LABORATORIO DI PROGRAMMAZIONE 2
Corso di laurea in matematica

Sistemi Operativi : introduzione

Marco Lapegna
Dipartimento di Matematica e Applicazioni
Universita' degli Studi di Napoli Federico II

wpage.unina.it/lapegna

cos'e' un sistema operativo

Chi ha mai usato uno strumento elettronico **facendo riferimento direttamente alle componenti elettroniche?**





Quindi, in generale

UN
SISTEMA OPERATIVO
E'

un **ambiente software** che agisce da **intermediario** tra l'utente e l'hardware di un computer.

Dal punto di vista dell'utente, un S.O.

- Permette l'esecuzione dei programmi e rende più semplice la soluzione di possibili problemi legati alla gestione della macchina
- Rende il sistema di calcolo semplice da usare.

E' un ambiente per eseguire programmi in modo facile ed efficiente.

Dal punto di vista dell'hardware, un S.O.

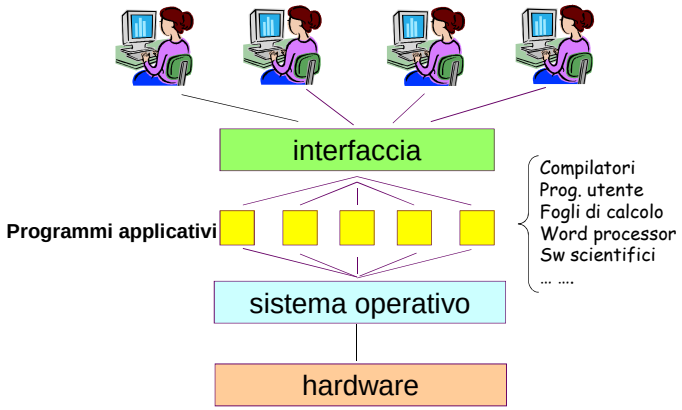
- Gestisce le risorse: controlla ed alloca le risorse hardware (in modo equo ed efficiente).
- Controlla l'esecuzione dei programmi utente e le operazioni sui dispositivi di I/O facendo fronte ad eventuali errori

E' un insieme di programmi sempre in esecuzione e a diretto contatto con l'hardware (Kernel)

Componenti di un sistema di calcolo

1. Hardware — fornisce le risorse fondamentali di calcolo (CPU, memoria, device di I/O).
2. Sistema Operativo — controlla e coordina l'utilizzo delle risorse hardware da parte dei programmi applicativi dell'utente.
3. Programmi Applicativi — definiscono le modalità di utilizzo delle risorse del sistema, per risolvere i problemi di calcolo degli utenti (compilatori, database, video game, programmi gestionali).
4. Interfaccia — permette agli utenti di accedere alle risorse del sistema attraverso un linguaggio di comando, desktop grafici o altro
5. Utenti — persone, altri macchinari, altri elaboratori.

Componenti di un sistema di calcolo

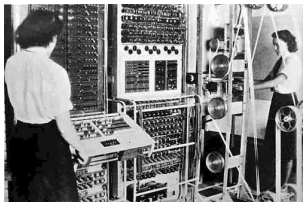


Quali sono le funzioni di un S.O. ?

La struttura dei moderni sistemi operativi e' il frutto di aggiunte di programmi introdotti nel tempo per far fronte a specifiche esigenze

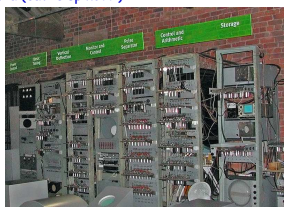
Uno sguardo alla storia del calcolo …

Fino al 1948-49



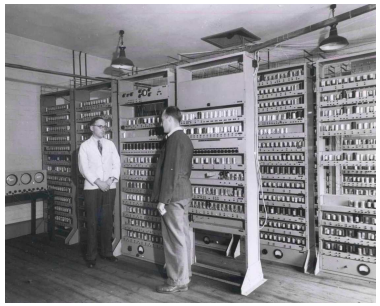
Colossus (Inghilterra, 1944): usato per decriptare codici durante la guerra. Elettronico, programma non in memoria (cavi e spinotti)

Zuse Z3 (Germania, 1941). Primo calcolatore elettromeccanico, programma in memoria



SSEM (Inghilterra, 1948). Non un calcolatore, ma un dispositivo per sperimentare nuove tecnologie per le memorie (elettronico e programma in memoria)

EDSAC (Inghilterra, 1949)



Primo calcolatore elettronico con programma in memoria

Ispirato dai lavori di John Von Neumann

La memorizzazione del programma avveniva mediante lettura di nastri perforati

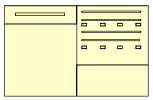
Macchina di Von Neumann: calcolatore con CPU separata dalla memoria che e' capace di contenere istruzioni e dati

Anni 1940-1950

- interazione diretta uomo – calcolatore
- assenza di qualunque tipo di software di sistema

Svantaggi:

- Elaborazione molto lenta e inefficiente
- Alta possibilita' di errori
- gestione inefficiente del sistema



Soluzione
Schede e nastri magnetici

Un centro di calcolo negli anni 50

Un lettore di nastri

IBM 701

Alcuni problemi

- La nascita di numerosi strumenti di I/O, ognuno con caratteristiche differenti
  (buffer, flag, registri, bit di controllo, bit di stato,…)

Una semplice operazione di I/O poteva essere composta di numerose istruzioni di linguaggio macchina
(magari da ripetere molte volte da molti utenti)

Scrivere una procedura specifica per ogni dispositivo
(Driver del dispositivo)

Nascita delle librerie di I/O

Primi linguaggi di programmazione

- Fortran (1957), Algol (1960) e Cobol (1961)
- Necessita' di un traduttore (compilatore)

- Passi da eseguire:
  1. Montare il nastro del compilatore
  2. Tradurre il programma utente
  3. Smontare il nastro del compilatore
  4. Collegare il programma utente tradotto con le librerie di I/O
  5. Montare il nastro con il programma utente
  6. Esecuzione del programma

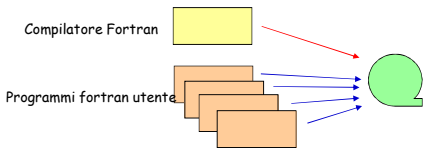
Ancora troppi passi (tutti manuali !!!)

Sistemi batch

Programmi con caratteristiche simili
possono essere raggruppati in lotti (batch)

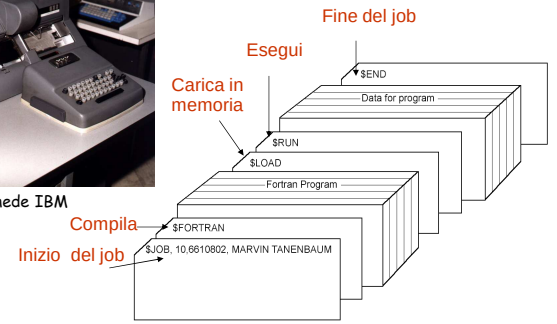
Si riduce il tempo di preparazione

Esempio : un insieme di programmi Fortran



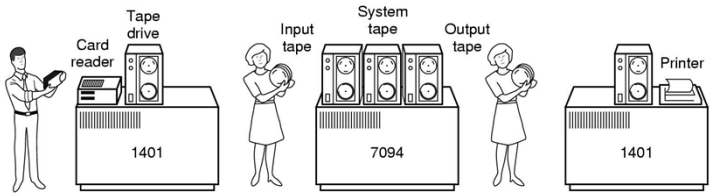
Struttura di un tipico job Fortran

Perforatrice di schede IBM



La struttura del job e' definita dalle schede di comando

Sistemi batch (a lotti)

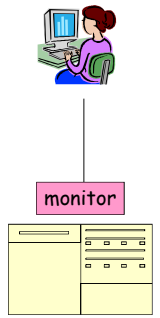


- Un lotto di job sotto forma di schede viene trascritto su di un nastro
- Il nastro viene montato nel sistema centrale (7094) che li elabora uno alla volta. I risultati sono trascritti su di un ulteriore nastro (nel frattempo e' possibile cominciare a creare un altro nastro di job sul sistema ausiliario 1401)
- Il nastro con i risultati viene montato sul sistema ausiliario 1401 che li stampa (nel frattempo il sistema centrale 7094 elabora altri job su un altro nastro)

Monitor residente

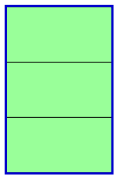
Con i sistemi batch il controllo viene trasferito automaticamente da un job al successivo da un programma residente in memoria chiamato **monitor che legge le schede di comando e chiama le opportune routine**.

- Monitor residente:
  - Legge le schede di controllo
  - Chiama le relative routine di servizio
  - Usa il programma utente e i dati come input di tali routine



primo embrione di sistema operativo

Il monitor residente



Interprete delle schede di controllo

Caricatore (copia il programma dal nastro alla memoria)

Sequenzializzatore (gestisce il passaggio da un programma all'altro)

Il monitor garantisce le sequenza automatica dei programmi cosi' come era indicato nelle schede di controllo

La lentezza dell'intervento umano viene sostituita dal calcolatore stesso

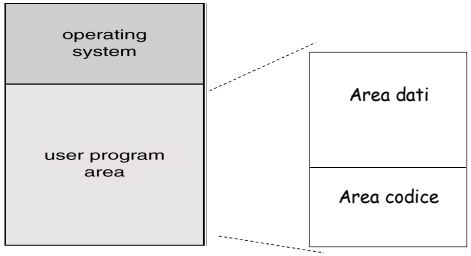
Primi sistemi operativi per sistemi batch

- **GM OS** (1955)
  - Per alcuni e' il primo sistema operativo in assoluto per sistemi batch. Sviluppato dalla General Motors per l'IBM701, basato su monitor residente
- **SAGE** (1957 ?)
  - Semi-Automatic Ground Environment system. Programma di controllo per sistemi IBM. Primo sistema operativo real time sviluppato in ambito militare.
- **Fortran Monitor System** (1958)
  - Sistema operativo sviluppato dall'aviazione americana per calcolatori IBM. Primo con supporto per un linguaggio ad alto livello.
- **SOS** (1959)
  - Sistema operativo sviluppato dall' IBM SHARE Users Group per l'IBM 709

Sistemi batch: caratteristiche

- Presuppongono un operatore ≠ utente
- Assenza di interazione fra utente e job a *run-time*.
- Presuppongono come periferica di ingresso un lettore di schede o nastri.
- Riducono il tempo di setup riunendo in lotti (*batch*) job simili.
- Aumento del throughput
- viene eseguito un solo job alla volta fino al suo completamento

Uso della memoria in un sistema batch negli anni 50



Sistemi batch : problemi

- le operazioni di elaborazione e di I/O non possono essere svolte contemporaneamente.
- E' possibile tenere in memoria un solo job alla volta
- Lentezza dei lettori di schede e nastri rispetto alla CPU (anche 3 ordini di grandezza).

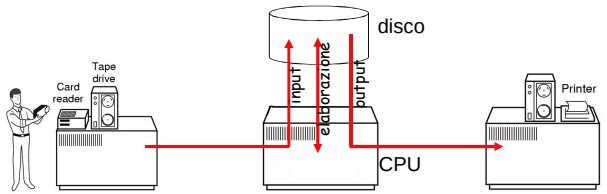
Uso inefficiente della CPU

Soluzione: uso di memorie di massa veloci ad accesso diretto (dischi)

Un centro di calcolo negli anni '60

OS/360 primo s.o. portabile su una famiglia di calcolatori

Spooling



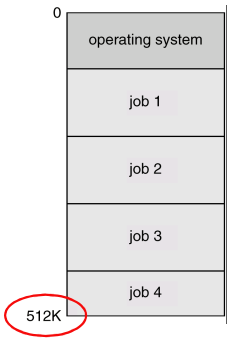
- I job e i dati vengono caricati automaticamente dal lettore di schede al disco
- Mentre viene eseguito un job, il SO…
  - Legge il prossimo job dal lettore di schede su un'area del disco (job queue).
  - Stampa l'output di job eseguiti precedentemente, copiandoli dal disco su un nastro.

Sovrapposizione di I/O e elaborazione

Sistemi multiprogrammati

Più job vengono mantenuti nella memoria principale contemporaneamente in particolari strutture (job pool) e l'uso della CPU viene diviso fra loro

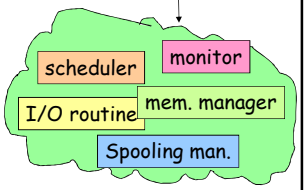
(Scheduling della CPU)

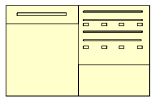


Uso della memoria un un sistema multiprogrammato negli anni 60

S.O. per calcolatori con multiprogrammazione

- la multiprogrammazione impone al SO…
  - Presenza di routine per I/O.
  - Gestione della memoria
  - Scheduling della CPU —
  - Gestione dei dischi e delle unita' di I/O.





Sistemi multiprogrammati: problemi

Se nel job pool e' presente un job molto lungo, eventuali job piu' piccoli devono attendere la fine di tale job anche se devono usare la CPU per poco tempo

tempo medio di elaborazione elevato

Soluzione: sospendere l'esecuzione di job che superano un fissato tempo limite (time slice)

Sistemi time-sharing

- un job viene sospeso quando
  - deve effettuare una operazione di I/O
  - ha esaurito il suo tempo limite
- La CPU viene commutata tra più job che vengono mantenuti contemporaneamente in memoria e sul disco
- I job sono sottoposti a *swap-in* dal disco alla memoria ed a *swap-out* dalla memoria al disco.

Riduzione del tempo medio di attesa

Sistemi interattivi

Con la multiprogrammazione e il time sharing si riducono i tempi medi di attesa e ogni utente ha la sensazione di essere l'unico utente del sistema

SISTEMI INTERATTIVI

- permettono la comunicazione on-line tra utente e sistema; quando il SO termina l'esecuzione di un comando, si aspetta il successivo comando da tastiera.
- devono essere sempre disponibili per l'accesso a dati e codice da parte degli utenti (File system on-line).

problemi

Il time sharing permette l'accesso alla cpu a molti programmi residenti in memoria

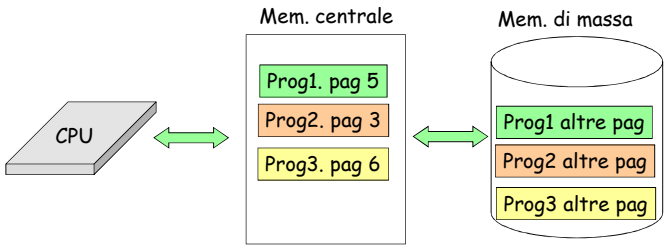
La memoria ha una capacita' limitata

Come fare a tenere numerosi programmi (anche grandi) contremporaneamente in memoria?

soluzione

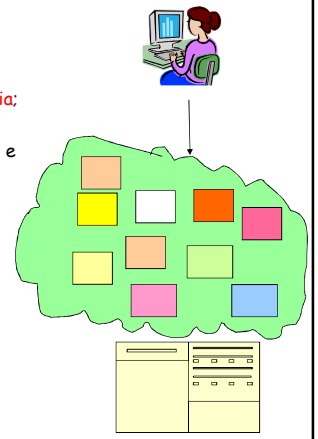
Dividere il programma in pezzi (pagine) e conservare:
- in memoria centrale solo la pagine con la sezione di codice da eseguire
- in memoria di massa il resto delle pagine

MEMORIA VIRTUALE



S.O. per calcolatori con time-sharing

- Il time-sharing impone al SO…
  - La gestione e la protezione della memoria;
  - La gestione della memoria virtuale;
  - La gestione di un file system on-line e della memoria secondaria di supporto;
  - La presenza di meccanismi per l'esecuzione concorrente, la comunicazione e la sincronizzazione dei job;
  - La presenza di meccanismi per evitare i deadlock.
  - La presenza di differenti unita' di I/O



Sistemi operativi negli anni '60

- Atlas (Inghilterra, 1961)
  - gestione particolarmente efficiente delle memorie e uso della memoria virtuale
- CTSS (California, 1961)
  - Il "bisnonno di Linux"
- XDS-940 (California, 1965)
  - Tra i primi ad utilizzare il time sharing e introduzione della duplice modalita' di esecuzione (utente / sistema)
- THE (Olanda, 1968)
  - Progettazione accurata e gestione efficiente e semplice dei programmi in esecuzione con i semafori
- RC4000 (Danimarca, 1970)
  - Introduzione di moderni meccanismi di comunicazione tra i programmi in esecuzione

Storia di Unix (1)

John McCarthy e Herb Teager (Stanford, 1961) pongono le basi per il primo S.O. time sharing: il CTSS (Compatible Time Sharing System) Sistema prototipale, piccolo e con poche funzionalita'

Fernando José Corbató (MIT, 1965), tra i progettisti di CTSS dirige il progetto MAC per il S.O. MULTICS (MULTIplexed Information and Computing Service) Evoluzione del CTSS, ma grande, complesso e poco efficiente

Kenneth Thompson (Bell labs, 1969), dopo il ritiro della Bell Labs dal progetto MULTICS, sviluppa UNIX semplificando il progetto MULTICS.
S.O. con time sharing e memoria virtuale. Disponibile per calcolatori medio/grandi.

Storia di Unix (2)

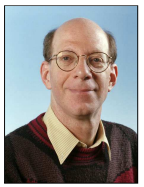
Dennis Ritchie (Bell Labs, 1972), sviluppa il linguaggio C e riscrive UNIX nel nuovo linguaggio. Primo S.O. portabile su differenti piattaforme

Bill Joy (Berkley, 1978), a partire dal codice che i Bell Lab furono tenuti a distribuire a causa di leggi antitrust, guida un gruppo di ricercatori nello sviluppo di BSD Unix con funzionalita' di rete TCP/IP. Una delle piu' diffuse distribuzioni negli anni 80.

Paul Allen e Bill Gates (1980), alla Microsoft sviluppano XENIX, prima versione commerciale per microcomputer. Seguiranno le prime versioni commerciali per processori Intel: Venix, QNX, Idris,..
Poco diffuse per la scarsa potenza dei processori per PC

Storia di Unix (3)

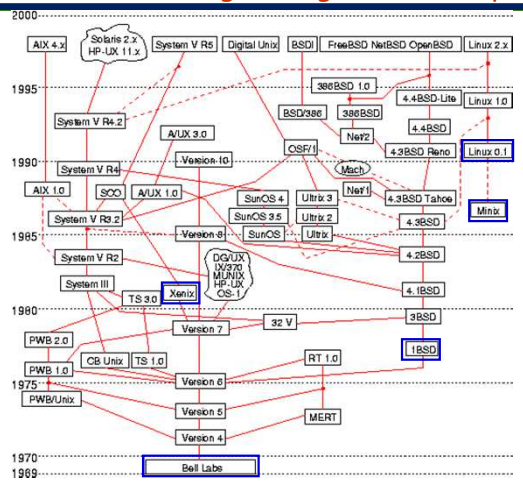
Richard Stallman (Stanford, 1985), lancia il progetto GNU per la distribuzione libera e gratuita del software di Unix. Sviluppa solo alcuni tool (compilatori, debugger, ..)



Andrew Tanenbaum (Amsterdam, 1986), professore alla Vrije University sviluppa Minix, piccolo S.O. Unix per processori Intel (ora sufficientemente potenti) . Sviluppato per usi didattici - gratuito

Linus Torvalds (Finlandia, 1992) sviluppa Linux a partire da Minix e dai tools realizzati nel progetto GNU. Free e open source. Motivo: costi troppo alti per le licenze . Versione per processori Intel poi adottata da tutte le grandi industrie

L'albero genealogico di Unix (parziale)



I Personal Computer

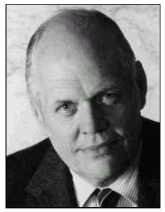
- Avvento dei microprocessori negli anni 70
- miglioramento dei processi tecnologici

Obiettivo:
Costruire calcolatori economici e facili da usare (personal computer)

Federico Faggin (1971)

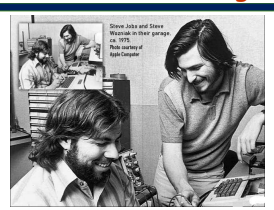
- Introduzione di numerosi mezzi di I/O (tastiera, mouse, monitor, piccole stampanti).
- Possono adottare tecnologie gia' sviluppate per i sistemi operativi per mainframe;
- un solo utente utilizza il PC e quindi non sono necessarie tecniche sofisticate per l'utilizzo della CPU, né sono richieste funzioni avanzate di protezione.

Chi ha avuto ragione?

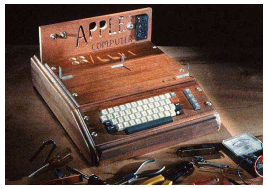


"There is no reason anyone would want a computer in their home." –

Ken Olsen, presidente, e fondatore della Digital Equipment Corp., 1977



Steven Wozniak, Steve Jobs



L'Apple I (1977)

S.O. per personal computer

- MS-DOS (MicroSoft Disk Operating System -1980)
- Windows (1985), Windows 95, Windows 98, Windows Me
- Windows NT, Windows 2000, Windows XP

- MacOS

- Minix (1987)
- Linux (1994)

Caratteristiche:
- facilita' di uso mediante GUI
- efficienti per elab. testi, fogli elettronici,…
- piccole dimensioni

Storia di Windows (1)

1980 - Microsoft, dietro commessa della IBM produce un S.O. per personal computer chiamato PC-DOS 1.0. Per i PC IBM compatibili viene prodotta una versione simile chiamata MS-DOS

1982 – DOS V.1.2 con driver per floppy disk

1983 – DOS V.2.0 con driver per hard disk e gestione del file system

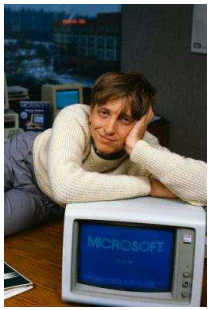
1984 – DOS V.3.0 con supporto di rete

1985 – Windows 1.0 prima interfaccia grafica

1987 – Windows 2.0 con funzionalita' di multitasking

1988 – DOS V.4.0 con shell e gestione della memoria estesa

1991 – DOS V.5.0 gestione piu' efficiente della memoria



B. Gates – 1977

Storia di Windows (2)

1992 – Windows 3.1. oltre 3 milioni di copie in due mesi

1993-94 – Windows NT 3 e NT4. S.O. object oriented per server di alto livello. Stessa interfaccia grafica di Windows 3.1 ma kernel completamente riscritto

1995 – Windows 95: S.O. per applicazioni a 32 bit. Non piu' una interfaccia grafica come Windows 3.1 ma un completo S.O.

1996-1998 – Windows CE 3.0. S.O. per computer palmari

1998 – Windows 98. browser integrato, supporti Java e HTML e vari supporti hardware (USB, firewire, DVD,…)

2000 – Windows 2000 con supporto molto efficiente per le connessioni di rete

2001 – Windows XP: supporto ai multiprocessori

2009 – Windows 7: supporto multitouch e per CPU multicore

B. Gates – 2004

L'Unione fa la forza

Problemi grandi ⟺ Utilizzo di molte risorse

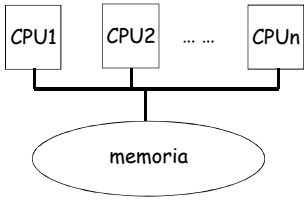
calcolatori paralleli (*Tightly coupled system*)

Un sistema di unità processanti omogenee, strettamente collegate che comunicano per risolvere problemi su larga scala in maniera efficiente

Principale obiettivo:
Ridurre i tempi di elaborazione

- calcolatori a memoria condivisa
- calcolatori a memoria distribuita

calcolatori paralleli a memoria condivisa

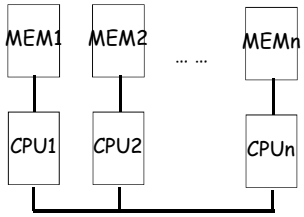


Le CPU condividono la memoria e il clock

Principali svantaggi:
- sincronizzazione degli accessi alla memoria
- scarsa scalabilita'

Seymour Cray e il Cray X-MP (1984)

Calcolatori paralleli a memoria distribuita



Cosmic Cube con 64 schede Intel 8086 + memoria (1983)

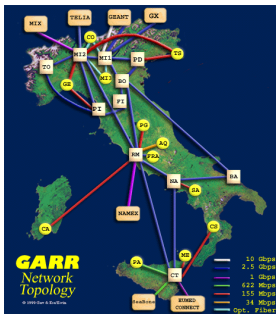
Ogni CPU ha una propria memoria e comunica mediante una rete

Principali svantaggi:
- reti lente
- comunicazione tra le CPU

Goeffrey Fox

Le reti geografiche

- 1968 primi progetti di una rete di computer dell'ARPA per connettere 12 universita' e centri di ricerca. Bandwidth = 56 Kbits



Rete GARR che connette univ. e centri di ricerca italiani

1994	2Mbit/sec
1998	32Mbit/sec
2002	2.5Gbit/sec
2006	10Gbit/sec

Crescita di 1000 volte in 8 anni (x2 in 9 mesi)

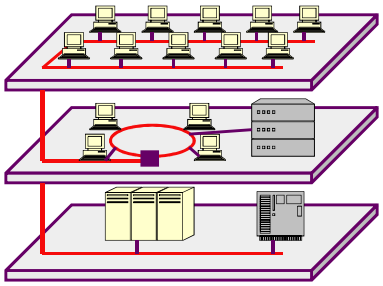
Utilizzare i calcolatori connessi ad una rete come una unica risorsa di calcolo (calcolo distribuito)

Sistemi distribuiti (*Loosely coupled system*)

Un sistema di unità processanti non omogenee, autonome, indipendenti, geograficamente distribuite che sono aggregate per risolvere problemi su larga scala in maniera efficiente

Principale obiettivo:
Aggregare risorse

Un sistema distribuito e' ...



Una Rete Aziendale composta da differenti calcolatori collegati tra loro tra reti differenti

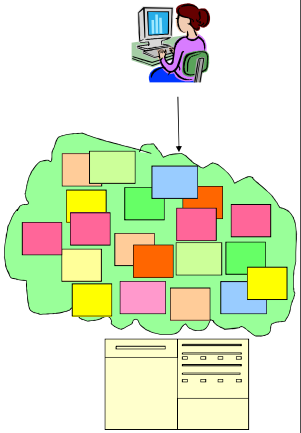
- Richiedono un'infrastruttura di rete.
- La rete può essere una LAN (*Local Area Network*) o una WAN (*Wide Area Network*).

S.O. per calcolatori paralleli/distribuiti

- Sistemi operativi distribuiti
  - Minor autonomia fra computer;
  - unico sistema operativo che controlla e gestisce in maniera trasparente l'intera rete di computer.
- Sistemi operativi di rete
  - Consentono la condivisione di file;
  - Garantiscono uno schema di comunicazione;
  - Vengono eseguiti indipendentemente per ciascun computer in rete.

S.O. per calcolatori paralleli/distribuiti

- i calcolatori paralleli/distribuiti impongono al SO…
  - fault tolerance
  - sincronizzazione
  - gestione eterogeneita'
  - infrastruttura di rete
  - protocolli di comunicazione
  - bilanciamento del carico
  - sicurezza
  - Accesso a risorse remote



il calcolatore piu' veloce al mondo (2014)?

Tianhe-2 al National Supercomputer Center in Guangzhou (Cina)

3120000 unita' di calcolo indipendenti
architettura ibrida distribuita/condivisa
1024000 GB
oltre 33 000 000 000 000 000 = 33×10^{15} operazioni al secondo
sistema operativo Linux

Sistemi operativi tempo-reale

- Spesso utilizzati per applicazioni dedicate,
  - Telecomunicazioni,
  - Difesa militare
  - Controllo traffico aereo/ferroviario,
  - Controllo di sistemi industriali

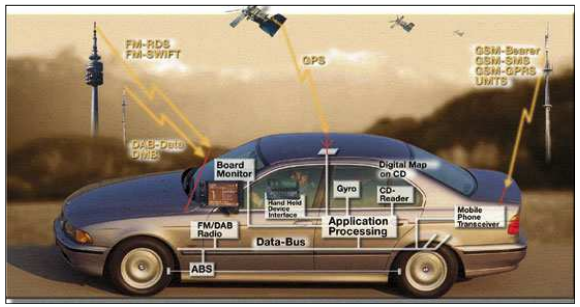
Sistemi operativi tempo-reale

Sono caratterizzati da tempi di risposta certi

- Hard real-time:
  - Memoria secondaria limitata o totalmente assente, dati memorizzati in memorie volatili o di sola lettura (ROM).
  - Non realizzano il time-sharing. Le funzionalità hard real-time non sono supportate dai SO general purpose.
- Soft real-time:
  - I task critici hanno priorità sugli altri task e la mantengono fino al completamento dell'esecuzione.
  - Utile nelle applicazioni che richiedono caratteristiche avanzate del SO (multimedia, realtà virtuale), ma non per controllo industriale e robotica.

Sistemi embedded (o integrati)

- sono i computer delle automobili, delle lavatrici, delle centraline di allarme, dei bancomat …
- sistemi molto semplici che svolgono mansioni molto specifiche
- danno priorita' alla gestione dei dispositivi fisici



Sistemi operativi per smartphone e tablet

- Caratteristiche:
  - Consumo contenuto
  - Piccole dimensioni

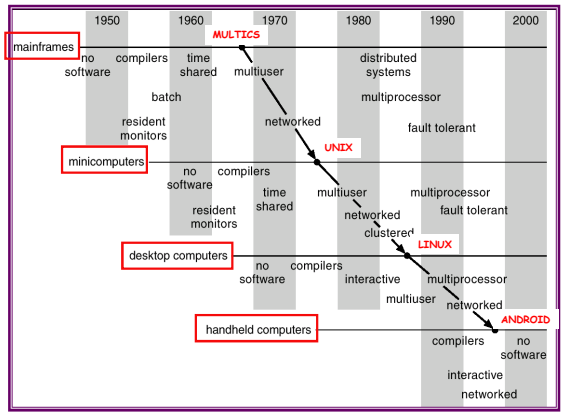
Android

BlackBarry OS

windows phone

iOS

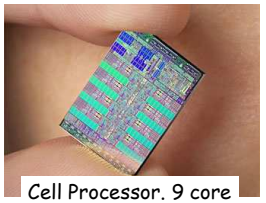
Migrazione temporale di concetti e caratteristiche dei SO



Altro esempio di "migrazione"

1984

Cray X-MP. 4 CPU



2006

Cell Processor. 9 core

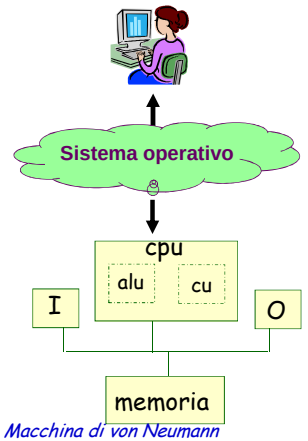
LABORATORIO DI PROGRAMMAZIONE 2
Corso di laurea in matematica

Sistemi Operativi : interazione tra software e hardware

Marco Lapegna
Dipartimento di Matematica e Applicazioni
Universita' degli Studi di Napoli Federico II

wpage.unina.it/lapegna

Compiti del S.O.



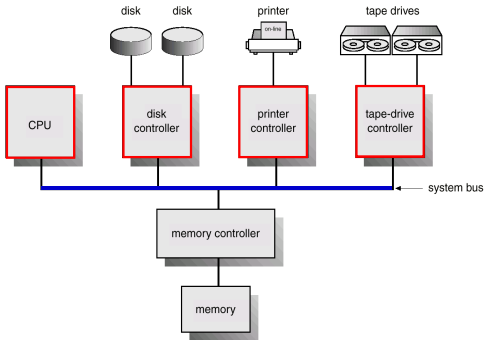
Macchina di von Neumann

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

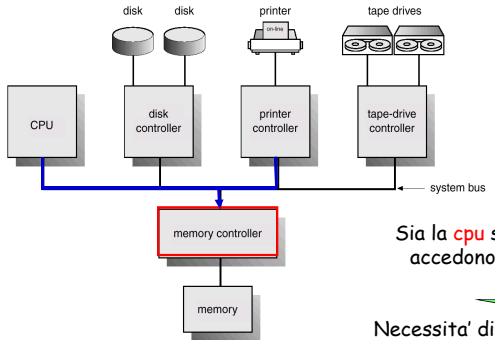
Architettura di un Sistema di Calcolo



Controller
=
dispositivo hw per la gestione delle periferiche

- la cpu e i controller operano in modo concorrente
- la cpu e i controller comunicano attraverso un bus

Architettura di un Sistema di Calcolo



Sia la cpu sia i controller accedono alla memoria

Necessita' di sincronizzazione degli accessi

Controller della memoria

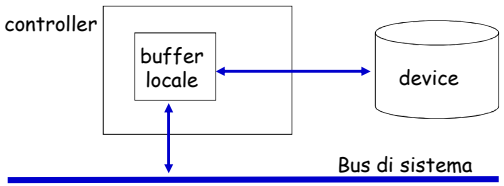
la CPU

Compiti della CPU:
- recuperare una istruzione dalla memoria
- Decodificarla per determinare tipo e operandi
- eseguirla

Tutte le CPU hanno un insieme di registri:
- per contenere dati in transito da e per la memoria
- program counter (indirizzo della prossima istruzione)
- stack pointer (indirizzo dello stack)
- program status word (insieme di bit di controllo)
- per delimitare lo spazio di indirizzamento del programma in esecuzione

I Dispositivi di I/O

- Ogni *controller* ha un suo buffer locale.
- Le operazioni di I/O avvengono dal dispositivo verso i buffer locali e viceversa.
- La CPU sposta i dati dalla memoria ai buffer e viceversa.



problema

Come fa la cpu a sapere che i dati sono stati trasferiti nel buffer locale del controller?

I soluzione:
La CPU interroga in continuazione il controller

Tale tecnica, chiamata attesa attiva (o polling), tiene impegnata la CPU fino al completamento dell'operazione di I/O

II soluzione:
Il controller avvisa la CPU della fine dell'operazione di I/O mediante un evento

Gli eventi

Gli eventi (oppure segnali, oppure interruzioni) sono il principale meccanismo con cui si sincronizzano le azioni di un moderno sistema operativo

La ricezione di un evento avvisa la CPU che deve fare qualcosa

Se non ci sono processi, dispositivi da servire o utenti con cui interagire il sistema operativo rimane inattivo nell'attesa di un evento
(i S.O. sono *events driven*)

eventi: dipendono dall'architettura

In generale due tipi di eventi

esempio: processori Intel

evento segnalato da una componente hardware (disp. di I/O, memoria, timer)

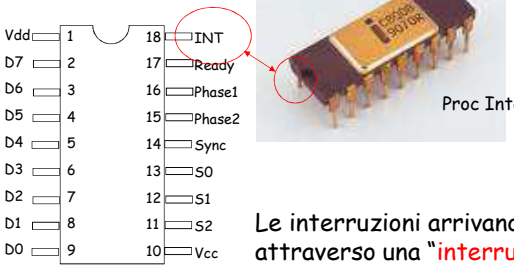
segnali di interruzione (interrupt)

evento segnalato da un programma in esecuzione

Segnali di eccezione
(trap → es: div per 0
fault → es: overflow
abort → es: errore in un evento)

In generale si parla sempre di interruzioni

La interrupt line



Proc Intel 8008 (1974)

Le interruzioni arrivano alla CPU attraverso una "interrupt line"

Alla fine di ogni ciclo macchina la CPU controlla la presenza di un segnale sulla interrupt line

Come viene gestito una interruzione

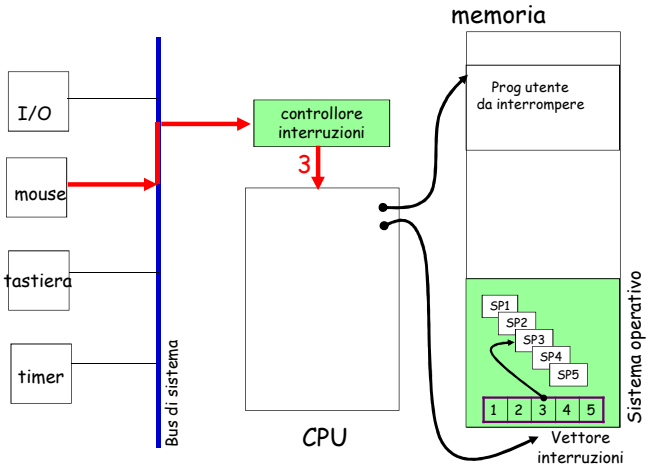
- Un dispositivo attiva una comunicazione elettrica con un controllore delle interruzioni della CPU
- Il controllore delle interruzioni informa la CPU codificando il tipo di interruzione
- La CPU interrompe l'esecuzione dell'istruzione corrente e salva lo stato del programma in esecuzione (dati e registri della CPU).
- trasferisce il controllo ad una *interrupt service routine* (ISR) il cui codice si trova in una prefissata zona della memoria
- Le ISR (Interrupt Service Routine) eseguono il codice specifico per gestire il segnale appena giunto (ad es. trasferisce il dato dal buffer locale del controller alla memoria)
- Al termine dell'esecuzione della ISR, il S.O. ripristina lo stato del programma e riprende l'esecuzione

La gestione delle interruzioni

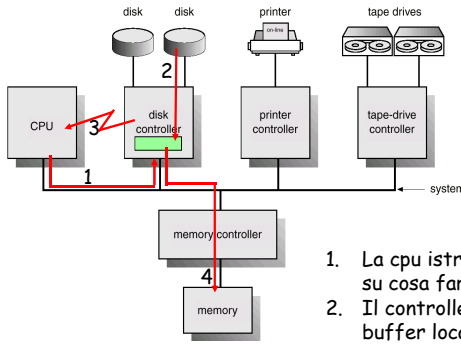
L'efficienza di un Sistema Operativo dipende dall'efficienca con cui sono gestite le interruzioni

Per rendere efficiente la gestione delle interruzioni si usa di solito una sola ISR che gestisce una tabella di puntatori contenente gli indirizzi delle varie procedure di servizio (vettore delle interruzioni)

Gestione delle interruzioni

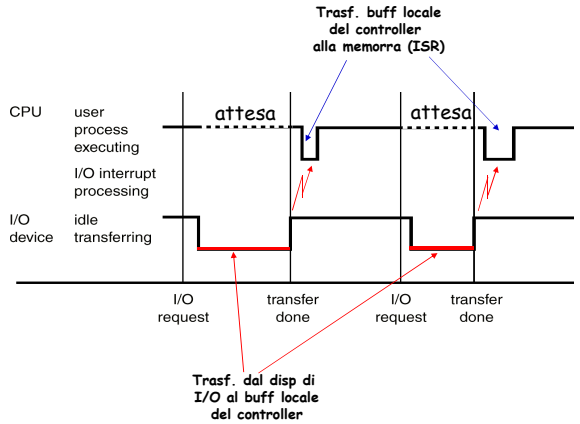


Esempio: lettura di un dato (I/O sincrono)



1. La cpu istruisce il controller su cosa fare e attende
2. Il controller carica i dati nel buffer locale
3. Il controller manda un interrupt di fine I/O alla cpu
4. La cpu trasferisce il dato dal buffer locale alla memoria

Time Line per l'I/O di un singolo processo



Problema

Se la CPU effettua una operazione di I/O, deve attendere il completamento del trasferimento dei dati dal dispositivo al alla CPU (I/O sincrono)

Ma i dispositivi sono molto piu' lenti della CPU

Uso inefficiente della CPU

I/O sincrono

- vantaggi:
  - una sola richiesta di I/O pendente alla volta
  - la CPU riconosce subito da quale dispositivo arriva il segnale di interruzione

MA

- svantaggi:
  - la CPU rimane inattiva per tutta la durata dell'I/O
  - no a operazioni I/O in parallelo
  - no a sovrapposizione di I/O e calcolo

I/O asincrono

Invece di rimanere inattiva, la CPU potrebbe essere impiegata a gestire altri programmi (multiprogrammazione)

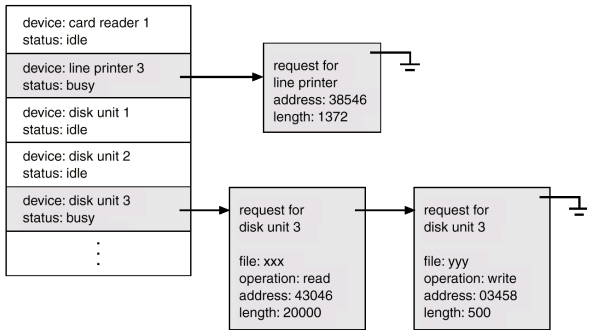
Dopo l'avvio dell'I/O il controllo deve tornare subito alla CPU (I/O asincrono)

se c'e' un solo processo in esecuzione, tale processo deve comunque attendere la fine dell'operazione di I/O

Possibilita' di numerose richieste di I/O pendenti

Device-Status Table

La presenza di piu' operazioni di I/O pendenti impone la presenza di una tabella di stato dei dispositivi



problema

Alcuni dispositivi di I/O veloci
(ad es. un lettore di nastri o dischi)
sono capaci di tasferire dati
a una velocita' simile a quelle della memoria

La CPU e' costretta a interrompere di continuo il suo lavoro per gestire le continue interruzioni in presenza di numerosi dati di I/O

Importanza del controllore della memoria

DMA (Direct Memory Access)

- E' un dispositivo dedicato in grado di accedere autonomamente alla memoria
- La CPU avvia la procedura di I/O avvisando il DMA.
- Il DMA accede direttamente alla memoria per effettuare le operazioni di I/O, trasferendo blocchi di dati (da 128 a 4096 byte)
- La CPU è libera di procedere autonomamente
- viene generata una sola interruzione per blocco di dati

La memoria centrale

La memoria centrale e' il supporto su cui sono conservati dati e istruzioni

E' vista dalla CPU come una sequenza lineare di locazioni con un indirizzo

Le uniche operazioni permesse sui dati residenti in memoria sono load e store

E' il solo dispositivo di memorizzazione di grandi dimensioni direttamente accessibile dalla CPU

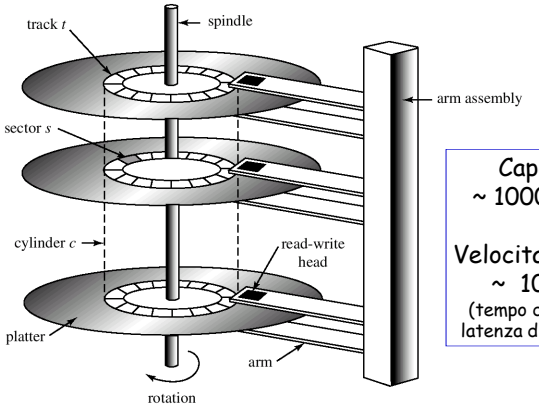
Memorie di massa

La memoria centrale e' un dispositivo volatile

Ha una capicita' dell'ordine dei 10^9 byte (Gbyte) e non e' sufficiente a contenere i modo permanente tutti i dati e i programmi di un sistema operativo

necessita' di dispositivi piu' capienti e non volatili
(memoria di massa / dischi magnetici)

Dischi magnetici



Capacita'
~ 1000 Gbyte

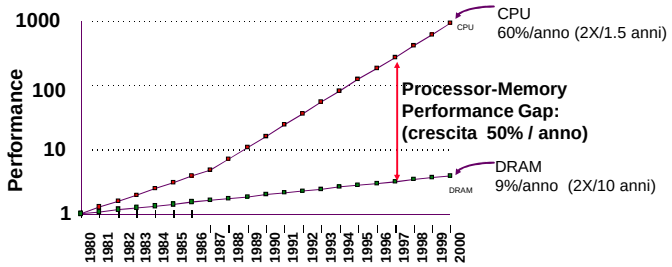
Velocita' accesso
~ 10^{-6}sec.
(tempo di ricerca + latenza di rotazione)

Gap tra memoria e CPU

Tempo per una operazione aritmetica
1 ciclo di clock (~ 10^{-9} sec)

Tempo di accesso alla memoria
4-5 cicli di clock (~ 10^{-8} sec)

Rallentamento della CPU



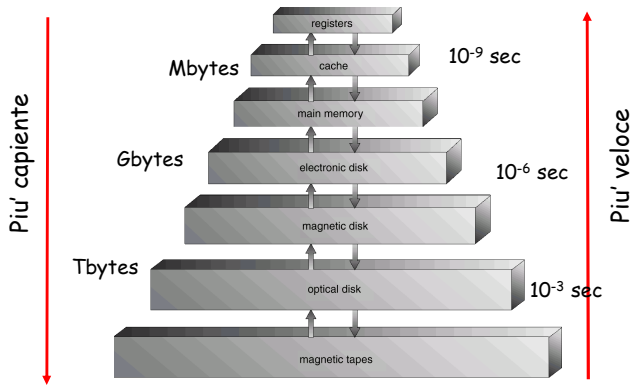
Registri e cache

Soluzione:
uso di dispositivi di memorizzazione 'on chip' capaci di supportare la velocita' operativa della CPU
(cache e registri)

Al momento del loro uso, blocchi di dati sono copiati dalla memoria nella cache

Caratteristiche:
- limitata capacita' (~1 Mbyte)
- alta velocita' di accesso (~ 10^{-9} sec)

La gerarchia delle Memorie



Caching

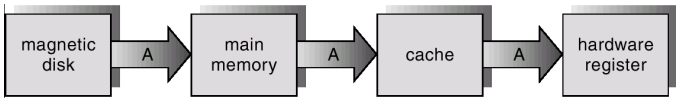
Caching – utilizzo di una memoria a piu' alta velocità per mantenere informazioni cui si accede più spesso.

- Richiede un politica di gestione della cache.
- Provoca una replicazione dei dati, quindi necessita di una politica di gestione che garantisca la *consistenza* dei dati (di solito gestita dall'hardware).

E' un concetto che puo' essere applicato a piu' livelli

- La memoria centrale puo' essere vista come una *cache* per i dischi magnetici
- I dischi possono essere visti come una cache per i supporti di backup

Migrazione di un dato dal disco ai registri



Problemi :

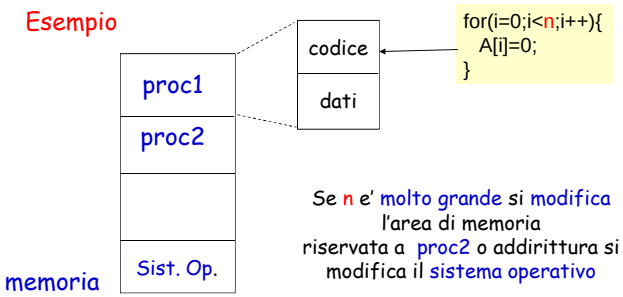
- dimensioni della cache (piu' grande = piu' costoso)
- criteri di aggiornamento ('prevedere' quali dati saranno utilizzati dalla cpu)

Gestione efficiente della cache ➔ 80% dei dati deve trovarsi nella cache quando servono

problema

La condivisione delle risorse (memoria, cpu, I/O) da parte dei processi di un sistema operativo, comporta che un errore in un programma puo' alterare il funzionamento di tutto il sistema

Esempio



Se n e' molto grande si modifica l'area di memoria riservata a proc2 o addirittura si modifica il sistema operativo

Protezione delle risorse

L'esistenza di risorse condivise richiede che il sistema operativo garantisca che un programma scorretto non possa effettuare operazioni non consentite.

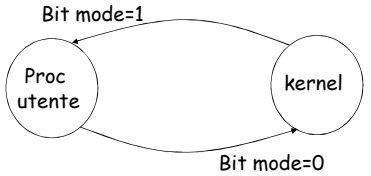
Le istruzioni possono essere eseguite in due modalità:

1. *User mode* – (un utente qualsiasi puo' eseguire un insieme ristretto di istruzioni).
2. kernel *mode* (anche *monitor mode*, *system mode* o *superuser mode*) – (il sistema operativo puo' eseguire tutte le istruzioni).

(dual mode operation)

Supporto hardware per la dual mode operation

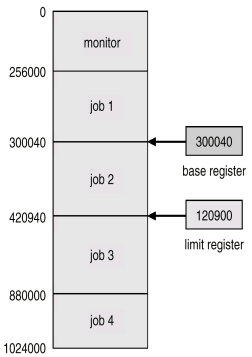
- La CPU deve essere dotata di un *Mode bit* che indica lo stato corrente: system (0) or user (1).
- Quando giunge una interruzione o avviene un errore il sistema passa in modalità "sistema" e viene attivata la procedura di servizio.



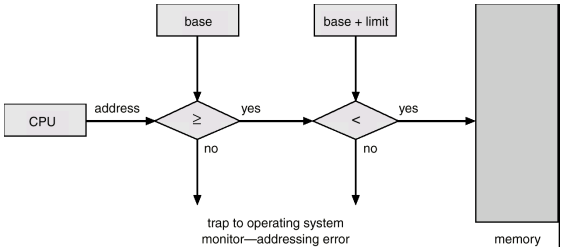
Esistono *istruzioni privilegiate* che possono essere eseguite solo in modalità superutente.

Supporto hardware per la protezione della memoria

- E' necessario garantire protezione della memoria , come minimo per le informazioni che sitrovano nel vettore delle interruzioni e nelle ISR.
- La CPU deve possedere due registri per stabilire le locazioni di memoria cui ogni programma ha diritto ad accedere.:
  - **Base register** – memorizza il più piccolo indirizzo di memoria cui l'accesso è lecito.
  - **Limit register** – Contiene la dimensione massima di memoria ad accesso consentito
- La memoria al di là dell'intervallo indicato è *protetta*.



Protezione della memoria



- Quando una istruzione viene eseguita in modalià sistema ha accesso completo a TUTTA la memoria.
- Le istruzioni per caricare i valori dei registri *base* e *limit* sono di tipo privilegiato.

Supporto hardware per la protezione della CPU

- *Timer* – genera una interruzione dopo un intervallo di tempo specificato, per garantire che il sistema operativo mantenga il controllo del sistema.
  - Il Timer viene decrementato ad ogni colpo di clock.
  - Quando il timer raggiunge il valore zero, viene generata una interruzione e il controllo passa al s.o..
- Viene in genere utilizzato per realizzare sistemi di tipo *time sharing*.
- E' utilizzato anche per calcolare l'ora attuale.
- Il Caricamento del timer è una istruzione privilegiata.

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

come organizzare tali sottosistemi

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

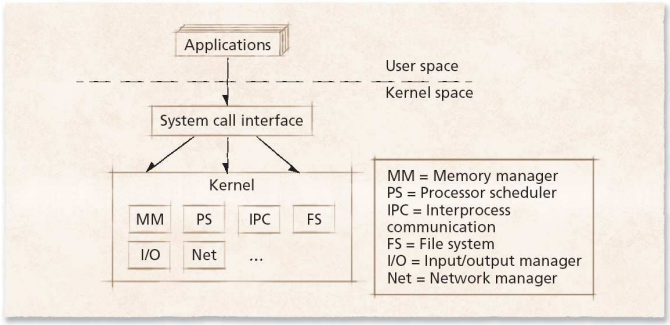
Architettura monolitica

- 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



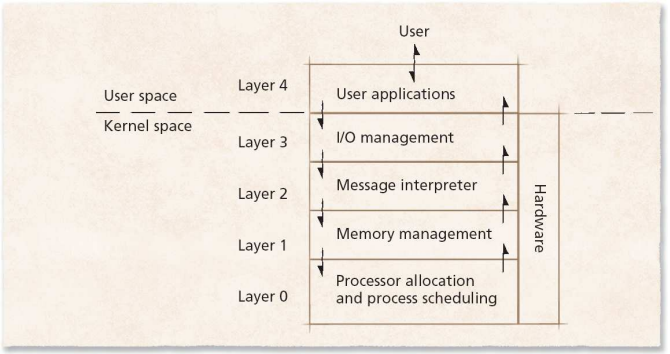
Architettura stratificata

- 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

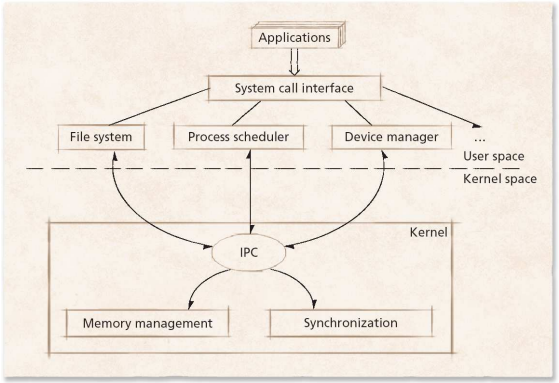


Architettura microkernel

- 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

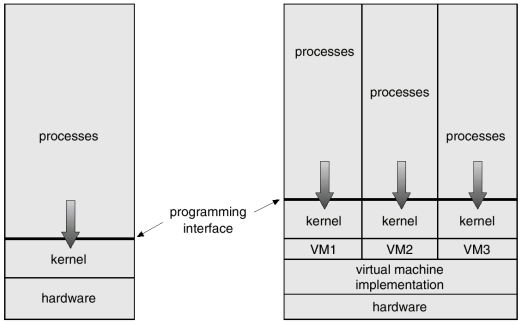


Macchine virtuali

- 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

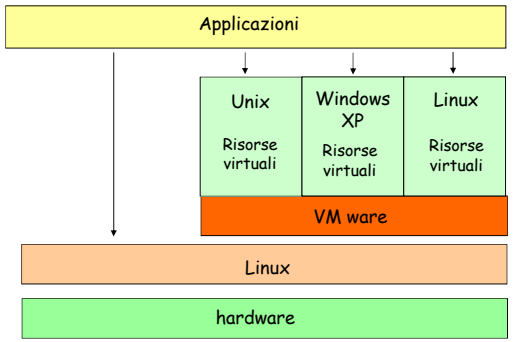
- Le risorse del computer fisico vengono condivise in modo da creare le macchine virtuali.
  - Lo scheduling della CPU può creare l'illusione che gli utenti abbiano un loro proprio processore.
  - Lo spooling e il file system possono fornire lettori di schede virtuali e stampanti in linea virtuali.
  - 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.



interazione processi – sistema operativo

e' possibile per un programma accedere direttamente alle componenti del sistema operativo?

Esempi:
- generare nuovi programmi
- accedere ai dispositivi di I/O
- accedere alla memoria secondaria
- usare la memoria centrale
- ...

I programmi richiedono servizi ai sistemi operativi

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.

Servizi del sistema operativo

- Per l'esecuzione di programmi (es. fork(), exec(),..)
- Per le operazioni di I/O (es. scanf(), printf(),)
- Per la manipolazione del file system (es. open(), close(), seek()).
- Per la gestione della memoria (es. malloc(), calloc(), ..)

le system call eseguono compiti del sistema operativo per conto del programma utente

eseguite in modalita' "system"

Le system call

Quando un programma effettua una system call, il S.O.

1. manda una interruzione alla CPU
2. pone il bit mode =0 (modalita' sistema)
3. esegue la procedura di servizio
4. ripone il bit mode=1 (modalita' utente)
5. restituisce il controllo del sistema al programma interrotto

Una chiamata di sistema viene gestita come una interruzione