programmi. I programmi possono essere suddivisi in:
  - Manipolazione di file e directory
  - Supporto a linguaggi di programmazione
  - Caricamento ed esecuzione di programmi
  - Comunicazioni
  - Programmi applicativi
- Possono far parte del codice dell'interprete o costituire un programma independente

programmi di sistema vs chiamate di sistema

Esempio: copia di un file

In Unix:

>> cp file1 file2

Programma di sistema

- stampa un messaggio che chiede i nomi dei file
- leggi i nomi dei file
- apri il file sorgente
- crea il file di destinazione
- leggi dal file sorgente
- scrivi sul file destinazione
- chiudi il file sorgente
- chiudi il file destinazione
- stampa un messaggio di successo

Chiamate di sistema

Riassumendo

Dal punto di vista dell'utente il S.O. fornisce servizi per
- l'esecuzione di programmi
- la manipolazione del file system
- la gestione dei dispositivi di I/O
- la comunicazioni tra processi
- il rilevamento di errori

Dal punto di vista del sistema il S.O. gestisce risorse
- cpu
- memoria centrale e secondaria
- file system
- sistema di I/O
- networking
- hardware di protezione

Architettura dei sistemi operativi

Nel tempo le funzionalita' (e quindi la complessita') dei sistemi operativi sono cresciute enormemente

Necessita' di una metodologia nella progettazione

4 tipologie di organizzazione
- architettura monolitica
- architettura stratificata
- Architettura a macchina virtuale
- architettura a microkernel

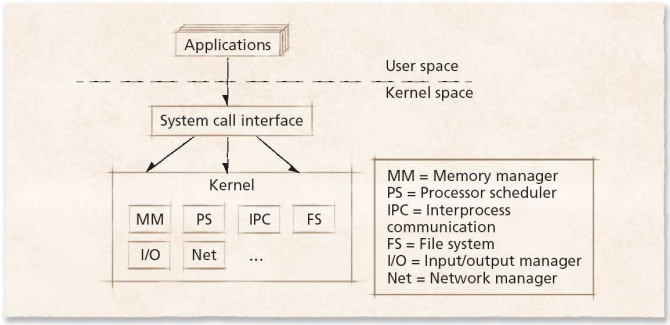
Architettura monolitica

- caratteristica dei primi sistemi operativi
- tutte le funzionalita' contenute nel kernel
- ogni componente puo' comunicare con tutte le altre
- esempi: MS/DOS, OS/360, linux

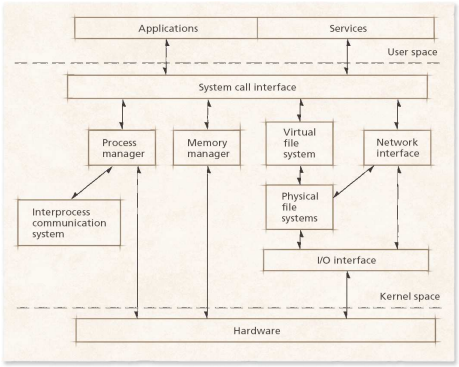
- Vantaggi:
  - efficienza

- Svantaggi
  - manutenzione e espandibilita' difficile
  - poco fault tolerant

Architettura monolitica



Esempio: Linux



Linux e' considerato un sistema con struttura monolitica, sebbene abbia caratteristiche di modularita'

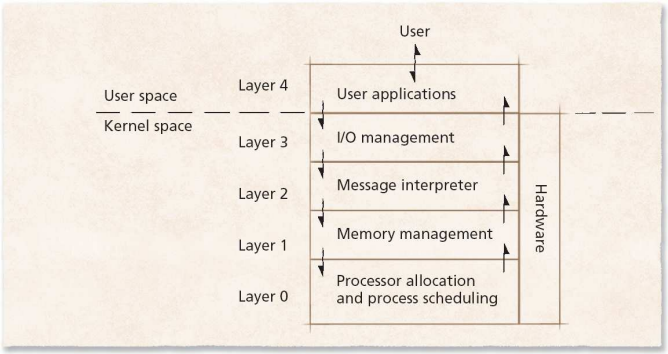
Architettura stratificata

- Il sistema operativo è suddiviso in un certo numero di strati (livelli), ciascuno costruito sopra agli strati inferiori.
- ciascuno strato comunica esclusivamente con gli strati immediatamente superiore e inferiore attraverso interfacce
- Funzionalita' ancora tutte nel kernel
- esempi: Windows XP, OS/2,

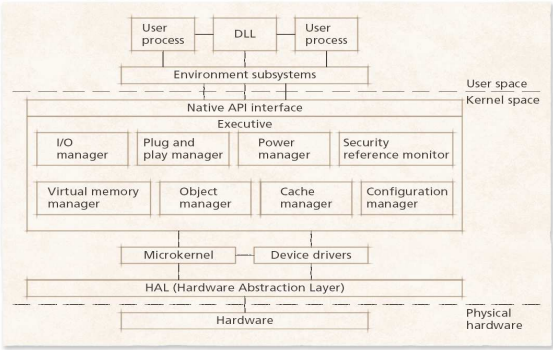
- Vantaggi
  - Modularita', facilita' di gestione

- Svantaggi
  - ancora sensibile ad attacchi esterni
  - Meno efficiente

Architettura stratificata



Esempio: Windows XP

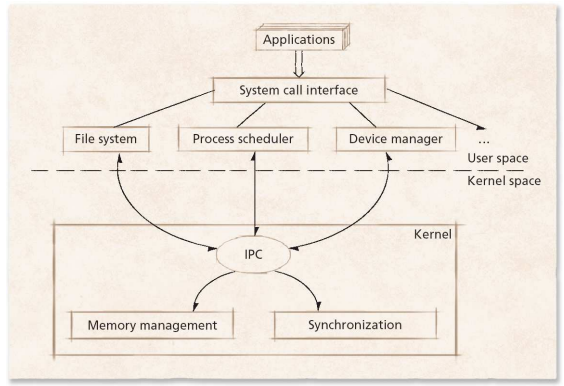


Architettura microkernel

- Metodologia piu' recente (anni 90)
- Quasi tutte le funzionalità del kernel vengono spostate nello spazio utente.
- Le comunicazioni tra i moduli del s.o. avvengono mediante scambio di messaggi (client/server) attraverso il kernel.
- Esempi: in parte Windows NT e Windows XP

- Vantaggi:
  - funzionalità del microkernel più semplici da estendere;
  - sistema più facile da portare su nuove architetture;
  - più affidabile (meno codice viene eseguito in modo kernel);
  - maggior sicurezza.

Architettura microkernel

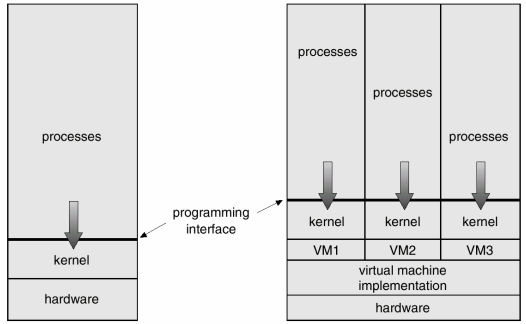


Macchine virtuali

- Una macchina virtuale porta l'approccio stratificato alle sue estreme conseguenze logiche. Sia l'hardware che il SO vengono trattati allo stesso modo.
- Una macchina virtuale crea una "immagine software" della macchina fisica sottostante.

In una macchina virtuale le risorse della macchina fisica vengono condivise in modo che il SO crei l'illusione che ciascun processo sia in esecuzione su un differente processore, con la sua propria memoria

Modelli di sistemi



tradizionale

Macchina virtuale

Macchine virtuali

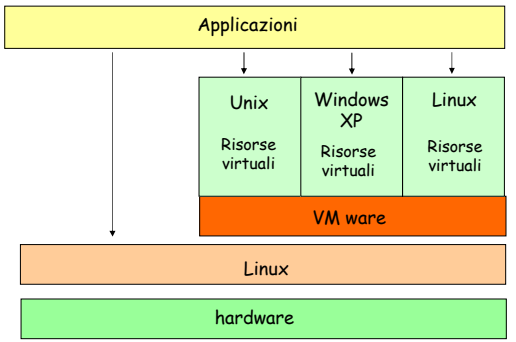
- Le risorse del computer fisico vengono condivise in modo da creare le macchine virtuali.
  - Lo scheduling della CPU può creare l'illusione che gli utenti abbiano un loro proprio processore.
  - Lo spooling e il file system possono fornire lettori di schede virtuali e stampanti in linea virtuali.
  - Un normale terminale per utente in time-sharing funziona come console per l'operatore della macchina virtuale.
- Esempio: s.o. VMS (Virtual Machine System) dell'IBM
  - Ogni utente "non vede" gli altri utenti

Vantaggi/svantaggi delle macchine virtuali

- ogni macchina virtuale è isolata da tutte le altre
  - fornisce una protezione completa delle risorse di sistema.
  - non permette una condivisione diretta delle risorse.
- Approccio indicato per la ricerca e lo sviluppo di sistemi operativi (Lo sviluppo del sistema è effettuato sulla macchina virtuale e così non disturba il normale funzionamento del sistema.)
- Possibilita' di simulare CPU differenti e aumentare la portabilita' di applicazioni esistenti
- difficile da implementare per il notevole sforzo richiesto per fornire un duplicato *esatto* della macchina sottostante.

Un esempio: VM ware

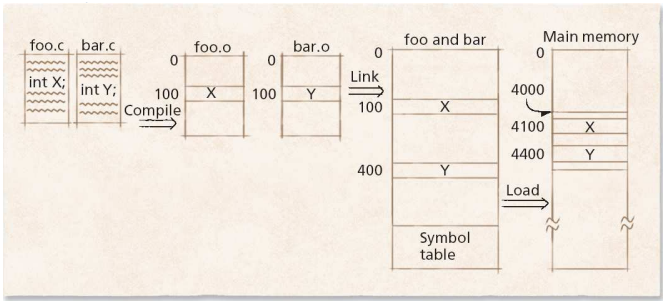
- Applicazione disponibile per Windows o Linux
- Disponibile per architetture Intel
- Consente l'emulazione di differenti s.o.



Alcuni concetti importanti nei sistemi operativi

- Prima che un programma scritto in linguaggio ad alto livello possa essere eseguito, sono necessarie alcune fasi:
  - **Compiling** : traduzione in liguaggio macchina
  - **Linking**: collegamento con altri moduli in linguaggio macchina da cui dipende (ad es. Le routine di I/O)
  - **Loading**: caricamento in memoria

Esecuzione di un programma



Compilazione

- Traduce un codice in linguaggio ad alto livello in codice macchina
  - Accetta un codice sorgente e ritorna un codice oggetto
  - Le fasi della compilazione sono:
    - Lexer
      - Separa i caratteri in parole chiave e nomi di variabili (tokens)
    - Parser
      - Raggruppa i tokens in statements sintatticamente corretti del linguaggio
    - Intermediate code generator
      - Converte statements in una sequenza di istruzioni macchina
    - Optimizer
      - Migliora l'efficienza dle codice e l'uso della memoria
    - Code generator
      - Produce il file oggetto contenente il codice in linguaggio macchina

Linking

- Linkers
  - Crea un singolo eseguibile a partire a piu' moduli
  - Integra moduli precompilati e librerie richieste dal programma
  - Assegna indirizzi relativi a differenti moduli
  - Risolve tutti i riferimenti esterni tra sottoprogrammi
  - Il collegamento puo' essere fatto prima dell'esecuzione (linking statico) o durante l'esecuzione (linking dinamico)

caricamento

- Il caricatore
  - Converte gli indirizzi relativi in indirizzi fisici
  - Sistema in memoria le istruzioni e i dati
- Tecniche per il caricamento:
  - Caricamento assoluto
    - Sistema il programma negli indirizzi specificati dal compilatore (assumendo gli indirizzi disponibili)
  - Caricamento rilocabile
    - Ridefinisce gli indirizzi del programma in base alla sua posizione effettiva in memoria
  - Caricamento dinamico
    - Carica i sottomoduli del programma quando servono