v 1.0
20120919

Sistemi Operativi

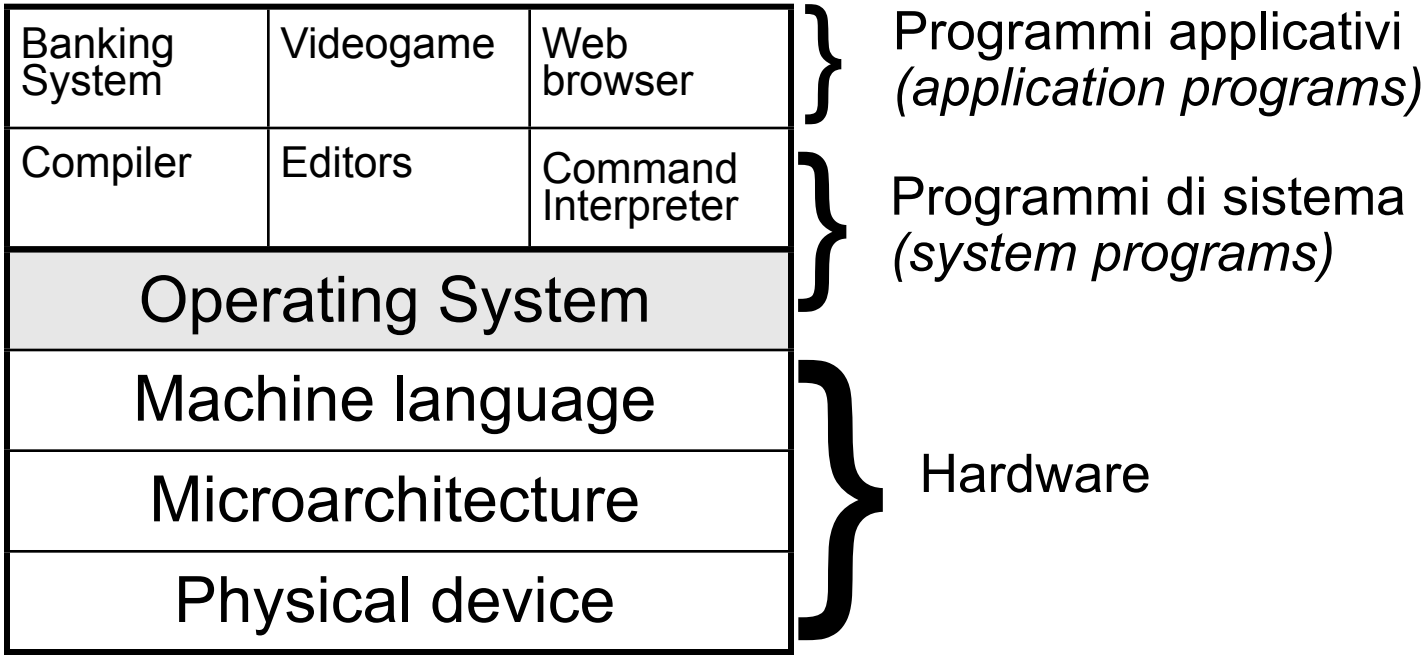
II Facoltà di Ingegneria - Cesena
a.a 2012/2013

[modulo 1a]

CONCETTI INTRODUTTIVI

SISTEMI OPERATIVI: INQUADRAMENTO

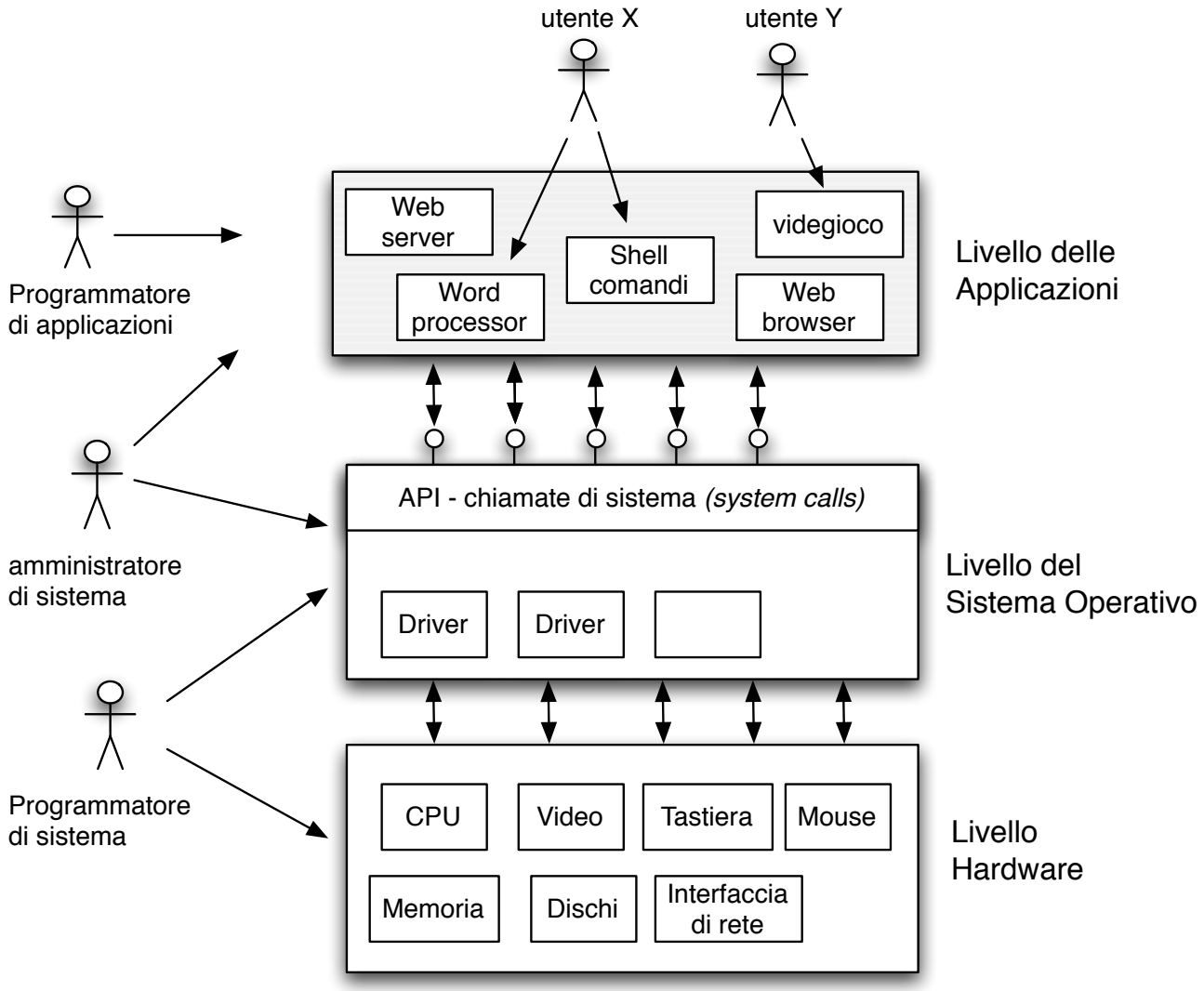
- Il **sistema operativo** (S.O) - in inglese *operating system (O.S.)* - è quella parte software di un sistema di elaborazione che controlla l'esecuzione dei programmi applicativi e funge da *intermediario* fra questi e la macchina fisica (hardware)



OBIETTIVI PRINCIPALI DI UN S.O.

- Eseguire programmi degli utenti e controllare la loro esecuzione, in particolare l'accesso alla macchina fisica
  - S.O. come **extended / virtual machine**
  - funzione di astrazione, controllo e protezione
- Rendere agevole ed efficace l'utilizzo delle risorse del computer
  - S.O. come **resource manager**
  - funzione di ottimizzazione
- *Abilitare e coordinare* le interazioni fra applicazioni, utenti, risorse
  - più applicazioni in esecuzione *concorrente*
  - più utenti che condividono e usano simultaneamente il sistema

SISTEMA DI ELABORAZIONE: LIVELLI ASTRATTI



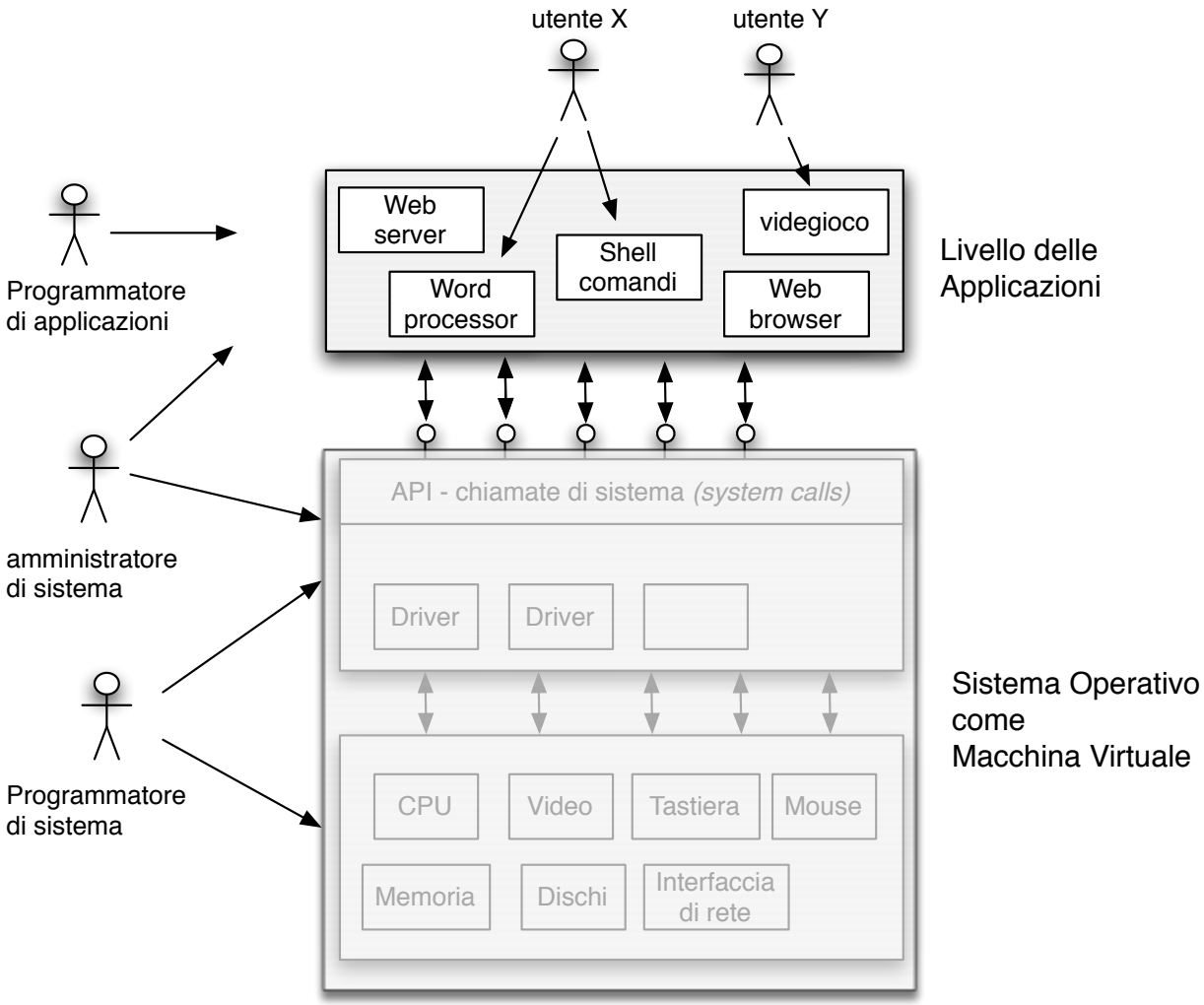
RUOLO DI MEDIAZIONE

- Il sistema operativo media l'interazione fra livello applicazione e livello hardware, controllando e coordinando l'accesso e l'uso del livello hardware richiesto dal livello applicazione, *e rendendo i due livelli il più possibile indipendenti fra loro*
  - fattorizzazione delle esigenze comuni alle applicazioni in servizi che le applicazioni possono direttamente usare astraendo dalla loro implementazione o realizzazione
- Ruolo fondamentale delle **API** (*Application Programming Interface*)
  - *interfaccia* di programmazione per i programmi
  - insieme di funzioni o *primitive* di base chiamate **chiamate di sistema** (*system calls)*

S.O. COME MACCHINA VIRTUALE

- Il sistema operativo maschera ai programmi applicativi la struttura reale della macchina fisica, facendo veder loro una **macchina virtuale (o *astratta)***
  - una macchina più semplice e ad un livello di astrazione maggiore rispetto al livello fisico
  - accessibile mediante le chiamate di sistema che nell'insieme costituiscono *l'interfaccia della macchina virtuale*
  - vista *top-down*

S.O. COME MACCHINA VIRTUALE



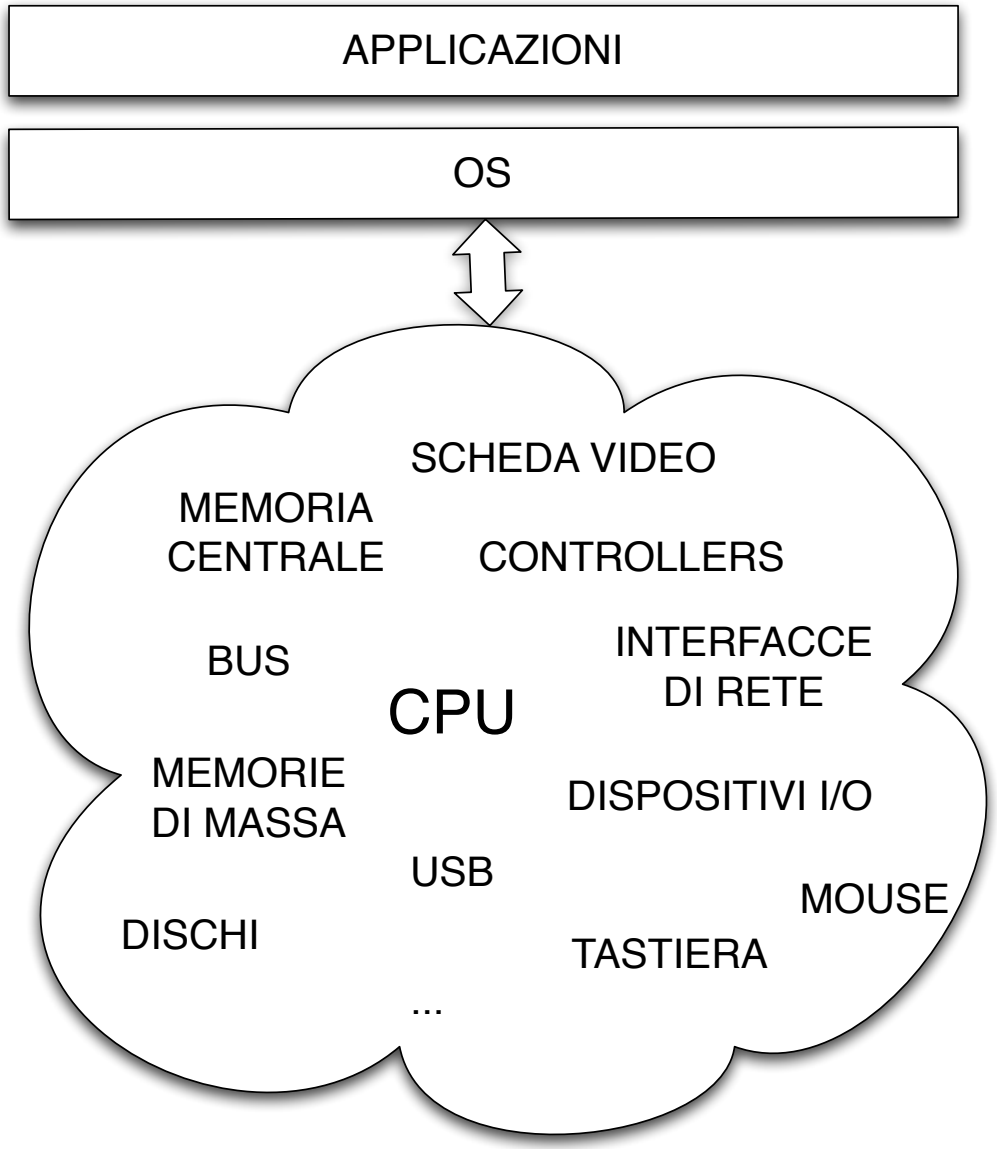
S.O. COME MACCHINA VIRTUALE - BENEFICI

- Agevolare la progettazione e programmazione delle applicazioni
  - il programmatore non deve conoscere necessariamente i dettagli specifici della macchina fisica per interagire con le risorse e può concentrarsi sulle funzionalità del programma
- Aumentare il livello di **portabilità** dei programmi
  - il medesimo programma può essere messo in esecuzione su macchine virtuali con la medesima interfaccia che girano su macchine fisiche diverse
  - semplificazione del *porting* di un programma da un sistema operativo ad un altro

S.O. COME GESTORE DI RISORSE

- Gestione risorse
  - condivise e utilizzate da *più programmi*
    - anche concorrentemente
  - eventualmente appartementi a *più utenti*
    - che utilizzano la medesima macchina
- Funzionalità di
  - **ottimizzazione** degli accessi
    - algoritmi di *scheduling*
  - **protezione** e **sicurezza**
    - evitare che un programmi in esecuzione possa provocare malfunzionamenti al sistema e ad altri programmi in esecuzione
    - garantire protezione e sicurezza dei dati degli utenti
- Vista *bottom-up*
  - dalle risorse (bottom) ai programmi applicativi

S.O. COME GESTORE DI RISORSE



VIRTUALIZZAZIONE DELLE RISORSE

- **Memoria virtuale**
  - fare in modo che un programma in esecuzione veda la memoria a disposizione come uno spazio **lineare** virtualmente illimitato
  - utilizzo trasparente (per i programmi in esecuzione) della memoria di massa come estensione di quella principale
- **File system virtuale**
  - accedere e modificare file nel file system in modo uniforme a prescindere dal tipo specifico di file system e dalla effettiva locazione dei file stessi
    - locali o remoti

ESECUZIONE DI PROGRAMMI

- **Multi-programmazione** (**multi-programming**)
  - capacità di caricare in memoria centrale più programmi che vengono eseguiti in modo da ottimizzare l'utilizzo della CPU
  - se un programma in esecuzione è impegnato in una operazione di I/O e non ha bisogno della CPU, la CPU viene allocata ad un altro programma in esecuzione
  - permette una prima forma di **multi-tasking,** ovvero di esecuzione concorrente di più programmi
- **Time-sharing** (o **multi-tasking**)
  - estensione logica della multi-programmazione funzionale all'esecuzione di programmi che richiedono interazione con l'utente
  - capacità di eseguire più programmi concorrentemente (a prescindere da operazioni di I/O), con condivisione della CPU fra i programmi in esecuzione secondo determinate strategie di schedulazione
  - piena realizzazione del concetto di multi-tasking

MODALITA' DI FUNZIONAMENTO

- Funzionamento **interrupt-driven**
  - entra in esecuzione in seguito ad eventi che sono associati all'occorrenza di **interruzioni hardware** o di **trap software**
    - interruzioni hardware
      - segnali inviati da dispositivi (es: tastiera, timer, dischi,...)
      - **asincrone**
    - trap / interruzioni sw:
      - richieste da parte dei programmi di eseguire servizi (chiamate di sistema)
      - oppure **eccezioni** generate da errori nei programmi
      - **sincrone**
- Sfruttamento di due modalità operative distinte della CPU
  - **user-mode**
    - esecuzione del programmi utente
  - **kernel-mode** (super-visor mode, provileged mode, system mode)
    - esecuzione delle parti di programma del sistema operativo

ATTIVITA' PRINCIPALI DI UN S.O.

- **Gestione dei processi**
- **Gestione della memoria**
- **Gestione della persistenza delle informazioni** (storage)
  - gestione del File System
  - gestione dei dischi
  - caching
- **Gestione dell I/O**
- **Gestione della rete**
- **Gestione della protezione e sicurezza**
- **Gestione utenti**

GESTIONE DEI PROCESSI (1/2)

- I sistemi operativi sono anzitutto ambienti che supportano l'esecuzione di programmi / applicazioni
  - l'astrazione con cui nei sistemi operativi si definisce e identifica un *programma in esecuzione* prende il nome di **processo**
- Un processo tipicamente necessita di determinate risorse per funzionare correttamente
  - tra le risorse abbiamo la CPU, che ne permette l'esecuzione, la memoria, files, device di I/O
- Il sistema operativo fornisce meccanismi / servizi fondamentali per le gestione di processi, in particolare per
  - la creazione / esecuzione
  - terminazione / sospensione
  - protezione

GESTIONE DEI PROCESSI (2/2)

- I sistemi operativi moderni permettono l'esecuzione di più processi concorrentemente
  - multi-tasking
- Meccanismi a supporto di forme di **cooperazione** fra processi
  - **comunicazione**
    - scambio di messaggi fra processi
  - **sincronizzazione**
    - coordinazione delle azioni dei processi, in particolare nell'accesso a risorse condivise
- Meccanismi per gestire forme di **competizione** fra processi
  - accesso **mutuamente esclusivo** a risorse
  - **sezioni critiche**

GESTIONE DELLA MEMORIA PRINCIPALE

- Altra risorsa fondamentale gestita dal sistema operativo è la **memoria principale**
  - la memoria può essere vista come un array di parole, ognuna con il proprio indirizzo
    - è volatile: il suo contenuto non persiste in caso di shutdown del sistema hardware o di failure
  - funge da risorsa che mette a disposizione operazioni per memorizzare e accedere velocemente dati, condivisi fra CPU e I/O device
- La gestione della memoria da parte del S.O. include le seguenti attività:
  - tener traccia di quali zone di memoria sono attualmente usate e da quale processo
  - decidere quali processi caricare in memoria centrale quando c'è spazio disponibile
  - allocare / deallocare quantità di memoria a seconda delle richieste

GESTIONE DELLA MEMORIA SECONDARIA

- Siccome la memoria principale è volatile e non sufficientemente capiente per contenere in modo permanente tutte le informaziomi e programmi che servono, è fornita allo scopo la **memoria secondaria** (*secondary storage*) come la maggior parte dei sistemi moderni adotta *dischi* come mezzo di memorizzazione secondario, sia per dati, sia per programmi
- La gestione della memoria secondaria comporta fra le attività principali:
  - gestione dello spazio libero
  - allocazione della memoria
  - scheduling dei dischi

GESTIONE DI FILE E FILE SYSTEM

- Un **file** è una collezione di informazioni, tipicamente sequenziale, definite dal suo creatore.
  - I file sono usati per rappresentare una gran varietà di informazioni, da programmi (in forma di sorgenti e binari), a puri dati.
  - una **directory** è un file che funge da contenitore (logico) di file e directory consentono di organizzare il file system in modo gerarchico.
- Per **file system** si intende il sistema adottato per gestire i file su un dispositivo di memoria di massa
  - vari tipi di file system: UFS, NTFS, FAT32,...
- La gestione dei file si compone delle seguenti attività / servizi:
  - creazione e cancellazione di file
  - creazione e cancellazione di directory
  - supporto per primitive (servizi) per la manipolazione di file
  - mapping dei file in memoria secondaria
  - backup dei file su supporti di memorizzazione non volatili

GESTIONE DEI DISPOSITIVI DI I/O

- Il sottosistema di I/O in un S.O. si occupa di fornire supporto per la comunicazione fra device esterni e componenti del sistema operativo stesso.
- E' costituito in generale da:
  - un sistema di buffer-caching
  - interfaccia generale per i **device-driver**
  - driver per gli specifici hardware device

GESTIONE DELLA RETE (NETWORKING)

- Un sistema di rete è concepibile come una collezione di processori che *non condividono né memoria, né un clock*
  - ogni processore ha la propria memoria locale e un proprio clock
- Il computer (**host**) dove risiede un processore in un sistema distribuito viene chiamato **nodo**
  - i processori in un sistema distribuito sono connessi mediante una rete di comunicazione
  - *protocolli di comunicazione* definiscono le regole con cui avviene la comunicazione
- Un sistema distribuito fornisce l'accesso ad un utente alle varie risorse, distribuite per i nodi del sistema distribuito
- L'accesso a risorse condivise consente di avere in generale:
  - speed-up della computazione
  - migliore data availability
  - migliore reliability

GESTIONE UTENTI

- I sistemi operativi moderni sono **multi-utente**, ovvero permettono l'accesso (simultaneo) al medesimo computer da parte di più utenti
- E' di fondamentale importanza allora fornire **politiche di controllo** degli accessi alle risorse e più in generale di *protezione*
  - evitare che gli utenti possano danneggiare volontariamente o volontariamente le risorse degli altri utenti (es: file) e del sistema operativo in generale
- Allo scopo i sistemi operativi moderni permettono di definire un insieme di **ruoli** con cui suddividere gli utenti che usano il sistema
  - a seconda del ruolo, è definito l'insieme delle operazioni che è possibile fare sulle risorse (es: leggere, scrivere, cancellare files..), incluso il sistema operativo stesso
- In ogni sistema è definito il ruolo di amministratore (detto anche **root** o **supervisor**) che ha i diritti per compiere qualsiasi operazione sul sistema

CENNI STORICI: EVOLUZIONE DEI S.O.

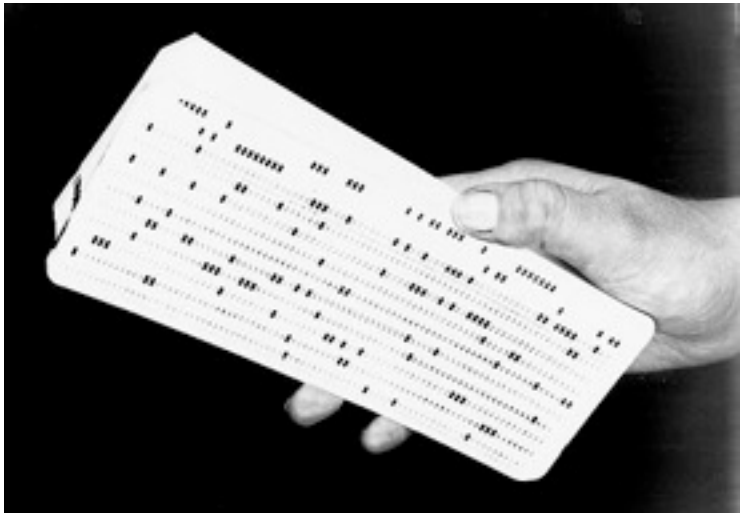
- Prima generazione: 1945-1955
  - sistemi a valvole, no sistemi operativi o linguaggi di programmazione
- Seconda generazione: 1955-1965
  - sistemi a transistori, e primi sistemi batch
- Terza generazione: 1965-1980
  - sistemi basati su circuiti integrati
  - sistemi batch multi-programmati
- Quarta generazione: 1980-oggi
  - personal computers
- Direzioni
  - virtualizzazione & cloud, Internet, architetture multi-core...

PRIMA GENERAZIONE: 1945-1955

- Macchine a valvole, enormi
  - utilizzate per scopi scientifici e militari
  - calcoli (tabelle del seno, logaritmi..)
- Nessuna idea di sistema operativo e di linguaggi di programmazione
  - Interazione diretta uomo-macchina
- Input attraverso delle *plugboard*

SECONDA GENERAZIONE: 1955-1965

- Comparsa dei primi *mainframe*
  - dalle valvole ai *transistor*
  - riduzione delle dimensioni
- Prime periferiche di I/O
  - INPUT: lettori di *schede perforate* (*cards*) o *nastri perforati*
  - OUTPUT: stampanti e perforatori di schede



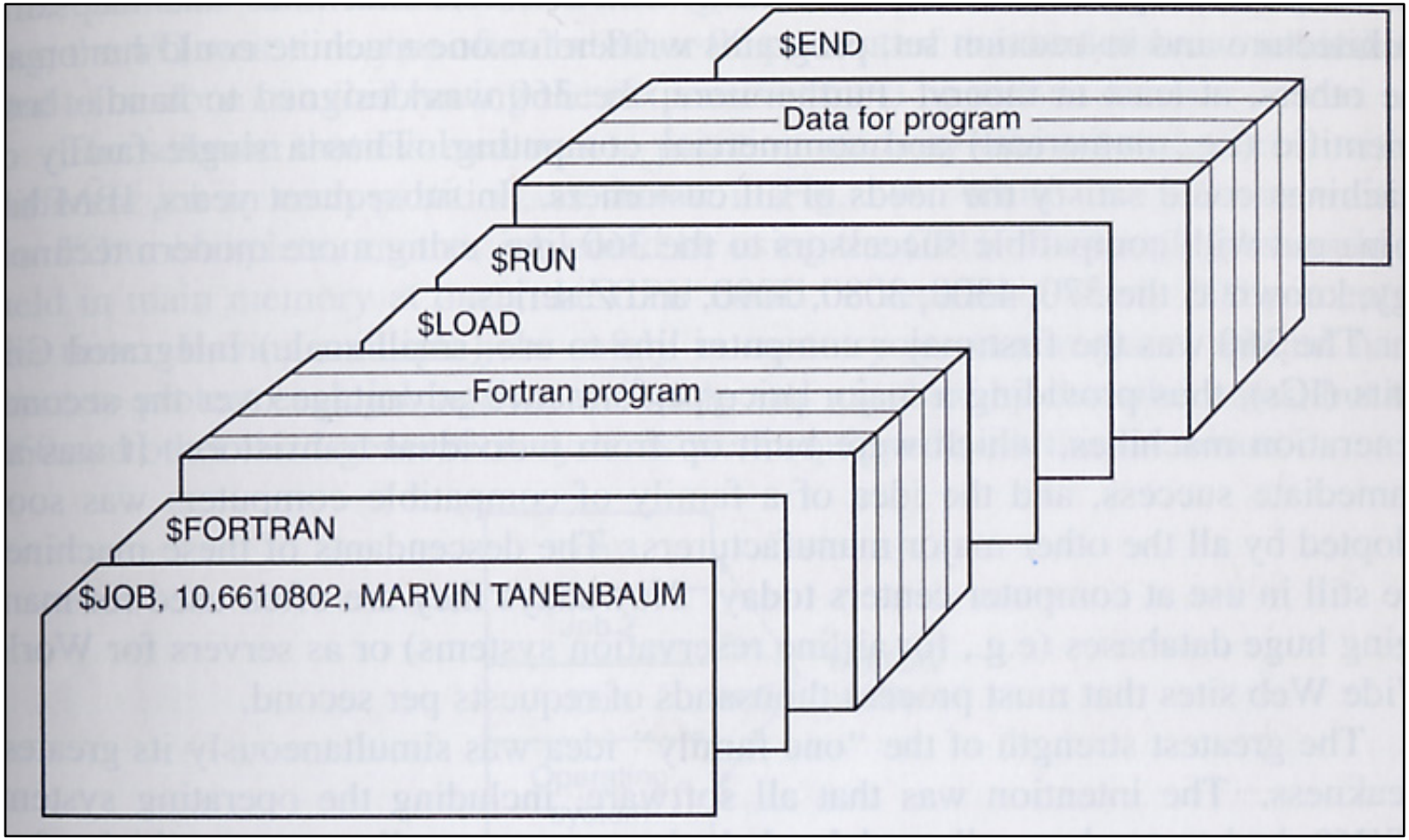
LA NOZIONE DI *JOB*

- Un programma o un insieme di programmi da eseguire venivano chiamati *job*
  - eseguiti una alla volta, in sequenza, senza interazioni con l'utente durante l'esecuzione
  - descritti in termini di dati, programmi e informazioni di controllo scritte su schede perforate
- L'esecuzione molto laboriosa
  - compilazione del programma utilizzando un **compilatore** *(compiler)*
    - INPUT: sorgente su schede perforate, OUTPUT: schede perforate
  - caricamento del programma compilato utilizzando un **caricatore** *(loader)*
  - esecuzione del programma caricato
- Efficienza ridottissima (~1%) dovuta alla necessità di un continuo intervento umano
  - fornire / prelevare schede perforate (input/output)

IL PRIMO SISTEMA OPERATIVO: IL MONITOR

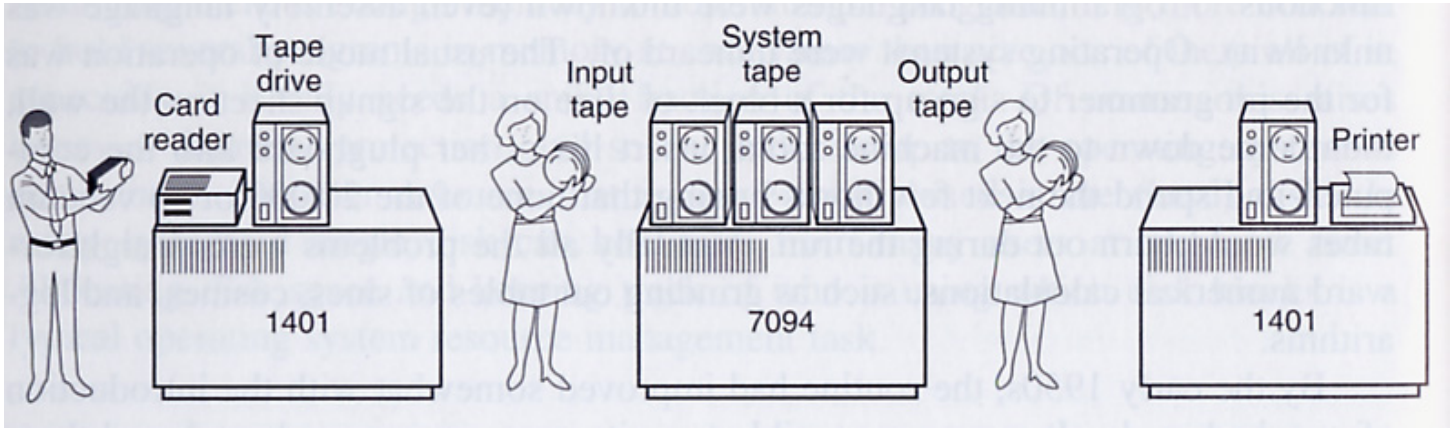
- Obiettivo: “automatizzazione” dell’esecuzione di un job per ridurre intervento umano
  - > introduzione delle prime memorie di massa: i *nastri magnetici*
  - > introduzione di un programma speciale detto **monitor**
    - programma sempre residente in memoria
    - automatizzazione caricamento / esecuzione programmi di sistema
    - introduzione del BIOS (Basic Input/Output System)
      - semplici routine di gestione dei dispositivi di I/O
- Nuova procedura per eseguire un job:
  - gli utenti programmatori fornivano ad opportuni operatori il proprio pacco di schede contenenti i programmi da eseguire
    - programmi + dati
  - fra le schede: programmi per il controllo dell’esecuzione scritte con linguaggio di controllo *Job Control Language* **(JCL)**
  - Il monitor interpretava i programmi in JCL per la compilazione, il caricamento ed esecuzione dei programmi del Job

ESEMPIO DI UN JOB con JCL



PRIMI SISTEMI BATCH

- Ulteriore ottimizzazione dei tempi: processamento per **batch**
  - processamento di interi lotti di pacchi di schede (*batch)* , corrispondenti a job anche di utenti diversi
  - intervento operatore necessario solo alla fine del batch



- Esempi di sistemi: IBM 1401, IBM 7094
- Aspetti negativi
  - “allontanamento” dalla macchina per l’utente e lunga attesa per avere i risultati
  - in particolare nel caso di batch con job di altri utenti lunghi e pesanti

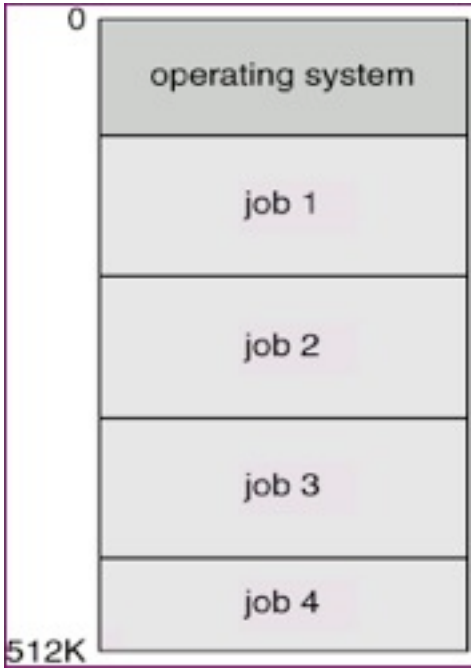
TERZA GENERAZIONE: 1965-1980

- A livello hardware
  - avvento dei primi elaboratori basati su circuiti integrati
  - introduzione delle memorie di massa basate su *dischi*
  - introduzione meccanismo delle interruzioni e del DMA (Direct Memory Access)
    - il trasferimento di informazioni da disco a memoria senza la necessità dell'intervento della CPU
- Primo miglioramento ottenuto sfruttando i dischi
  - tecnica dello *spooling*
    - Job dei batch preventivamente caricati su disco
    - durante l'esecuzione la CPU legge e scrive dati dal disco
    - possibilità di *scheduling* dei Job

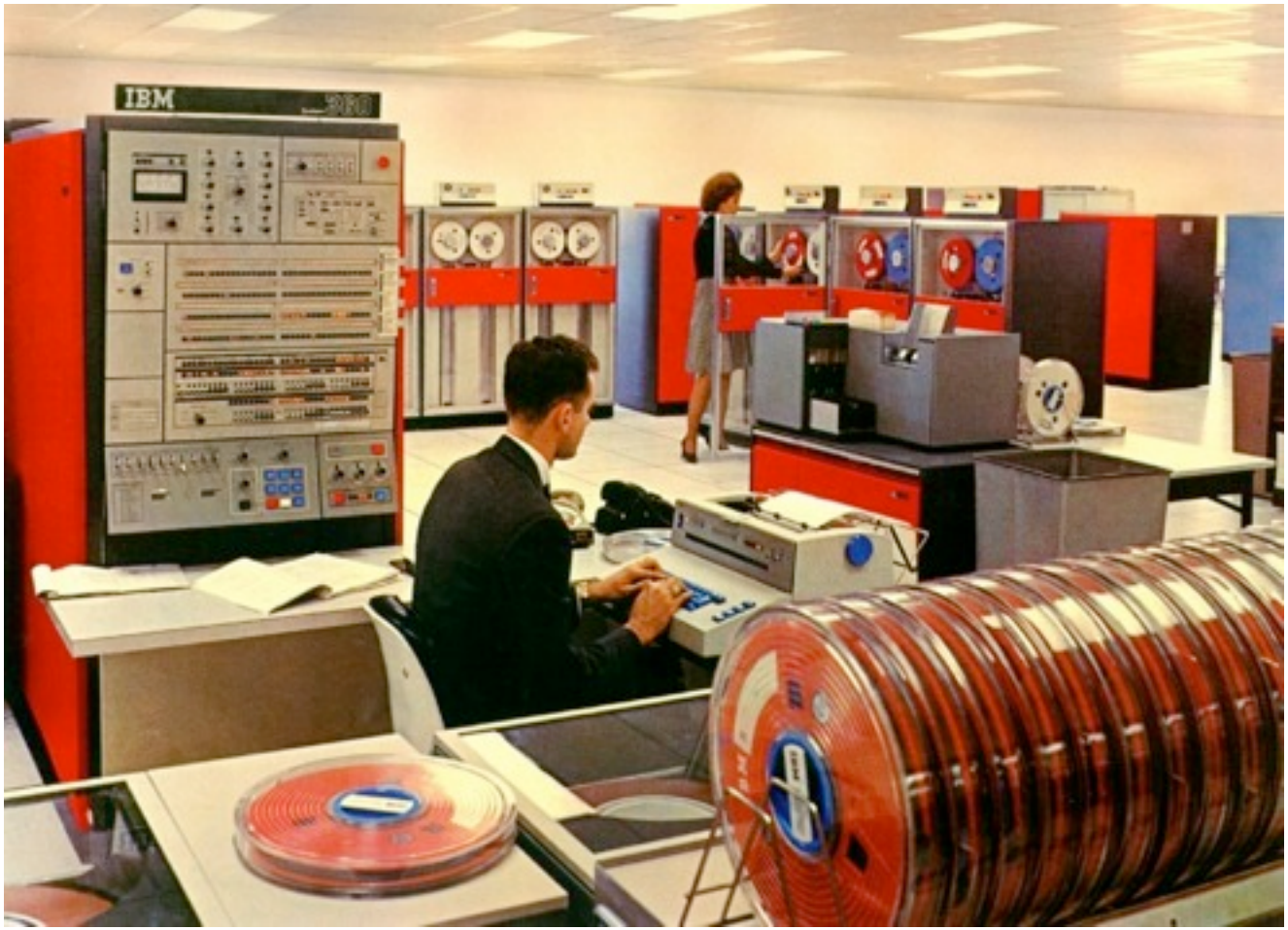
PRIMI DISCHI DA 8 POLLICI...

SISTEMI BATCH MULTIPROGRAMMATI

- Obiettivo: ridurre i tempi di attesa dovuti alle operazione di I/O
  - più lunghe di vari ordini di grandezza rispetto alle operazioni della CPU
  - minimizzare lo spreco della CPU, in attesa
- Introduzione dell tecnica della **multiprogrammazione**
  - caricamento in memoria centrale di *più programmi (*in zone diverse)
  - quando un programma in esecuzione richiede una operazione di I/O la CPU viene passata ad un programma pronto per l'esecuzione
    - *cambio di contesto*
  - in questo modo i programmi in memoria evolvono "contemporaneamente"
    - forma di **multi-tasking**
  - aumento dell'efficienza notevole
- Primo esempio di sistema operativo complesso
  - in esecuzione su mainframes
  - sistemi IBM / 360

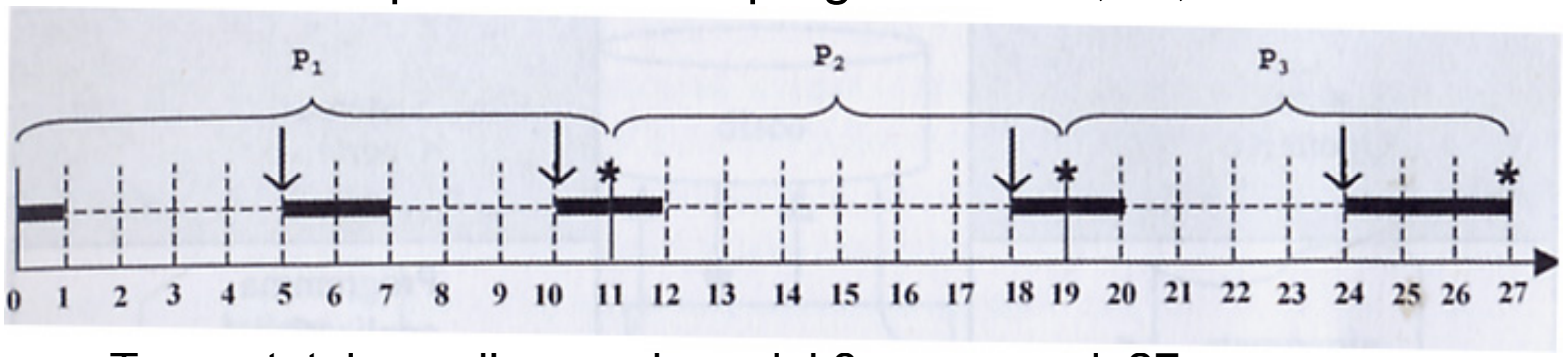


IBM 360: FOTO RICORDO



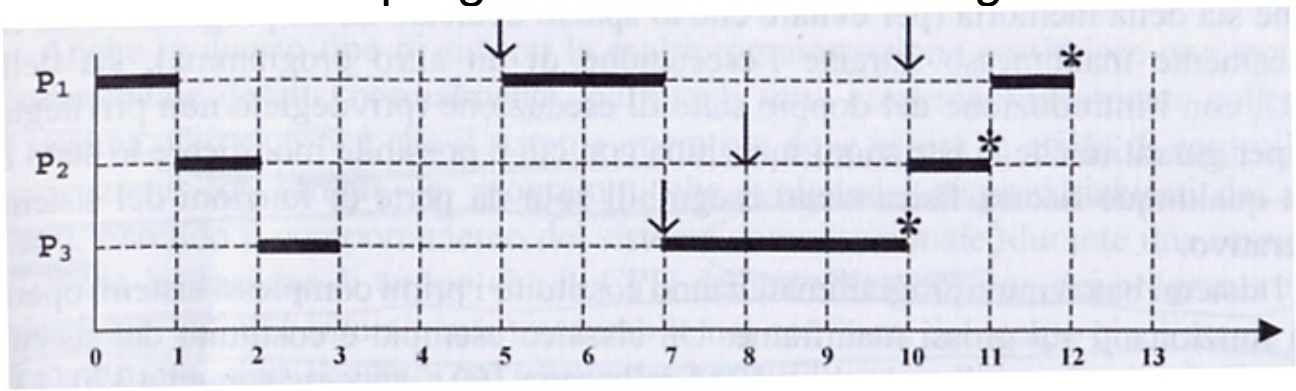
MULTITASKING: UN PRIMO ESEMPIO

- Esecuzione sequenziale di tre programmi: P1,P2,P3



  - Tempo totale per l'esecuzione dei 3 programmi: 27
    - P1 a t={1,7} esegue operazioni di I/O che durano rispettivamente {4,3} unità di tempo, e termina a t=11. P2 inizia a t=11, esegue I/O a t=12, che dura per 6 unità e termina in t=19. P3 inizia a t=19, a t=20 esegue una I/O di 4 e termina in t=27

- Stessa esecuzione di programmi in *multi-tasking*



  - Tempo totale: 12 unità di tempo !

DAL CALCOLO AD APPLICAZIONI INTERATTIVE

- Evoluzione del tipo di applicazioni da eseguire
  - da applicazioni di puro calcolo ad *applicazioni interattive*
    - necessità di interazione con utente durante l'esecuzione del programma
    - esempi: sistemi gestionali
  - impossibilità su sistemi batch
- Nuovi requisiti per i sistemi operativi
  - necessità di gestire interazione dinamica utente / sistema
    - interprete comandi con cui dinamicamente l'utente impartisce comandi al sistema ed esegue applicazioni
  - necessità di gestire più utenti che operano sul medesimo sistema in proprie *sessioni di lavoro*
- Complessità del sistema operativo
  - gestione esecuzione di programmi di utenti diversi
    - nuovi obiettivi: minimizzazione tempo risposta per gli utenti
  - primi passi verso la nozione di *macchina virtuale*
    - ogni utente vede la macchina come se fosse dedicata totalmente a lui

SISTEMI TIME-SHARING

- Supporto multi-utenza e multi-programmazione mediante una **suddivisione del tempo** (*time-slicing*)
  - il sistema operativo assegna la CPU per un certo quanto di tempo (time-slice) ad un programma per essere eseguito
  - al termine del quanto di tempo, se il programma non è terminato il sistema operativo toglie la CPU al programma e la assegna ad un altro
- Aumento notevole della complessità dei sistemi
  - gestione utenti (*account)* multipli, collegati al sistema mediante appositi terminali
  - priotezione del sistema, degli utenti e delle applicazioni in esecuzione
- Primi sistemi time-sharing:
  - CTSS (Compatible Time Sharing System), MIT
  - **MULTICS** (MIT), derivato da CTSS
    - da cui sono derivati VMS (Digital) e sistemi UNIX
    - l'ultimo sistema MULTICS ha chiuso i battenti nel 2000

FOTO RICORDO:
UNA INSTALLAZIONE DI MULTICS

MINICOMPUTERS E NASCITA SISTEMI UNIX

- Alla fine della terza generazione iniziano a diffondersi *mini-computers*
  - meno potenti dei sistemi mainframe, ma molto più ridotti e meno costosi
  - Linea **PDP** della Digital
    - PDP-11 modello di punta
- Sviluppo sistemi operativi per minicomputer: nascita di **UNIX**
  - prima versione sviluppata da Ken Thompson e Ritchie per un PDP-7
  - molto meno complesso di MULTICS
    - il nome UNIX fu scelto appositamente
  - ideato in ambito di ricerca, scritto in un linguaggio di alto livello appositamento ideato allo scopo, **il Linguaggio C**

FOTO RICORDO (1972): THOMPSON E RITCHIE SU UN PDP-11

QUARTA GENERAZIONE: 1980-OGGI

- Dai mainframe e minicomputer ai **personal computer**, inizialmente chiamati microcomputer
  - sviluppo grazie all'avvento dei microprocessori
  - diffusione enorme
- Comparsa sistemi cosiddetti *desktop*
  - pensati per un 'utilizzo 'personale' (home, office) del computer
    - lo scenario di partenza è quindi un singolo utente che interagisce con una o più applicazioni che concorrentemente sono in esecuzione sulla singola macchina.

PRIMI PC E RELATIVI (D)OS

- Anni ‘70
  - 1975, Intel 8080 e Z80, 8 bit
    - sistema operativo CP/M (control program for microcomputers) della Digital
  - Motorola 6800, 8 bit
  - MOS Tech, 6502 8 bit
    - Apple II con proprio sistema operativo, Apple DOS
    - add-on per far girare CP/M, ora venduto da una piccola company chiamata Microsoft (!)
- Primi anni ’80
  - Intel 8088 e 8086, cuore dei sistemi PC IBM con sistema operativo MS-DOS venduto da Microsoft
    - estensione di DOS, acquistato da un’altra azienda e migliorato
  - Home computing:
    - Famiglia Commodore (Vic 20, 64, 128) e ZX Spectrum
- Tutti i S.O. erano a *linea di comando*, no interfaccia grafica

ALCUNE FOTO RICORDO: MSDOS...

The entire staff of Microsoft, 1978

```
MS-DOS Command release 1.00, version 1.19

C:dir
CHKDSK   COM      1754    8-16-83   12:56p
COMMAND  COM      4986    1-18-84    2:01p
CONFIGUR COM     19724    5-03-84   10:33a
DEBUG    COM      6003    8-16-83    1:02p
DISKCOMP COM      5344   11-11-83    1:37p
DISKCOPY COM      5728   12-13-83    1:37p
EDLIN    COM      2313    8-16-83   12:56p
EXE2BIN  EXE      1280    8-16-83    1:01p
FILCOM   COM      8320    8-16-83   12:56p
FORMAT   COM      3856    4-24-84    3:50p
IO       SYS      1713   12-29-83    1:54p
LIB      EXE     32128    8-16-83    1:00p
LINK     EXE     41856    8-16-83    1:00p
MSDOS    SYS      6138    8-16-83   12:49p
PRINT    COM      1740    8-16-83    1:43p
RDCPM    COM      3548   12-13-83    1:28p
SYS      COM       914    1-18-84    2:50p
       17 File(s)
C:_
```

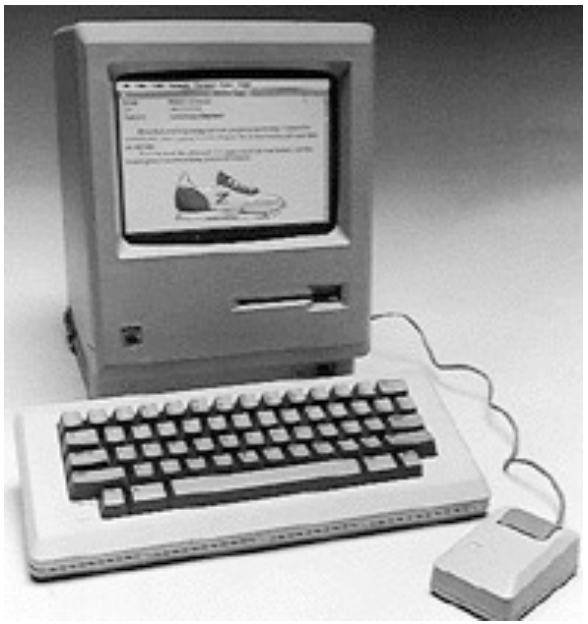
...COMMODORE 64...

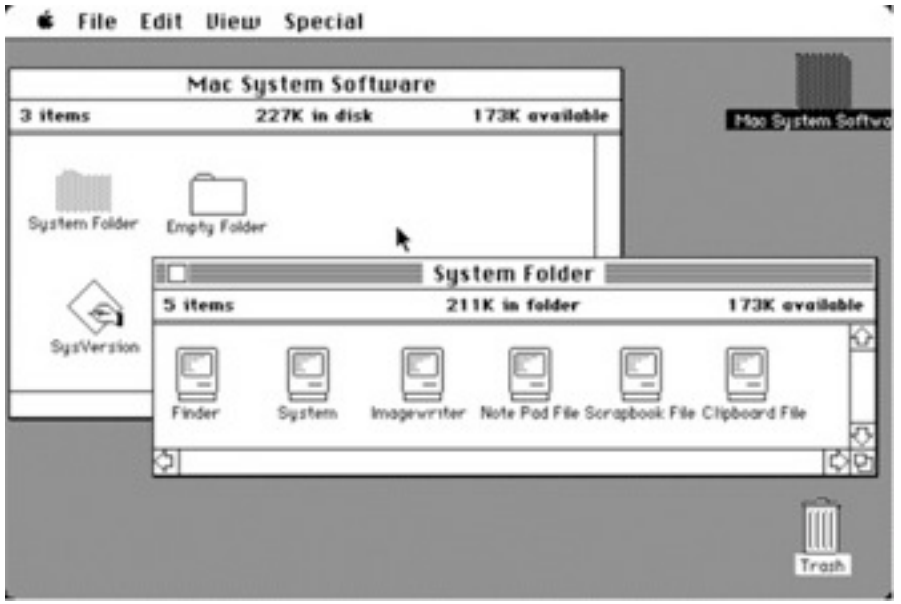
```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
```

PRIMI SISTEMI CON GRAPHICAL USER INTERFACE (GUI)...

- Primi sistemi operativi con *interfaccia grafica* (GUI)per l'interazione con l'utente
  - Apple **Macintosh**, Commodore **Amiga**, **Atari**
    - tutti basati su processori Motorola 68000 a 16 bit e GUI su ROM
- Vista la concorrenza, Microsoft introduce la famiglia **Windows** per sistemi Intel a 16 bit (8086, 80286)
  - inizialmente come pura interfaccia grafica su MS-DOS
  - diviene un vero sistema operativo con Windows NT (32 bit)

...PRIMI MACINTOSH (1984)...





CPU:
an 8 MHz Motorola 68000

Memory:
128 KB DRAM, 64KB ROM

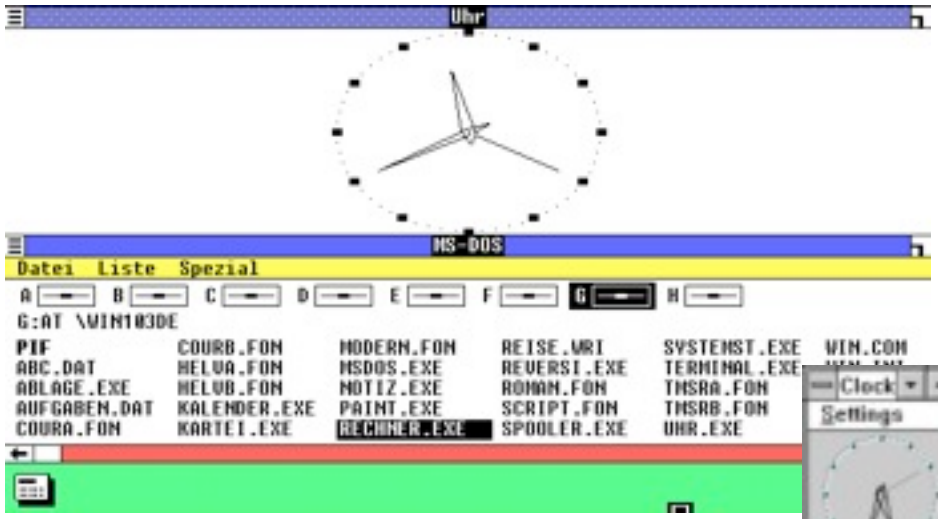
Video:
one-bit black-and-white,
9-inch CRT, resolution of 512x342 pixels

Costo: ~$2,500

FOTO RICORDO: JOBS & WOZNIAK

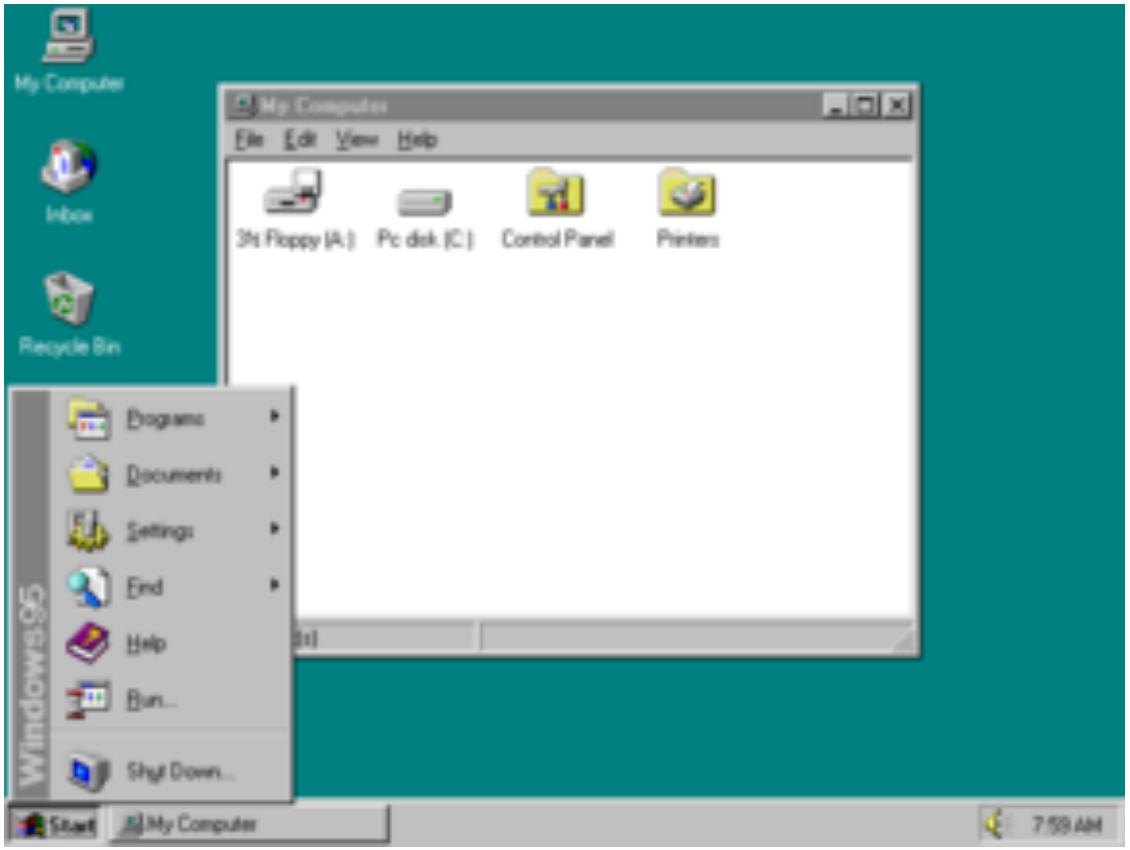
...COMMODORE AMIGA...

...PRIMI SISTEMI WINDOWS (1.0 - 3.11)



...WINDOWS 95 (1995)...

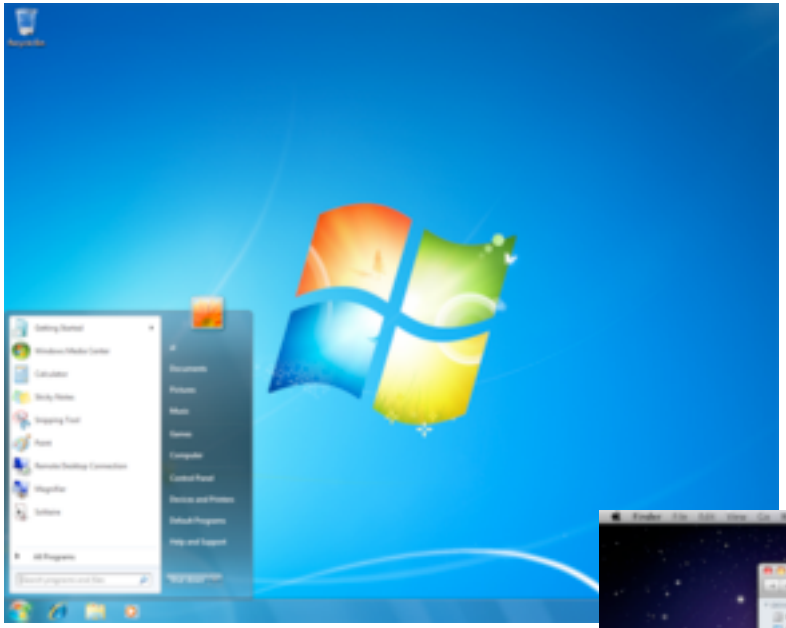
- 32 bit S.O.
- Pre-emptive multitasking
- GUI rinnovata
- Long-File names
- ...

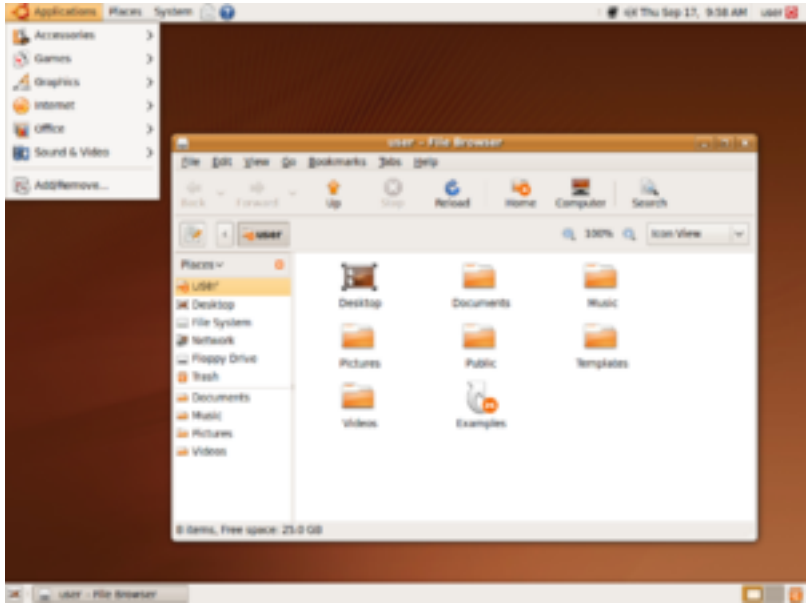


FINE ANNI 90 - VERSO I GIORNI NOSTRI

- Sviluppo hardware
  - architetture a **32 bit** e **64 bit**
  - frequenza processori ~ GH
  - memoria RAM ~ GB
  - memoria secondaria ~ 10^2 GB / TB
  - acceleratori grafici
  - networking pervasivo
  - ...
- Sviluppo delle principali famiglie (ambito desktop)
  - **Microsoft Windows** (da NT)
    - Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8
  - **Linux** e altri sistemi famiglia UNIX
    - Solaris, FreeBSD
  - S.O. Apple: Apple **Mac OS X**
    - core Unix-based + Core Mach 3.0 (S.O. di ricerca sviluppato al CMU)

WINDOWS, LINUX, Mac OS X





MICROSOFT WINDOWS - CENNI STORICI

- A metà degli anni 80 Microsoft e IBM sviluppano cooperativamente OS/2
  - in assembly language, per macchine Intel 80286
- Nel 1988 Microsoft abbandona il progetto e parte con uno nuovo: **NT** (New Technology) operating system, in grado di supportare sia OS/2, sia standard API POSIX (UNIX)
  - portato avanti da Dave Cutler, ex-ingegnere dei sistemi VAX / VMS
- NT passa dal supporto ad OS/2 al supporto delle Win32 API, nate con Windows 3.0: nasce **Windows NT 3.1**
- Esce **Windows NT 4** (versione interna 4.0), con interfaccia grafica di Windows 95
- Il tentativo di introdurre in NT caratteristiche proprie della famiglia Win 98 / Win 95 (es: supporto videogiochi) porta la famiglia NT ad essere molto instabile: cambio di strategia, esce **Windows 2000 (**versione interna 5.0)
  - obiettivo di recuperare stabilità, ma con limitata compatibilità verso win 95 / 98

MICROSOFT WINDOWS - CENNI STORICI

- Nell'ottobre del 2001 esce **Windows XP** (versione interna 5.1)
  - obiettivo: miglioramento stabilità e al contempo capacità di eseguire in modo efficiente applicazioni Win 95 -like
  - primo S.O. di Microsoft ad avere una versione anche a 64bit
  - **Windows Server 2003** (versione interna 5.2)
- Nel gennaio del 2007 esce **Windows Vista** (versione interna: 6.0)
  - nuove funzionalità
  - obiettivo: miglioramento aspetti sicurezza
  - parziale re-ingegnerizzazione sull'infrastruttura .NET
  - interfaccia grafica Windows Aero
  - **Windows Server 2008** (Marzo 2008)
- Estate 2009 / Fine 2009: **Microsoft Windows 7**
  - miglioramento delle funzionalità introdotte con Windows Vista
- **Windows 8** (2012)
  - tra i cambiamenti sostanziali: interfaccia Metro
  - convergenza con il mobile

UNIX - CENNI STORICI

- Prima versione sviluppata da **Ken Thompson** nel 1969, nei laboratori del gruppo di ricerca Bell, su PDP-7
  - coadiuvato da Ritchie, proveniente dal progetto MULTICS
- La terza versione fu scritta nel **linguaggio C**, inventato appositamente (Kerningham & Ritchie)
- Prima versione disponibile fuori dai Bell labs: Versione 6, 1976
- Nel 1978 viene distribuita la **versione 7**, da cui derivano le versioni moderne
  - Nasce lo USG, Unix Support Group, per gestire quello che non era più solo un oggetto di ricerca, ma un sistema largamente usato
  - AT/T supporta USG
- Viene portato velocemente su numerosi altri sistemi e diffuso fra università e imprese.
  - Es: AT&T con gruppo di ricerca nell'Università di Berkley
- A Berkley lavorano Bill Joy e Ozalp Babaouglu, che aggiungono supporto a memoria virtuale: nasce UNIX 3BSD, famiglia **BSD**

UNIX - CENNI STORICI

- DARPA (militari US) finanzia Berkley e nasce la versione 4.2BSD, con supporto per la rete e protocolli come il TCP/IP, alla base di ARPANET (in futuro Internet). L'ultima versione BSD (4.4BSD) è nel 1993
- UNIX è stato poi alla base per numerosi altri S.O. scritti da aziende importanti: AIX (UNIX di IBM), XENIX (UNIX di Microsoft), OSF/1..
- **GNU** Project (Stallman, 1984 / 1985)
  - obiettivo di creare un sistema UNIX-compatibile a partire da unicamente free-software
  - nascita della Free Software Foundation e della licenza GNU
- Esplosione di versioni ed estensioni...
  - Solaris - S.O. della Sun - deriva da UNIX
  - **FreeBSD** deriva da 4.3BSDLite (dal 1994 fino alla vers. 4.2, 2001), è una versione largamente utilizzata per sperimentazioni e ricerche
  - il sistema operativo OpenStep - ideato da Steve Jobs - che integra sistemi UNIX e sistemi Apple Mac..

UNIX E STANDARD POSIX

- Date le varie versioni e diramazioni, la compatibilità è stato da subito un problema fondamentale.
- Allo scopo è stato definito lo standard **POSIX**,
  - un insieme di specifiche che definiscono le API di UNIX (in C)
  - se un sistema operativo è POSIX compliant, allora è possibile compilare su quel sistema qualsiasi programma che rispetti le specifiche POSIX

LINUX

- Nasce come progetto universitario di uno studente finlandese, Linus Torvald, nel 1991
  - scrive un kernel per macchine Intel 386, compatibile UNIX
  - corso di Sistemi Operativi alla Vrije Univ. tenuto dal Prof. Tanenbaum
- Da subito, Torvald rende il codice disponibile liberamente in rete, per chiunque volesse commentare e contribuirne allo sviluppo
- Da allora il sistema si è sviluppato tanto da diventare il principale S.O. con Windows per sistemi desktop, server e mobile
  - es: piattaforma Android per smartphone è basata su Linux Kernel
- Contribuiscono allo sviluppo di Linux molte persone, fra mondo accademico, industriale e di ricerca
  - filosofia open-source

(LINUS TORVALD - PRIMA E DOPO)

LINUX KERNEL

- Sorgenti del kernel nelle varie versioni disponibili in http://www.kernel.org/
- E' scritto quasi interamente in C
  - formidabile portabilità,
  - dimensioni
    - ~10K loc nel 1991 (v.0.01)
    - più di 6M loc con la v.2.6
- Recenti sviluppi / versioni
  - dal 2003-2011: versione **2.6**
    - ultima versione: 2.6.39
  - luglio 2011: introdotta la versione **3.0**
    - nessun cambiamento sostanziale rispetto alla 2.6.39
    - celebrativo dei 20 anni

Mac OS X

- E' il sistema operativo per computer della famiglia Apple
  - successore di Mac OS 8 / 9
  - cambiamento radicale, nuova generazione che deriva da OpenStep sviluppato dal team di Steve Jobs su macchine NeXT
- Reingegnerizzazione completa del sistema operativo
  - multitasking preemptive (vs cooperativo, Mac OS 9)
  - compatibilità con sistemi UNIX
  - **Darwin** è il nome del kernel, open-source
    - basato su kernel Mach (Carnegie Mellon University)
    - integra parti di BSD
- Ultime versioni
  - Mac OS X Leopard (10.5, anno 2006)
  - Mac OS X Snow Leopard (10.6, anno 2008)
  - OS X Lion (2011), Mountain Lion (2012)

UNIX & DERIVATI - FOTO DI FAMIGLIA

D. Ritchie

B. Kerninghan

R. Stallman

Linus Torvald

K. Thompson

Bill Joy

Tanenbaum

Steve Jobs

SISTEMI OPERATIVI OPEN SOURCE

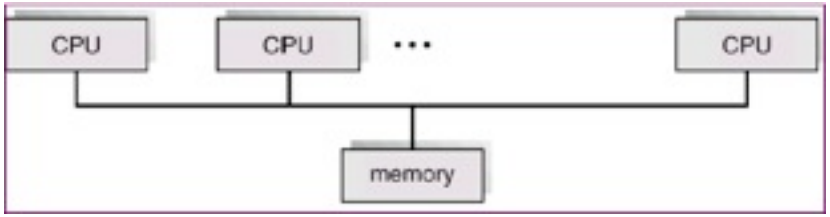
- Linux
  - http://www.kernel.org/pub/linux/kernel
  - distribuzioni: RedHat, SUSE, Fedora, Debian, Slackware, Ubuntu
- BSD Unix
  - distribuzioni: FreeBSD, OpenBSD, DragonflyBSD
- Darwin (Mac OS)
  - http://www.opensource.apple.com/darwinsource
- Solaris
  - sistema operativo Unix-based di Sun
  - http://www.opensolaris.org
- + tutti quelli di ricerca (centinaia...)

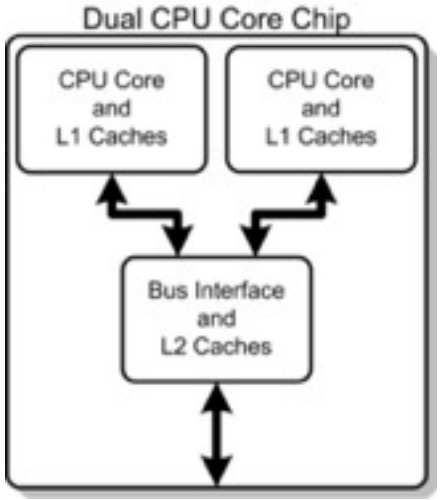
SVILUPPI RECENTI / STATO DELL'ARTE

- Sistemi operativi per architetture a **64 bit**
- Sistemi operativi per architetture **multi-processore**
  - architetture **multi-core** in particolare
- Tecnologie di **virtualizzazione**
  - macchine virtuali, infrastrutture di virtualizzazione
- Sistemi operativi **di rete** e **distribuiti**
  - "the network is the computer"
  - *web-based* operating systems
  - **cloud computing**

ARCHITETTURE MULTICORE

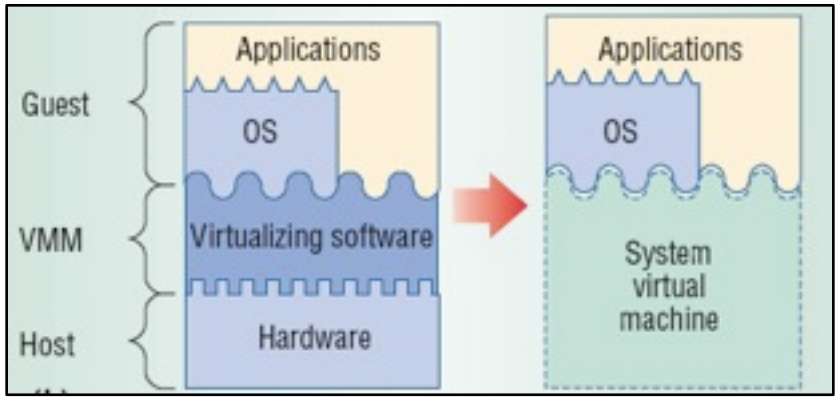
- Introduzione di architetture *multiprocessore*
  - più CPU che condividono la medesima memoria centrale
    - es: super-computer
  - recentemente: architetture **multicore**
    - es: Intel Core Duo 2
  - parallelismo ‘reale’
    - programmi in esecuzione simultanea su CPU diverse
- Aumento di complessità per il sistema operativo
  - gestione di più CPU
  - gestione esecuzione di funzioni del sistema operativo da parte di programmi in esecuzione su CPU diverse
- Previsione nei prossimi dieci anni: da decine a centinaia di core
  - prototipo con 80 core di Intel (2007)





VIRTUALIZZAZIONE

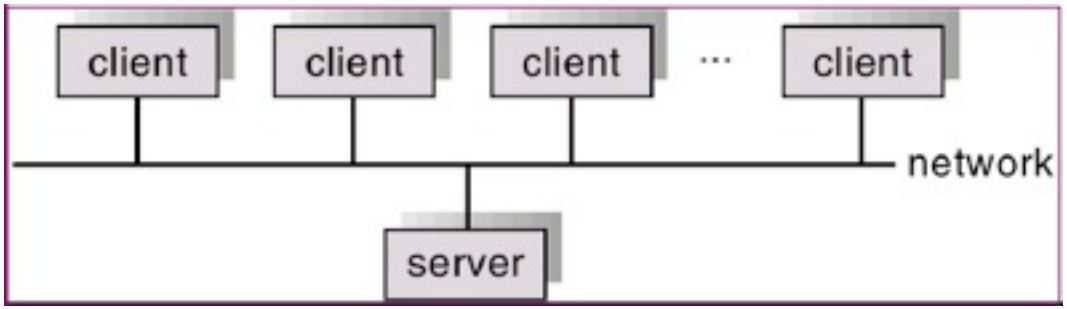
- Macchina Virtuali (VM)
  - implementazione software di una macchina (computer) che esegue programmi come una macchina reale (fisica)
  - macchina virtuale di sistema (*system o hardaware virtual machine)*
    - macchina virtuale che supporta l'esecuzione di un completo sistema operativo, emulando l'hardware di un intero sistema di elaborazione



- Sulla medesima macchina fisica possono essere messe in esecuzione più macchine virtuali, eventualmente con S.O. eterogenei
  - isolamento, portabilità

SISTEMI OPERATIVI DI RETE

- Introduzione e diffusione pervasiva delle *reti di calcolatori*
  - sistemi di elaborazione che non condividono memoria e che possono *comunicare* mediante opportune infrastrutture e protocolli di rete
    - es. Infrastruttura Internet (IP), protocolli TCP/IP, ...
- Sistemi operativi di rete
  - ogni nodo della rete ha il proprio sistema operativo
    - eterogeneità
  - il sistema operativo fornisce funzionalità per la comunicazione via interfacce di rete e supporto per protocollli di comunicazione
  - le applicazioni sfruttano tali funzionalità per interagire e scambiare informazioni
  - Architetture *client-server* e *peer-to-peer*.
    - Esempi:
      - Browser + WebServer
      - File Servers



WEB-BASED OPERATING SYSTEMS

- Ambienti desktop pensati per **cloud computing**
  - sistemi distribuiti basati su web
- Caratteristiche
  - accesso alle applicazioni e gestione file via Web Browser
  - esecuzione applicazioni e programmi come normali S.O.
    - esecuzione parzialmente o totalmente remota, nel cloud
    - “web persistence”
- Esempi
  - progetto WebOS (UC Berkeley) e WOS
  - eyeos
    - scritto in PHP, Javascript, XML
    - piattaforma per scrivere applicazioni Web
    - contiene applicazioni Word processing, foglio elettronico,...

NEXT GUI?

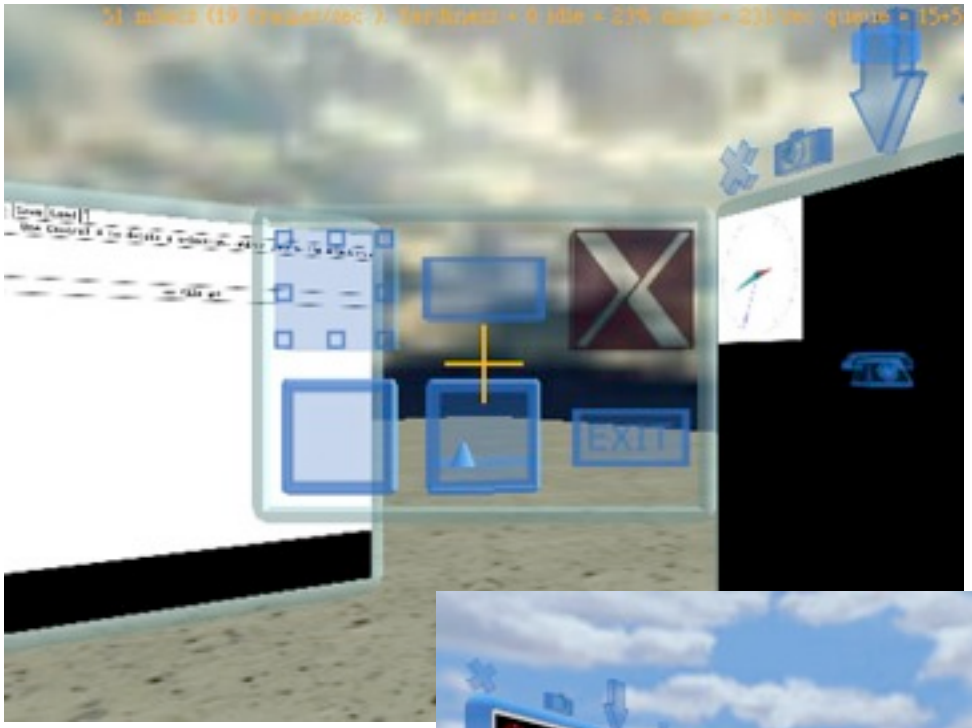
- La shell grafica è oggi una componente molto importante dei sistemi operativi moderni
- Tuttavia le GUI attuali sono pressoché basate su metafore sviluppate più di 20 anni fa
  - pensate principalmente per sistemi standalone
  - scrivania, cartella, cestino....
- Nella ricerca si studiano GUI innovative, basate su metafore nuove
  - GUI 3D, in cui si rappresenta l'ambiente in cui l'utente interagisce con le applicazioni come spazio virtuale tridimensionale
    - es: Progetto Looking Glass di Sun Microsystem
  - Ambienti virtuali tridimensionali per S.O. fortemente orientati alla rete
    - Progetto Croquet

PROGETTO LOOKING GLASS (2003)

http://www.sun.com/software/looking_glass/

PROGETTO CROQUET

http://www.opencroquet.org



SISTEMI SPECIAL-PURPOSE

- Sistemi operativi visti fino ad ora sono detti **general-purpose**
  - esecuzione di applicazioni di tipo diverso
- Esistono sistemi **special-purpose**
  - sistemi operativi sviluppati per *contesti specifici*
- Esempi principali
  - sistemi per **dispositivi mobili**
    - smart phone, tablet
  - sistemi operativi **real-time**
    - sistemi **embedded**

SISTEMI HANDHELD

- In questa categoria ricadono i sistemi operativi sviluppati per device definiti **personal digital assistant (PDA),** come palmari, pocket-PC e cellulari...
  - accezione più diffusa correntemente (2011): **smartphone** e **tablet**
- Aspetti centrali
  - interazione con l'utente
    - design opportune **UI** (User Interface)
    - impatto anche sui sistemi desktop
  - uso ottimizzato risorse (CPU, memoria)
    - aspetto meno stringente oggi rispetto ad anni fa, in virtù del progresso tecnologico in ambito mobile
  - gestione sensori
  - uso massiccio della rete e delle tecnologie di comunicazione
    - web, cloud

S.O. PER SISTEMI “MOBILE”

- Per i sistemi “mobile” (handheld)
  - smart phones / cellulari, tablets
- Sistemi operativi e piattaforme
  - alcuni sono nati appositamente per il mobile
    - **PalmOS** e **Symbian** (non più usati)
  - altri derivano dal *porting* di sistemi operativi (o un loro sottoinsieme) tradizionali.
    - **iOS**
      - famiglia Mac OS X
    - **Windows Phone 7/8**
      - famiglia Microsoft
    - Google **Android**
      - basata su Linux Kernel e macchina virtuale Dalvik per esecuzione programmi Java-like

PalmOS & MS Pocket PC

PalmOS

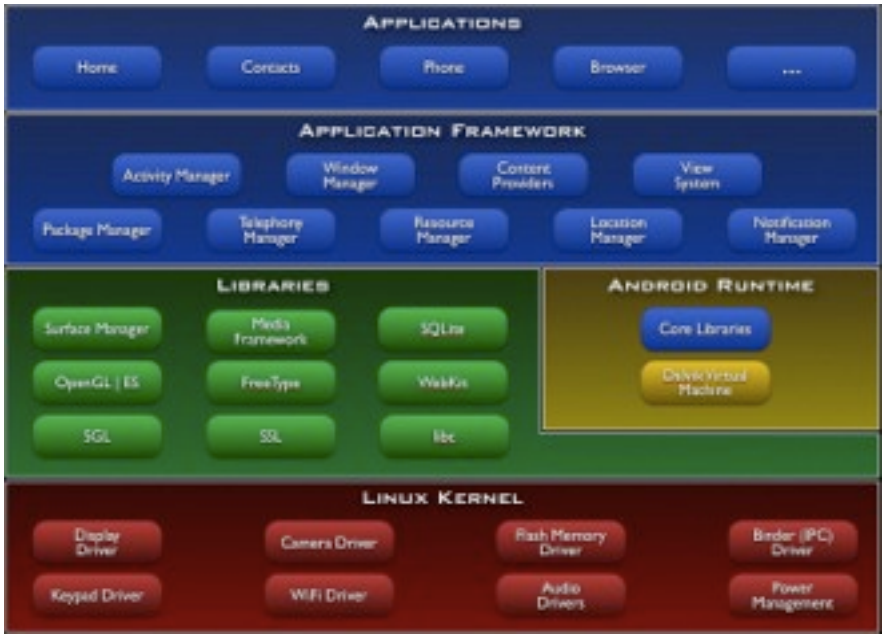
Microsoft PocketPC

WINDOWS PHONE e iPHONE/iOS

GOOGLE ANDROID

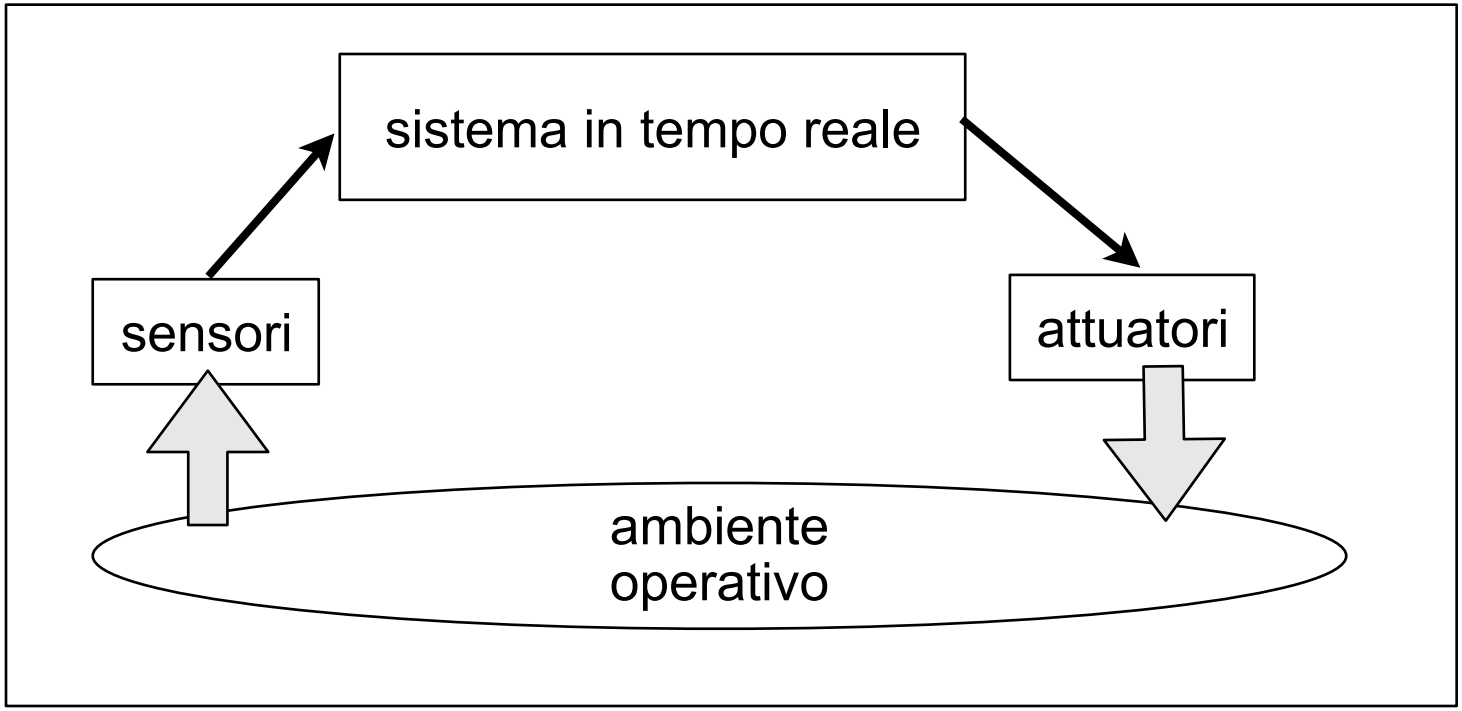
http://code.google.com/android/

- Piattaforma aperta per sistemi mobile
  - sviluppata da Open Handset Alliance
    - gruppo che include più di 30 aziende del settore IT e mobile
- Basato su kernel Linux e macchina virtuale Dalvik per sviluppo ed esecuzione di applicazioni scritte in linguaggio Java-like



SISTEMI REAL-TIME / EMBEDDED

- Sistemi Operativi specializzati per sistemi di elaborazione applicati a problemi di *controllo*
  - controllo del corretto funzionamento di un sistema *fisico*
  - **sistemi embedded**



SISTEMI SOFT E HARD REAL-TIME

- Presenza di vincoli temporali nell'esecuzione dei vari programmi (*task*)
  - sistemi **hard real-time**
    - nessuna violazione dei vincoli temporali è permessa
    - applicazioni di controllo critiche
      - es: Macchine biomedicali
  - sistemi ***soft real-time***
    - definizione di priorità
    - i sistemi operativi moderni generalmente soft-real time
  - algoritmi specializzati per lo scheduling dei task

- Esempi
  - VxWorks
  - RT-Linux
  - QNX

NOTE CONCLUSIVE:
S.O. E LINGUAGGI DI PROGRAMMAZIONE

- C'è una stretta relazione fra l'architettura dei sistemi - sistemi operativi in particolare - e i linguaggi / modelli computazionali utilizzati per la loro programmazione
- Dunque al di là della programmazione, i modelli / paradigmi dei linguaggi hanno un notevole impatto sull'architettura dei sisitemi operativi
- Il linguaggio utilizzato storicamente - e ancora oggi in molti casi - per lo sviluppo di sistemi operativi è il **linguaggio C**
  - linguaggio procedurale, imperativo, progenitore del C++ ed in parte di Java
  - i sistemi Unix-like - Linux, MINIX - sono implementati in C

```
#include <stdio.h>

main(){
  printf("Hello world!\n");
}
```

SISTEMI OPERATIVI OBJECT-ORIENTED, BASATI SU MACCHINA VIRTUALE

- Lo sviluppo di linguaggi OO ha influenzato notevolmente anche l'evoluzione dei sistemi operativi più diffusi, ispirando progettazioni più affini al modello OO.
  - Ad esempio Solaris, sistema operativo UNIX-like di Sun Microsystem.
  - Mac OS X e Windows XP hanno parti progettate in termini OO e sviluppate in C++ / Objective C
- Nella ricerca esistono sistemi operativi la chi architettura è stata *totalmente* progettata in termini OO e quindi realizzati in linguaggi OO.
  - esempi sono: Choices (C++), Spring (C++), ApertOS (CLOS)...
  - JNode e JX (linguaggio Java)
  - **Microsoft Singularity** (Sing#, estensione di C#)
- Questi sistemi esibiscono caratteristiche più avanzate dei sistemi operativi mainstream, in particolare in termini di configurabilità, sicurezza ed estendibilità dinamica.