AlmaWeb
Universita` di Bologna

Master in Tecnologia del Software Libero e Open Source

Corso di *Sistemi Operativi*

Prof. Anna Ciampolini –aciampolini@deis.unibo.it

Dott. Stefano Zacchiroli – zack@bononia.it

A.A. 2006 - 2007

Cos’e` un sistema operativo?

Cos'e` un Sistema Operativo ?

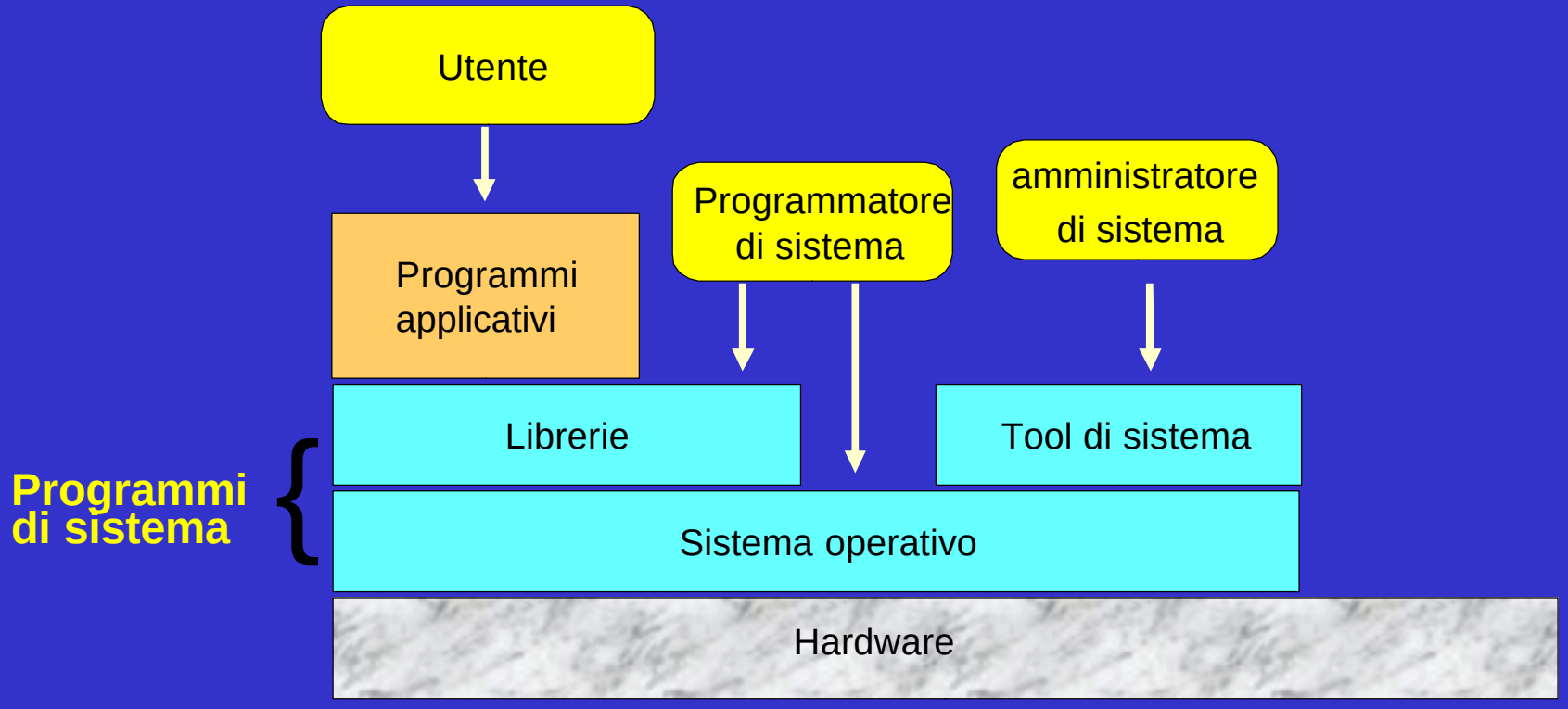
- È un programma (o un insieme di programmi) che agisce come **intermediario tra l'utente e l'hardware** del computer:
  - fornisce un **ambiente di sviluppo e di esecuzione** per i programmi
  - **gestisce** le risorse del sistema di calcolo
  - fornisce una **visione astratta** dell'architettura

Cos'e` un Sistema Operativo ?

Il Sistema Operativo e l'Hardware

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

Aspetti importanti di un S.O.

- **Organizzazione e struttura:**
  - quali servizi offre il sistema?
  - da quali componenti e` costituito?
  - quali relazioni tra le componenti?
- **Gestione delle Risorse:**
  - quali risorse vengono gestite?
  - come vengono rappresentate?
  - come vengono gestite?
    - condivisione
    - protezione
    - sicurezza
    - ...

Aspetti importanti di un S.O.

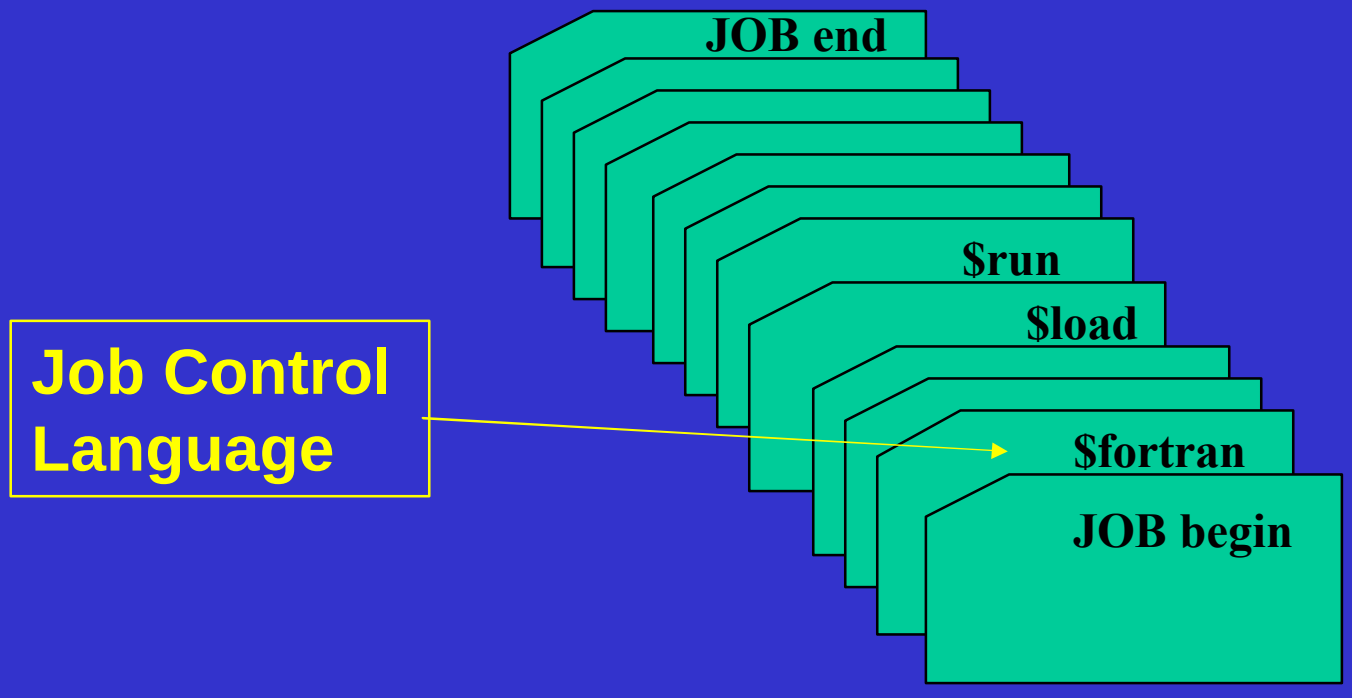
- **Efficienza:** le risorse vengono gestite dal sistema in modo *ottimale ?*
- **Affidabilità:** come reagisce il S.O. a malfunzionamenti (HW/SW) ?
- **Estendibilità:** è possibile aggiungere/modificare funzionalità?
- **Conformità a Standard:** portabilita`, estendibilita`, apertura

Evoluzione dei Sistemi di Elaborazione e dei Sistemi Operativi

- **Prima generazione** (anni ‘50)
  - macchine a valvole
  - linguaggio macchina
  - assenza del sistema operativo
- **Seconda Generazione** (‘55-’65): **Sistemi batch semplici**
  - avvento dei transistori
  - linguaggio di alto livello (fortran)
  - input mediante schede perforate
  - aggregazione di programmi in **lotti** (batch) con esigenze simili
  - sistema operativo: monitor residente

Sistemi batch semplici

Batch: insieme di programmi (*job*) da eseguire **in modo sequenziale**.



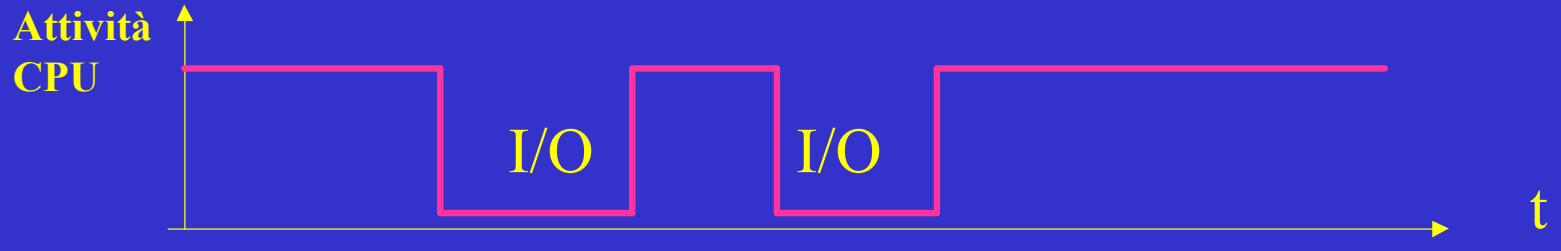
Sistemi batch semplici

Ruolo del Sistema Operativo:

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:

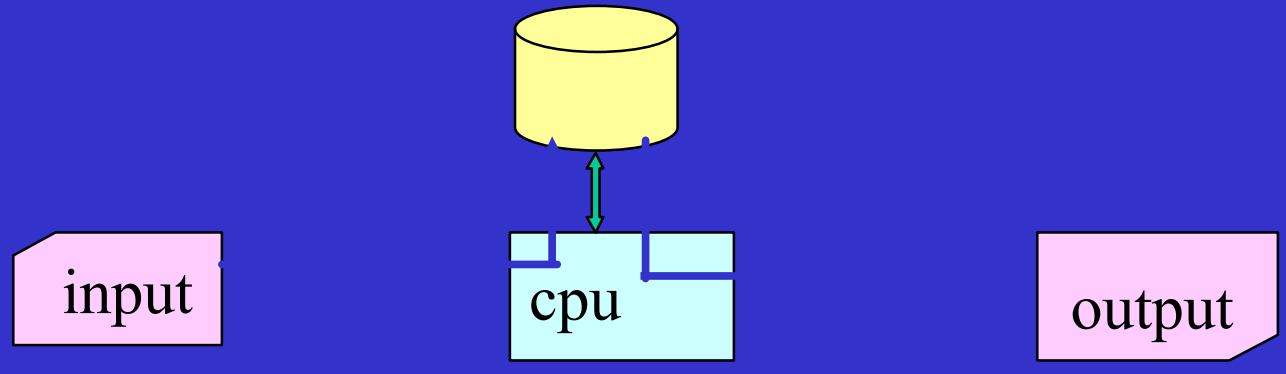
Configurazione della memoria centrale in sistemi batch semplici

memoria inutilizzata

Spooling
(Simultaneous Peripheral Operation On Line)

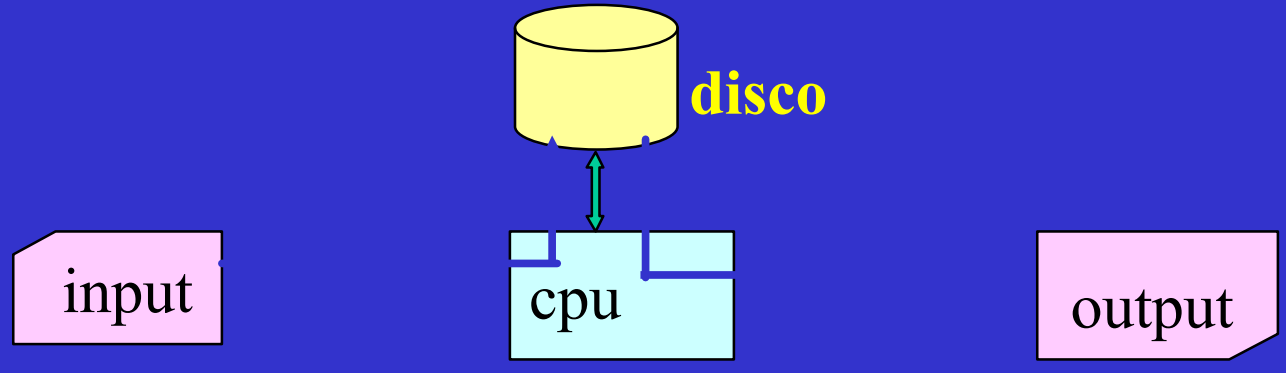
Obiettivo: **aumentare l'efficienza** del sistema.

Avvento dei dischi: $\Rightarrow$ I/O in parallelo con l'attività della CPU:



Spooling

- 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



Spooling

- contemporaneità di I/O e computazione
- 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.

Multiprogrammazione:

- **Pool di job** pre-caricati 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 (*cambio di contesto*, ***context switch***)

Sistemi Batch multiprogrammati

Il sistema operativo consente l'esecuzione di più job ***contemporaneamente***:

- Se la macchina e` monoprocessore, 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:

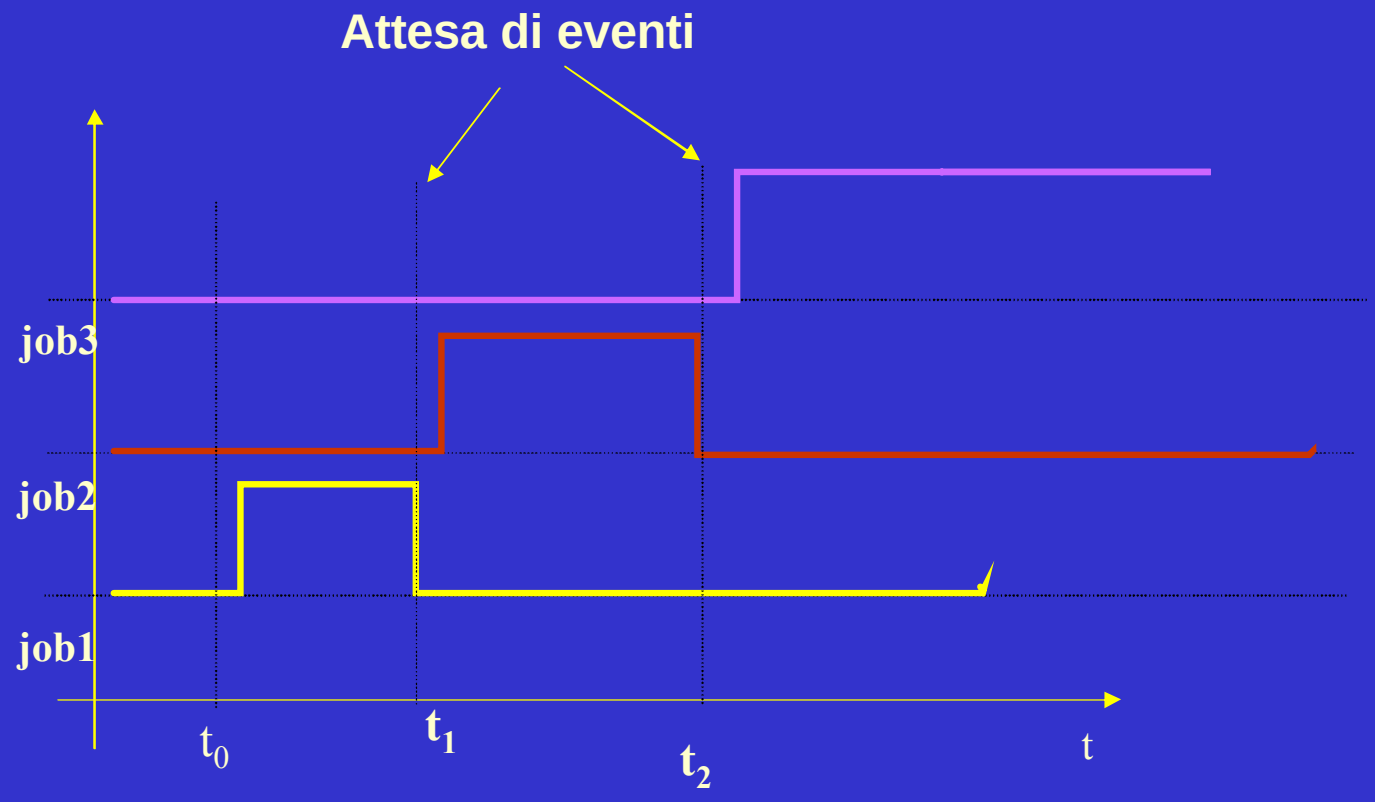
Il **S.O. effettua delle scelte** tra tutti i job:

- 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

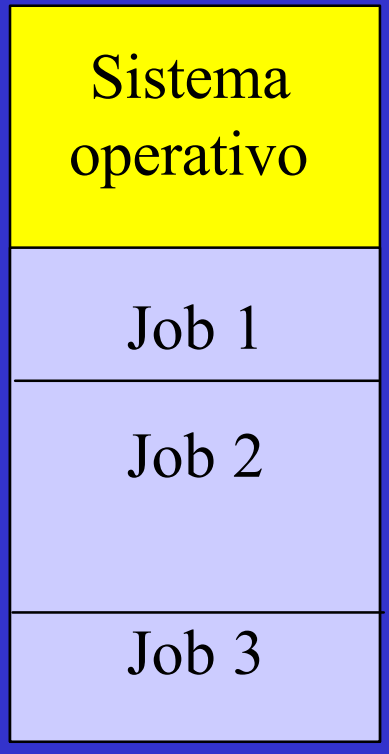
Scheduling

Sistemi Batch Multiprogrammati



Sistemi Batch multiprogrammati

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

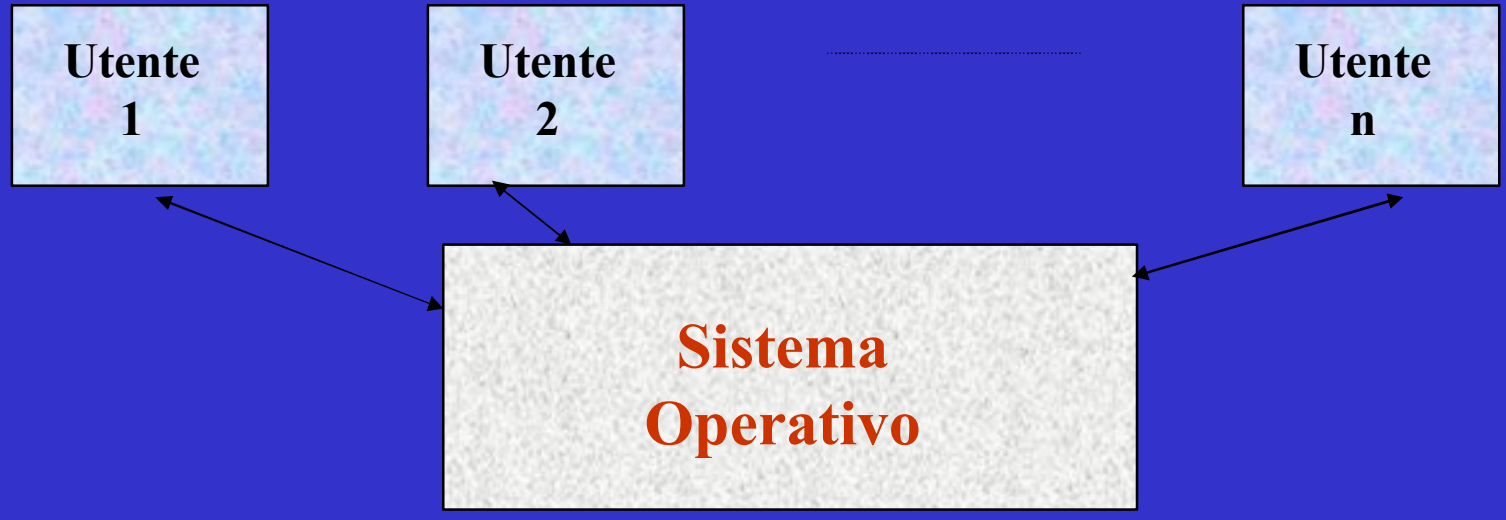


Configurazione della memoria centrale in sistemi batch multiprogrammati

Necessità di protezione !

Sistemi Interattivi (Multics, 1965)

- Obiettivi:
  - **interattività** con l'utente
  - **multi-utenza:** più utenti utilizzano contemporaneamente il sistema.



Soluzione: sistemi operativi ***Time Sharing***

Sistemi a divisione di tempo (time sharing)

- **Multiutenza**: il sistema presenta ad ogni utente una macchina ***virtuale*** completamente dedicata

- **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 a divisione di tempo (time sharing)

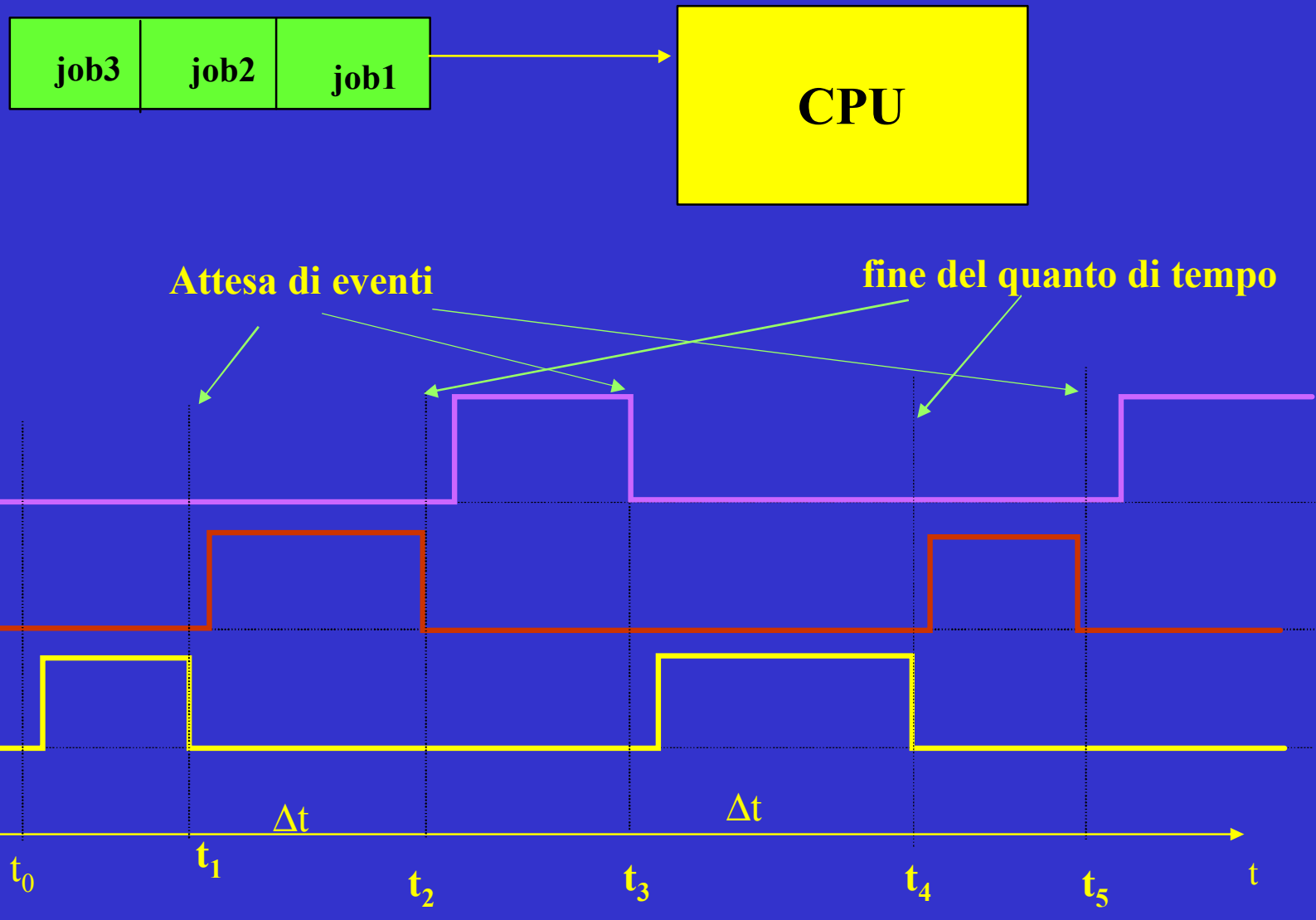
- L'attività della CPU è **dedicata a job diversi** che si alternano nell'uso della risorsa
- La frequenza di commutazione della CPU è tale da fornire l'illusione ai vari utenti di una **macchina completamente dedicata** (***macchina virtuale***).

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**:
  - separazione degli spazi assegnati ai diversi job
  - molteplicità job + limitatezza della memoria:

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

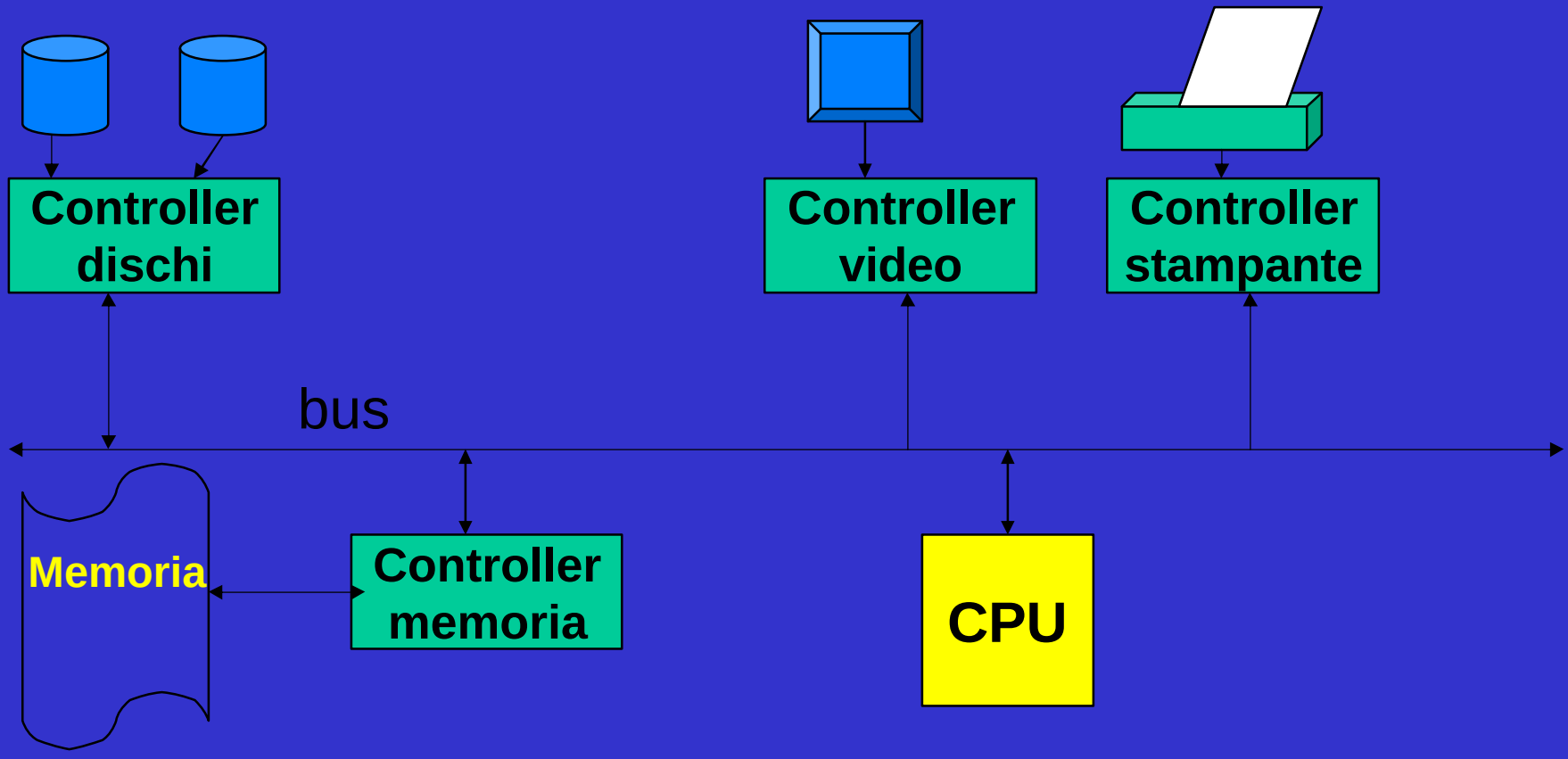
Sistemi Time Sharing

- **Multics** MIT, anni 60
- **VMS** (Digital), **Unix e GNU/Linux**
- **Windows**
- **MacOS**
- **...**

Richiami sul funzionamento di un sistema di elaborazione*

* **Per approfondimenti:** Silberschatz, Galvin: Cap.2.

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
  - ...
- ad ogni interruzione è associata una **routine di servizio (*handler*),** per la gestione dell'evento
- Al verificarsi di un'interruzione, il sistema reagisce in modo asincrono eseguendo la routine di servizio

Funzionamento di un sistema di calcolo

- **Interruzioni hardware (o esterne)**: i dispositivi possono inviare segnali al S.O.:
  - Ad esempio, per richiedere il trasferimento di informazioni da/verso il dispositivo.
- **Interruzioni software (o interne, *trap*)**: i programmi in esecuzione nel sistema possono generare interruzioni SW:
  - quando i programmi tentano l'esecuzione di operazioni non lecite (ad es., accessi illegali alla memoria)
  - ***system call:*** richiesta di l'esecuzione di servizi al S.O.

Gestione delle interruzioni

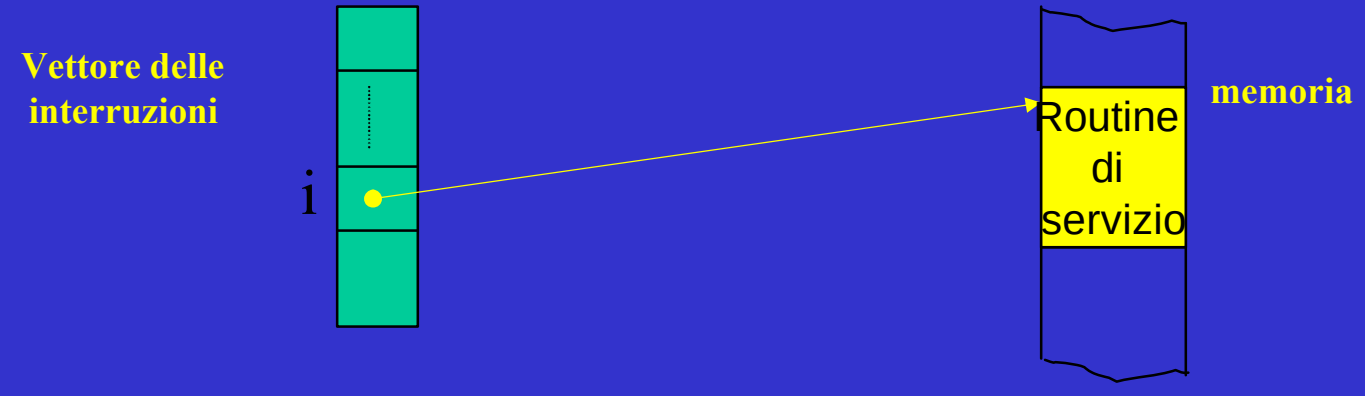
Alla ricezione di un'**interruzione**, il S.O.:

1] interrompe la sua esecuzione => salvataggio dello ***stato*** del **processo in esecuzione** in memoria (contenuto dei registri di CPU, stack di sistema, ecc.)

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

3] ripristina lo stato salvato (del "processo in esecuzione")

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



I/O

Quali meccanismi presiedono all'I/O?

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):

1. la CPU scrive nei registri speciali del dispositivo le specifiche dell'operazione da eseguire
2. il Controller interpreta il contenuto dei registri e provvede a trasferire i dati richiesti dal dispositivo al buffer
3. Al termine del trasferimento: invio di **interruzione** alla CPU
4. la CPU esegue l'operazione di I/O, tramite la routine di servizio: trasferimento dal buffer del controller alla memoria centrale.
5. Se i dati trasferire non sono finiti, si ripete dal punto 1

➔ **1 interruzione per ogni trasferimento dispositivo/buffer**

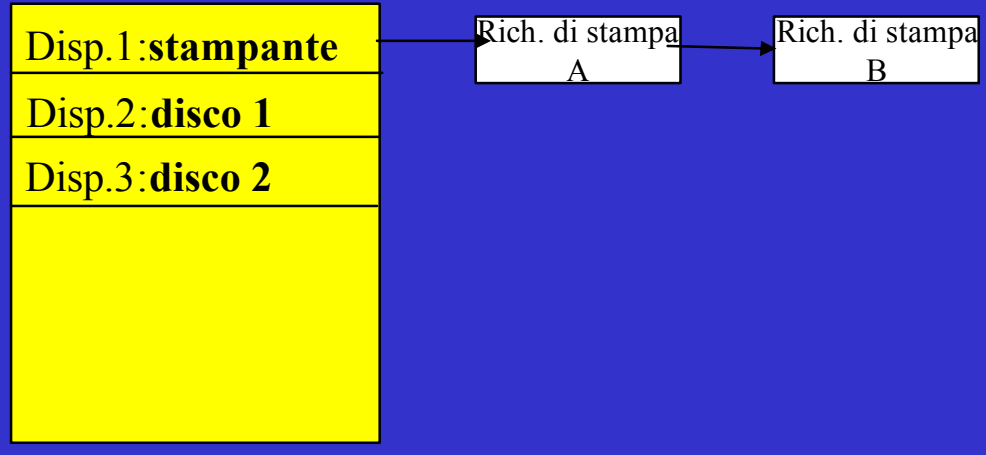
I/O

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**



DMA

Ogni operazione di I/O puo` richiedere piu` trasferimenti di seguito.

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

Nelle architetture moderne, per renalizzare in modo piu` efficiente le operazioni di I/O da/verso dispositivi veloci viene utilizzato un dispositivo dedicato al controllo delle operazioni di I/O: ***DMA controller:***

- sostituisce la CPU nel controllo dei trasferimenti dispositivo-buffer-memoria

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

DMA

I/O con DMA:

per ogni operazione di I/O:

1. La CPU trasferisce nei registri del *DMA controller* i dati relativi al trasferimento da effettuare
2. Il DMA controller avvia l'operazione di I/O
3. Solo alla fine dell'operazione di I/O (cioe`, al termine della sequenza di trasferimenti dispositivo/memoria) il DMA controller invia un'**interruzione** alla CPU.

→ 1 interruzione per ogni operazione di I/O

Protezione

Multiprogrammazione e multiutenza rendono necessari alcuni meccanismi HW per esercitare la **protezione sulle risorse**

- 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.

Architetture Dual Mode

Molte architetture prevedono un duplice modo di funzionamento (*dual* mode), che consente di differenziare due tipi di esecuzione:

- **user** mode: programmi di utente
- **kernel** mode (***supervisor***, ***monitor*** mode): sistema operativo

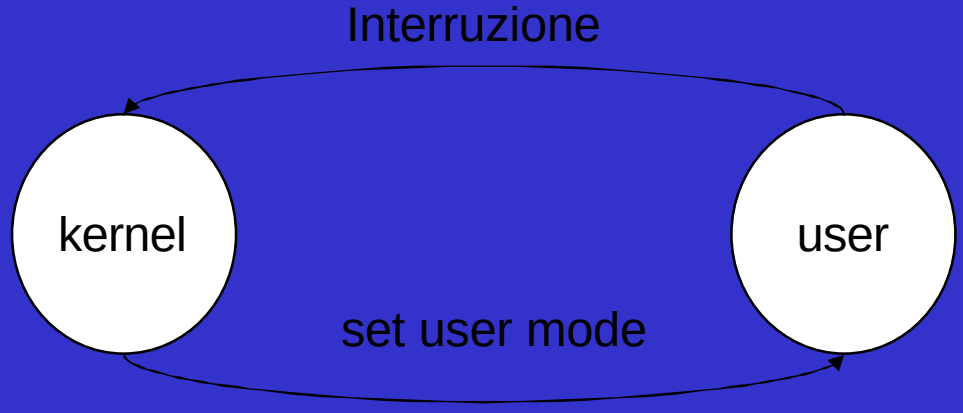
Realizzazione: l'architettura prevede un ***bit di modo***

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

Dual Mode

- Il Mode bit indica il modo di esecuzione corrente: kernel (0) o user (1).

- Al verificarsi di un'interruzione, l'hardware commuta al kernel mode.



Dual mode

Istruzioni privilegiate: possono essere eseguite soltanto se il sistema si trova in **kernel mode**. Sono quelle più *pericolose;* ad esempio:

- ✓ Operazioni di I/O
- ✓ Operazioni di gestione della memoria (accesso a strutture dati di sistema per la gestione della memoria)
- ✓ istruzione di ***shutdown*** (arresto del sistema)
- ✓ …

Dual mode

- Ogni programma utente esegue in *user mode*
- Quando un programma utente tenta l'esecuzione di una istruzione privilegiata, viene generato un *trap*.

Come puo` un programma di utente richiedere l'esecuzione di istruzioni privilegiate (ad esempio, I/O)?

➔ chiamata a *system call*

System Call

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

1. **invio di un'interruzione software al S.O.**
2. **Salvataggio dello stato (PC, registri, bit di modo, etc.) del programma chiamante e trasferimento del controllo al S.O.**
3. **Il S.O. esegue in modo kernel l'operazione richiesta (gestione dell'interruzione)**
4. **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
IRET
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***

Organizzazione dei Sistemi Operativi

Struttura dei S.O.

Quali sono le componenti di un S.O. ?

Quali sono le relazioni mutue tra le componenti ?

Struttura del Sistema Operativo

Sistema operativo = insieme di componenti

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

Processo

Processo = programma in esecuzione

- il programma è un'entità ***passiva:*** un insieme di byte contenente la codifica binaria delle istruzioni da eseguire.
- il processo è un'entità ***attiva***:
  - è ***l'istanza*** di un programma in esecuzione
  - è **l'unità di lavoro** all'interno del sistema: ogni attività all'interno del S.O. è rappresentata da un processo

Gestione dei Processi

In un sistema multiprogrammato:

più processi possono esistere contemporaneamente

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 un 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**
- **memoria virtuale: offrire spazi logici di indirizzi di dimensioni indipendenti dalla dimensione effettiva della memoria**

Gestione dei File

Il sistema operativo fornisce una visione logica uniforme della memoria secondaria, indipendente dal tipo e dal numero dei dispositivi effettivamente disponibili:

- **realizza il concetto astratto di file, come unità di memorizzazione logica**
- **fornisce opportune astrazioni per l'organizzazione dei file (direttori, partizioni)**
- **realizza i meccanismi per la gestione di file, direttori e partizioni:**
  - ✓ creazione/cancellazione di file e direttori
  - ✓ manipolazione di file/direttori
  - ✓ associazione tra file e dispositivi di memorizzazione secondaria

Gestione dell'I/O

Il Sistema Operativo si occupa della gestione delle periferiche di I/O:

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

Protezione e Sicurezza

Processi e utenti possono accedere alle risorse del sistema contemporaneamente.

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

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

Risorse da proteggere: **file, memoria, processi dispositivi, ecc.**

Protezione e Sicurezza

Sicurezza:

Se il sistema e` collegato 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 programmi (API) è rappresentato dalle ***system call***:

- mediante la system call il processo richiede al sistema operativo l'esecuzione di un **servizio** (in 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 usa 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**
- **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?

Varie soluzioni:

- **struttura *monolitica***
- **struttura *modulare***
- ***microkernel***
- ***exokernel***

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 diverse componenti avviene mediante il meccanismo di **chiamata a procedura**.

Esempi: *MS-DOS,UNIX, GNU/Linux*

Sistemi Operativi Monolitici

Principale Vantaggio: basso costo di interazione tra le componenti.

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*)

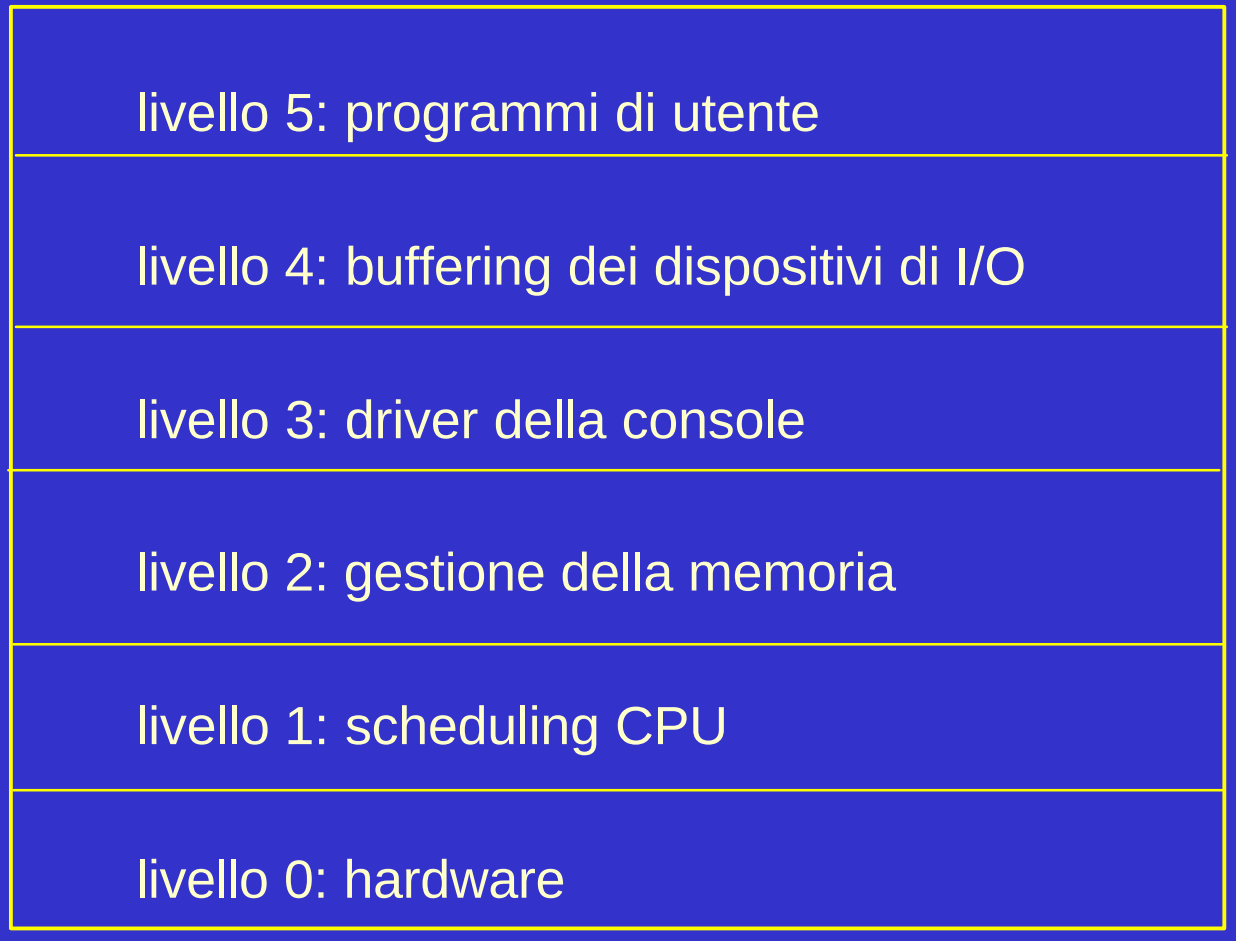
(THE, Dijkstra1968)

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

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 del Sistema Operativo (kernel)

"E` la parte del sistema operativo che esegue in ***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*)

- Tipicamente, le funzioni del nucleo sono:
  - Creazione/terminazione dei processi
  - **scheduling** della Cpu
  - gestire il **cambio** di contesti
  - Sincronizzazione/comunicazione tra processi
  - Gestione della memoria
  - Gestione dell'I/O
  - Gestione delle **interruzioni**
  - realizzare le **system call**.

Sistemi Operativi a Microkernel

- La struttura del nucleo è ridotta a poche funzionalità di base.
- il resto del SO è rappresentato da processi di utente

Caratteristiche:

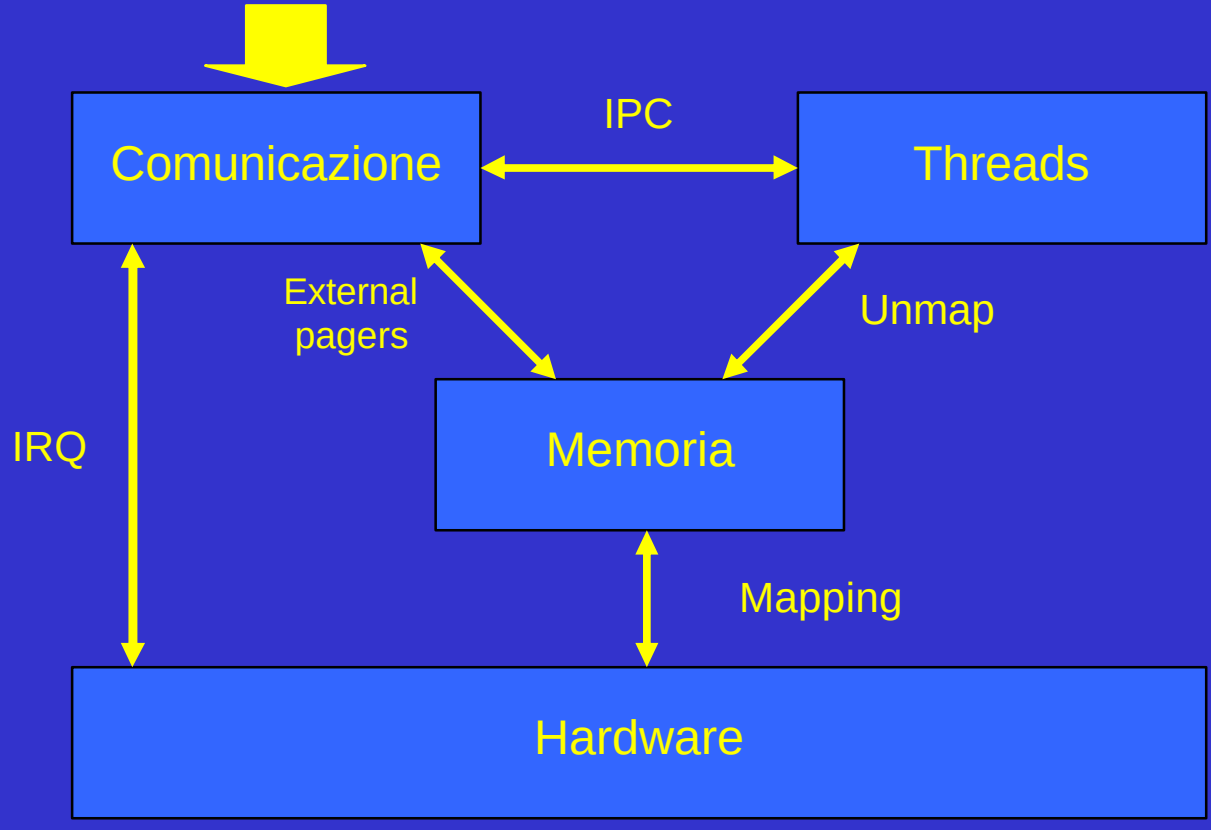
- affidabilità (separazione tra componenti)
- possibilità di estensioni e personalizzazioni
- scarsa efficienza (molte chiamate a system call)

ESEMPI: L4, Mach, Hurd, Windows NT

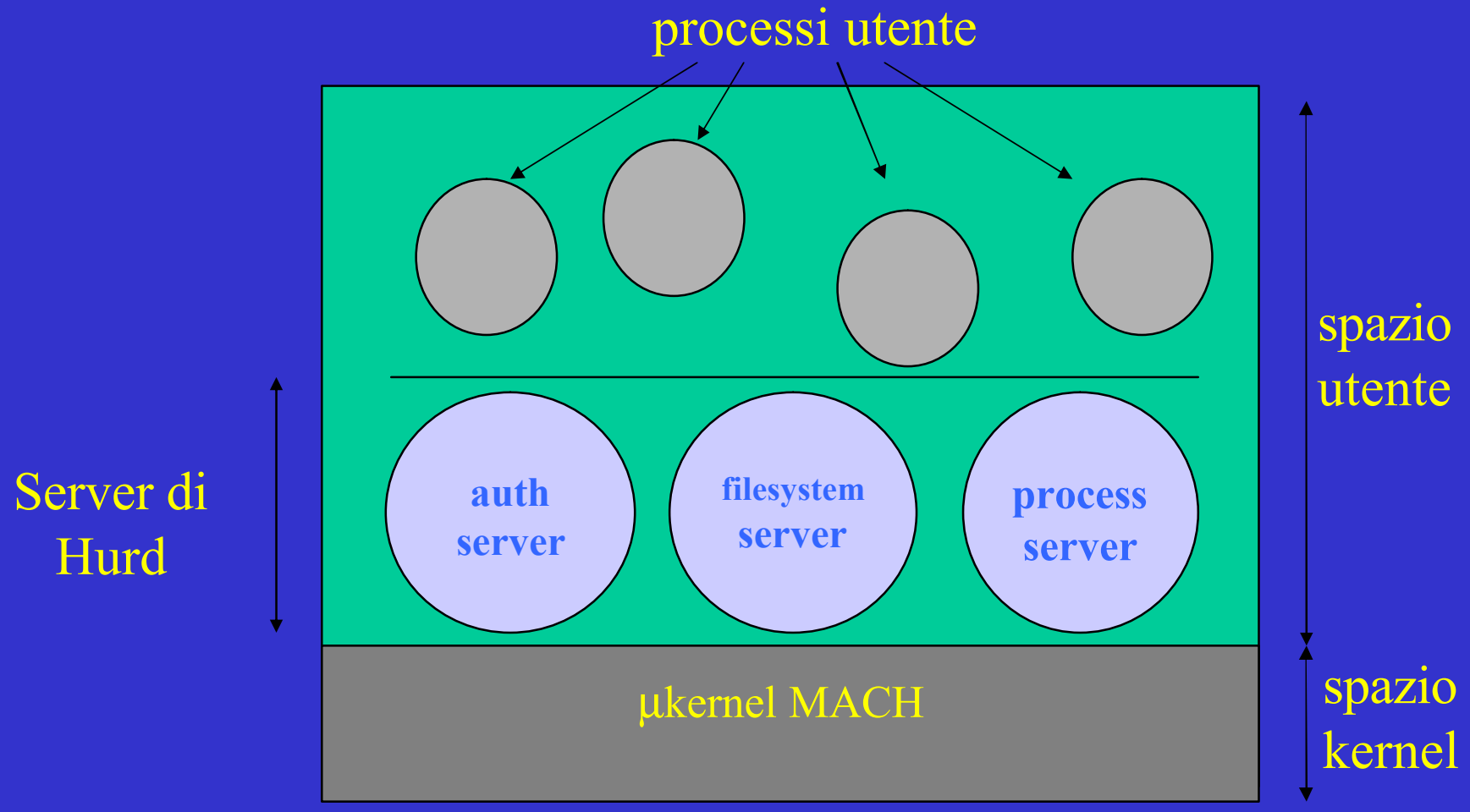
L4 mkernel

http://os.inf.tu-dresden.de

- gestione dei thread
- allocazione della memoria (pager esterni)
- Inter Process Communication



GNU/Hurd


processi utente
Server di Hurd
auth server
filesystem server
process server
µkernel MACH
spazio utente
spazio kernel

Macchine virtuali

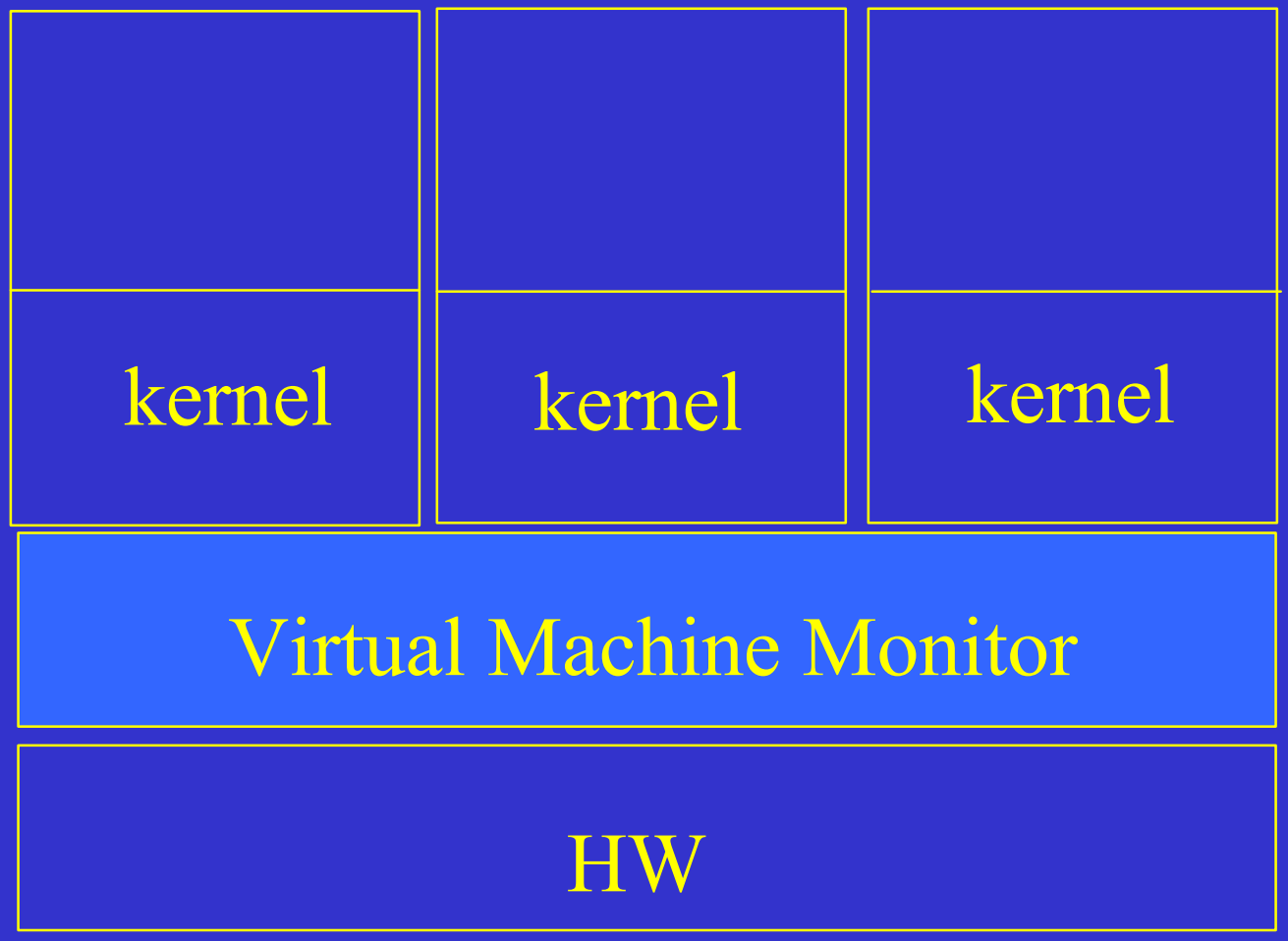
- il kernel del sistema operativo presenta una vista *astratta* (e in generale diversa) delle risorse presenti a livello di architettura.
- ogni *processo* puo` ottenere l'accesso alle risorse virtuali
- le modalita` di utilizzo delle risorse virtuali possono essere differenti dalle modalita` previste dalla risorsa reale

ES: memoria virtuale & memoria fisica

Sistemi Operativi per la Virtualizzazione

- la macchina fisica viene trasformata in n interfacce (**macchine virtuali**), ognuna delle quali e` una replica della macchina fisica:
  - dotata di tutte le istruzioni del processore (sia privilegiate che non privilegiate)
  - dotata delle risorse del sistema (memoria, dispositivi di I/O).

➔ Su ogni macchina virtuale è possibile installare ed eseguire un sistema operativo (eventualmente diverso da macchina a macchina).



Compito del VMM è consentire la condivisione da parte di più macchine virtuali di una singola piattaforma hardware.

Virtualizzazione e Emulazione

Emulazione:

- eseguire applicazioni (o SO) compilate per un'architettura su di un’altra**.**
- uno strato software emula le funzionalita` dell'architettura; il s.o. esegue sopra tale strato (a livello user). Le istruzioni macchina privilegiate e non privilegiate vengono emulate via SW. (Bochs, Qemu)

Virtualizzazione:

- definizione di contesti di esecuzione multipli (macchine virtuali) su di un singolo processore, partizionando le risorse;

Tecniche di Virtualizzazione

virtualizzazione completa: le istruzioni non privilegiate vengono eseguite direttamente a livello HW; solo le istruzioni privilegiate vengono traslate dinamicamente; il set di istruzioni delle macchine virtuali e` lo stesso della macchina fisica.(VMware)

paravirtualizzazione: l'hardware virtuale esposto dal VMM e` funzionalmente simile, ma non identico, a quello della sottostante macchina fisica. Definizione di una Applications Programming Interface (Virtual Hardware API). Il kernel dei sistemi operativi nelle macchine virtuali deve essere modificato. (xen)

Vantaggi virtualizzazione

- consolidamento HW: utilizzo efficiente dell'hardware in server farm
- ambienti di esecuzione isolati: possibilita` di testing di applicazioni senza mettere a rischio altre applicazioni
- in ambito didattico: laboratori virtuali di sistemi operativi; ogni studente amministra la propria macchina virtuale

Unix & Linux

Storia di Unix

- **1969**: AT&T, sviluppo di un ambiente di calcolo multiprogrammato e portabile per macchine di medie dimensioni.
- **1970**: prima versione di UNIX (multiprogrammata e monoutente) interamente sviluppata nel linguaggio assembler del calcolatore PDP-7.
- **Anni 1970**: nuove versioni, arricchite con altre caratteristiche e funzionalità. Introduzione del supporto alla multiutenza.

Unix e il linguaggio C

- **1973**: Unix viene realizzato nel linguaggio di programmazione C:
  - Elevata portabilità
  - Leggibilità
  - Diffusione presso la comunità scientifica e accademica.

- **Anni 80**: la grande popolarità di Unix ha determinato il proliferare di versioni diverse. Due famiglie:
  - Unix System V (AT&T Laboratories)
  - Unix Berkeley Software Distributions, o BSD (University of California at Berkeley)

Organizzazione di Unix

Caratteristiche di Unix

- multi-utente
- time sharing
- kernel monolitico
- Ambiente di sviluppo per programmi in linguaggio C
- Programmazione mediante linguaggi comandi
- portabilità

POSIX

- **1988**: POSIX (*Portable Operating Systems Interface*) è lo standard definito dall'IEEE. Definisce le caratteristiche relative alle modalità di utilizzo del sistema operativo.
- **1990:** POSIX viene anche riconosciuto dall' International Standards Organization (ISO).
- **Anni 90:** Negli anni seguenti, le versioni successive di Unix SystemV e BSD (versione 4.3), si uniformano a POSIX.

Introduzione a GNU/Linux

- GNU project:
  - **1984**: Richard Stallman avvia un progetto di sviluppo di un sistema operativo ***libero*** **compatibile con Unix:**

"GNU is Not Unix"

  - Furono sviluppate velocemente molte utilita` di sistema:
    - editor Emacs,
    - Compilatori: gcc,
    - shell: bash,
    - ...
  - lo sviluppo del kernel (Hurd), invece, subi` molte vicissitudini e vide la luce molto piu` tardi (1996)

GNU/Linux

- **1991**: Linus Tornvalds realizza un kernel Unix-compatibile (Minix) e pubblica su web i sorgenti
- In breve tempo, grazie a una comunita` di hacker in rapidissima espansione, Linux acquista le caratteristiche di un prodotto affidabile e in continuo miglioramento.
- **1994**: Linux viene integrato nel progetto GNU come kernel del sistema operativo: nasce il sistema operativo GNU/Linux

GNU/Linux

Caratteristiche:

- Open Source / Free software
- multi-utente, multiprogrammato e multithreaded
- Kernel monolitico con possibilita` di caricamento dinamico di moduli
- estendibilita`
- affidabilita`: testing in tempi brevissimi da parte di migliaia di utenti/sviluppatori
- portabilità