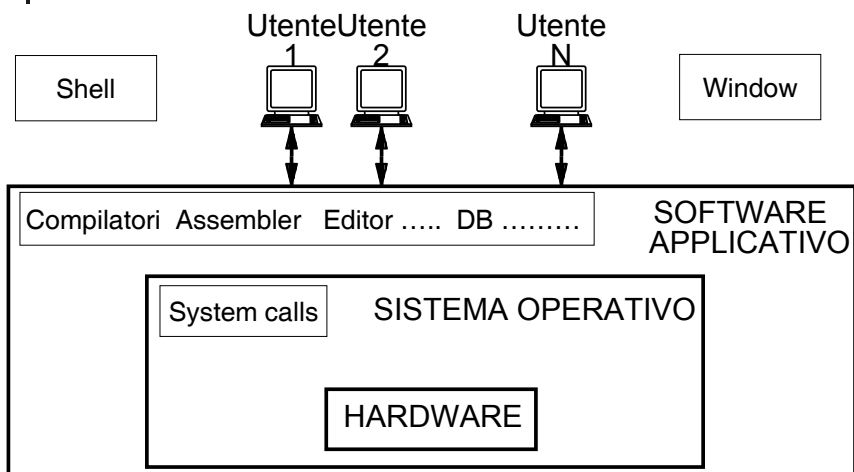


# Introduzione ai sistemi operativi

## Che cos'è un S.O.?



## Funzioni di un S.O

- Funzioni svolte da un sistema operativo:
  - Assegnazione delle risorse (CPU, memoria, etc.)
  - risoluzione dei conflitti
  - scelta dei criteri di assegnazione
- Controllo dell'esecuzione
  - system call
  - macchina estesa (o virtuale).
- Necessità di nascondere all'utente i dettagli HW legati ad un dispositivo
- Kernel – (nucleo) è il cuore del sistema operativo: il solo "programma" che gira sempre (tutti gli altri sono programmi applicativi).

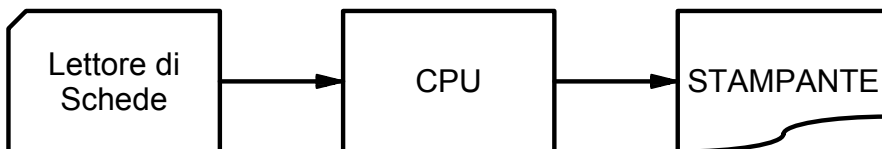
## Evoluzione dei S.O.

- Prima generazione
- Seconda generazione
  - Sistemi batch
  - Buffer di I/O
  - Spooling
- Terza generazione
  - Multiprogrammazione
  - Divisione di tempo, Sistemi interattivi
- Quarta generazione
  - Dos, Unix, Sistemi distribuiti, S.O. di rete

## Sistemi monoprogrammati (dedicati)

- Gestione sequenziale nel tempo dei programmi
  - Utilizzo CPU =  $T_p / T_t$  ( $T_p$  tempo di esecuzione del programma,  $T_t$  tempo di permanenza del programma nel sistema)
- Throughput: numero di programmi eseguiti nell'unità di tempo
- Bassa utilizzazione delle risorse

## Sistemi batch

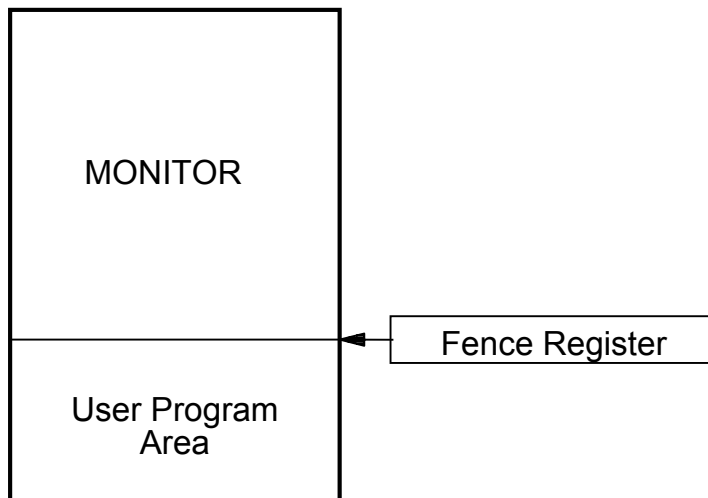


- Primo esempio di Sistema Operativo
- Un operatore disaccoppia il sistema dall'utente
- Utente  $\neq$  operatore
- Si usa un card reader

## Sistemi batch

- Automatic job sequencing – trasferisce automaticamente il controllo da un job ad un altro.
- Monitor residente
  - Inizialmente il controllo è del monitor
  - Il monitor trasferisce il controllo a un job
  - Quando il job finisce il controllo ritorna al monitor

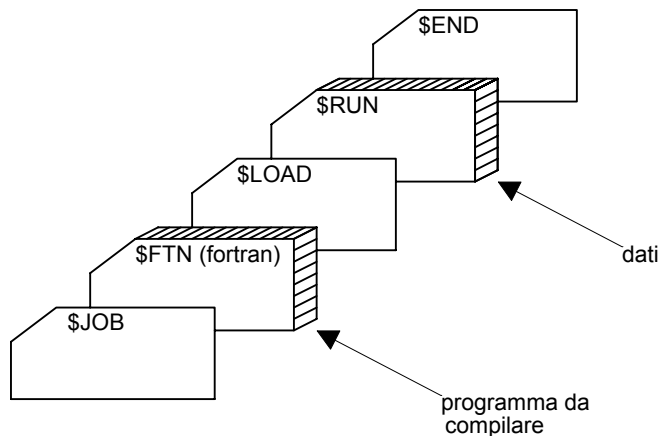
## Layout di memoria per sistemi batch



# Schede di controllo

- Problemi
  - 1. Come dare i comandi al monitor? Es. compilazione di un programma Fortran oppure esecuzione di un programma già compilato.
  - 2. Come fa il monitor a distinguere:
    - (a) Un job da un altro job?
    - (b) dati da programmi?
- Soluzione
  - Introduzione di schede di controllo

# Schede di controllo

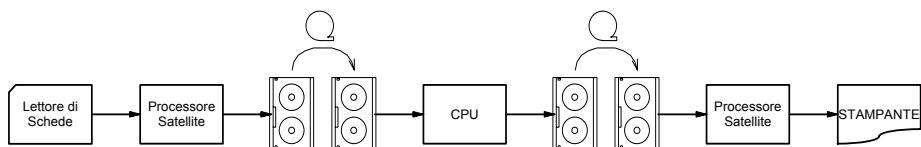


## Monitor residente

- Componenti del monitor residente:
  - Interprete delle schede di controllo – responsabile della lettura e dell'esecuzione delle istruzioni su ciascuna scheda
  - Loader – carica i programmi di sistema e quelli applicativi in memoria.
  - Device driver – conoscono le caratteristiche e le proprietà di tutte le periferiche di I/O del sistema di elaborazione.

## Sistemi batch off-line

Si velocizza l'elaborazione caricando i job in memoria da nastro magnetico, mentre la lettura delle schede e la stampa dell'output è effettuata off-line



## Spooling

---

- Si sovrappone l'I/O di un job all'elaborazione di un altro job.
- Durante l'esecuzione di un job, il SO
  - Legge il prossimo job dal card reader in un'area del disco (job queue - pool).
  - Stampa l'output di un job precedente prelevando le linee da disco e inviandole alla stampante.

## Sistemi multiprogrammati

---

- Gestione contemporanea di più programmi indipendenti
- Migliore utilizzazione delle risorse.
- Maggiore complessità del S.O.:
  - algoritmi per la gestione delle risorse.
  - protezione degli ambienti per i diversi programmi.
- Occupazione teorica della CPU=100%

## Sistemi time sharing

- Astrazione di più macchine virtuali
- Ogni utente ha un proprio programma in memoria
- Migliorano i tempi di attesa per i programmi corti
- Problema del Context Switching

## Sistemi paralleli

- Sistemi multiprocessore con più di una CPU in stretta comunicazione.
- Tightly coupled system - I processori hanno la memoria ed il clock in comune; la comunicazione avviene tipicamente utilizzando la memoria condivisa.
- Vantaggi dei sistemi paralleli:
  - Incremento del Throughput (job/ora)
  - Economia
  - Incremento dell'affidabilità



## Sistemi SMP

---

- Symmetric multiprocessing (SMP)
  - Ciascun processore esegue una copia identica del SO.
  - Più processi possono essere eseguiti contemporaneamente senza riduzione di prestazioni.
  - La maggior parte dei SO moderni gestiscono il SMP.



## Sistemi AMP

---

- Asymmetric multiprocessing (AMP)
  - A ciascun processore è assegnato un task specifico; il processore master schedula e alloca il lavoro per i processori slave.
  - Più comune su sistemi molto grandi.

## Sistemi real time

- Usati spesso come controllori in applicazioni dedicate quali il controllo di esperimenti scientifici, sistemi di controllo industriale, sistemi per l'acquisizione di dati, immagini mediche, ecc..
- Hanno vincoli temporali ben definiti: la correttezza di un task dipende dal tempo in cui un'operazione viene eseguita.

## Sistemi distribuiti

- Distribuiscono l'elaborazione su diversi processori.
- Loosely coupled system – ciascun processore ha una memoria locale; i processori comunicano con gli altri attraverso linee di comunicazione quali bus ad alta velocità o linee telefoniche.
- Vantaggi dei sistemi distribuiti.
  - Condivisione delle risorse
  - Incremento della velocità di calcolo – condivisione del carico
  - Affidabilità
  - Comunicazioni

## Componenti di un S.O.

- Gestore dei processi.
- Gestori della memoria principale e secondaria.
- Gestore dei dispositivi di I/O.
- Gestore dei file.
- Sistema di protezione.
- Gestione della comunicazione tra sistemi distribuiti.
- Interprete dei comandi.

## System call

- I programmi comunicano con il S.O. e richiedono ad esso particolari servizi tramite le chiamate di sistema (system call).
- Esempi: gestione dei file, sincronizzazione tra processi, invio di messaggi, etc.
- Importante possono essere disponibili:
  - direttamente come istruzioni nei linguaggi ad alto livello (esempio C)
  - Come istruzioni assembler

## System call

---

- Si usano tre metodi generali per il passaggio dei parametri tra un processo e il S.O.
  - Passaggio di parametri in registri.
  - Attraverso una tabella in memoria il cui indirizzo è passato come parametro in un registro.
  - Push dei parametri in uno stack da parte del programma, e pop dello stack da parte del kernel.

## Organizzazione di un S.O.

---

- Sistemi monolitici
  - Dos, Unix
- Sistemi a livelli
  - OS/2
- Macchine virtuali
  - VM/370
  - VMware

## Sistemi monolitici

---

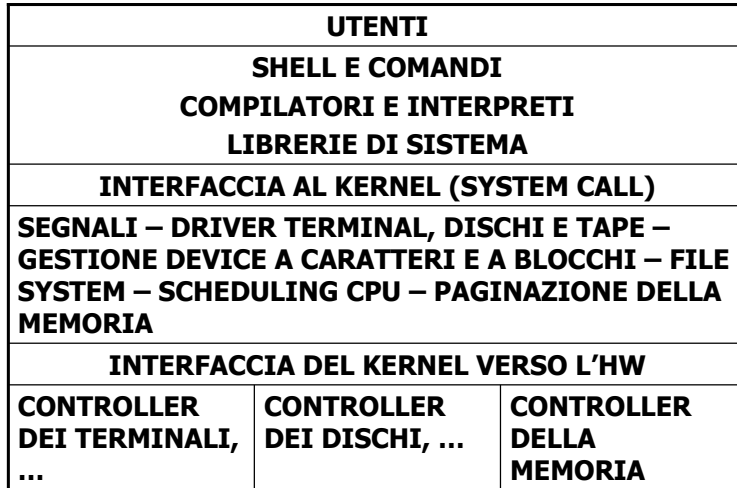
- Il S.O. è scritto come una collezione di procedure.
  - ogni procedura può chiamare qualunque altra procedura del sistema (nessun livello di gerarchia).
- Quando un programma utente invoca una system call, il controllo passa al S.O. che esegue la chiamata in modo kernel (o modo superuser)

## S.O. monolitici - Unix

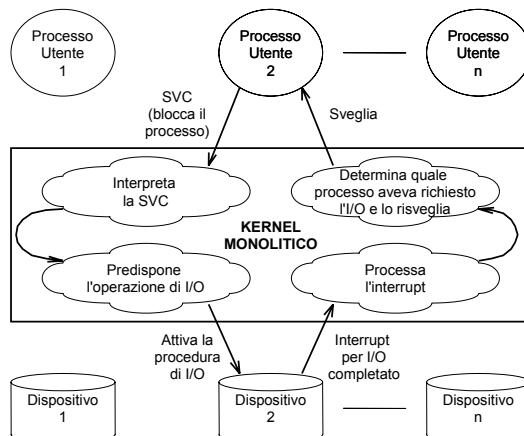
---

- UNIX consiste di 2 parti.
  - programmi di sistema
  - il kernel
    - È l'interfaccia al livello fisico dell'HW, attraverso le system call
    - Fornisce le funzioni relative al file system, scheduling della memoria e molte altre funzioni

# S.O. monolitici - Unix



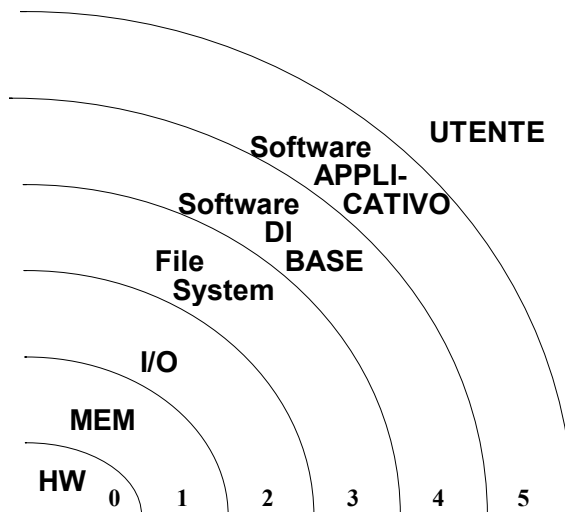
# S.O. monolitici - Unix



## Sistemi a livelli

- Le funzioni del sistema sono organizzate a livelli gerarchici.
- Ogni livello definisce un tipo di servizio e le modalità di utilizzo da parte del livello superiore.
- Ogni livello definisce una macchina virtuale, aggiungendo nuove funzionalità.
- Difficoltà nella realizzazione pratica della gerarchia.

## Esempio di S.O. a livelli



## Macchine virtuali

- Una macchina virtuale crea l'illusione di processi multipli, ciascuno eseguito da un proprio processore e su una propria memoria (virtuali).
- Il VM/370 crea più macchine virtuali; ogni macchina può ospitare un S.O. differente.
- VMware gestisce macchine virtuali:
  - Linux su Windows (NT/2000/XP) nativo
  - Windows su Linux nativo
  - ...

## Obiettivi di progetto

- Obiettivi dell'utente
  - facile da apprendere ed usare, affidabile, sicuro e veloce.
- Obiettivi del progettista
  - facile da progettare, implementare, e mantenere
  - flessibile, affidabile, senza errori ed efficiente.



## Meccanismi e politiche

---

- I meccanismi determinano come fare qualcosa.
- Le politiche decidono cosa deve essere fatto.
- La separazione delle politiche dai meccanismi è un principio molto importante perché consente la massima flessibilità se si decide di cambiare le decisioni politiche.