

Evoluzione dei sistemi operativi

In questa lezione impareremo...

- la storia dei sistemi operativi
- le caratteristiche dei sistemi operativi
- a classificare i sistemi operativi

■ Cenni storici

Sappiamo che è praticamente impossibile per l'utente utilizzare direttamente l'hardware di un calcolatore: anche se scrivessimo i programmi in linguaggio binario ci troveremmo di fronte a due problemi:

- come immettere programmi nella macchina (**Input**);
- come estrarre dalla memoria del calcolatore i risultati dei calcoli eseguiti per comunicarli all'esterno (**Output**).

Queste problematiche sono state le prime a essere affrontate dai progettisti sin dagli anni Quaranta del secolo scorso: i primi tentativi di facilitare l'interazione uomo-macchina si fecero creando dei programmi e/o sottoprogrammi di uso comune messi a disposizione dei programmatori per essere richiamati nelle diverse applicazioni. Nacque così il **software di base**, cioè un corredo di programmi di utilità per poter integrare nei programmi l'utilizzo dell'hardware senza dover ogni volta riscrivere tutte le funzioni di basso livello.

La storia dei sistemi operativi segue quindi parallelamente l'**evoluzione dell'hardware**: i passi da gigante fatti dalla tecnologia e gli sviluppi nel campo delle architetture e delle apparecchiature periferiche rendono necessario lo sviluppo di nuovo software di base.

Possiamo fare una prima classificazione dei sistemi operativi sulla base della tipologia di colloquio uomo-macchina dalle origini dell'informatica fino a oggi:

- sistemi **dedicati**;
- sistemi **a lotti (batch)**;
- sistemi **interattivi**:
 - conversazionali;
 - in tempo reale;
 - transazionali.

■ Sistemi dedicati (1945-1955)

La prima fase, quella compresa tra il 1945 e il 1955, può essere definita come quella dei sistemi dedicati. I calcolatori erano valvolari, occupavano intere stanze, erano lenti, molto costosi e a uso quasi esclusivo di università o centri di ricerca.

Il **tempo macchina**, inteso come tempo di utilizzo della **CPU**, era una risorsa pregiata concessa a pochi utenti privilegiati per brevi periodi (turni), e durante il proprio turno i vari utenti avevano a disposizione l'intero sistema: il programmatore caricava i propri programmi (scritti in linguaggio macchina prima e successivamente in assembler) in memoria, li faceva eseguire e ne controllava l'esecuzione.

Non esisteva un **sistema operativo** vero e proprio: le poche funzioni disponibili per la gestione dell'hardware venivano inviate impostando interruttori e commutatori della console.

Non esistevano tastiera e monitor, e i risultati dei programmi si leggevano direttamente in alcuni registri rappresentati sul pannello tramite indicatori luminosi.

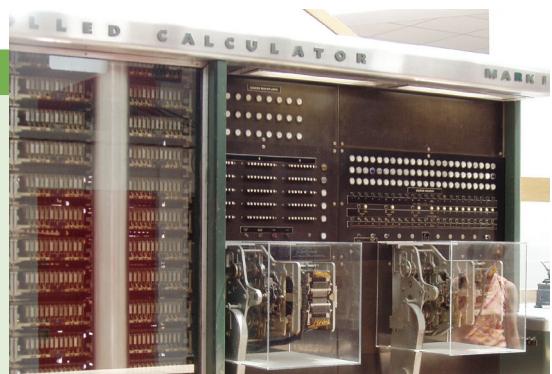


Zoom su...

MARK1, ENIAC E UNIVAC

Il primo calcolatore è forse quello costruito dall'università di Manchester e chiamato Universal Electronic Computer (successivamente rinominato **Mark1**) nel 1945: non esisteva alcun linguaggio assemblativo e veniva programmato con un alfabeto a 32 simboli, trasferiti come gruppi di 5 bit in memoria mediante un lettore/perforatore di nastro.

Negli Stati Uniti il primo calcolatore funzionante fu l'**ENIAC** (1945), una macchina costituita da 19.000 valvole e 1500 relays che occupava un intero locale; il peso complessivo superava le 30 tonnellate; la potenza consumata era di 200 KW. In origine l'**ENIAC** veniva programmato intervenendo manualmente su interruttori e connessioni dei cavi e solo in un secondo tempo, nel 1948, si iniziarono a scrivere primitive software conservate in tabelle di memoria precursori delle odiere **ROM**.



L'**UNIVAC 1** (1951) è una macchina con un importante significato storico per l'industria informatica perché è stato il primo prodotto a essere venduto al pubblico, cioè creato con uno scopo commerciale. Era dotato di una tastiera di una macchina da scrivere per l'input ed equipaggiato con diverse unità a nastro magnetico. È stato usato per predire correttamente il risultato delle elezioni presidenziali del 1952.

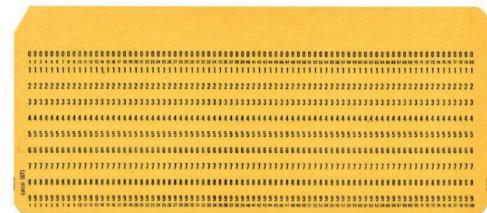


In questi primi calcolatori il sistema operativo consisteva essenzialmente in alcuni programmi per il trasferimento di blocchi di dati tra la memoria principale a quella ausiliaria, costituita da nastri magnetici.

■ Gestione a lotti (1955-1965)

La gestione appena descritta permetteva un basso utilizzo della CPU in quanto la maggior parte del tempo veniva sprecata per caricare i programmi: in un sistema molto costoso sicuramente tale situazione non era soddisfacente.

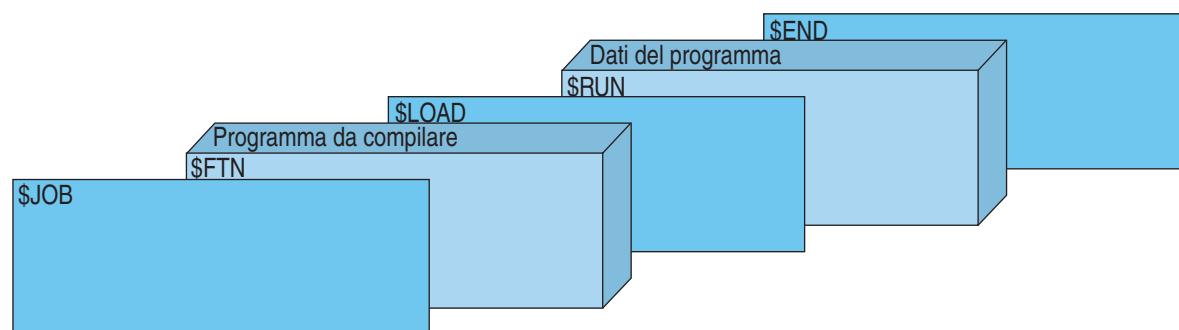
Il passo successivo fu quello di introdurre un sistema automatico di caricamento dei programmi: si utilizzarono le **schede perforate** come supporti su cui memorizzare i programmi, i comandi con le chiamate di sistema operativo e i lettori di schede come dispositivi di input.



Ogni **scheda perforata** conteneva un'**istruzione**.

L'utente non utilizzava più la console del calcolatore, ma la sua interazione avveniva semplicemente inserendo un pacco di schede (lotto) nell'apposito lettore: questo tipo di gestione fu proprio chiamata **gestione a lotti** (**batch processing**) o elaborazione a lotti.

I lavori degli utenti erano costituiti da pacchi di schede perforate dove ogni lotto (chiamato lavoro o **job**) iniziava con una scheda di identificazione dell'utente e consisteva in una sequenza di **passi** o **step**, dove i singoli step erano le funzioni richieste dall'utente al sistema operativo, come la compilazione del programma, il caricamento, l'esecuzione ecc.



Il primo sistema operativo di questo tipo fu sviluppato da General Motors per IBM e prese il nome di “**bath monitor**” (o semplicemente **monitor**) e aveva come compito essenziale quello di trasferire il controllo da un job appena terminato a un nuovo job in partenza.

Le schede di controllo utilizzavano un particolare linguaggio per comunicare con il calcolatore: il **JCL** o **Job Control Language**.

Un job è delimitato da due schede speciali di controllo: **\$JOB** e **\$END**.

Per esempio il comando **\$FTN** attivava il compilatore **Fortran**, **\$RUN** richiedeva l'esecuzione, **\$END** terminava l'elaborazione ecc.

Il processore alternava quindi l'esecuzione di istruzioni di controllo ai programmi: gli eventuali dati dovevano essere inclusi nel pacco di schede e quindi i singoli lavori prevedevano anche un'alternanza di schede di istruzioni, di controllo e di schede dati.

I risultati delle elaborazioni finivano stampati su un tabulato comune a tutti i job.

Le principali caratteristiche di questo tipo di sistema possono essere così elencate:

- il sistema operativo è **sempre residente** in memoria (monitor);
- in memoria centrale è presente **un solo job alla volta**;
- finché il job corrente non è terminato, il **successivo non può iniziare l'esecuzione**;
- **non è presente interazione** tra utente e job;
- se un job si sospende in attesa di un evento, la **CPU rimane inattiva**;
- ha una **scarsa efficienza**: durante l'I/O del job corrente, che sono per loro natura molto lente, la **CPU** rimane inattiva.

Il problema principale della scarsa efficienza di questo tipo di gestione era proprio legato al fatto che la **CPU** veniva **sempre altamente sottoutilizzata** dato che sia il lettore di schede sia la stampante, essendo componenti meccanici, erano di gran lunga più lenti del processore, che risultava per la maggior parte del tempo inattivo aspettando le “lentezze” delle sue periferiche.

Furono quindi introdotti perfezionamenti nella gestione a lotti per rimuovere tale collo di bottiglia:

- si passò dalle schede ai nastri magnetici, aumentando così la velocità di trasferimento;
- negli anni Cinquanta apparvero i primi calcolatori con canali di I/O autonomi e con le prime memorie a disco di grande capacità.

I calcolatori da strutture “monolitiche” iniziano a “scomporsi” in più componenti, come si può vedere dalla fotografia, che riprende un sistema **IBM 1401** del 1959.

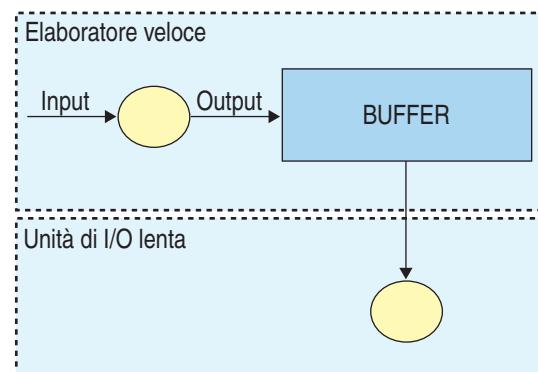


Buffering

Un primo rimedio per ovviare ai tempi di attesa delle periferiche fu l'introduzione della tecnica di **buffering**: un **buffer** è un'area di memoria intermedia dedicata al salvataggio temporaneo di informazioni, implementato direttamente nel **device driver** del dispositivo.

Con il **buffering** l'accesso alla periferica segue questa sequenza di operazioni:

- ▶ la periferica legge in Input/Output un blocco dati (blocco corrente) e lo passa alla CPU;
- ▶ mentre la **CPU** elabora il blocco dati corrente, la periferica legge il blocco dati successivo;
- ▶ la **CPU** produce output solo fino a quando il buffer è pieno e quindi si ferma;
- ▶ la periferica di uscita legge i dati prodotti dalla **CPU** prelevandoli dal buffer al suo ritmo.



Spooling

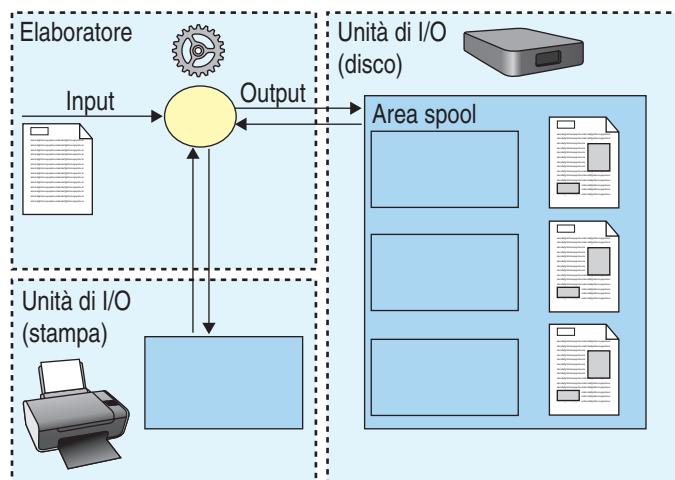
Nonostante l'introduzione dei buffer, rimasero alcuni problemi da risolvere:

- ▶ i buffer non possono essere molto grandi per motivi di costi;
- ▶ il buffer tende a riempirsi velocemente e quindi il problema dell'attesa non viene risolto;
- ▶ con **velocità di I/O << velocità CPU** non produce giovamenti;
- ▶ se il buffer si riempie, la CPU si blocca.

Si arrivò all'introduzione di meccanismi chiamati **dischi di spooling** (**Simultaneous Peripheral Operations On Line**), creando contemporaneità tra le attività di I/O e quelle di computazione: il disco viene impiegato come un **buffer** molto ampio, dove **leggere** in anticipo i dati e **memorizzare** temporaneamente i risultati, in attesa che il dispositivo di output sia pronto.

Invece cioè di scrivere e leggere dai dispositivi fisici si legge e si scrive su due **dispositivi virtuali**:

- ▶ il disco di **spooling di ingresso** contiene le schede lette dal lettore e contenenti **job** non ancora eseguiti (“**job virtuali**”);
- ▶ il disco di **spooling di uscita** contiene invece “**output virtuali**”, ossia tabulati già prodotti relativi a job eseguiti ma non ancora stampati.



In questo modo è possibile sovrapporre i tempi di elaborazione con quelli di **I/O**: mentre la **CPU** elabora il programma, i canali di I/O provvedono a riempire o svuotare i dischi di spooling e a gestire i trasferimenti tra la memoria e i dischi stessi.



■ Sistemi interattivi (1965-1980)

◀ Il DMA (Direct Memory Access, "accesso diretto alla memoria") è un meccanismo che permette ad alcuni dispositivi, come unità a disco, schede grafiche, schede audio e musicali, di accedere direttamente alla memoria del calcolatore per trasferire i dati senza far "sprecare" tempo di elaborazione alle CPU: a trasferimento ultimato richiamano l'attenzione della CPU generando un interrupt, cioè mandando un segnale asincrono che indica alla CPU che la periferica ha terminato il proprio lavoro e il risultato è disponibile per l'utilizzo. ►

Negli anni Sessanta IBM introduce i canali di I/O per ottenere il parallelismo effettivo delle operazioni: questi dispositivi, chiamati anche processori di I/O, hanno il compito di gestire totalmente le periferiche, così da lasciare alla CPU l'esecuzione dei programmi (oggi questo meccanismo si chiama ◀ DMA ▶ (Direct Memory Access)).

Multiprogrammazione

Con questo sistema si è data una prima risposta al problema costituito dai tempi morti di attesa: il secondo passo si è fatto introducendo la multiprogrammazione. L'osservazione che sta alla base di questa tecnica è il fatto che spesso, durante l'elaborazione, un job è costretto a fermarsi in attesa di un evento esterno e quindi

la CPU rimane inoperosa fino a quando questa situazione viene risolta (pensiamo per esempio a un errore nel codice in fase di test oppure a un programma che necessita che l'utente inserisca un dato dall'esterno per poter proseguire).

L'attesa può durare anche diversi minuti.

L'idea che ha fatto nascere la multiprogrammazione è quella di eliminare questi tempi tenendo pronti altri job in memoria che, nel caso uno si debba sospendere, prendano il suo posto nell'esecuzione.

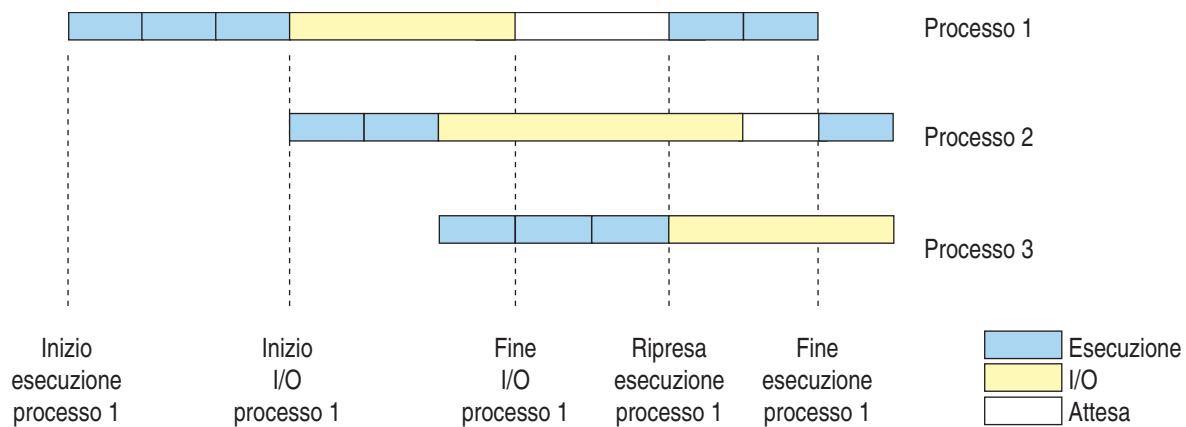
In questo caso il "tempo perso" risulta minimo, dovuto solo al ◀ cambio di contesto ▶.

◀ Con cambio di contesto (context switch) si intendono le operazioni tramite le quali il sistema operativo passa dall'esecuzione di un programma (processo) all'esecuzione di un altro processo, salvando lo stato del primo e predisponendo il sistema per poter mandare in esecuzione il secondo. Con processo si intende un'istanza del programma in evoluzione, cioè che è eseguito dal processore ►



MULTIPROGRAMMAZIONE

Nella multiprogrammazione sono caricati in memoria centrale contemporaneamente più job: mentre un job è in attesa di un evento e sospende la sua esecuzione, il sistema operativo assegna la CPU a un altro tra quelli pronti all'esecuzione.





Avere in memoria più programmi, come vedremo, significa anche gestire lo spazio della **RAM** suddividendola in parti e caricando i nuovi programmi che devono essere eseguiti ma “scaricando” quelli che terminano l’esecuzione.

Memoria
RAM

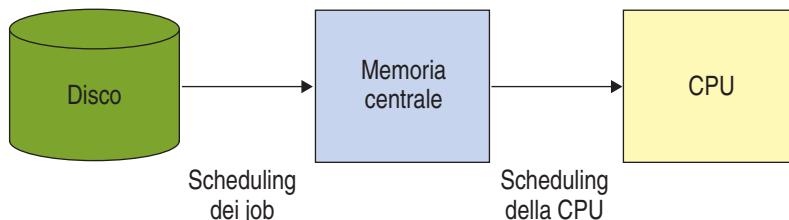
Il ruolo del sistema operativo diventa sempre più importante e complesso e le attività che deve svolgere divengono sempre più numerose: il sistema operativo deve **portare avanti** contemporaneamente l’esecuzione di più **job**.

Per poter evolvere simultaneamente, però, i programmi devono essere anche contemporaneamente residenti in memoria centrale: si parla ora di **multiprogrammazione**, nel senso che più job sono caricati in **RAM** e pronti a essere eseguiti.

Dato che **un solo job** utilizza in un dato istante la **CPU** e **più job** attendono (in memoria centrale) di acquisire la **CPU**, quando il job che sta utilizzando la **CPU** si sospende in attesa di un evento, il SO **decide** a quale job assegnare la **CPU** ed effettua lo scambio (**scheduling**).

Il **SO** effettua delle scelte tra tutti i **job**:

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



Vedremo in seguito in base a quali criteri vengono scelti i job da caricare e da mandare in esecuzione.

Un ulteriore passo avanti viene fatto con l’esigenza di soddisfare nuove tipologie di programmi: i programmi **interattivi**, quelli cioè che richiedono un’alternanza tra elaborazione e inserimento dati da parte dell’utente (il caso limite sono i videogiochi). In questi programmi il job si sospende continuamente per le operazioni di I/O.

Grazie alla multiprogrammazione, inoltre, i sistemi sono diventati **multitutente**. All’unità centrale sono connessi molti terminali: più utenti interagiscono contemporaneamente con il sistema e il SO deve essere in grado di consentire a più persone l’utilizzo del calcolatore, come se ciascuna fosse l’unica proprietaria (virtualizzazione del calcolatore).

Time sharing

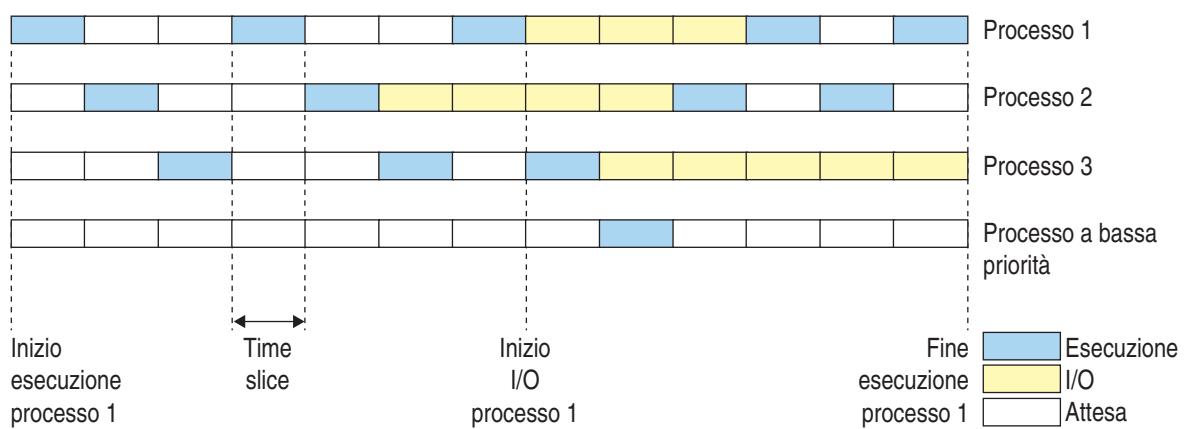
Nel 1963 il **MIT** (Massachusetts Institute of Technology) mise a punto un nuovo concetto di sistema operativo (il **Compatible Time Sharing System, CTSS**), che può essere considerato il primo dei sistemi operativi di tipo **interattivo**, operante su un sistema IBM7094 al quale erano connessi contemporaneamente più terminali.

Il termine **time sharing** (a partizione di tempo) sta a indicare il meccanismo di funzionamento: a ogni utente connesso con il suo terminale al sistema viene assegnata una porzione di tempo (**quanto**) di utilizzo **CPU** (**time slice**, tipicamente della durata di poche centinaia di milisecondi).

Se il **job** non si conclude nel tempo macchina a lui assegnato, viene temporaneamente sospeso e posto, assieme a tutti i job che sono nelle medesime condizioni, in una lista “di sospensione”, e il processore viene assegnato a un altro programma che era stato, come lui, precedentemente sospeso e così via.

In questo modo vengono eliminati i tempi di inattività del sistema dovuti alle operazioni di I/O, di editing e messa a punto dei programmi propri dell’attività interattiva.

Nell’esempio di cui alla figura sottostante sono riportati 4 processi in memoria che si alternano nell’utilizzo della **CPU** secondo la suddivisione del tempo in “**time slice**”: si può osservare che l’ultimo processo è “meno importante” dei primi, gli viene assegnata una priorità più bassa e accede con meno frequenza alla **CPU** rispetto agli altri.



Con il **time sharing** un job che sta utilizzando la CPU può terminare la sua elaborazione sostanzialmente per due motivi:

- ha terminato il tempo a lui assegnato, cioè il suo **time slice**;
- ha bisogno di qualche risorsa non disponibile (per esempio di un dato di input o di una periferica).

Questi meccanismi (politiche di gestione della CPU) saranno oggetto della prossima unità didattica.

Il concetto di **time sharing** fu reso famoso dal calcolatore **PDP 10** costruito da **Digital Equipment Corporation (DEC)** e utilizzato in diverse università americane, tra le quali ricordiamo **MIT AI Lab**, **Stanford AI Lab** e **Carnegie Mellon**.

Alla fine degli anni Sessanta **IBM** presenta un progetto innovativo di computer, il **System/370**, con la memoria principale basata su circuiti integrati, l'aritmetica in floating point e il supporto per la memoria virtuale con il nuovo sistema operativo **VM/370**.

L'utilizzo condiviso dell'elaboratore ha portato anche un nuovo problema: la **sicurezza** dei dati e dei programmi, questione ancora oggi centrale nell'informatica.

Digital risponde con la linea **VAX**, un'evoluzione del **PDP**, introducendo l'indirizzamento virtuale, il demand paging e il sistema operativo **Unix**, scritto per la parte a basso livello in linguaggio macchina e per la parte ad alto livello in **C**, linguaggio di programmazione progettato da **Ken Thompson, Brian Kernighan e Dennis Ritchie**.

Avere più utenti che condividono il calcolatore e le sue risorse ha creato l'esigenza di proteggere il lavoro di ogni utente in quanto un job potrebbe involontariamente (o deliberatamente) danneggiare il programma (o i dati) degli altri utenti: nasce quindi anche il problema della **riservatezza**, per garantire che ciascuno possa accedere solamente ai dati di propria competenza.

■ Home computing (anni Settanta)

Negli anni Settanta il calcolatore non viene più visto solo come uno strumento di calcolo, ma anche di intrattenimento e divertimento: si punta al mercato casalingo e pioniere in questo settore fu **Nolan Bushnell**, fondatore del primo grande colosso dei videogiochi: **Atari** (1972).

A esso si affiancarono **Commodore**, **Sinclair** e altre case produttrici di **home computer**, machine con architetture hardware poco costose (con un sistema operativo dedicato) programmabili generalmente in **BASIC**, che usavano come monitor il televisore domestico e avevano come periferiche di massa dapprima nastri magnetici (anche normali cassette audio) e in un secondo tempo unità con singolo floppy disk.



Si diffusero rapidamente sia per il costo modesto sia per la molteplicità di programmi disponibili: nasce il mercato dei **videogiochi**, un settore commerciale tuttora di grande importanza per l'informatica.

■ Sistemi dedicati (anni Ottanta)

Una nuova tipologia di sistemi operativi dedicati si presenta negli anni Ottanta con l'avvento del personal computer: si ripercorre la storia dei sistemi operativi su macchine personali, cioè su piccoli sistemi di calcolo (piccoli di dimensioni, ma con potenza di calcolo superiore ai mainframe degli anni Settanta).

Nel 1982 Microsoft rilascia **MS-DOS**, un sistema operativo per i microprocessori della famiglia x86 INTEL (PC IBM e compatibili). La tecnologia influenza notevolmente il sistema operativo: i monitor iniziano a essere grafici, cioè non più solo monocromatici a cristalli verdi (o ambra), e la loro risoluzione non è più a linee di caratteri (80×25) ma a matrice di punti (pixel).



Prende piede il **software grafico interattivo**, con nuovi strumenti di interazione con l'utente come il **mouse**, e quindi nuove tipologie di sistemi operativi: il primo esempio lo troviamo con il sistema operativo **MacOS** (acronimo di **Macintosh Operating System**) di **Apple** (1984), che è il capostipite di una nuova generazione di SO visuali dedicati (l'anno successivo, nel 1985, anche Microsoft rilascerà la prima versione del proprio sistema operativo **Windows**: la 1.0).

Questo tipo di sistemi operativi prende anche il nome di **sistemi monoutente** e il nome stesso indica che forniscono una macchina virtuale semplice, per un solo utente alla volta, che facilita l'applicazione di un vasto numero di pacchetti software, permettendo nel contempo all'utente di sviluppare e mettere in esecuzione programmi per proprio conto.

Nei sistemi operativi monoutente si dà maggior rilievo alla disponibilità di un linguaggio di comando facilmente utilizzabile, di un file system semplice e di risorse di I/O per dischi e dispositivi vari.

■ Sistemi odierni e sviluppi futuri

Nell'informatica è sempre difficile fare previsioni data la rapida evoluzione del settore: previsioni semplici sono quelle di un continuo e costante aumento dei pc personali, della connessione alla rete e dei computer indossabili (**wearable computer**) e i maggiori sforzi dei progettisti saranno rivolti alla progettazione di sistemi operativi per queste tre classi di macchine. Possiamo individuare e illustrare alcune tipologie di sistemi operativi oggi disponibili per i diversi settori di applicazione dell'informatica:

- Sistemi Operativi per **Server**;
- Sistemi Operativi in **Tempo Reale**;
- Sistemi Operativi **Embedded**;
- Sistemi Operativi per **Smart Card**;
- Sistemi Operativi per **Smartphone**.



SO per server

Anche la più piccola organizzazione oggi ha i PC collegati in rete locale, quindi i SO per rete sono e saranno oggetto di sviluppo.

Hanno come caratteristica principale quella di servire molti utenti in contesti di rete, mettendo a disposizione accessi concorrenti e servizi, dall'utilizzo delle stampanti e dei dischi all'accesso a Internet e alla posta elettronica ecc.

SO real-time

Sono sistemi operativi altamente specializzati, sviluppati per quei dispositivi hardware sui quali le applicazioni hanno forti vincoli **real-time**, cioè interagiscono con l'ambiente esterno attraverso periferiche e garantiscono a dati di ingresso risposte in un **tempo utile** rispetto alle costanti di tempo proprie dell'ambiente esterno.

Riguardano generalmente le applicazioni di controllo di processo e dei sistemi di telecomunicazione e sono sostanzialmente di due tipi:

- **hard real-time**: hanno una **deadline hard**, cioè un limite temporale rigido che non deve mai essere superato perché le conseguenze di un eventuale fallimento potrebbero essere disastrose (per esempio controllo aereo);
- **soft real-time**: permettono che qualche volta si "sfori" la deadline, quindi i vincoli temporali non sono tassativi e le conseguenze di un eventuale fallimento non sono disastrose (per esempio rete telefonica).

Sistemi embedded

Non esiste una definizione universalmente riconosciuta per i sistemi **embedded**, e una tra quelle più utilizzate è quella riportata di seguito.



SISTEMA EMBEDDED

In generale con sistema embedded si intende un dispositivo incapsulato all'interno del sistema da controllare progettato per una determinata applicazione e supportato da una piattaforma hardware su misura.

Sono sistemi generalmente privi di interazione umana, capaci di resistere a eventi dannosi, a vibrazioni e shock e di ripartire in modo autonomo dopo un'eventuale interruzione; sono di dimensioni ridotte e realizzati con un hardware dedicato per svolgere un particolare compito, hanno poca memoria e generalmente devono avere un basso consumo.

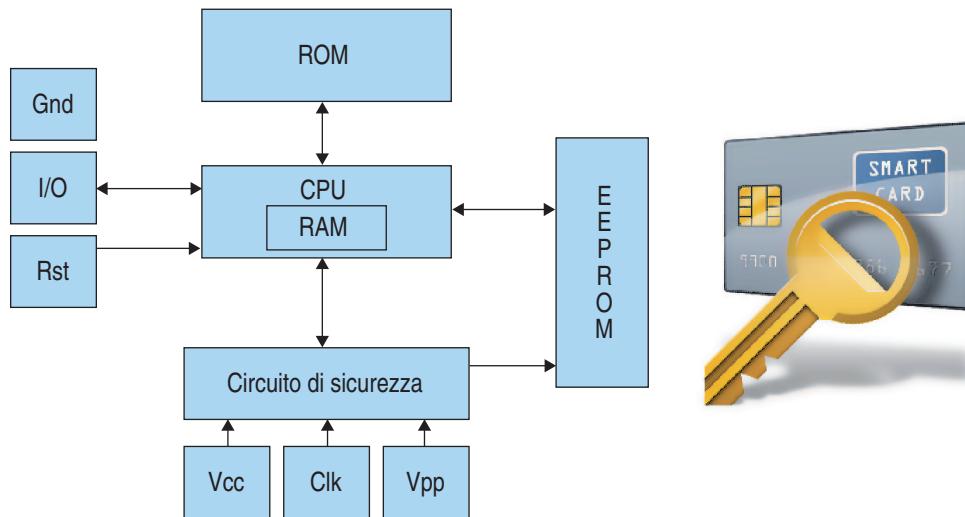
Rientrano tra queste applicazioni i telefoni cellulari, gli elettrodomestici, i palmari ecc.

Sistemi su Smart Card

Possono essere visti come un caso limite dei sistemi embedded estremi: sono generalmente di plastica, della dimensione di una carta di credito e incorporano nel loro interno un microchip con notevoli capacità di elaborazione e di memorizzazione.

All'interno del chip possono essere integrati molteplici componenti, quali microprocessori, memorie, ROM, RAM, antenne ecc., come un vero personal computer.

Le **Smart Card**, oggi largamente diffuse, hanno bassi costi di realizzazione e vengono usate per applicazioni di ogni genere: per utilizzare i mezzi di trasporto pubblici (come le **Oyster card**, biglietto elettronico usato nell'area di Londra), come carte di credito e di debito, come schede di sicurezza per l'accesso agli edifici e per applicazioni sanitarie o documenti di identità, fino alle raccolte punti dei supermercati.



Le **carte di vicinanza**, cioè quelle che utilizzano l'antenna per scambiare informazioni su distanze fino ai due metri, oggi non sono ancora di uso corrente, ma saranno utilizzate nel prossimo futuro per il ticketing e per persone con disabilità (per esempio per coloro che hanno limitate capacità d'uso di macchinari come le obliteratrici degli autobus).

I **chip** sono computer completi e sin dalle prime versioni di **Smart Card** i produttori stessi hanno sviluppato e proposto **sistemi operativi dedicati**: solo dal 1997 sono disponibili SO sviluppati da terze parti, cioè software house non produttrici anche dell'hardware (**MULTOS** e **Windows** per Smart Card).

Dai sistemi operativi scritti in linguaggio assembler si è passati a una soluzione simile all'implementazione di una **Java Virtual Machine** sulla memoria di una **Smart Card**: oggi uno dei sistemi più utilizzati è il **SmarTEC OS**, che risiede sulla **ROM** della carta ma che permette modifiche al codice presente sulla memoria **EEPROM** da parte dello sviluppatore.

Sistemi per smartphone

Con uno **smartphone** non si telefona come con un semplice cellulare, si comunica via email, lo si utilizza per navigare in Internet, per sentire musica e fotografare, per cercare un indirizzo od un locale e farsi guidare a destinazione dall'applicazione di navigazione satellitare installata sul telefono.



Lo **smartphone** è di fatto un personal computer tascabile e come ogni personal computer ha un sistema operativo che gestisce ogni sua potenzialità: oggi sono cinque le principali piattaforme per **smartphone**: **Android**, il “leader indiscusso”, **iOS**, “l’user friendly”, il **BlackBerry**, particolarmente indicato per il business, **Symbian** e **Windows Phone 7**.

Ci sono inoltre altri **SO** per cellulari, come **Bada** e **webOS**, ma non coprono un segmento di mercato significativo.

Verifichiamo le conoscenze

1. Risposta multipla

1 Quale di queste affermazioni è vera?

- a. Mark1 non aveva alcun linguaggio assemblativo
- b. ENIAC non aveva alcun linguaggio assemblativo
- c. UNIVAC non aveva alcun linguaggio assemblativo
- d. Macintosh non aveva alcun linguaggio assemblativo

2 Con il termine monitor si intende:

- a. un sinonimo dello schermo
- b. un sistema operativo
- c. una primitiva di sistema
- d. una parte hardware

3 Uno step di un job in un sistema batch è ad esempio:

- a. il caricamento del programma
- b. l'esecuzione del programma
- c. la terminazione di un programma
- d. la compilazione di un programma

4 Nei sistemi batch JCL sta a significare:

- a. Java Control Language
- b. Job Center Language
- c. Job Center Line
- d. Job Control Language

5 Il time-slice è dell'ordine di:

- a. poche decine di millisecondi
- b. poche centinaia di secondi
- c. qualche secondo
- d. pochi decimi di secondo

6 Windows è un sistema operativo:

- a. multiprogrammato
- b. multiutente
- c. time sharing
- d. batch

7 Quale di questi sistemi non è Hard real-time ?

- a. Controllo navale
- b. Forno a microonde
- c. Ascensore
- d. Air bag

8 Quale affermazione sui sistemi Smart Card è falsa?

- a. È un sistema embedded
- b. Sono programmate in java
- c. Sono usate come abbonamenti a mezzi pubblici (Oyster card)
- d. Non hanno un SO dedicato

2. Vero/falso

1 La prima fase compresa tra il 1945 e il 1955 è caratterizzata da SO dedicati.

V F

2 L'UNIVAC 1 era dotato di una lettore di schede perforate.

V F

3 La gestione a lotti è sinonimo di batch processing.

V F

4 Il SO sceglie tra tutti i job quali caricare/scaricare dalla memoria centrale.

V F

5 Il buffering è una variante dello spooling.

V F

6 In un sistema multiprogrammato più processi sono in esecuzione contemporaneamente.

V F

7 Le console per videogiochi sono sprovviste di sistema operativo.

V F

8 Windows è il primo SO a finestre.

V F

9 Tra i sistemi embedded rientrano anche le carte a microchip.

V F

10 Symbian è un sistema operativo per sistemi mobili.

V F