UNIVERSITA' DEGLI STUDI DI FOGGIA
DIPARTIMENTI
DI AREA MEDICA

CdLS in Odontoiatria e Protesi Dentarie

Corso di Informatica

Prof. Crescenzio Gallo
crescenzio.gallo@unifg.it

Funzioni dei Sistemi Operativi

Le funzioni principali del SO

- **Gestire** le risorse dell'elaboratore (CPU, memoria, dispositivi)
- **Controllare** che le operazioni vengano eseguite in modo regolare
- **Determinare** dove memorizzare dati e programmi
- **Coordinare** la comunicazione tra i vari componenti del computer
- **Gestire** l'interazione tra l'utente e i programmi applicativi

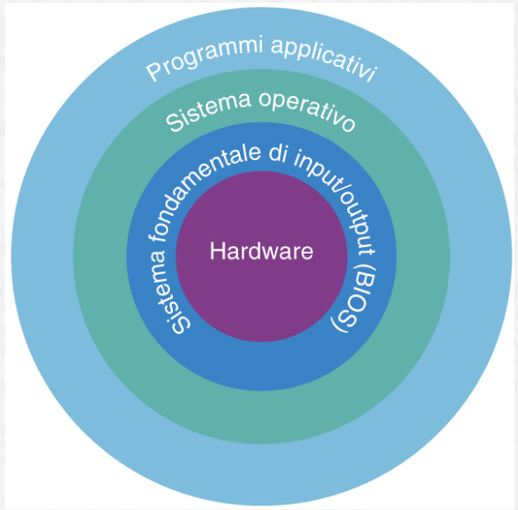
Il ruolo del BIOS

BIOS: **B**asic **I**nput **O**utput **S**ystem

Programma memorizzato su un chip di memoria non volatile (ROM/Flash)

Funzioni

- Interpreta i dati immessi tramite la tastiera
- Visualizza i caratteri sullo schermo
- Gestisce le comunicazioni attraverso le porte del computer



Fornisce un collegamento tra il software ed i componenti hardware dell'elaboratore

L'U.E.F.I.

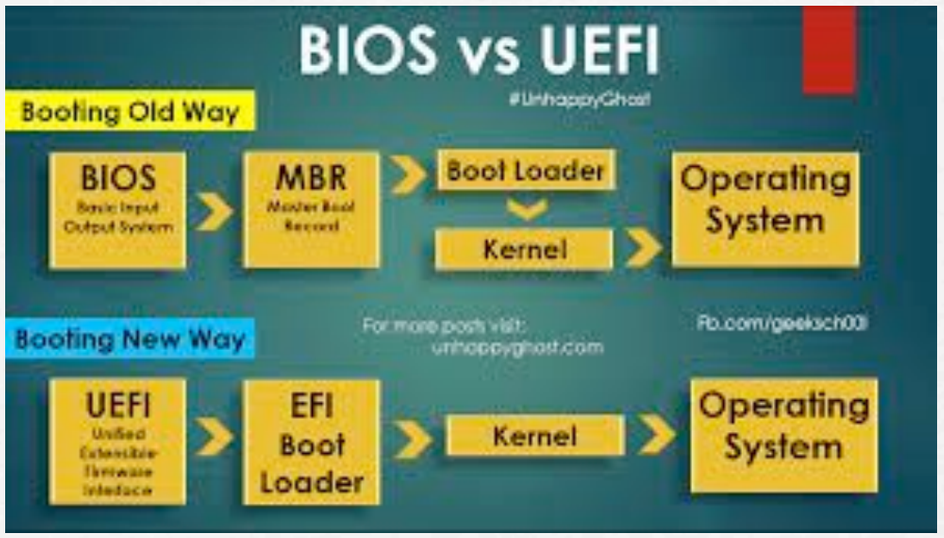
I moderni personal computer (basati su Linux, Mac OS X e Windows 8/10) integrano un nuovo BIOS detto **UEFI** (*Unified Extensible Firmware Interface*) proposto da Intel per superare le limitazioni del vecchio BIOS e che, come quest'ultimo, è un insieme di routine - scritte in una memoria non volatile presente sulla scheda madre - che forniscono le funzionalità basilari per l'accesso e l'utilizzo delle varie componenti del computer.

UEFI consente ai produttori di personal computer di integrare nel firmware della scheda madre nuove funzionalità ed applicazioni tra le quali strumenti di diagnostica, servizi per la crittografia e la gestione dei consumi energetici.

Le specifiche di UEFI sono state pensate per rendere "il nuovo BIOS" indipendente dall'hardware e per mettere a disposizione del sistema operativo servizi di boot e runtime direttamente accessibili.

Differenze tra BIOS e UEFI

UEFI supera le limitazioni del vecchio BIOS consentendo, tra l'altro:



- boot da dischi particolarmente capienti (di capacità superiore ai 2 Terabytes) utilizzando **GPT** (*GUID Partition Table*). GPT è parte integrante dello standard UEFI e porta con sé tutta una serie di novità rispetto all'utilizzo del MBR (*Master Boot Record*);
- architettura indipendente da CPU e driver;
- ambiente preOS (accessibile cioè all'infuori del sistema operativo, prima della fase di boot di quest'ultimo) molto più completo e versatile rispetto al passato (si pensi che di solito UEFI offre anche il supporto della connettività di rete);
- design modulare;
- eliminazione della necessità di un *bootloader* (fatta eccezione per utilizzi più avanzati);
- esecuzione di moduli firmati (funzionalità **Secure Boot**).

UEFI Secure Boot

La funzionalità **Secure Boot**, se attivata (lo è, di solito, in maniera predefinita su UEFI), s'incarica di prevenire il caricamento dei driver o dei loader per il sistema operativo che non sono firmati utilizzando una firma digitale valida e riconosciuta (non presente nel database delle firme conservate nel firmware della scheda madre).

Il Secure Boot mira in primis a bloccare malware che tentassero di prendere possesso del sistema nella fase pre-boot, ma **potrebbe costituire un problema se si volesse installare una distribuzione Linux o anche lo stesso Windows**. Va detto, comunque, che alcune tra le principali distribuzioni Linux possono godere di una firma digitale valida, riconosciuta da Secure Boot.

Nel caso in cui non fosse possibile effettuare il boot dal supporto d'installazione di una distribuzione Linux, di Windows o di altri strumenti software, si potrà agevolmente **disabilitare Secure Boot da UEFI**.

L'interprete dei comandi (shell)

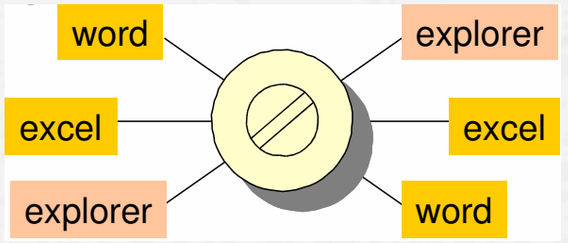
E' la componente del SO che si occupa di **acquisire l'input dell'utente e di interpretarlo**.

Esempio: doppio click sull'icona di un'applicazione.

➡ *Il SO interpreta questi comandi caricando ed eseguendo l'applicazione.*

Il Multitasking

- I vecchi sistemi operativi potevano eseguire soltanto *un programma per volta*.
- I sistemi operativi moderni consentono di utilizzare *diversi programmi contemporaneamente*.

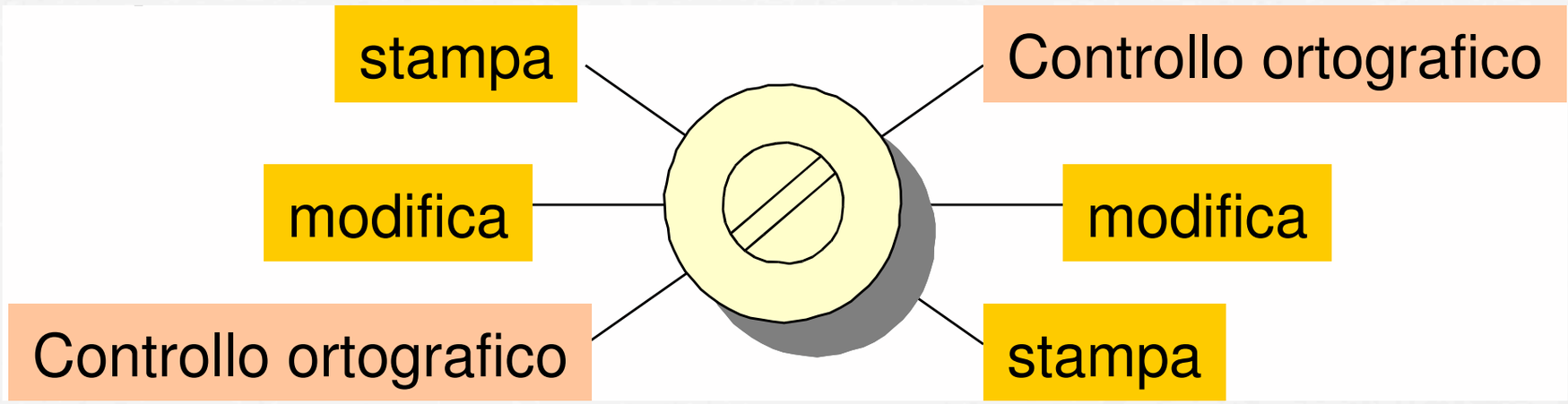


Ma come si fa con una sola CPU (o Core)?

- **Multitasking preempitive**: *l'accesso al processore* è *regolato dal SO.*

➡ Il SO memorizza tutti i dati relativi ai programmi che deve gestire.
➡ Il SO gestisce la memoria in maniera tale che ogni programma possa accedere solo alla zona di memoria che gli è stata assegnata.

Il Multithreading



I programmi, in genere, elaborano i dati e i comandi in **ordine sequenziale**.

Solo quando un'operazione è stata portata a termine possono iniziarne un'altra.

In un SO **multithreading** i programmi applicativi possono svolgere più operazioni *(thread)* parallelamente.

Il Multithreading

Un **thread** (abbreviazione di *thread of execution*, filo di esecuzione) è l'unità granulare in cui un processo può essere suddiviso.

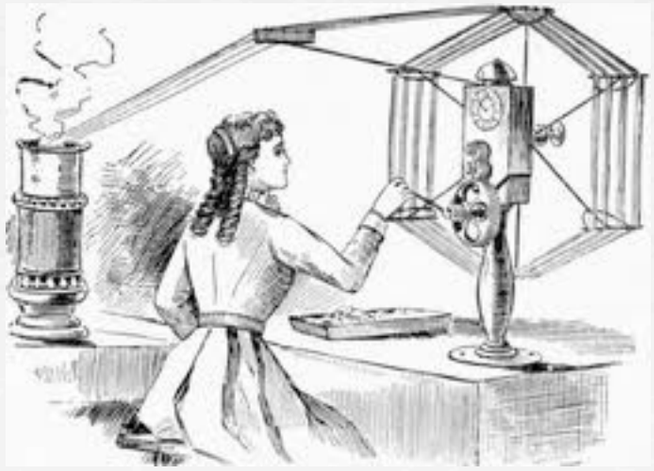
Un processo ha sempre almeno un thread (sé stesso), ma in alcuni casi un processo può avere più thread che vengono eseguiti in parallelo.

Mentre i processi sono fra loro **indipendenti** (utilizzano diverse aree di memoria ed interagiscono solo mediante appositi meccanismi di comunicazione) i thread **condividono** le medesime informazioni di stato, la memoria ed altre risorse di sistema.

Mentre la creazione di un nuovo processo è sempre onerosa per il sistema in quanto devono essere allocate le risorse necessarie alla sua esecuzione, il thread è parte del processo, e quindi una sua nuova attivazione viene effettuata in tempi ridottissimi.

Fiber vs Thread

- Un modo alternativo di scomporre un processo in attività elementari è quello che fa uso di **fiber** (fibra).
- Come i thread, i fiber condividono uno spazio di indirizzi comune (quello del processo) ma utilizzano il **multitasking cooperativo**, mentre i thread implementano il multitasking preemptive.
- I thread dipendono dallo scheduler dei thread del kernel per interrompere un thread occupato e ripristinarne un altro.
- I fiber "passano" autonomamente l'esecuzione ad un altro fiber e non attraverso lo scheduler.
- I fiber possono quindi essere considerati come dei thread nello spazio dell'utente e non del sistema operativo.



Il Multiprocessing

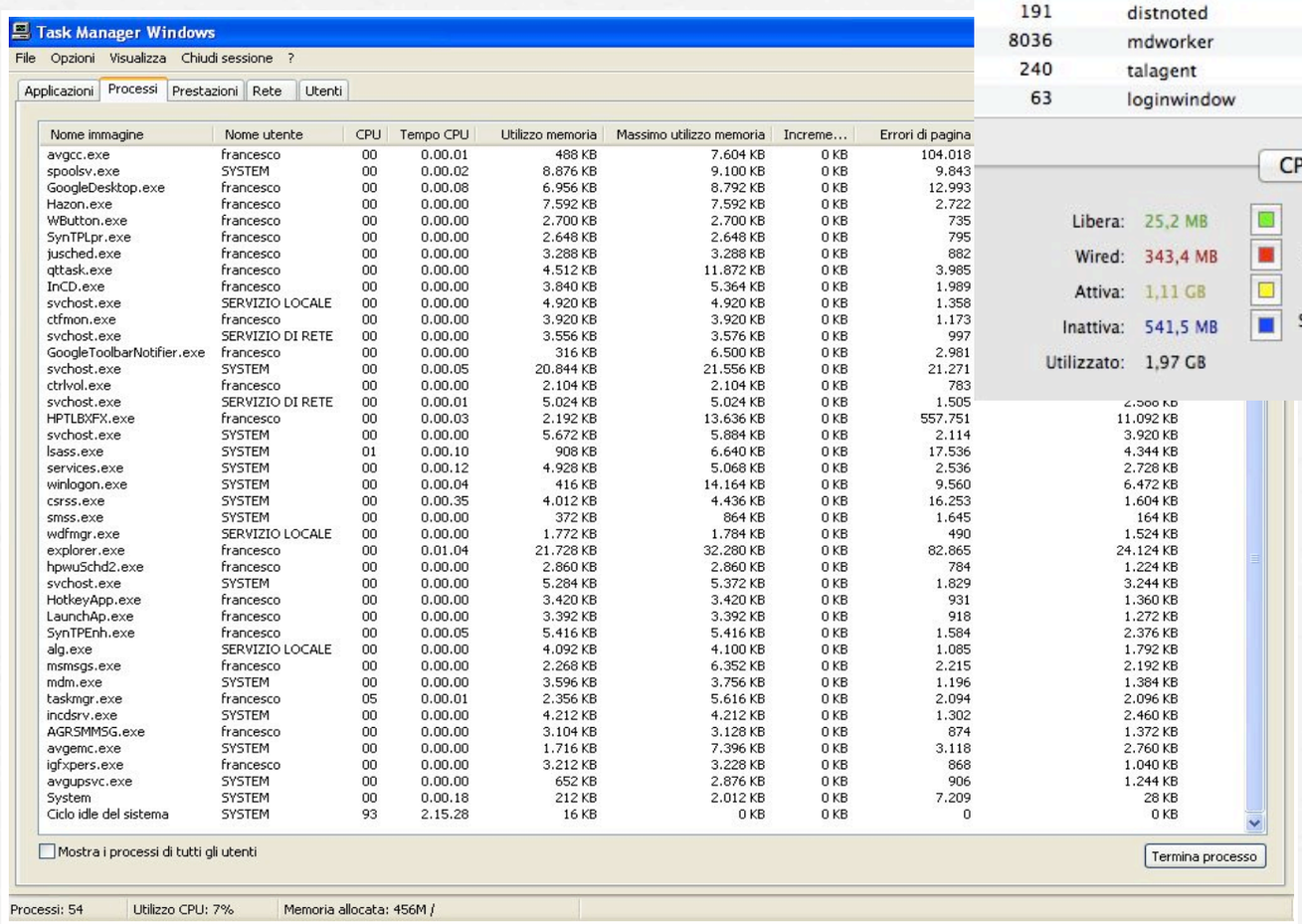
Il **multiprocessing** (*multielaborazione*) indica la presenza fisica di più processori nello stesso sistema di elaborazione.

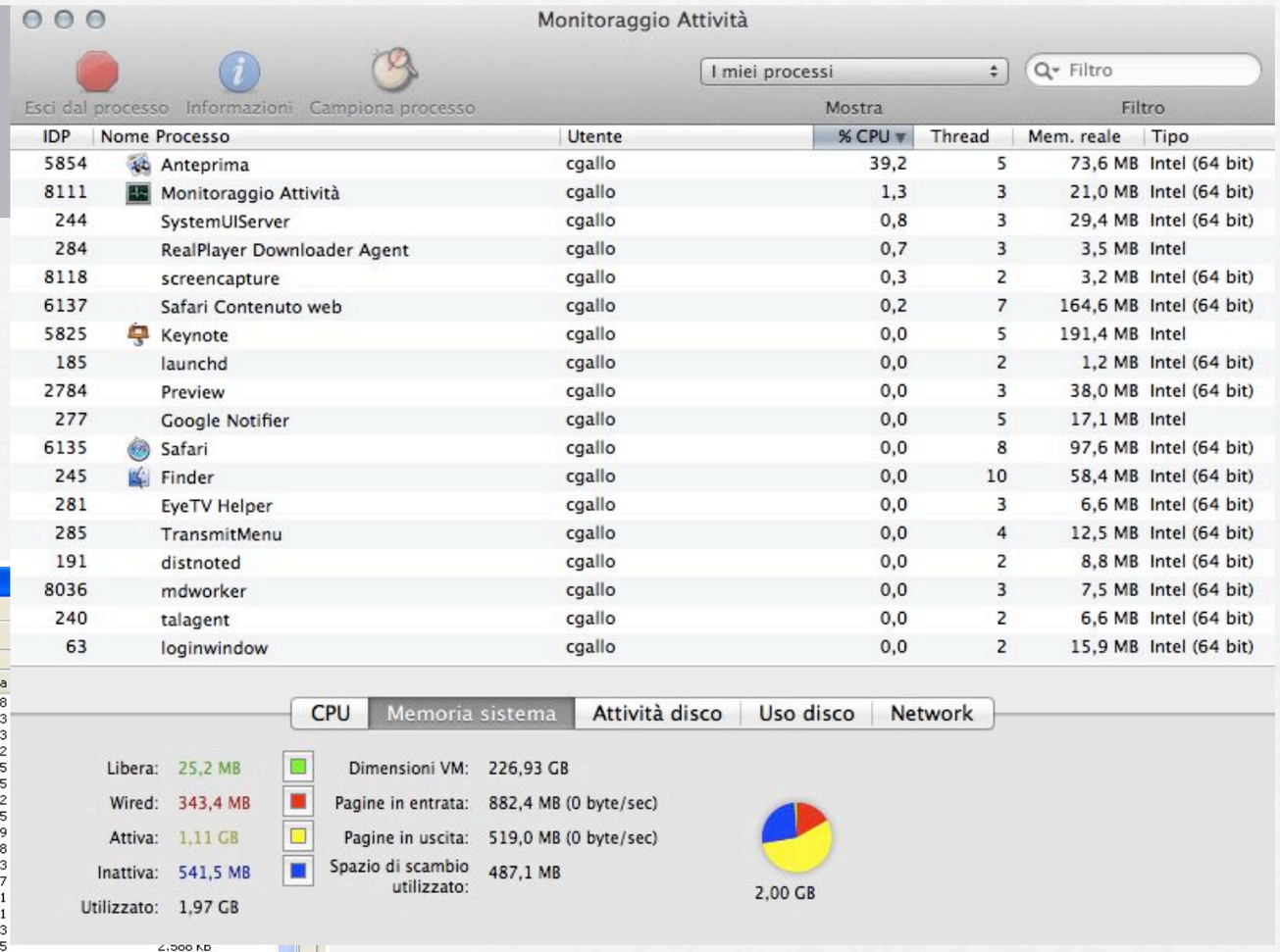
Il Sistema Operativo tipicamente gestisce il multiprocessing in due modalità:

1. **Symmetric Multiprocessing (SMP)**
   - ogni processore può eseguire un processo diverso (e relativi eventuali thread)
   - ogni CPU condivide le risorse del sistema
2. **NUMA** (*non-uniform memory access*)
   - ogni processore ha le sue risorse "private"
   - un processore può accedere rapidamente alla propria memoria locale, più lentamente alle memorie degli altri processori o alla memoria condivisa

Il Task Manager

Nei Sistemi Operativi i processi in esecuzione possono essere visualizzati per mezzo di opportune utilities.





Il "monitoraggio attività" di Mac OS X

La "gestione attività" di Windows

Il Time-sharing

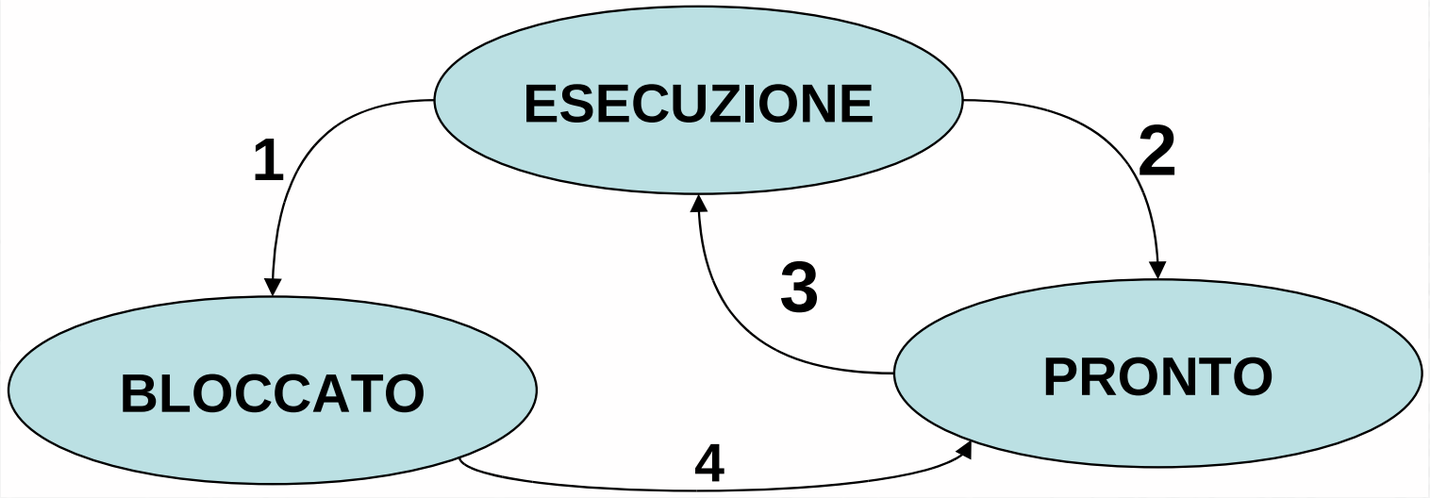
Il **time-sharing** consente a più utenti di usare la stessa risorsa di calcolo (CPU) contemporaneamente e indipendentemente l'uno dall'altro.

Il time-sharing è stato reso possibile dal fatto che la velocità di calcolo della CPU è stata fin dagli inizi molto maggiore dei tempi di reazione dell'uomo.

La problematica principale del time-sharing è data dalla suddivisione del tempo di calcolo tra i vari utenti.

La gestione dei processi

- Nel **time-sharing** il tempo macchina della CPU viene discretizzato: in pratica, la risorsa tempo viene suddivisa in parti uguali chiamate **ticks** (istanti).
- Ogni processo viene eseguito per un certo numero di ticks e poi messo in attesa.
- Eventuale gestione statica delle priorità (**real time**).



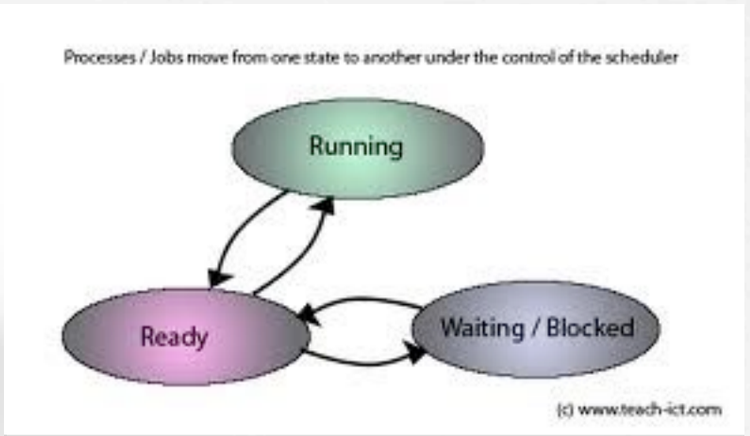
1. Il processo in esecuzione viene bloccato perché ha richiesto un I/O (**WAIT**)
2. Viene scelto un altro processo pronto da mandare in esecuzione
3. Il processo viene eseguito (**RUN**)
4. L'attesa per I/O è terminata: il processo ritorna in stato di pronto (**READY**)

Lo scheduler

Lo **scheduler** gestisce l'assegnazione della CPU ai processi in esecuzione.

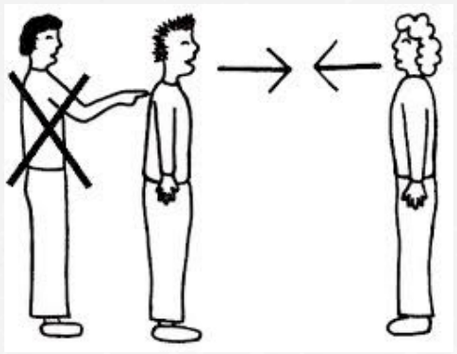
La transizione tra i processi (materialmente effettuata dal **dispatcher**) è possibile tramite un meccanismo hardware chiamato **interruzione** (interrupt).

Il meccanismo degli interrupt è per l'appunto quello che consente di interrompere un processo in esecuzione per l'avvio di un altro processo.



Le interruzioni

- Il meccanismo delle interruzioni è implementato in hardware.
- Quando arriva un'interruzione la CPU salva in maniera automatica le parti essenziali del programma in esecuzione e fa partire lo scheduler che poi porta a termine il lavoro.
- Le **interruzioni** possono essere di due tipi.
  - **Software** (o interne): *generate all'interno del processore da una particolare istruzione (ad es. una chiamata di sistema)*
  - **Hardware** (o esterne): *generate all'esterno del processore dalle periferiche (ad es. l'hard-disk comunica che ha terminata la lettura di un cluster)*



Un'interruzione può essere "interrotta" solo da una a più alta priorità.

Le interruzioni sono servite in maniera LIFO.

Tipologie di Sistemi Operativi

Riassumendo, dal punto di vista dell'elaborazione da parte del processore possiamo suddividere i Sistemi Operativi nelle seguenti tipologie.

MULTIPROGRAMMAZIONE

Ad ogni interrupt viene selezionato il processo a più elevata priorità. Assegnando priorità inferiore ai processi ad alto utilizzo della CPU (che tendono a monopolizzarla) e priorità superiore a quelli con più operazioni di input/output (che impegnano la CPU per tempi molto brevi) si ottiene una equilibrata ripartizione della CPU tra i vari processi.

TIME SHARING (Partizione di tempo)

Tutti i processi utente hanno la stessa priorità. Ad intervalli regolari viene generato un interrupt (oltre a quelli generati dalle operazioni di Input/Output). Ad ogni interrupt viene selezionato il "prossimo" processo pronto in una coda di tipo circolare.

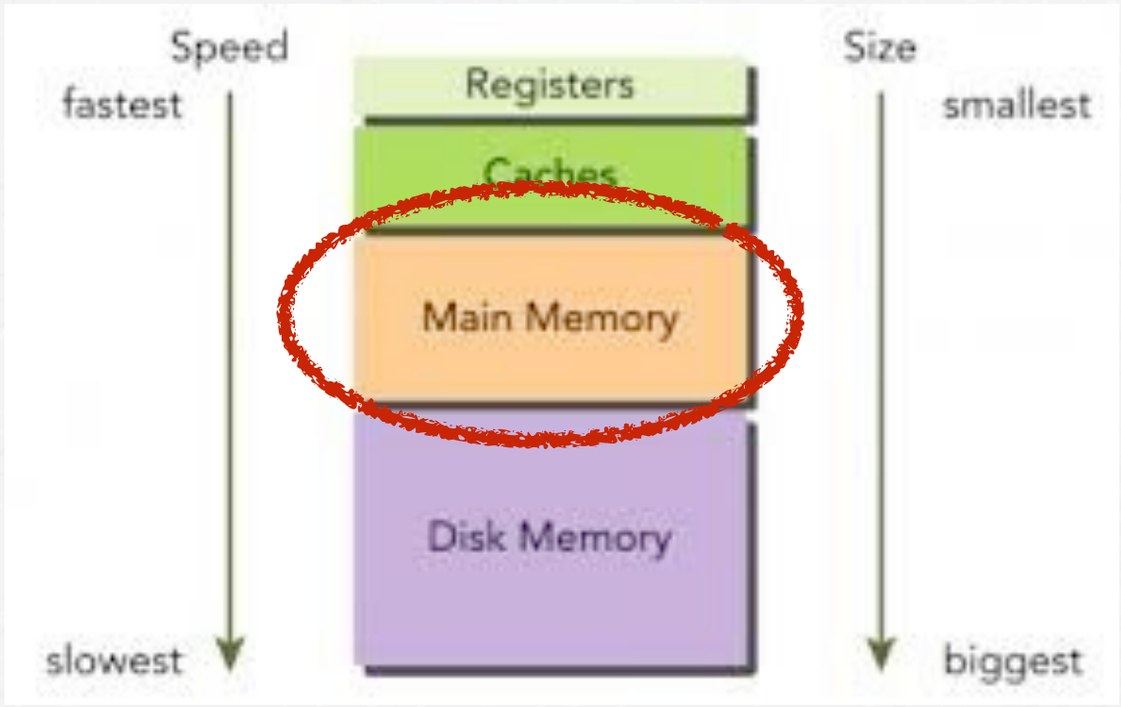
Tipologie di Sistemi Operativi

REAL-TIME

Sono sistemi operativi che consentono la gestione del tempo nel senso che consentono di determinare il tempo massimo di esecuzione di un processo, data una configurazione di processi pronti. Ad ogni processo è assegnata una priorità fissa; viene generato un interrupt ad intervalli regolari, ed i processi possono generare interrupt (oltre a quelli di Input/Output). Ad ogni interrupt viene selezionato il processo a più alta priorità. Questo consente di terminare il processo a più alta priorità al massimo entro il suo tempo di esecuzione più l'intervallo massimo intercorrente tra due interrupt.

N.B.: *I sistemi operativi real-time sono adatti alla gestione del tempo. Essi rispondono alle sollecitazioni di input (generalmente segnali provenienti da strumentazione o da intervento umano) in tempo utile. Il "tempo reale" non significa velocemente o immediatamente. Spesso, sistemi colloquiali o interattivi - che rispondono agli input - vengono erroneamente indicati come sistemi in tempo reale.*

Gestione della memoria centrale



La parte del SO che si occupa della **gestione della memoria** è il *memory manager*.

Il suo **compito** è:

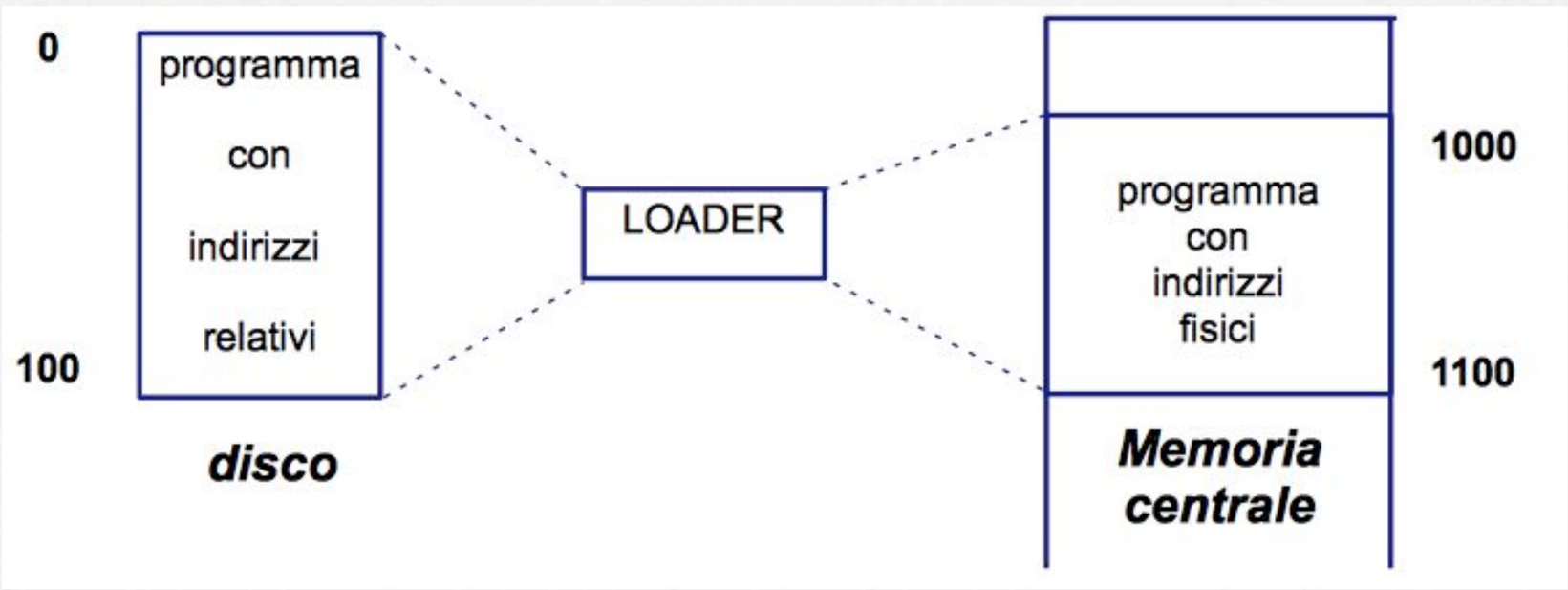
- *conoscere*, ad ogni istante, quali parti della memoria sono in uso e quali no;
- *allocare* memoria ai processi quando ne hanno bisogno;
- *deallocare* la memoria dei processi che hanno terminato.

Gestione della memoria centrale

Il gestore della memoria implementa meccanismi di:

- **partizionamento fisso** (memoria centrale suddivisa a priori) o **variabile**;
- **segmentazione** e **paginazione**;
- **memoria virtuale** (evoluzione del precedente);
- **protezione**, cioè impedire che un programma possa accedere ad uno spazio di memoria esterno a quello che gli è stato assegnato.

Al momento dell'esecuzione il programma viene caricato nella memoria centrale da un modulo del S.O., il **loader**, che trasforma gli indirizzi relativi in indirizzi fisici (rilocazione).



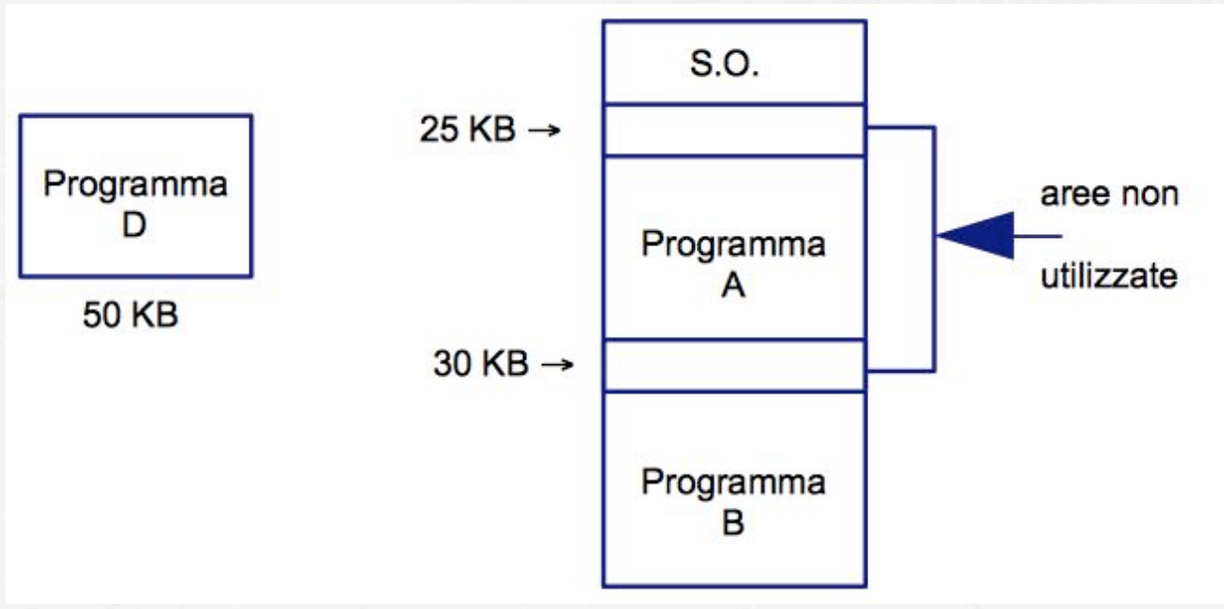
Per eseguire un processo, il S.O. deve trovare una zona di memoria libera e di dimensioni sufficienti.

La frammentazione

Il problema della **frammentazione** della memoria centrale sorge nel momento in cui un processo richiede l'uso della memoria e non esiste una zona libera di dimensioni identiche: ciò implica che si hanno delle aree residue (frammenti) non utilizzate a causa della loro piccola dimensione.

Esso è meno presente nel partizionamento fisso, mentre con il partizionamento variabile si possono avere frammenti anche di piccolissime dimensioni.

Esempio: Il programma D di 50KB non può essere caricato in memoria, e quindi i 55KB di memoria libera non sono utilizzabili.



Vediamo alcune possibili soluzioni al problema della frammentazione...

La segmentazione

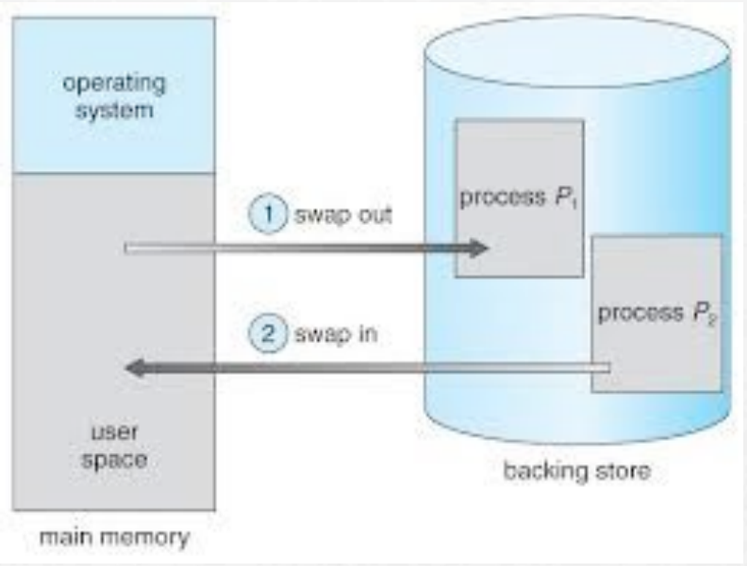
Mediante la **segmentazione** si suddivide il programma in parti (segmenti) di grandezza uguale ai frammenti di memoria disponibili (l'operazione di traduzione di un indirizzo relativo in indirizzo fisico diventa naturalmente più complessa).

Si può anche caricare solo un segmento del programma in memoria centrale e caricare gli altri quando è necessario (ossia solo quando si fa riferimento ad un indirizzo di un altro segmento).

Lo swapping

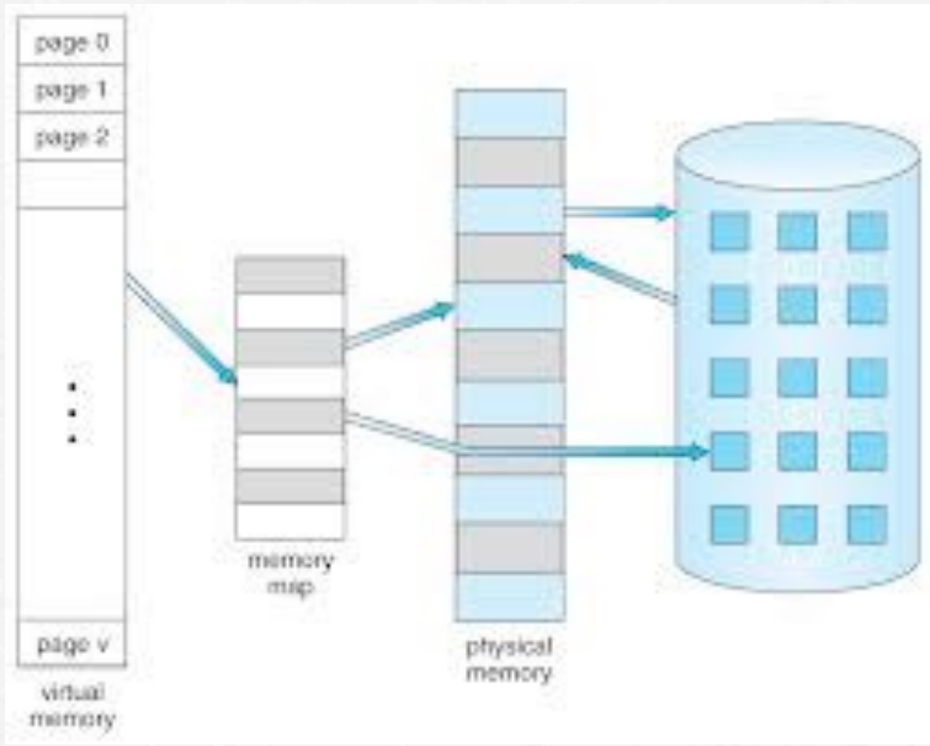
Nello **swapping** si usa la memoria di massa per memorizzare temporaneamente lo stato in memoria di processi (o di segmenti relativi a processi) non in esecuzione, e ripristinarlo quando necessario.

Le prestazioni del sistema in termini di tempi di risposta sono influenzate negativamente a causa delle onerose operazioni di salvataggio e ripristino su/da disco.



Questa tecnica consente anche di realizzare la gestione della memoria mediante **overlay**, che consiste nella suddivisione di un programma in una piccola parte fissa (root) sempre presente in memoria, mentre gli altri segmenti del programma vengono caricati alternativamente all'occorrenza in una stessa area libera di memoria (detta appunto "area di overlay").

La paginazione



Nella **paginazione** la memoria centrale viene suddivisa in parti relativamente piccole (di alcuni KByte, dette "pagine fisiche") tutte della stessa dimensione, e i programmi sono suddivisi in parti della stessa dimensione (pagine logiche) che vengono caricate nelle pagine fisiche libere.

Si evita in tal modo lo spreco di memoria dovuto alla frammentazione, riducendolo alla sola parte relativa a pagine riempite solo parzialmente (ultima pagina di ogni programma).

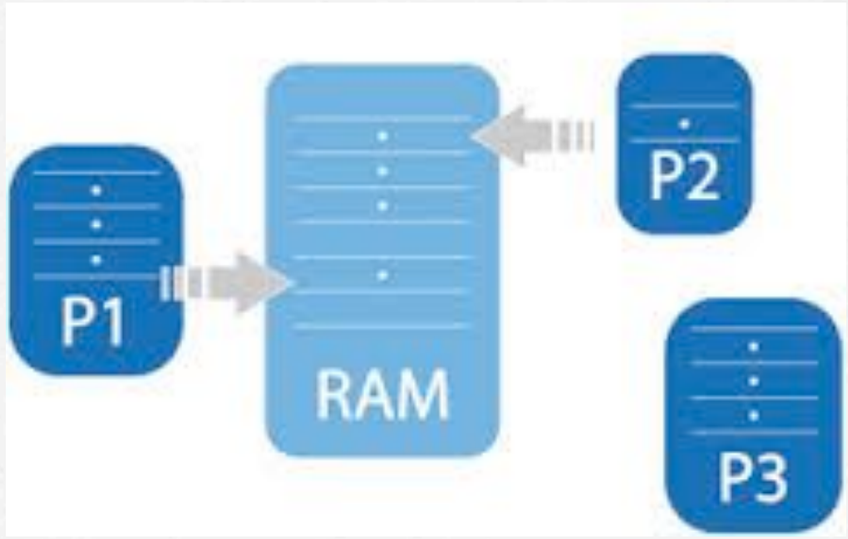
La memoria virtuale

- Il S.O. tenta di tenere in RAM più processi possibile per aumentare la multi-programmazione.
- Tuttavia, per una certa quantità di RAM disponibile in un sistema, il numero di processi che vi possono stare dipende dalla loro dimensione.
- Può accadere che la quantità di memoria disponibile sia minore di quella richiesta dai processi in esecuzione in quel momento.
- La Memoria Virtuale è un insieme di tecniche per permettere l'esecuzione di processi la cui immagine (codice, dati) non è completamente nella memoria RAM.

La memoria virtuale

L'idea è quella di tenere in memoria solo le parti dei programmi (pagine) effettivamente utilizzate in quel momento, mentre le altre vengono spostate sull'hard disk (in Windows nel PAGEFILE.SYS) e caricate in memoria in memoria solo quando sono effettivamente richieste.

La memoria virtuale utilizza uno **spazio di indirizzamento virtuale** (segmento-pagina-offset).



La memoria virtuale

I criteri di scelta delle pagine da accantonare sono basati sulla probabilità di successiva richiesta della pagina, calcolata dalla storia del suo utilizzo e considerazioni collaterali (principi di **località** spaziale e temporale).

L'algoritmo **LRU** (Least Recently Used) fa uso di Reference Bit e Change Bit per "scaricare" prima le pagine non utilizzate e/o non modificate (che non necessitano di scrittura su disco).

Terminologia

page fault: mancanza in memoria di una pagina referenziata;

page in: caricamento in memoria dal (l'area di swap del) disco di una pagina;

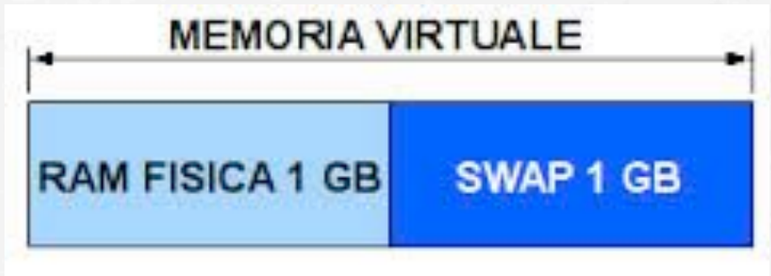
page out: rimozione di una pagina dalla memoria (previo salvataggio su disco).

La memoria virtuale

Il risultato è che la memoria di massa viene utilizzata per procurare un'espansione virtuale della memoria centrale.

Naturalmente le prestazioni del sistema decrescono per le operazioni di salvataggio/ripristino delle pagine.

A titolo indicativo la memoria virtuale non deve superare più del 50% la memoria reale, per ottenere prestazioni accettabili.

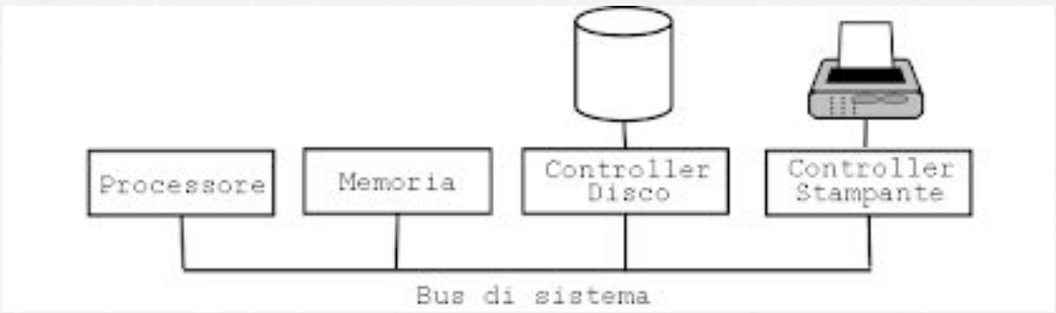


La gestione dell'I/O

Il terzo livello del S.O. consente di trattare le unità di ingresso/uscita (I/O) in modo semplice, nascondendo i dettagli della loro struttura fisica. Ciò viene ottenuto mettendo a disposizione dei vari livelli superiori una serie di comandi di I/O ad alto livello.

In genere un comando di questo tipo conterrà le seguenti informazioni:

- la zona di memoria centrale interessata all'operazione;
- il nome dell'unità di I/O utilizzata;
- eventuali parametri necessari alla gestione dell'unità di I/O (esempio: settore, traccia e cilindro per un disco rigido).

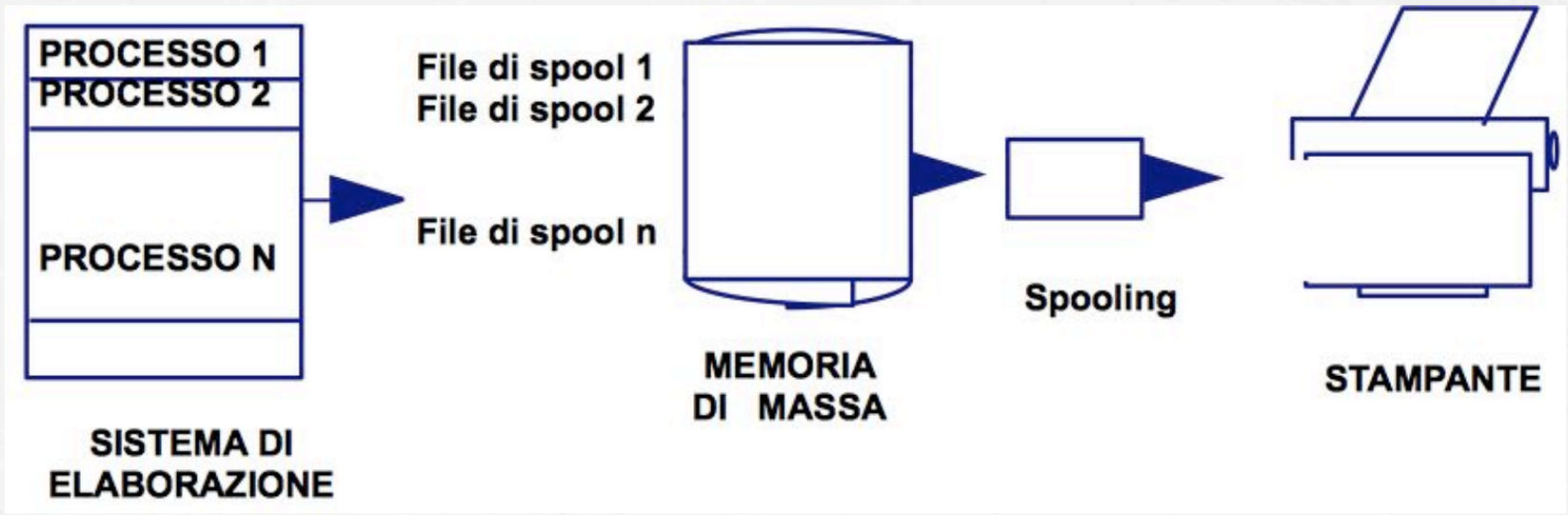


La gestione dell'I/O - SPOOL

Al termine dell'esecuzione del comando, il S.O. informa l'utente sull'esito dell'operazione (esito positivo, errore di trasmissione, unità fuori servizio, ecc.).

Una funzione importante realizzata a questo livello è lo **SPOOL** (Synchronous Peripheral Operations On Line), che consente di evitare che dati inviati dai vari processi ad unità di uscita (ad esempio la stampante) vengano forniti all'ambiente esterno nello stesso ordine in cui vengono eseguiti i comandi d'uscita.

In questo caso, infatti, i dati d'uscita relativi ai vari processi verrebbero mescolati in modo praticamente casuale.

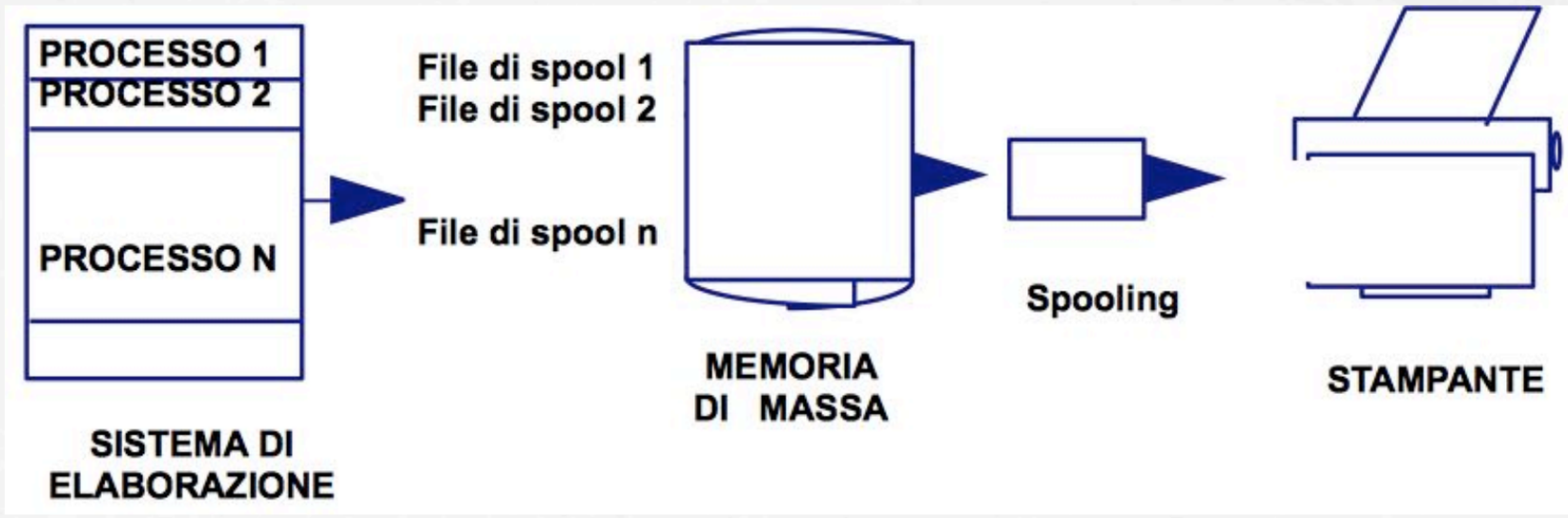


La gestione dell'I/O

Lo SPOOL utilizza la memoria a disco per tenere separati i dati di uscita relativi a processi diversi.

Solo quando un processo è terminato, e quindi i suoi dati di uscita sono stati tutti prodotti, si ha il trasferimento al dispositivo (ad es. la stampante).

È anche teoricamente possibile gestire mediante lo SPOOL dispositivi di ingresso non condivisibili (ad es. uno scanner che deve acquisire pagine per processi e/o utenti differenti).



Il File Management System

- Le unità di memoria di massa forniscono il supporto fisico per la memorizzazione permanente dei dati, e presentano *caratteristiche estremamente diverse* a seconda della casa costruttrice e del tipo di unità.
- Il **File System** offre una *visione logica uniforme* della memorizzazione dei dati basata su un'unità di memoria logica, il file, definita indipendentemente dalle caratteristiche fisiche delle particolari unità.
- Il **file** è un insieme di informazioni, registrate nella memoria di massa, identificato da un nome, eventualmente seguito da un'estensione (che denota il tipo del file).

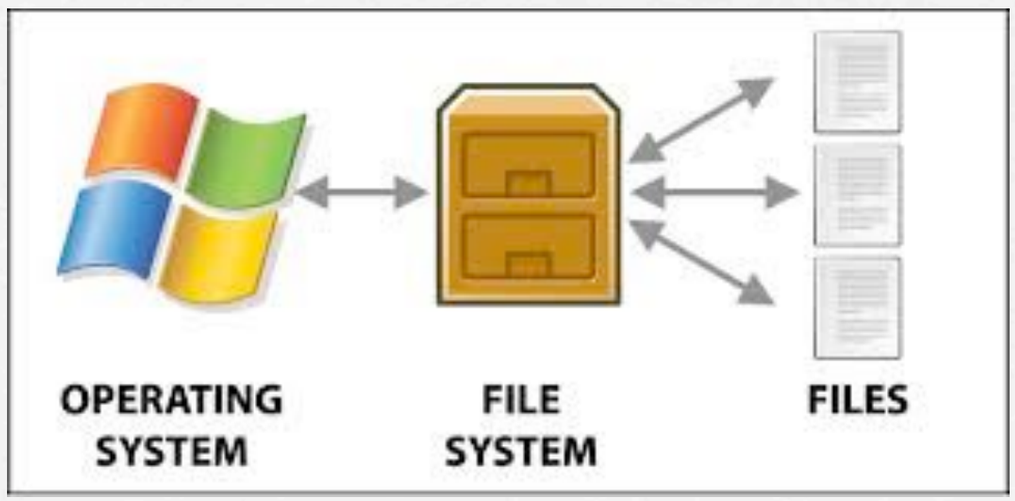
Dal punto di vista dell'utente, un file è la più piccola porzione (logica) di memoria secondaria: i dati, cioè, possono essere scritti nella memoria secondaria solo all'interno di un file.

Il File Management System

Il **File System** è quindi l'organizzazione logica e fisica degli archivi sui supporti di memoria esterna, così come il sistema operativo li struttura.

L'insieme dei programmi del S.O. che gestisce i file è detto **File Management System**. Sue tipiche funzioni sono:

- creazione e cancellazione dei file;
- accesso ai file (lettura e scrittura);
- gestione dello spazio della memoria di massa (nastro/disco);
- protezione dei file da accessi non autorizzati;
- copia, cambio nome, spostamento, etc.



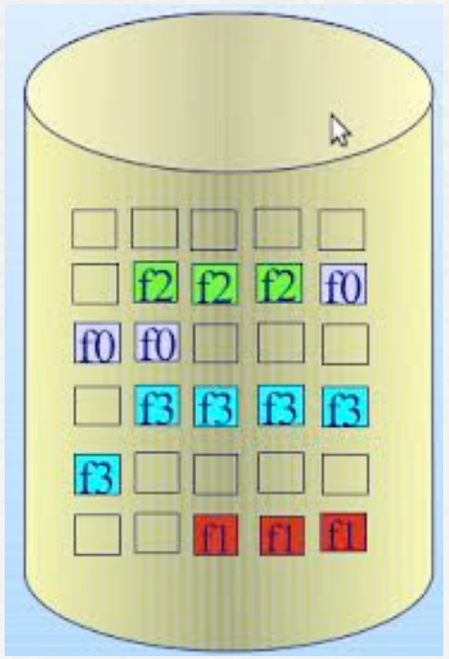
Il File Management System

I file system possono essere classificati, secondo il modo in cui lo spazio sulle memorie di massa viene assegnato ai file, in file system ad allocazione contigua e sparsa.

In un file system ad **allocazione contigua** l'intero file viene registrato (tipicamente su disco) in un blocco unico, cioè i suoi record si succedono in maniera fisicamente consecutiva. Ciò semplifica notevolmente la gestione (c'è solo bisogno di tenere traccia della posizione iniziale del file e della sua lunghezza), e velocizza l'accesso (dato il numero r del record cui accedere, l'ampiezza d in byte del singolo record - essendo tutti della stessa lunghezza - e la posizione iniziale i_0 del file sul supporto, è semplice calcolare la posizione x occupata dal record come: $x = i_0 + (r\text{-}1)\cdot d$.

Lo svantaggio fondamentale consiste nell'impossibilità di espandere un file, qualora subito dopo di esso lo spazio sia occupato da un altro file: in tal caso è necessario effettuare la **compattazione** del disco, spostando verso l'inizio o la fine in maniera opportuna i file per creare lo spazio necessario.

Tale operazione può anche durare ore, a seconda della dimensione del disco.

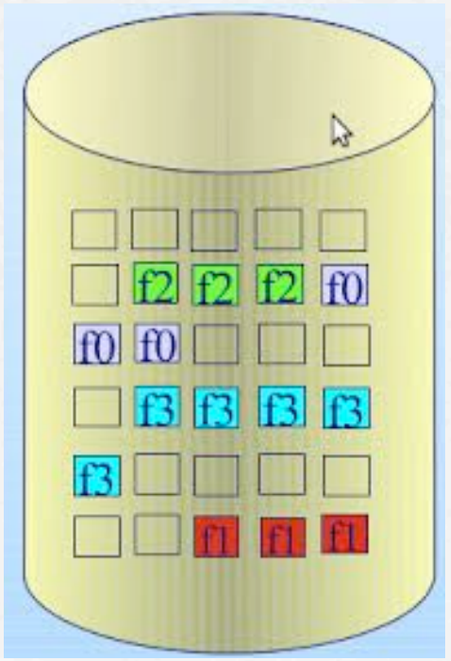


Il File Management System

Un file system ad **allocazione sparsa** assegna lo spazio ai file in parti fisicamente contigue (dette estensioni) man mano che occorre: in tal modo non c'è più l'inconveniente della compattazione per espandere un file che ha raggiunto la sua massima estensione poiché viene semplicemente allocato un altro blocco per il file in un'altra zona libera del disco.

Ciò però complica la gestione, perché il sistema deve tenere traccia di tutte le estensioni di ogni file (cioè dove iniziano e quanto sono lunghe), nonché concatenarle logicamente tra di loro per poter ricercare un qualunque record.

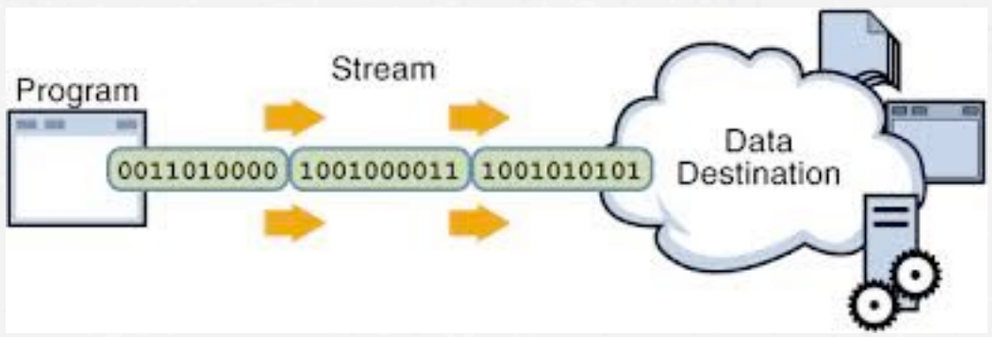
Tale ricerca, inoltre, non può più essere immediata (come nel caso dell'allocazione contigua), bensì va prima determinata l'estensione contenente il record cercato e quindi è possibile (all'interno del blocco contiguo rappresentato dall'estensione individuata) la ricerca diretta.



Il File Management System

Dal punto di vista della **strutturazione logica** interna dei file, i file system possono essere di tipo:

- **byte-stream** (flusso di byte): il file è visto dal sistema semplicemente come un flusso continuo di byte, senza alcuna strutturazione logica. È quindi a carico dell'utente (cioè dei programmi applicativi) accedere al record desiderato specificandone la posizione in byte relativamente all'inizio del file. I sistemi operativi UNIX-like hanno file system di questo tipo
- **record-oriented** (orientati al record): il sistema conosce la strutturazione in record di un file, ma solamente per quanto riguarda la lunghezza in byte dei record che lo compongono. È quindi possibile, per le applicazioni, accedere ad un record qualunque semplicemente specificandone il numero, e non più la distanza in byte relativamente all'inizio del file. Tale tipo di file system è caratteristico dei minicomputer di fascia media (midrange)



Il File Management System

database (o anche field-oriented, orientato ai campi): il sistema è a conoscenza non solo della struttura in record dei file, ma anche della struttura interna dei record in campi.

In tal modo tutte le informazioni sulla definizione del file sono interne al file stesso e non più di pertinenza dei programmi applicativi, che accederanno ai campi mediante il loro nome e non più mediante la posizione di un campo all'interno del record.

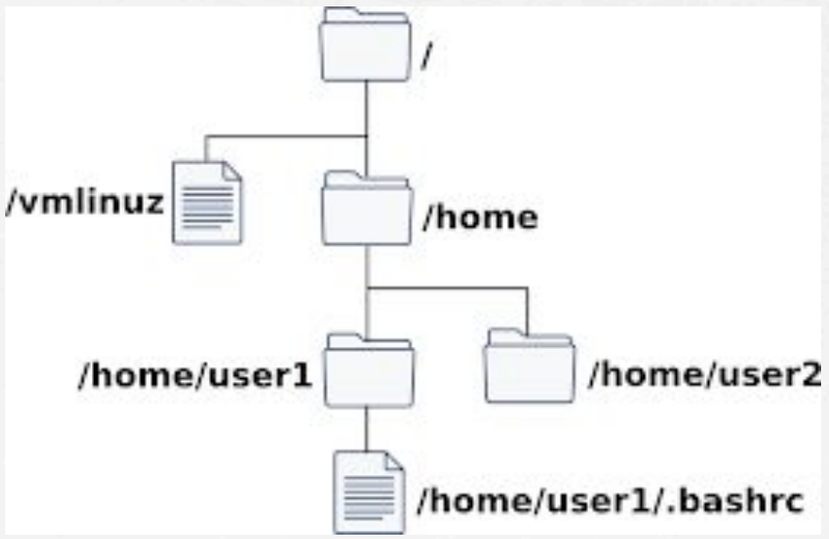
È il massimo grado di libertà e di indipendenza dalla strutturazione fisica dei dati, e permette la realizzazione nativa di organizzazioni generali e sofisticate di gestione dei dati note con il termine di database (basi di dati).

Tipici esempi di tali file system sono quelli integrati nel sistema commerciale IBM System I (ex AS/400) e delle "database machine" di Teradata; i software di gestione di database relazionali (RDBMS) più noti sono Oracle, Informix e DB2 (IBM), SQL Server (Microsoft), Sybase (SAP), MySql, PostgreSQL (questi ultimi due open source).

Il File Management System

Infine, dal punto di vista del **raggruppamento logico** i file system possono essere di tipo:

- **piatto** (flat) — i file non sono raggruppabili, ma sono tutti allo stesso livello all'interno del disco;
- **gerarchico** — in tal caso i file sono logicamente raggruppabili in directory o cartelle, che a loro volta possono contenere altri file o directory e così via, simulando l'organizzazione in classificatori/cartelle dei documenti/file in un ufficio.



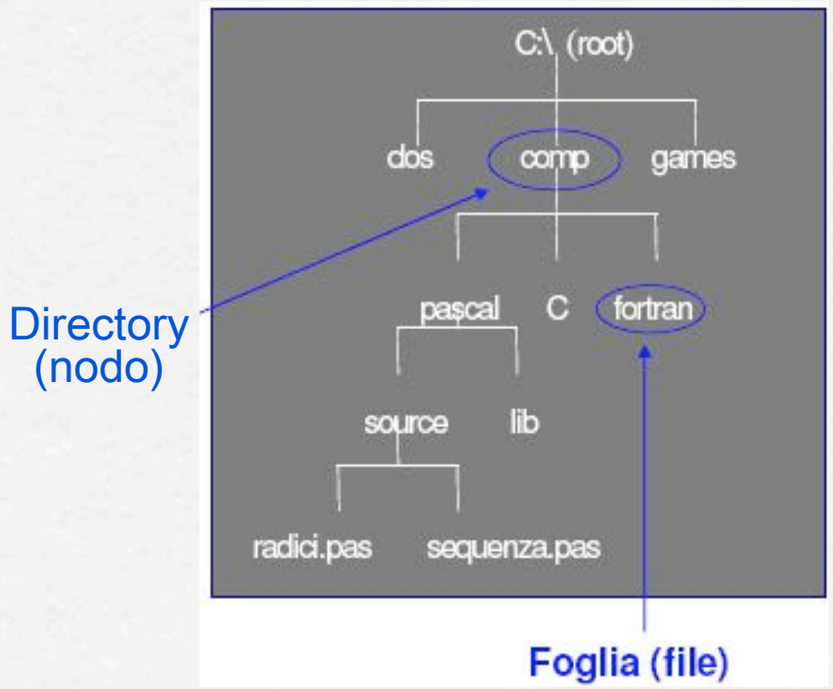
Il File Management System

Le Directory (o Cartelle)

- Le directory sono anch'esse dei file.
- Una directory può contenere anche delle directory.
- Una directory può essere vista come una tabella.
- Le directory consentono di raggruppare tra loro i file.

Organizzazione Logica dei File

- L'insieme dei file presenti in memoria di massa è organizzato secondo una struttura gerarchica ad albero, in cui i nodi intermedi costituiscono le directory (che raggruppano altri files e directory secondo un criterio di omogeneità), mentre le foglie rappresentano i file.
- All'interno di tale struttura, un particolare file è univocamente identificato dal path (o percorso) che localizza la directory in cui il file è memorizzato.



Il File Management System

La Gestione dei File

- ✓ I file presenti su una periferica possono essere cancellati, aumentati (o diminuiti) in termini di dimensioni, creati, etc.
- ✓ Come fa un file system a gestire lo spazio di memoria presente sulle periferiche per soddisfare queste esigenze?

Organizzazione Fisica dei File su Disco

Da un punto di vista fisico, la registrazione del file sul disco viene realizzata dal sistema operativo disponendo il contenuto del file su un insieme di cluster possibilmente contigui.

La registrazione dei dati è organizzata in maniera sequenziale, per cui le operazioni di lettura e scrittura possono avvenire solo a partire dall'inizio e procedendo verso la fine.

Il File Management System

La deframmentazione

- ✓ riorganizza i cluster del disco per rendere contigui quelli appartenenti allo stesso file
- ✓ migliora l'efficienza del disco
- ✓ In Windows XP è l'utility DEFRAG, in Win7/8/10 e Mac OS X è automatica.

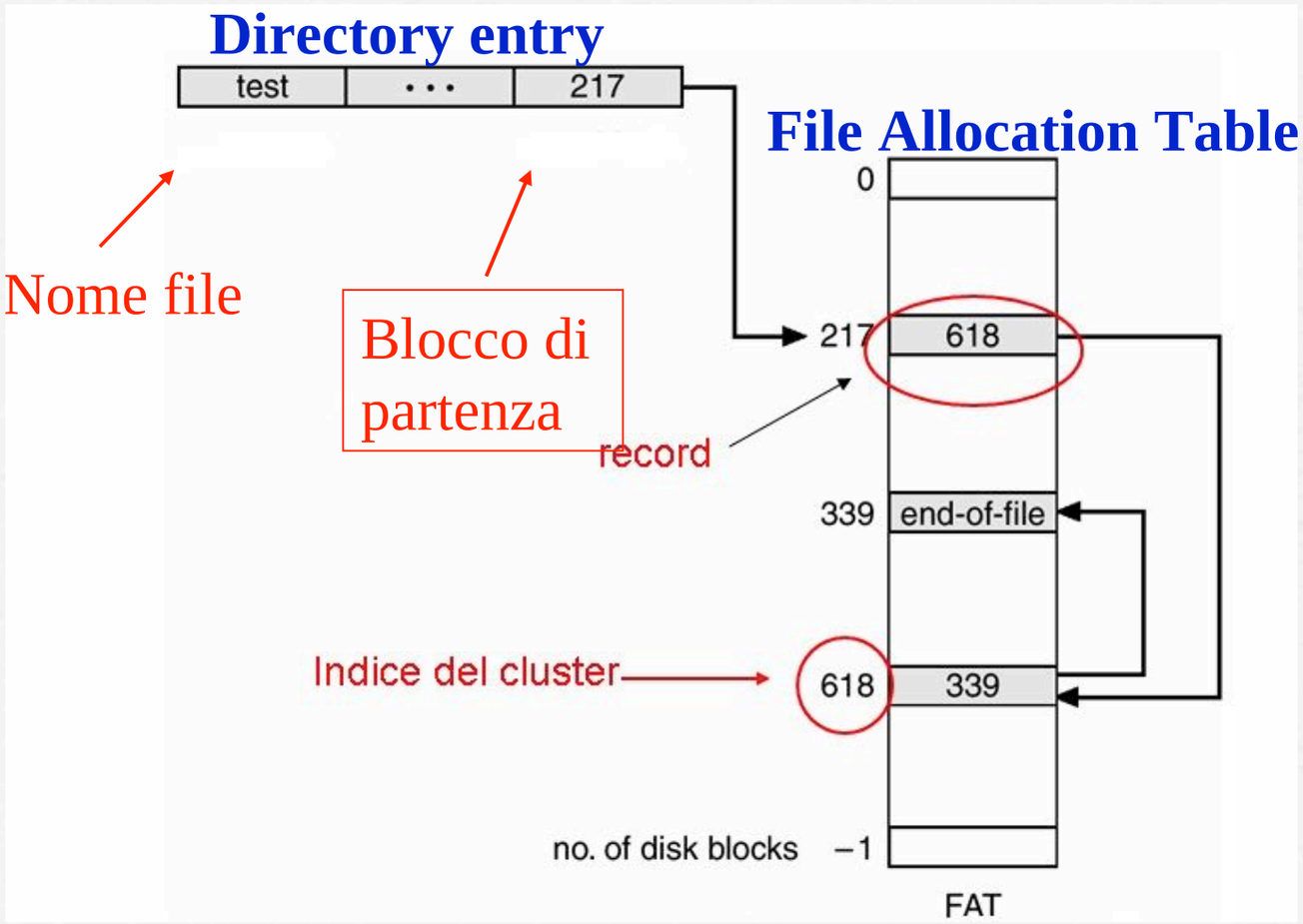
La scansione del disco

- ✓ In Windows è l'utilità di sistema SCANDISK, in Mac OS X è l'Utility Disco.
- ✓ Controlla gli errori in file e cartelle
- ✓ Controlla gli errori fisici sulla superficie del disco
- ✓ Tenta di riparare gli eventuali errori trovati

Tipi di File System — FAT

File Allocation Table (FAT)

È comparsa inizialmente con il S.O. MsDos; attualmente nella versione FAT32 (32 bit di indirizzamento) è utilizzata per compatibilità tra periferiche di varia natura (pendrive USB, SmartTV, …).



Tipi di File System — NTFS

NTFS (New Technology File System)

NTFS è una evoluzione della FAT, utilizzato nei sistemi Windows basati su kernel NT. È transazionale ("journaled"), con controllo degli accessi (ACL), nomi lunghi (fino a 255 caratteri Unicode), elevate dimensioni e flessibilità.

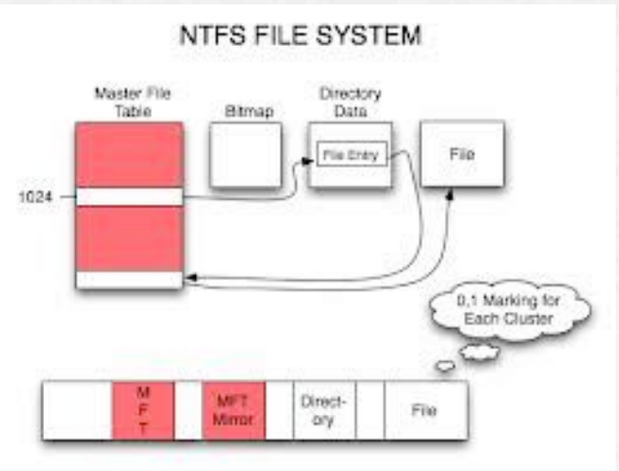
Un volume NTFS supporta fino 2^32 file per volume, ha un limite teorico di 2^64 cluster e può gestire file che raggiungono i 2^64 bytes di dimensione.

Utilizzando cluster di 64 KB la dimensione massima di un volume NTFS in un sistema Windows è di 256 TB, che si riduce a 16 TB con cluster di 4 KB. La dimensione massima di un singolo file è di 16 TB contro i 4 GB di FAT e FAT32. Sono supportati nativamente i volumi sparsi e il mirroring. Sono disponibili anche gli hardlink (come i link in Unix/Linux).

Le performance di NTFS sono invece leggermente inferiori a quella di FAT e di FAT32. A partire da Windows 2000, è inoltre possibile montare un volume NTFS come sottodirectory di un altro volume NTFS.

NTFS permette inoltre di utilizzare trasparentemente delle opzioni di compressione e di crittografia (chiamato anche EFS=Encrypting File System).

In NTFS sono stati aggiunti i cosiddetti "punti di reparse", ovvero dei meccanismi che consentono le giunzioni (junctions) tra directory, altrimenti impossibili per la struttura del file system.



Tipi di File System — HFS e HFS+

HFS (Hierarchical File System)

HFS è stato introdotto da Apple nel 1985 come nuovo file system per i computer Macintosh, in sostituzione del precedente Macintosh File System (MFS) usato dai primi modelli di Mac e che consentiva solo una struttura di memorizzazione piatta.

HFS permette nomi di file lunghi fino a 31 caratteri, l'aggiunta di metadati e la memorizzazione separata di dati e risorse riguardanti lo stesso file ("fork"). Benché HFS sia una tecnologia di tipo proprietario, è ben documentata (come NTFS) e sono disponibili diverse soluzioni per accedere a dischi formattati con HFS per la maggior parte dei sistemi operativi moderni.

Nel 1998 Apple ha introdotto **HFS Plus** (detto anche HFS+) introducendo tra le altre migliorie la transazionalità, i nomi lunghi fino a 255 caratteri, le ACL e una più efficiente allocazione dello spazio (come in NTFS). HFS è ancora utilizzabile nelle versioni correnti di Mac OS, ma a partire da Mac OS X non è più possibile avviare il sistema da un disco HFS.

HFS Plus viene inoltre utilizzato come uno dei formati da iPod e iPhone. Nella documentazione ufficiale il formato HFS Plus è denominato Mac OS Extended (Mac OS esteso).

HFS Plus usa i B*-tree (alberi binari bilanciati) per memorizzare la maggior parte dei metadati del disco.

Altri tipi di File System

- **Ext4** — Extended File System 4, rilasciato come stabile dal kernel Linux 2.6.28 (già presente dalla versione 2.6.19 come ext4dev).
- **ExFAT** — Conosciuto anche come FAT64, creato da Microsoft e pensato appositamente per memorie flash.
- **HPFS** — High Performance File System, usato su OS/2.
- **ISO 9660** — Usato su dischi CD-ROM e DVD-ROM (anche con estensioni Rock Ridge e Joliet).
- **Journaled File System** (JFS) — Disponibile su sistemi GNU/Linux, OS/2, e AIX.
- **UDF** — Universal Disk Format, file system a pacchetti usato su supporti WORM/RW, CD-RW e DVD.