Universita` di Bologna

Corso di Laurea in Ingegneria Informatica

Sistemi Operativi L-A

A.A. 2003-2004

Prof. Anna Ciampolini

Cos’e` un Sistema Operativo ?

- È un programma (o un insieme di programmi) che agisce come **intermediario tra l’utente e l’hardware** del computer:
  - fornisce un **ambiente di sviluppo e di esecuzione** per i programmi
  - fornisce una **visione astratta** dell’HW
  - **gestisce** le risorse del sistema di calcolo

Il Sistema Operativo e l'Hardware

- Il sistema operativo interfaccia i programmi con le risorse HW:
  - **CPU**
  - **memoria** volatile e persistente
  - **dispositivi** di I/O
  - **Rete**
  - **…**
- Il S.O. *mappa* le risorse HW in **risorse logiche**, accessibili attraverso interfacce ben definite:
  - ✓ processi (CPU)
  - ✓ file (dischi)
  - ✓ Memoria virtuale (memoria)...

Cos’e` un Sistema Operativo ?

Cos’è un sistema operativo?

- Un programma che **alloca** le risorse del sistema di calcolo ai programmi e agli utenti:
  - ✓ CPU
  - ✓ Memoria
  - ✓ dischi
  - ✓ dispositivi di I/O
  - ✓ ...
- Un programma che **controlla** dispositivi e programmi allo scopo di garantire un funzionamento corretto ed efficiente.

Aspetti importanti di un S.O.

- **Struttura**: come è organizzato un S.O. ?
- **Condivisione:** quali risorse vengono condivise tra utenti e/o programmi? In che modo?
- **Protezione:** il S.O. deve impedire *interferenze* tra programmi/utenti. Con quali metodi?
- **Efficienza**
- **Affidabilità:** come reagisce il S.O. a malfunzionamenti (HW/SW) ?
- **Estendibilità:** è possibile aggiungere funzionalità al sistema ?
- **Conformità a Standard:** portabilita`, estendibilita`, apertura

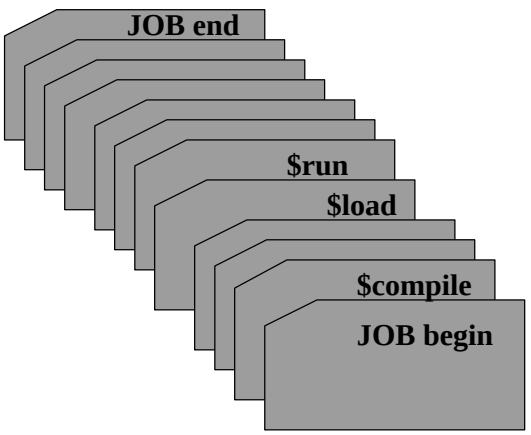
Evoluzione dei Sistemi Operativi

- **Prima generazione** (anni ‘50)
  - linguaggio macchina
  - dati e programmi su schede perforate
- **Seconda Generazione** (‘55-’65):

  Sistemi batch semplici
  - linguaggio di alto livello (fortran)
  - input mediante schede perforate
  - aggregazione di programmi in **lotti** (batch) con esigenze simili

Sistemi batch semplici

Batch: insieme di programmi (*job*) da eseguire **<u>in modo sequenziale</u>**.

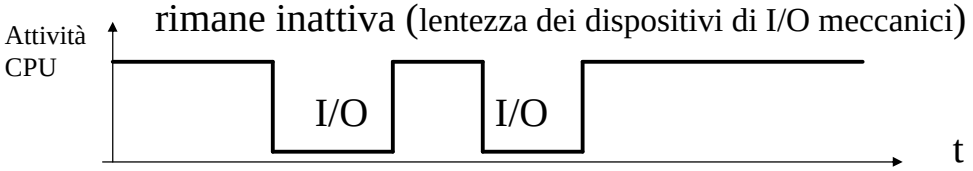


Sistemi batch semplici

Compito del Sistema Operativo (*monitor*):
trasferimento di controllo da un job (appena terminato) al prossimo da eseguire.

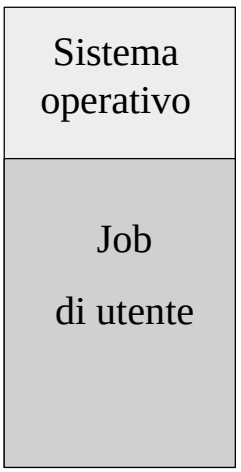
Caratteristiche dei sistemi batch semplici:

- sistema operativo residente in memoria (monitor)
- assenza di interazione tra utente e job
- scarsa efficienza:durante l'I/O del job corrente, la CPU rimane inattiva (lentezza dei dispositivi di I/O meccanici)



Sistemi batch semplici

In memoria centrale, ad ogni istante, è caricato (al più) un solo job:



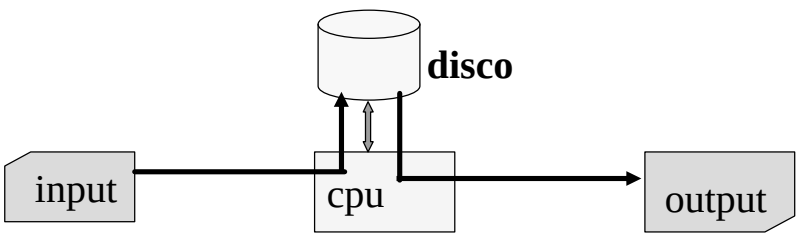
Configurazione della memoria centrale in sistemi batch semplici

Sistemi batch semplici

Spooling
(Simultaneous Peripheral Operation On Line)

Obiettivo: **aumentare l'efficienza** del sistema.

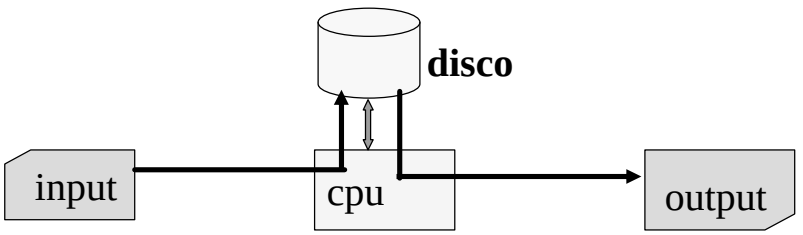
Avvento dei dischi + DMA: ⇒I/O **in parallelo** con l'attività della CPU:



Sistemi batch semplici

Spooling : contemporaneità di I/O e computazione

- il disco viene impiegato come un **buffer** molto ampio, dove:
  - **leggere** in anticipo i dati
  - **memorizzare** temporaneamente i risultati (in attesa che il dispositivo di output sia pronto)
  - caricare **codice** e **dati** del job successivo: -> possibilità di **sovrapporre I/O** di un job **con elaborazione** di un altro job



Sistemi Batch semplici

Problemi:

- finché il job corrente non è terminato, il successivo non può iniziare l'esecuzione
- se un job si sospende in attesa di un evento, la CPU rimane **inattiva**
- non c'è interazione con l'utente.

Sistemi Batch multiprogrammati

- **Sistemi batch semplici**: l'attesa di un evento causa l'inattività della CPU.

  Þ **Multiprogrammazione:**

  Pool di job contemporaneamente presenti su disco:

  - il S.O. seleziona un sottoinsieme dei job appartenenti al pool da caricare in memoria centrale:

    più job in memoria centrale

  - mentre un job è in **attesa di un evento**, il sistema operativo assegna la CPU a un altro job

Sistemi Batch multiprogrammati

Il sistema operativo è in grado di ***portare avanti*** l'esecuzione di più job contemporaneamente:

- Ad ogni istante:
  - **un solo job** utilizza la CPU
  - **più job** (in memoria centrale) attendono di acquisire la CPU.
- Quando il job che sta utilizzando la CPU si sospende in attesa di un evento:
  - il S.O. ***decide*** a quale job assegnare la CPU ed effettua lo scambio (**scheduling**)

Sistemi Batch multiprogrammati

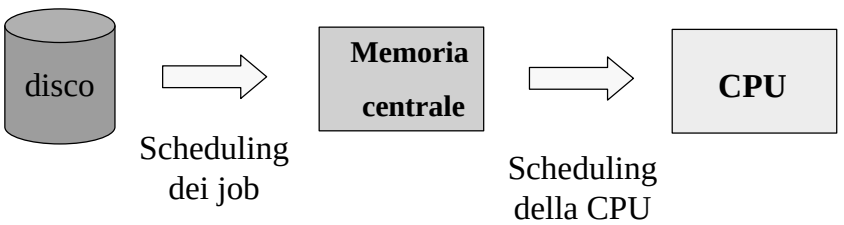
Scheduling

Il **S.O. effettua delle scelte** tra tutti i job:

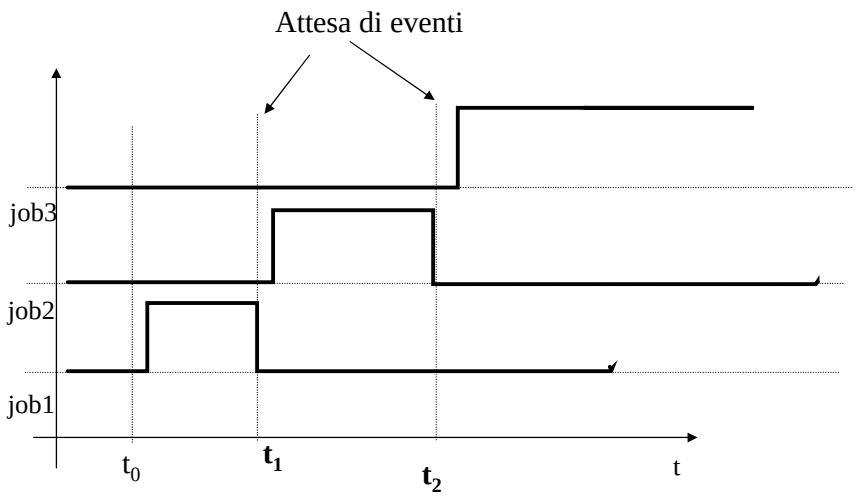
- quali job caricare in memoria centrale: **scheduling dei job** (*long term scheduling*)
- a quale job assegnare la CPU: **scheduling della CPU** o (*short term scheduling*)

Sistemi Batch multiprogrammati

Scheduling

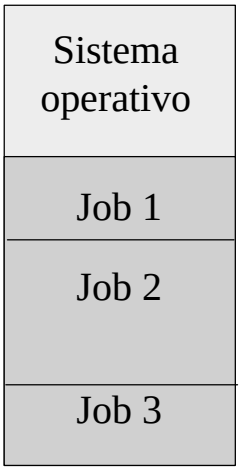


Sistemi Batch Multiprogrammati



Sistemi batch multiprogrammati

In memoria centrale, ad ogni istante, possono essere caricati più job:

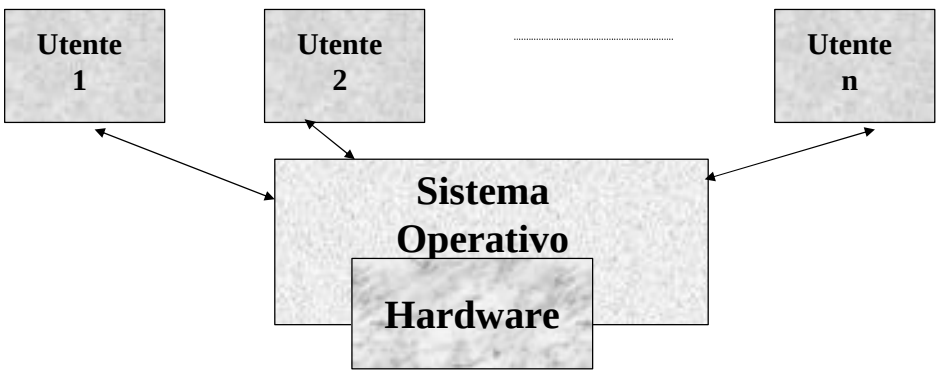


Configurazione della memoria centrale in sistemi batch multiprogrammati

Necessità di protezione !

Sistemi Time-Sharing (Multics, 1965)

- Nascono dalla necessità di:
  - **interattività** con l'utente
  - **multi-utenza:** più utenti interagiscono contemporaneamente con il sistema.



Sistemi Time-Sharing

- **Multiutenza**: il sistema presenta ad ogni utente una macchina ***virtuale*** completamente dedicata a lui, per:
  - l'utilizzo della CPU
  - l'utilizzo di altre risorse (ad es., file system)
- **Interattività**:per garantire un'accettabile velocità di "reazione" alle richieste dei singoli utenti, il S.O. *interrompe* l'esecuzione di ogni job dopo un intervallo di tempo prefissato (***quanto di tempo***, o ***time slice***), ed assegna la CPU ad un altro job.

Sistemi Time Sharing (oppure, a *divisione di tempo*)

Sono sistemi in cui:

- L'attività della CPU è **dedicata a job diversi** che si alternano ciclicamente nell'uso della risorsa
- La frequenza di commutazione della CPU è tale da fornire l'illusione ai vari utenti di una macchina completamente dedicata (**macchina *virtuale***).

Cambio di contesto (***context switch***):

operazione di trasferimento del controllo da un job al successivo ⇨ costo aggiuntivo (*overhead*).

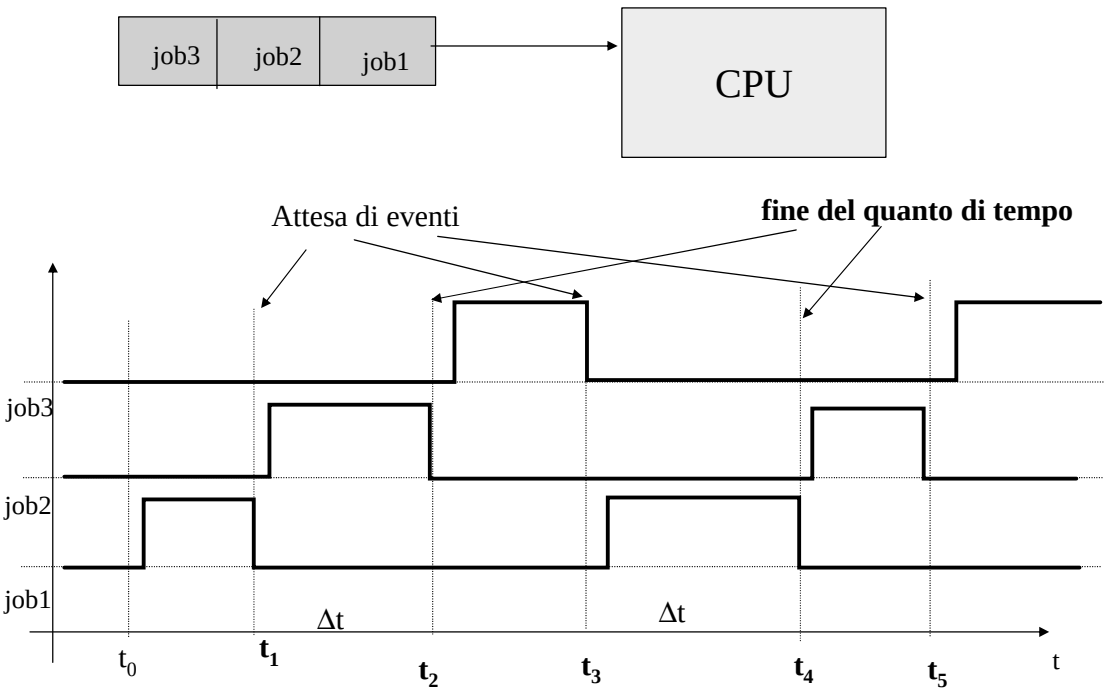
Sistemi Time Sharing

Estensione dei sistemi multiprogrammati:

Un job può sospendersi:

- perchè **in attesa di un evento**
- perchè è **terminato il *quanto di tempo***.

Sistemi Time Sharing



Time Sharing: requisiti

- **Gestione/Protezione** della memoria:
  - trasferimenti memoria-disco
  - separazione degli spazi assegnati ai diversi job
  - molteplicità job + limitatezza della memoria:

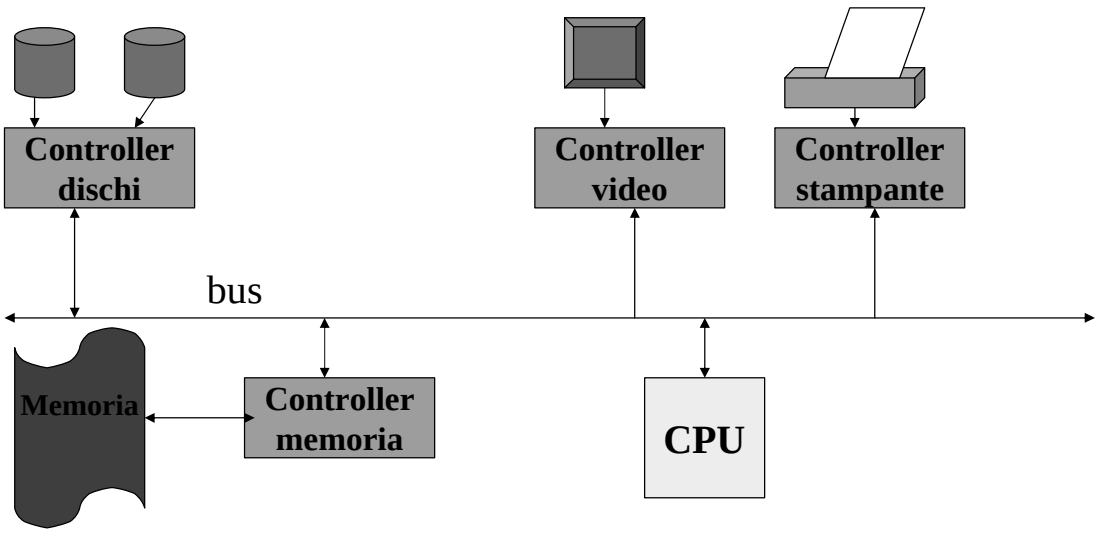
    Þ **memoria virtuale**
- Scheduling della CPU
- Sincronizzazione/comunicazione tra Job:
  - interazione
  - prevenzione/trattamento di blocchi critici (***deadlock***)
- Interattività: file system *on line* per permettere agli utenti di accedere semplicemente a codice e dati

Sistemi Operativi Attuali

- **MSDOS**: monoprogrammato, monoutente
- **Windows 95, 98**: multiprogrammato (time sharing), monoutente
- **Windows NT, 2000,..** : multiprogrammato, “multiutente”
- **MacOS**: multiprogrammato, multiutente
- **Unix/Linux**: Multiprogrammato/multiutente

Richiami sul funzionamento di un sistema di elaborazione

Architettura di un sistema di calcolo



- **Controller**: interfaccia HW delle periferiche verso il bus di sistema

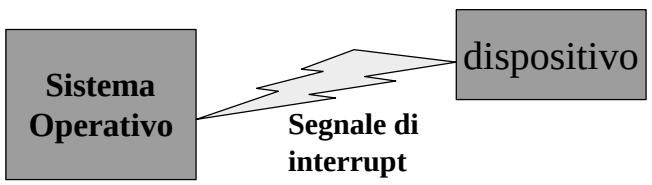
Funzionamento di un sistema di calcolo

Funzionamento a interruzioni:

- le varie *componenti* (HW e SW) del sistema interagiscono con il S.O. mediante ***interruzioni*** **asincrone *(interrupt)***
- ogni interruzione è causata da un **evento**; ad esempio:
  - ✓ richiesta di servizi al S.O.
  - ✓ completamento di I/O
  - ✓ accesso non consentito alla memoria
  - ✓ etc.
- ad ogni interruzione è associata una **routine di servizio (*handler*),** per la gestione dell'evento

Funzionamento di un sistema di calcolo

- Interruzioni hardware: i dispositivi inviano segnali al S.O.
  - Per richiedere l'esecuzione di servizi al S.O.



Funzionamento di un sistema di calcolo

- Interruzioni software: i programmi in esecuzione nel sistema possono generare interruzioni SW:
  - quando i programmi tentano l'esecuzione di operazioni non lecite (ad es., divisione per 0): ***trap***
  - ***system call:*** richiedono l'esecuzione di servizi al S.O.

Gestione delle interruzioni

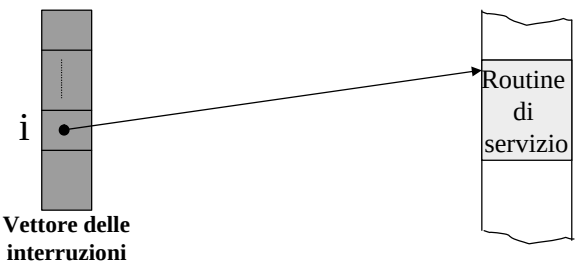
Alla ricezione di un'**interruzione**, il S.O.:

1] interrompe la sua esecuzione => salvataggio dello ***stato*** in memoria (locazione fissa, stack di sistema..)

2] attiva la routine di servizio all'interruzione (handler)

3] ripristina lo stato salvato

Per individuare la routine di servizio il S.O. può utilizzare un **vettore delle interruzioni** :



I/O

- Come avviene l'I/O in un sistema di elaborazione?

  Controller: interfaccia HW delle periferiche verso il bus di sistema

  - ogni controller è dotato di:
    - ✓ un **buffer** (ove memorizzare temporaneamente le informazioni da ***leggere*** o ***scrivere***)
    - ✓ alcuni **registri** speciali, ove memorizzare le specifiche delle operazioni di I/O da eseguire.

I/O

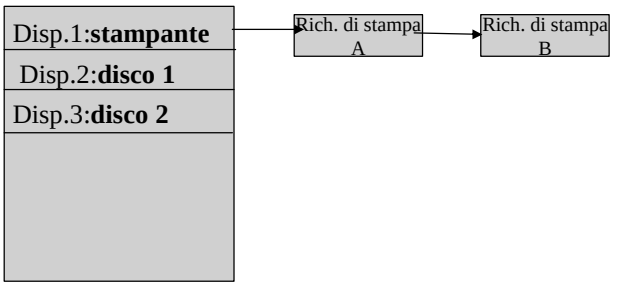
Quando un job richiede un'operazione di I/O (ad esempio, **lettura** da un dispositivo):

- ✓ la CPU scrive nei registri speciali del dispositivo coinvolto le specifiche dell'operazione da eseguire
- ✓ il Controller esamina i registri e provvede a trasferire i dati richiesti dal dispositivo al buffer
- ✓ invio di interrupt alla CPU (completamento del trasferimento)
- ✓ la CPU esegue l'operazione di I/O, tramite la routine di servizio (trasferimento dal buffer del controller alla memoria centrale)

I/O

2 tipi di I/O:

- **Sincrono**: il job viene sospeso fino al completamento dell'operazione di I/O
- **Asincrono**: il sistema restituisce immediatamente il controllo al job:
  - operazione di **wait** per attendere il completamento dell'I/O
  - possibilità di più I/O ***pendenti*** -> tabella di stato dei dispositivi



I/O asincrono = maggiore efficienza

DMA

I/O asincrono: se i dispositivi di I/O sono **veloci** (tempo di trasferimento dispositivo-buffer paragonabile al tempo di esecuzione della routine di servizio):

➔ l'esecuzione dei programmi può effettivamente riprendere soltanto alla fine dell'operazione di I/O

Direct Memory Access (DMA):

è una tecnica che consente di migliorare l'efficienza del sistema durante le operazioni di I/O.

DMA

Il trasferimento tra memoria e dispositivo viene effettuato direttamente, senza l'intervento della CPU.

Introduzione di un dispositivo HW per controllare l'I/O: **DMA controller**.

- **driver di dispositivo:** componente del S.O. Che:
  - ✓ copia nei registri del DMA controller i dati relativi al trasferimento da effettuare
  - ✓ invia al DMA controller il comando di I/O
- **interrupt** alla CPU (inviato dal DMA controller) solo alla fine del trasferimento dispositivo/memoria.

Protezione

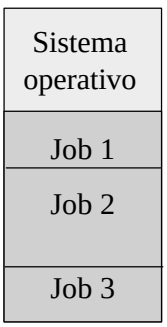
- Nei sistemi che prevedono multiprogrammazione e multiutenza sono necessari alcuni meccanismi HW per esercitare **protezione.**
- Le risorse allocate a programmi/utenti devono essere protette nei confronti di **accessi illeciti** di altri programmi/utenti:
  - ✓ dispositivi di I/O
  - ✓ memoria
  - ✓ CPU

  Ad esempio: accesso a locazioni esterne allo spazio di indirizzi del programma.

Protezione della memoria

In un sistema **multiprogrammato** o **time sharing**: ogni *job* ha un suo spazio di indirizzi:

- ➔ è necessario impedire al programma in esecuzione di accedere ad aree di memoria esterne al proprio spazio (ad es., del S.O. oppure di altri *Job*).



Se fosse consentito: un programma potrebbe modificare codice e dati di altri programmi o del S.O. !

Protezione

Per garantire protezione, molte architetture prevedono un duplice modo di funzionamento (*dual* mode):

- **user** mode
- **kernel** mode (***supervisor***, ***monitor*** mode)

Realizzazione: l'architettura prevede un ***bit di modo***

- **kernel: 0**
- **user: 1**

Dual mode

Istruzioni privilegiate: sono quelle più *pericolose* e possono essere eseguite soltanto se il sistema si trova in **kernel mode**:

- ✓ accesso a dispositivi di I/O (dischi, stampanti, schede di rete, etc.)
- ✓ gestione della memoria (accesso a strutture dati di sistema per la gestione della memoria)
- ✓ istruzione di ***shutdown*** (arresto del sistema)
- ✓ etc.

Dual mode

- **Il S.O. esegue in modo kernel.**
- **Ogni programma utente esegue in user mode:**
  - **Quando un programma utente tenta l'esecuzione di una istruzione privilegiata, viene generato un *trap*.**
  - **se necessita di operazioni privilegiate:**

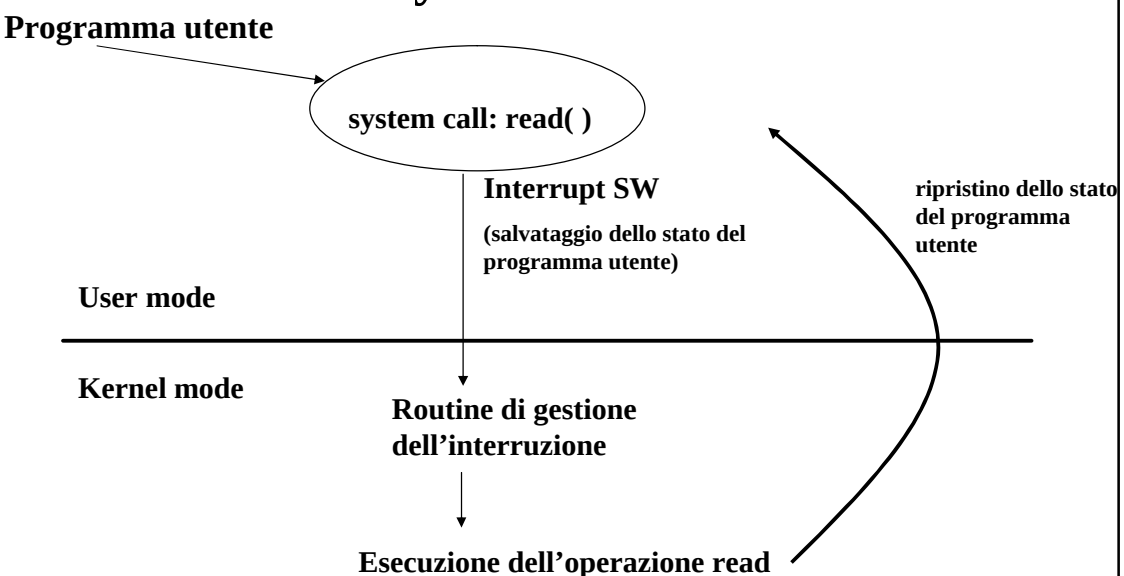
    chiamata a ***system call***

System Call

Per ottenere l'esecuzione di istruzioni privilegiate, un programma di utente deve chiamare una *System Call*:

- ✓ **invio di un'interruzione software al S.O.**
- ✓ **Salvataggio dello stato (PC, registri, bit di modo, etc.) del programma chiamante e trasferimento del controllo al S.O.**
- ✓ **Il S.O. esegue in modo kernel l'operazione richiesta**
- ✓ **al termine dell'operazione, il controllo ritorna al programma chiamante (ritorno al modo user)**

System Call


Programma utente
system call: read( )
Interrupt SW
(salvataggio dello stato del programma utente)
ripristino dello stato del programma utente
User mode
Kernel mode
Routine di gestione dell'interruzione
Esecuzione dell'operazione read

System Call

La comunicazione tra il programma chiamante ed il sistema operativo avviene mediante i parametri della system call: come vengono trasferiti?

- **mediante registri (problema: dimensione limitata)**
- **mediante blocchi di memoria indirizzati da registri**
- **mediante *stack di sistema***

Protezione della memoria

Il Sistema Operativo deve fornire gli strumenti per separare e proteggere gli spazi di indirizzi dei programmi:

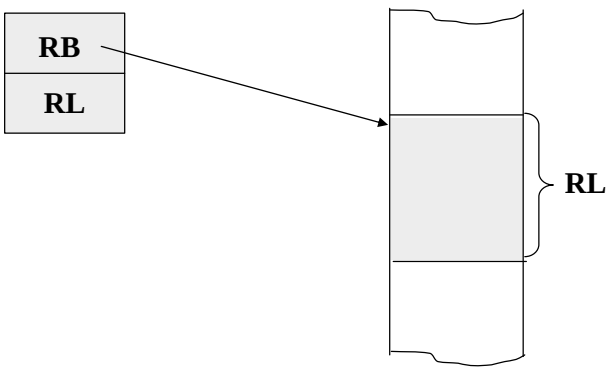
Registri *base* e *limite*

- memorizzano, per il programma in esecuzione (se viene allocato su parole contigue tra loro):
  - **l'indirizzo della prima parola (RB)**
  - **la dimensione (RL)**

  dello spazio degli indirizzi associato al programma.
- l'Hardware può controllare ogni indirizzo, per verificare se appartiene all'intervallo **[RB, RB+RL]**

Protezione della memoria

Registri *base* e *limite*



Protezione della CPU

Il S.O. deve evitare che un programma utente non monopolizzi la CPU (ad es., loop):

- **uso di timer, per interrompere il programma dopo un intervallo di tempo prefissato (time sharing)**
- **allo scadere dell'intervallo:**

  interrupt ⇨ cambio di contesto

Organizzazione dei Sistemi Operativi

Struttura dei S.O.

Quali sono le componenti di un S.O. ?

Quali sono le relazioni mutue tra le componenti ?

Componenti dei S.O.

- **Gestione dei processi**
- **gestione della memoria centrale**
- **gestione dei file**
- **gestione dell'I/O**
- **gestione della memoria secondaria**
- **protezione e sicurezza**
- **interfaccia utente/programmatore**

Processi

Processo = programma in esecuzione

- **il programma è un'entità passiva (un insieme di byte contenente le istruzioni che dovranno essere eseguite**
- **il processo è un'entità attiva:**
  - **è l'unità di lavoro all'interno del sistema: ogni attività all'interno del S.O. è rappresentata da un processo**
  - **è l'istanza di un programma in esecuzione:**

Processo=programma + contesto di esecuzione (PC, registri, etc)

Gestione dei Processi

In un sistema multiprogrammato:
più processi possono essere presenti nel sistema

Compiti del Sistema Operativo:

- **creazione/terminazione dei processi**
- **sospensione/ripristino dei processi**
- **sincronizzazione/comunicazione dei processi**
- **gestione del blocco critico (*deadlock*) di processi**

Gestione della Memoria Centrale

L'HW di sistema di elaborazione è equipaggiato con un'unico spazio di memoria accessibile direttamente da CPU e dispositivi.

Compiti del Sistema Operativo:

- **separare gli spazi di indirizzi associati ai processi**
- **allocare/deallocare memoria ai processi**
- **realizzare i collegamenti (*binding*) tra memoria logica e memoria fisica**
- **memoria virtuale: gestire spazi logici di indirizzi di dimensioni complessivamente superiori allo spazio fisico**

Gestione dei File

Ogni sistema di elaborazione dispone di uno o più dispositivi per la memorizzazione persistente delle informazioni (memoria secondaria)

Compiti del Sistema Operativo:

Il sistema operativo fornisce una visione logica uniforme della memoria secondaria (indipendente dal tipo e dal numero dei dispositivi):

- **realizza il concetto astratto di file, come unità di memorizzazione logica**
- **fornisce una struttura astratta per l'organizzazione dei file (direttorio)**

Gestione dei File

Il Sistema Operativo realizza inoltre:

- creazione/cancellazione di file e direttori
- manipolazione di file/direttori
- associazione tra file e dispositivi di memorizzazione secondaria

Spesso: file, direttori e dispositivi di I/O vengono presentati a utenti/programmi in modo uniforme.

Gestione dell'I/O

La gestione dell'I/O rappresenta una parte importante del Sistema Operativo:

- **interfaccia tra programmi e dispositivi**
- **Per ogni dispositivo: *device driver***
  - **routine per l'interazione con un particolare dispositivo**
  - **contiene conoscenza specifica sul dispositivo (ad es., routine di gestione delle interruzioni)**

Gestione della memoria secondaria

Tra tutti i dispositivi, la memoria secondaria riveste un ruolo particolarmente importante**:**

- **allocazione/deallocazione di spazio**
- **gestione dello spazio libero**
- ***scheduling* delle operazioni sul disco**

Di solito:

- la **gestione dei file** usa i meccanismi di gestione della memoria secondaria.
- la **gestione della memoria secondaria** è indipendente dalla gestione dei file.

Protezione e Sicurezza

In un sistema multiprogrammato, più entità (processi o utenti) possono utilizzare le risorse del sistema contemporaneamente: necessità di protezione.

Protezione: controllo dell'accesso alle risorse del sistema da parte di processi (e utenti), mediante:

- **autorizzazioni**
- **modalità di accesso**

Risorse da proteggere:

- ✓ **memoria**
- ✓ **processi**
- ✓ **file**
- ✓ **dispositivi**

Protezione e Sicurezza

Sicurezza:

Se il sistema appartiene ad una rete, la sicurezza misura l'affidabilità del sistema nei confronti di accessi (*attacchi*) dal modo esterno.

Interfaccia Utente

Il S.O. presenta un'interfaccia che consente l'interazione con l'utente:

- **interprete comandi** (***shell***): l'interazione avviene mediante una linea di comando
- **interfaccia grafica** (graphical user interface, ***GUI***): l'interazione avviene mediante ***click*** del mouse su elementi grafici; di solito organizzata a finestre.

Interfaccia Programmatore

L'interfaccia del SO verso i processi è rappresentato dalle ***system call***:

- mediante la system call il processo richiede al sistema operativo l'esecuzione di un servizio**.**
- la system call esegue istruzioni privilegiate: passaggio da modo ***user*** a modo ***kernel***

Classi di system call:

- ✓ gestione dei processi
- ✓ gestione di file e di dispositivi (spesso trattati in modo omogeneo)
- ✓ gestione informazioni di sistema
- ✓ comunicazione/sincronizzazione tra processi

***Programma di sistema* = programma che chiama system calls**

Struttura del Sistema Operativo

Sistema operativo = insieme di componenti

- ✓ **Gestione dei processi**
- ✓ **gestione della memoria centrale**
- ✓ **gestione dei file**
- ✓ **gestione dell'I/O**
- ✓ **gestione della memoria secondaria**
- ✓ **protezione e sicurezza**
- ✓ **interfaccia utente/programmatore**

Le componenti non sono indipendenti tra loro, ma interagiscono.

Struttura del Sistema Operativo

Come sono organizzate le varie componenti all'interno del sistema operativo?

Vari approcci:

- **struttura *monolitica***
- **struttura *modulare: stratificazione***
- ***microkernel***

Struttura Monolitica

Il sistema operativo è costituito da un unico modulo contenente un **insieme di procedure**, che realizzano le varie componenti:

l'interazione tra le componenti avviene mediante il meccanismo di chiamata a procedura.

Ad esempio:

- MS-DOS
- prime versioni di UNIX

Sistemi Operativi Monolitici

Principale Vantaggio: basso costo di interazione tra le componenti -> EFFICIENZA!

Svantaggio: Il SO è un sistema complesso e presenta gli stessi requisiti delle applicazioni ***in-the-large:***

- ✓ estendibilità
- ✓ manutenibilità
- ✓ riutilizzo
- ✓ portabilità
- ✓ affidabilità
- ✓ ...

Soluzione: organizzazione **modulare**

Struttura modulare

Le varie componenti del SO vengono organizzate in moduli caratterizzati da interfacce ben definite.

Sistemi Stratificati (a *livelli*)

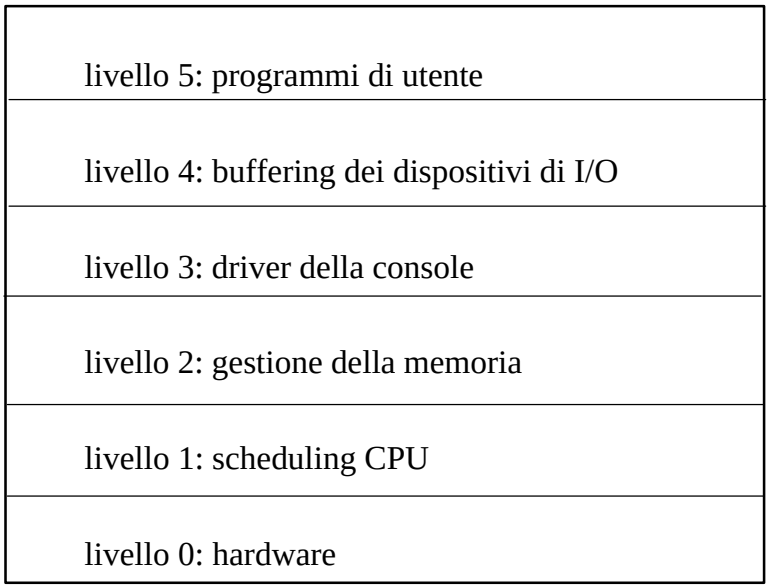
(THE, Dijkstra1968)

il sistema operativo è costituito da livelli sovrapposti, ognuno dei quali realizza un insieme di funzionalità:

- ogni livello realizza un'insieme di funzionalità che vengono offerte al livello superiore mediante un'interfaccia
- ogni livello utilizza le funzionalità offerte dal livello sottostante, per realizzare altre funzionalità

Struttura a livelli

Ad esempio: THE (5 livelli)



Struttura Stratificata

Vantaggi:

- **Astrazione:** ogni livello è un **oggetto astratto**, che fornisce ai livelli superiori una visione astratta del sistema (***Macchina Virtuale***), limitata alle astrazioni presentate nell'interfaccia.

- **Modularità:** le relazioni tra i livelli sono chiaramente esplicitate dalle interfacce ⇨ possibilità di sviluppo, verifica, modifica in modo indipendente dagli altri livelli.

Struttura Stratificata

Svantaggi:

- **organizzazione gerarchica** tra le componenti: non sempre è possibile -> difficoltà di realizzazione.
- **scarsa efficienza** (costo di attraversamento dei livelli)

Soluzione: limitare il numero dei livelli.

Nucleo (Kernel) del Sistema Operativo

E` la parte del sistema operativo che esegue in modo privilegiato (modo ***kernel***).

- È la parte più *interna* del sistema operativo, che si interfaccia direttamente con l'hardware della macchina.

- Le funzioni realizzate all'interno del nucleo variano a seconda del Sistema Operativo.

Nucleo del Sistema Operativo (kernel)

- Per un sistema multiprogrammato a divisione di tempo, il nucleo deve, almeno:
  - gestire il **salvataggio/ripristino** dei contesti
  - realizzare lo **scheduling** della Cpu
  - gestire le **interruzioni**
  - realizzare le **system call**.

Sistemi Operativi a Microkernel

- La struttura del nucleo è ridotta a poche funzionalità di base:
  - gestione della CPU,
  - della memoria
  - meccanismi di comunicazione.

il resto del SO è mappato su processi di utente

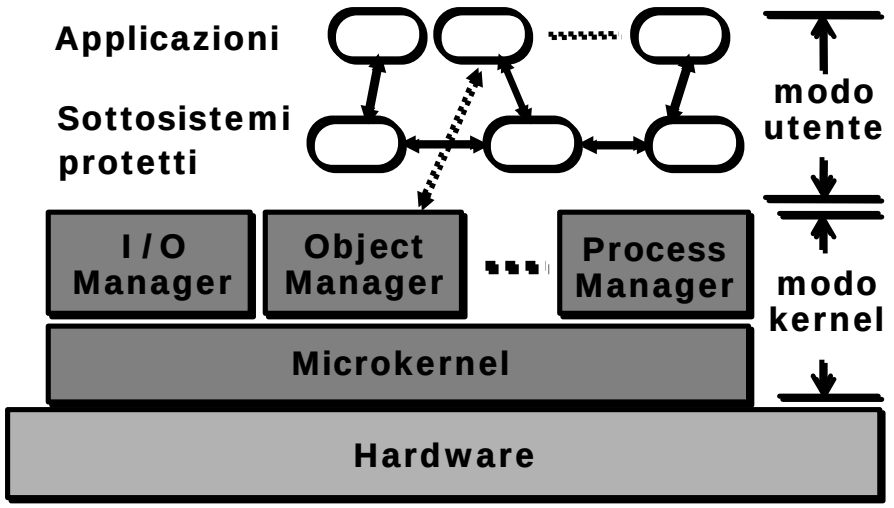
Caratteristiche:

- affidabilità (separazione tra componenti)
- possibilità di estensioni e personalizzazioni
- scarsa efficienza (molte chiamate a system call)

Esempi: Mach, L4, Hurd, Windows.

Organizzazione a Microkernel

Windows NT


Applicazioni
Sottosistemi protetti
I/O Manager
Object Manager
Process Manager
Microkernel
Hardware
modo utente
modo kernel

Organizzazione di Unix