THE O.S. INTERFACE

È la parte esterna del SO, attraverso la quale gli utenti richiedono l'esecuzione dei programmi (siano essi programmi "di utilità" dello stesso SO o programmi "applicativi") o accedono ai servizi del kernel.

L'interfaccia può essere di due tipi

.... a caratteri

L'interfaccia a caratteri o **shell** o **command interpreter** è un vero e proprio linguaggio di programmazione che permette all'utente di controllare la esecuzione di comandi, sia in modo interattivo che in modo "batch" (mediante "script" di shell). Tipici SO con interfaccia a caratteri sono il DOS, UNIX, LINUX, ecc.

.... grafica o iconica

L'interfaccia grafica è un sistema d'interazione uomo-macchina basato su simboli (icone) che rappresentano le risorse su cui è possibile operare. Tipici SO con interfaccia iconica sono WINDOWS (in tutte le sue versioni), MAC OS, UNIX vers. OSF Motif, LINUX vers. OSF Motif, ecc.

THE CHARACTER INTERFACE (*)

Una shell si basa sull'esecuzione di tre tipi di **comandi**:

1. **oggetti eseguibili (o comandi esterni)**
   Sono file contenenti programmi in formato eseguibile
   Esempio: sort
2. **comandi built-in (o comandi interni)**
   Sono comandi che realizzano funzioni semplici, eseguite direttamente dalla shell
   Esempio: cd
3. **script (o comandi batch)**
   sono "programmi" in linguaggio di shell

Quando si digita un comando, la shell verifica se si tratta di un comando built-in. In caso positivo, lo esegue. Altrimenti la shell crea (fork) un nuovo processo che esegua il comando.

Oltre all'esecuzione di comandi, la shell consente altre funzioni tipiche:
- **ridirezione** dell'input e dell'output
- **"pipelines"** di comandi
- **filtri**

(*) si fa riferimento ad una tipica interfaccia a caratteri di un SO UNIX-like

THE GRAPHICAL INTERFACE (*)

Un'interfaccia grafica si basa sulla manipolazione di "**risorse**"

Una risorsa è un **oggetto** che fornisce o elabora informazioni

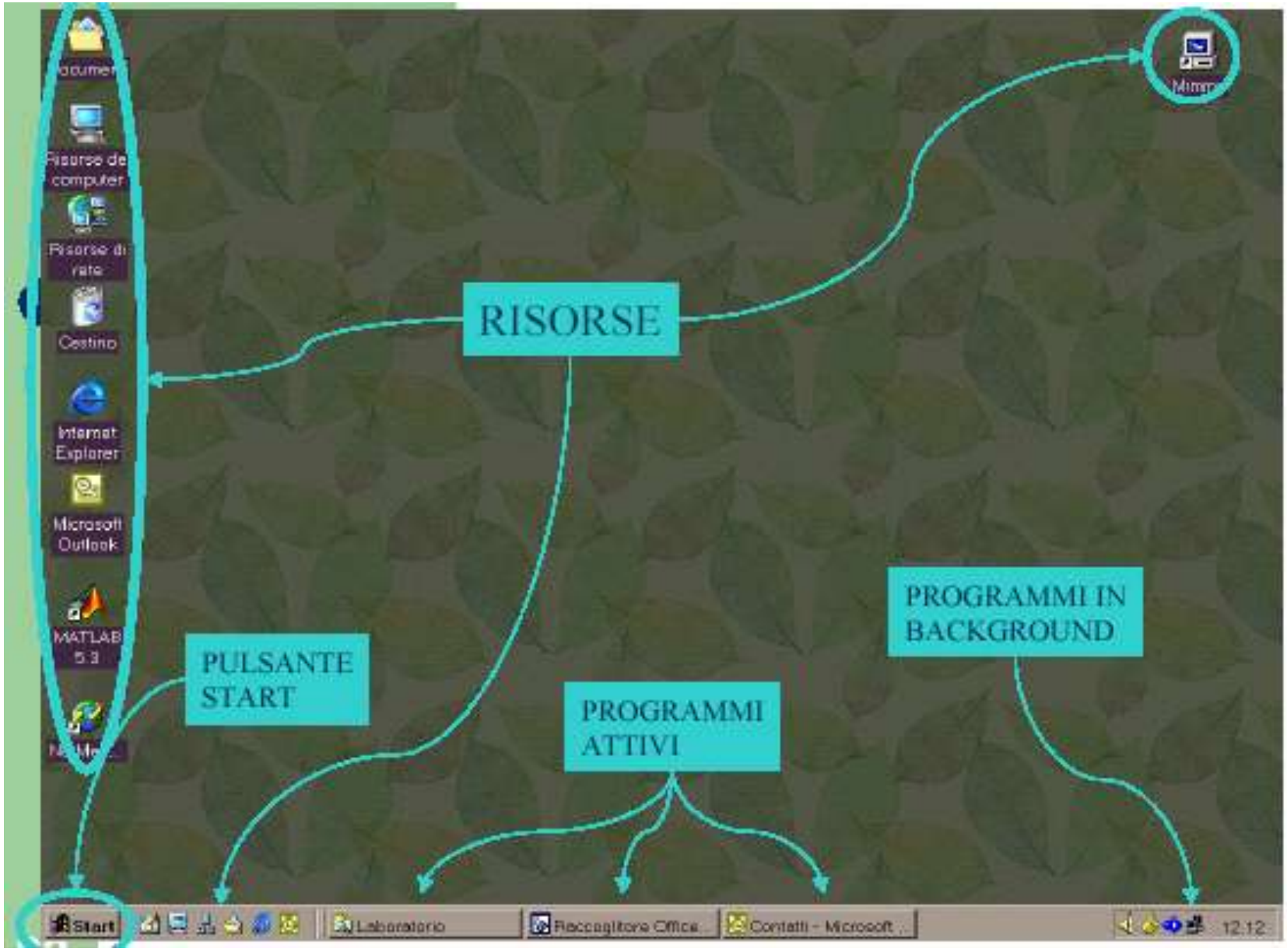
In Windows le risorse sono rappresentate da icone ed hanno un nome

Esempi di risorse
- I dischi e le stampanti
- Tutti i dispositivi esterni
- I file e le cartelle
- I programmi
- I componenti di Windows

Il desktop
È una risorsa speciale di Windows, realizzata come cartella in cui sono contenuti documenti (oggetti) di interesse generale ed organizzata come "scrivania" virtuale, su cui sono disposte le icone degli oggetti.

(*) si fa riferimento ad una tipica interfaccia grafica di un SO WINDOWS-like

THE WINDOWS DESKTOP



THE ACTIVITIES MANAGED BY AN O.S.

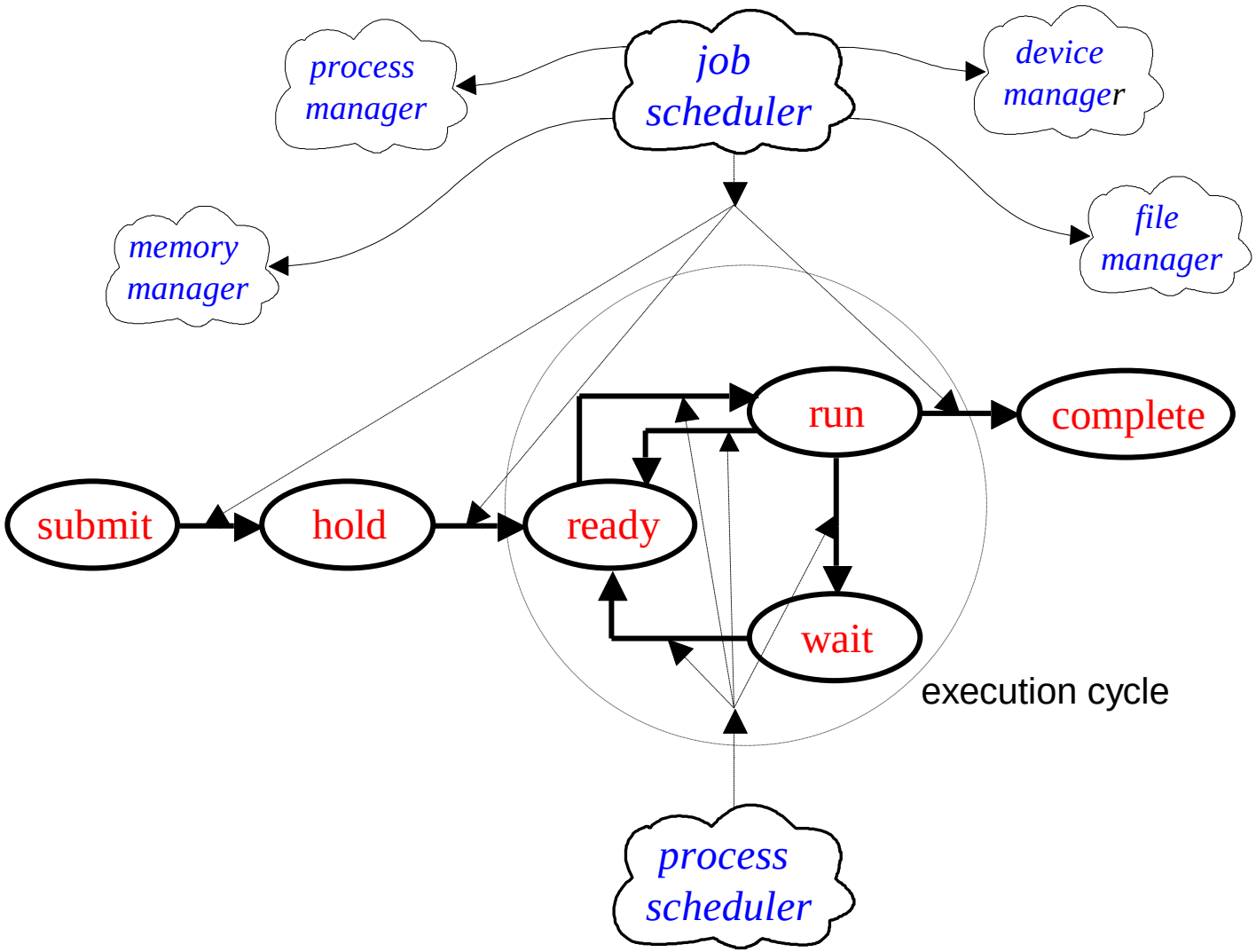
Un ***job*** è costituito da uno o più job-step (programmi) da eseguire in sequenza.

Un ***job-step*** (o programma) è costituito da uno o più task (processi cooperanti) che operano «contemporaneamente» contendendo fra loro, oltre che con gli altri processi costituenti gli altri programmi (processi concorrenti), nell'utilizzo delle risorse (CPU, memoria centrale, memoria di massa, dispositivi, file) presenti nel sistema di elaborazione.

Il sistema operativo stesso è costituito da processi che cooperano per rendere possibile l'esecuzione dei programmi.

Un ***task*** è una attività elementare ed indipendente, con associate le risorse (memoria, file, dispositivi) richieste per la sua esecuzione. Ad ogni task è associato un Task Control Block (TCB) o Process Control Block (PCB).

THE EXECUTION STEPS OF A PROGRAM



PROGRAM EXECUTION PROCEEDING

Il sistema operativo produce l'avanzamento dell'esecuzione dei programmi.

Dapprima le richieste sottoposte vengono raggruppate, costituendo la coda relativa allo **stato di submit**.

Dopo l'analisi delle risorse necessarie a soddisfare le richieste, queste vengono ordinate in relazione ai criteri di priorità (macroscheduling) del ***Job Scheduler***, costituendo la coda relativa allo **stato di hold**.

A partire dalla prima richiesta di esecuzione presente nella coda di hold, il Job Scheduler, tramite i gestori delle risorse, verifica la disponibilità delle risorse necessarie e, dopo averne avuto conferma, attiva il ***Process Scheduler***. Questo trasforma la richiesta di esecuzione in processo e costruisce un blocco di informazioni (*Process* o *Task Control Block*), che, progressivamente aggiornate, gli permetteranno di gestire il programma nel *ciclo di esecuzione*. Inizialmente il task viene immesso nella coda dello **stato di ready**. Tale coda è organizzata con criteri di priorità (microscheduling) con cui il Process Scheduler ordina i vari task che abbisognano della sola CPU per essere eseguiti.

Non appena la CPU si libera (o il rispetto delle priorità lo richiede), viene operato il cambio del contesto operativo (*context switching*) della CPU e un processo passa nello **stato di run** (ed uno passa in ready).

L'attesa di specifici eventi può costringere il S.O. a far transitare il processo nella coda corrispondente allo **stato di wait**.

Dopo aver ciclato tra gli stati del ciclo di esecuzione, il processo arriva a conclusione e transita nella coda dello **stato di complete**.

O.S. DESIGN AND IMPLEMENTATION

- ✓ Traditionally written in assembly language, operating systems can now be written in higher-level languages.
- ✓ Code written in a high-level language:
  - can be written faster.
  - is more compact.
  - is easier to understand and debug.
- ✓ An operating system is far easier to *port* (move to some other hardware) if it is written in a high-level language.

O.S. INSTALLATION AND USE (SYSTEM GENERATION)

- ✓ Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site.
- ✓ SYSGEN program obtains information concerning the specific configuration of the hardware system.
- ✓ *Booting* – starting a computer by loading the kernel.
- ✓ *Bootstrap program* – code stored in ROM that is able to locate the kernel, load it into memory, initialize all system functions (CPU registers, memory controllers, etc.,…) and start executing the first process, i.e. the so called ***initiator***, then waits for events (signaled from ***interrupts***) or user's requests.

PRIMI SISTEMI – (INIZIO ANNI '50): MAINFRAME SYSTEMS

- ***Struttura***
  - *Grandi calcolatori* funzionanti solo da console
  - *Sistemi single user*; il programmatore era anche utente e operatore
  - *schede di collegamento*, poi I/O su nastro perforato o schede perforate
- ***Primi Software***
  - *Assemblatori, compilatori, linker, loader*
  - Librerie di *subroutine comuni*
  - *Driver* di dispositivi
- ***Uso inefficiente di risorse assai costose***
  - Bassa utilizzazione della CPU
  - Molto tempo impiegato nel setup dei programmi

II GENERATION: TRANSISTORS AND BATCH SYSTEMS: MAINFRAME SYSTEMS

…. Easy Batch Systems

- User ≠ Operator
- Reduce setup time by *batching* (grouping) similar jobs
- Automatic job sequencing – automatically transfers control from one job to another. First rudimentary operating system.
- Resident monitor
  - initial control in monitor
  - control transfers to job
  - when job completes control transfers back to monitor

II GENERATION: TRANSISTORS AND BATCH SYSTEMS: MAINFRAME SYSTEMS

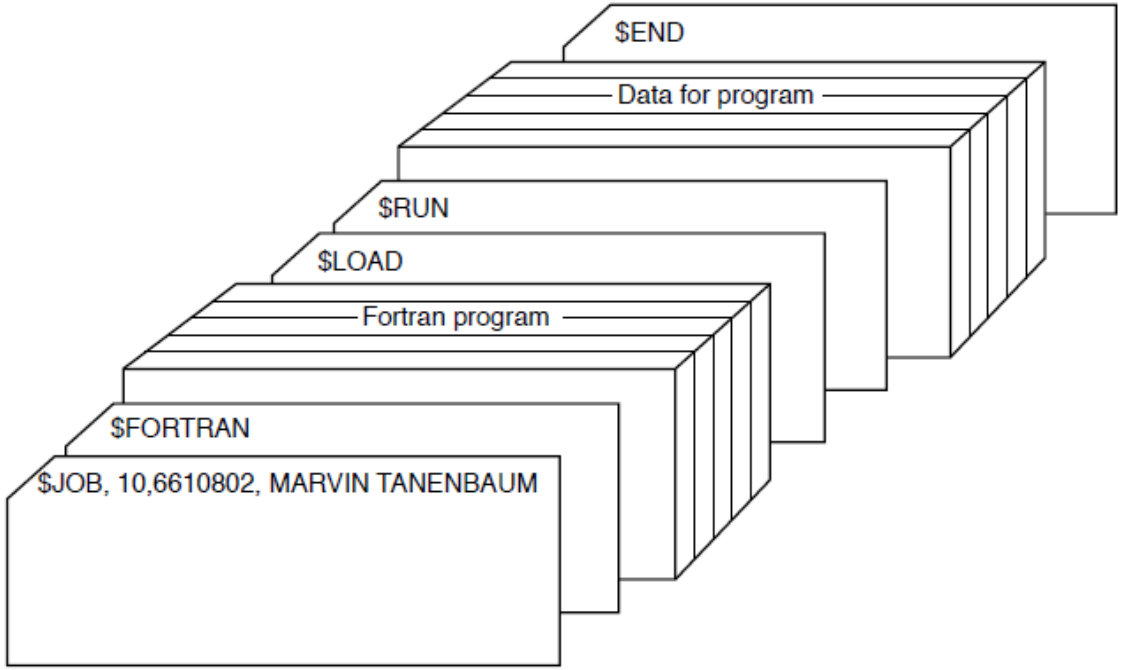
…. Easy Batch Systems

- ***Problemi***
  1. Come fa il monitor a sapere la natura del job (e.g., Fortran o Assembler?) o quale programma eseguire sui dati forniti?
  2. Come fa il monitor a distinguere un job da un altro oppure i dati dal programma

Soluzione: schede di controllo

- ***Schede di controllo***

  Schede speciali che indicano al monitor residente quali programmi mandare in esecuzione
  Caratteri speciali distinguono le schede di controllo dalle schede di programma o di dati.



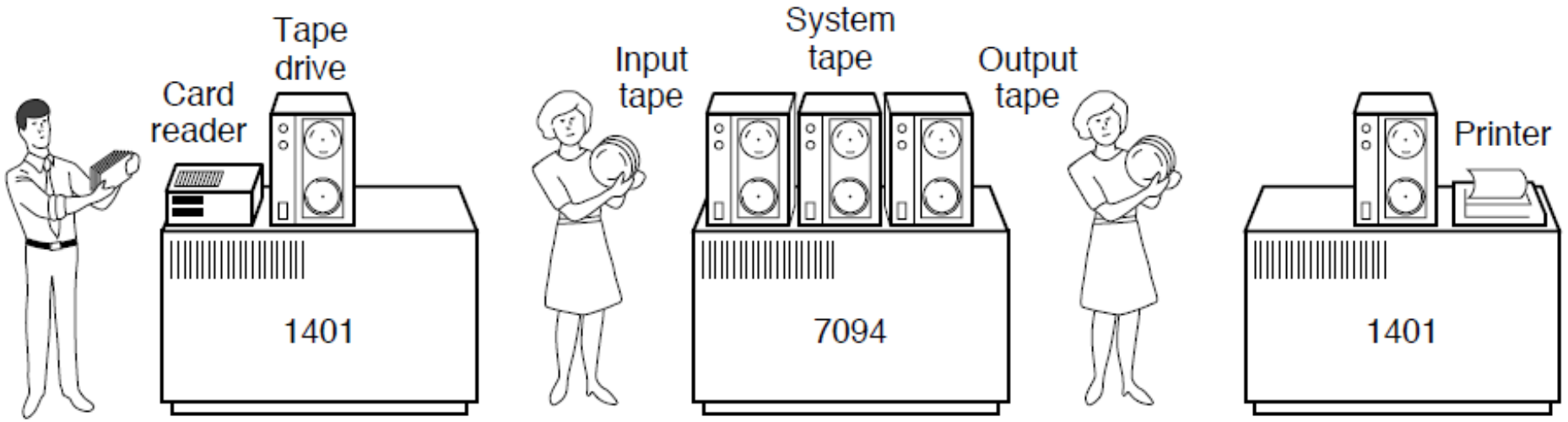
II GENERATION: TRANSISTORS AND BATCH SYSTEMS: MAINFRAME SYSTEMS

…. Easy Batch Systems

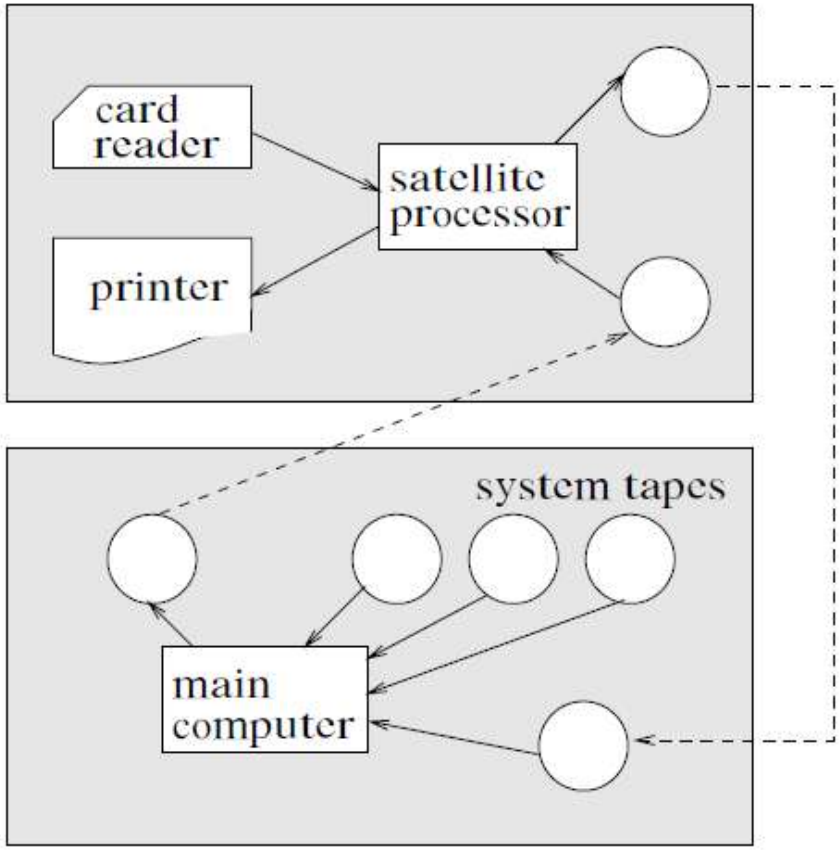
- ***Una parte del monitor residente è:***
  - Interprete delle schede di controllo – responsabile della lettura e esecuzione delle istruzioni sulle schede di controllo
  - Loader – carica i programmi di sistema e applicativi in memoria
  - Driver dei dispositivi – conoscono le caratteristiche e le proprietà di ogni dispositivo di I/O.

- ***Problema: bassa performance*** – I/O e CPU non possono sovrapporsi; i lettori di schede sono molto lenti.

Soluzione: operazioni off-line – velocizzare la computazione caricando i job in memoria da nastri, mentre la lettura e la stampa vengono eseguiti off-line



OPERAZIONI OFF-LINE



Il computer principale non è limitato dalla velocità dei lettori di schede o stampanti, ma solo dalla velocità delle unità nastro.
Non si devono fare modifiche nei programmi applicativi per passare dal funzionamento diretto a quello off-line
Guadagno in efficienza: si possono usare più lettori e più stampanti per una CPU.

O.S. EVOLUTION

MAINFRAME SYSTEMS

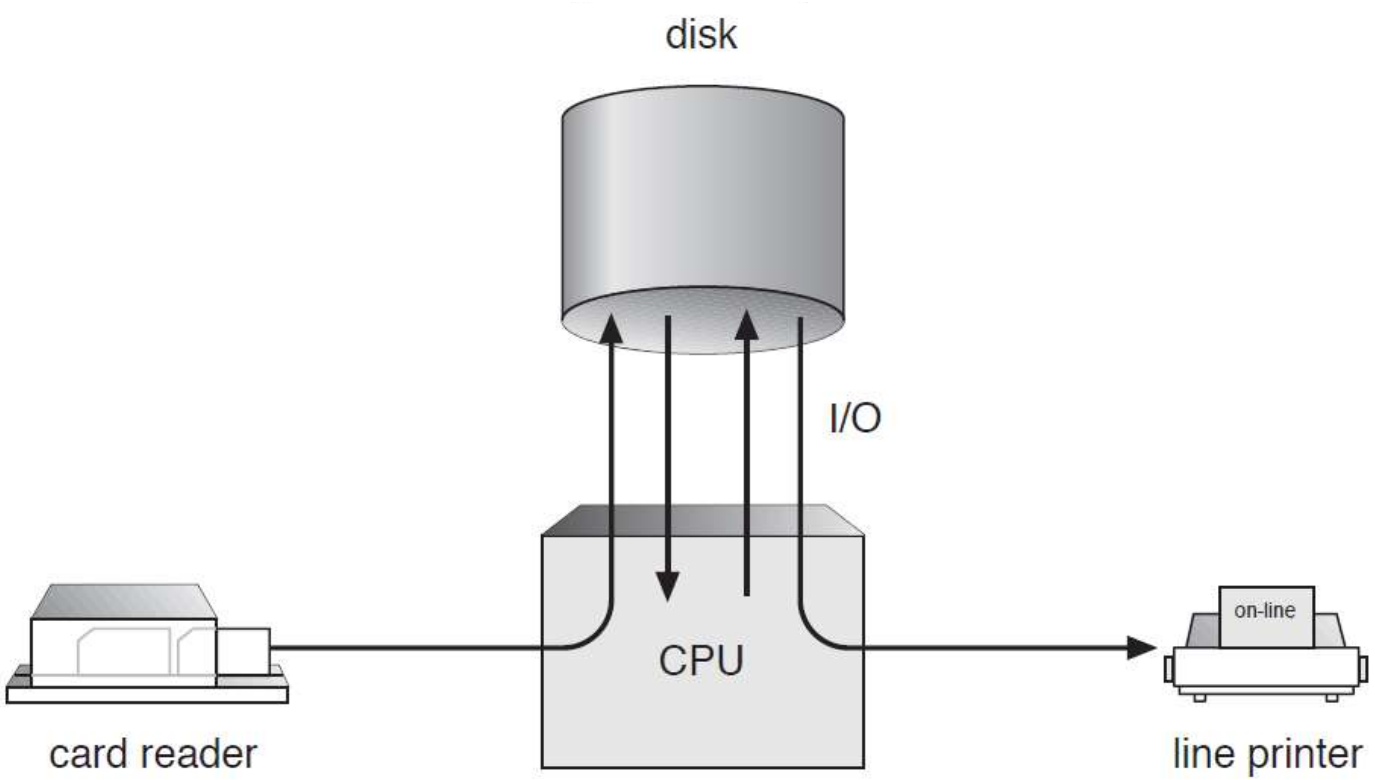
…. Sixty years: Multiprogrammed Batch Systems

☞ Several jobs are kept in main memory at the same time, and the CPU is multiplexed among them.

Caratteristiche dell'OS richieste per la multiprogrammazione

- Routine di I/O devono essere fornite dal sistema
- Gestione della Memoria – il sistema deve allocare memoria per più job
- Scheduling della CPU – il sistema deve scegliere tra più job pronti per l'esecuzione
- Allocazione dei dispositivi

Spool = Simultaneous Peripheral Operation On-Line



Sovrapposizione dell'I/O di un job con la computazione di un altro job.

Mentre un job è in esecuzione, il sistema operativo:

- legge il prossimo job dal lettore di schede in un'area su disco (coda dei job)
- trasferisce l'output del job precedente dal disco alla stampante

Job pool – struttura dati che permette al S.O. di scegliere quale job mandare in esecuzione al fine di aumentare l'utilizzazione della CPU.

O.S. EVOLUTION

MAINFRAME SYSTEMS

…. Seventy years: time-sharing (interactive)

- **Multiple users** simultaneously access the system through terminals
- **The CPU is multiplexed among several jobs** that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory).
- **A job swapped in and out of memory** to the disk.
- **Interaction between the user** and the system is provided; when the operating system finishes the execution of one command, it seeks the next "*control statement*" from the user's keyboard.
- **On-line file system** must be available for users to access data and code.

O.S. EVOLUTION

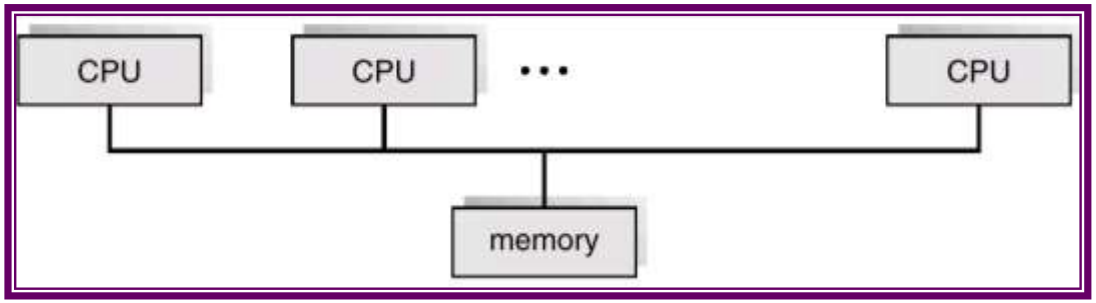
Eighty Years: DESKTOP SYSTEMS

- ***Personal computers*** – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Advanced user interface (GUI=Graphical User Interface)
- Can adopt technology developed for larger operating systems. Often individuals have sole use of computer and do not need advanced CPU utilization of protection features.
- May run several different types of operating systems (Windows System 7, MacOS, UNIX, Linux)

O.S. EVOLUTION

Ninety Years: PARALLEL SYSTEMS

- Multiprocessor systems with more than on CPU in close communication.
- *Tightly coupled system* – processors share memory and a clock; communication usually takes place through the shared memory.
- Advantages of parallel system:
  - Φ Increased *throughput*
  - Φ Economical
  - Φ Increased reliability
    - ✓ graceful degradation
    - ✓ fail-soft systems

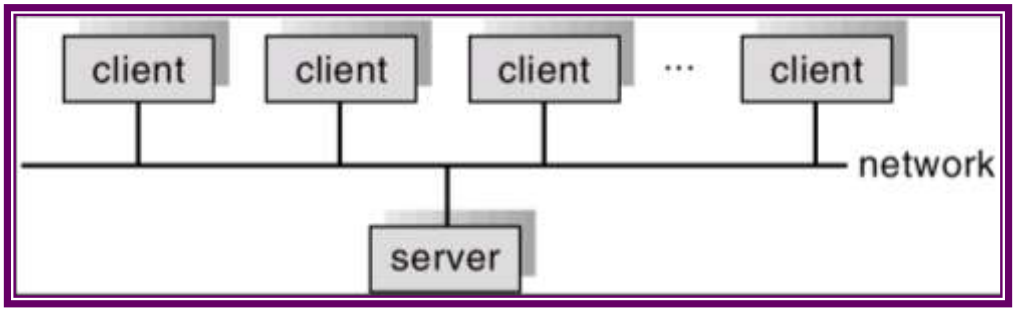


- *Symmetric multiprocessing (SMP)*
  - Φ Each processor runs and identical copy of the operating system.
  - Φ Many processes can run at once without performance deterioration.
  - Φ Most modern operating systems support SMP
- *Asymmetric multiprocessing*
  - Φ Each processor is assigned a specific task; master processor schedules and allocated work to slave processors.
  - Φ More common in extremely large systems

O.S. EVOLUTION

Ninety Years: COMPUTER NETWORKS

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
  - Φ Resources Sharing
  - Φ Computation speed up – load sharing
  - Φ Reliability
  - Φ Communications



- Requires networking infrastructure.
- Local area networks (LAN) or Wide area networks (WAN)

O.S. EVOLUTION

REAL-TIME SYSTEMS

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- Real-Time systems may be either *hard* or *soft* real-time.
- Hard real-time:
  - Secondary storage limited or absent, data stored in short term memory or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
  - Used in Robotics, Industrial control, on board software, . . .
  - Soft real-time
  - Limited utility in industrial control or robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

O.S. EVOLUTION

HAND-HELD SYSTEMS

- Personal Digital Assistants (PDAs)
- Cellular telephones
- Issues:
  - Limited memory
  - Slow processors
  - Small display screens.

The Future: DISTRIBUTED OPERATING SYSTEMS

In un sistema operativo distribuito, l'utente ha una visione unitaria del sistema di calcolo.

- Condivisione delle risorse (dati e computazionali)
- Aumento della velocità – bilanciamento del carico
- Tolleranza ai guasti

Un sistema operativo distribuito è molto più complesso di un SO di rete.

Esempi di servizi (non sistemi) di rete: NFS, P2P (KaZaA, Gnutella, . . .), Grid computing. . ., Cloud computing. . . .

FUTURE OPERATING SYSTEMS

Diversi obiettivi e requisiti a seconda delle situazioni

- Mainframe
- Supercalcolatori
- Server
- Multiprocessore
- Mobile Computer
- Real Time
- Palm or Embedded Computer
- Smart card

FUTURE OPERATING SYSTEMS

SISTEMI OPERATIVI PER MAINFRAME

- Enormi quantità di dati (> 1TB)
- Grande I/O
- Elaborazione “batch” non interattiva
- Assoluta stabilità (uptime > 99, 999%)
- Applicazioni: banche, amministrazioni, ricerca. . .
- Esempi: IBM OS/370, OS/390, Unix, Linux.

FUTURE OPERATING SYSTEMS

SISTEMI OPERATIVI PER SUPERCALCOLATORI

- Grandi quantità di dati (> 1TB)
- Enormi potenze di calcolo (es. NEC Earth-Simulator, 40 TFLOP)
- Architetture NUMA o NORMA (migliaia di CPU)
- Job di calcolo intensivo
- Elaborazione “batch” non interattiva
- Esempi: Unix, o ad hoc

FUTURE OPERATING SYSTEMS

SISTEMI PER SERVER

- Sistemi multiprocessore con spesso più di una CPU in comunicazione stretta.
- Degrado graduale delle prestazioni in caso di guasto (fail-soft)
- Riconfigurazione hardware a caldo
- Rilevamento automatico dei guasti
- Elaborazione su richiesta (semi-interattiva)
- Applicazioni: server web, di posta, dati, etc.
- Esempi: Unix, Linux, Windows

FUTURE OPERATING SYSTEMS

SISTEMI OPERATIVI PER COMPUTER PALMARI E SISTEMI EMBEDDED

- Per calcolatori palmari (PDA), cellulari, ma anche televisori, forni a microonde, lavatrici, registratori DVD, lettori musicali MP3, etc.
- Hanno spesso caratteristiche di real-time
- Limitate risorse hardware
  - Memoria ridotta
  - Processori lenti
  - Piccoli display

Esempio: SymbianOS, PalmOS, PocketPC, Windows mobile, Android, QNX.

FUTURE OPERATING SYSTEMS

SISTEMI OPERATIVI PER SMART CARD

- Girano sulla CPU delle smartcard
- Stretti vincoli sull'uso di memoria e alimentazione
- Implementano funzioni minime (es.: pagamento elettronico)
- Esempio: JavaCard

O.S. EVOLUTION

Migration of O.S. concepts and features