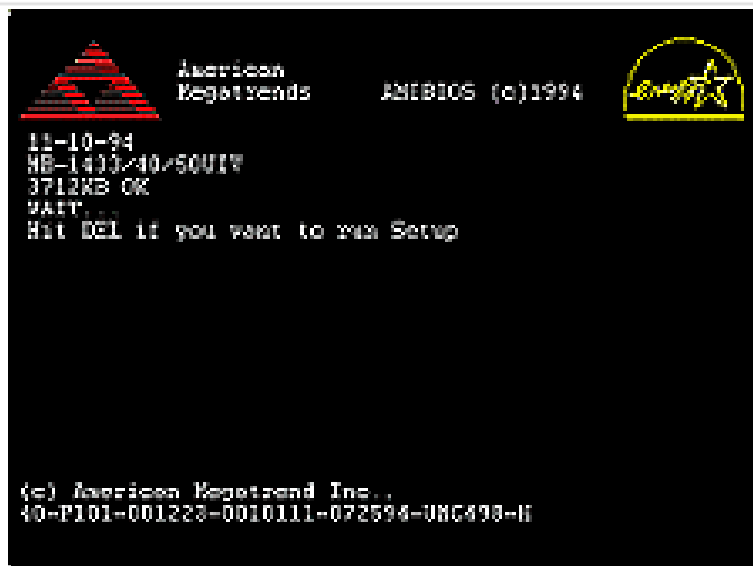

SISTEMI OPERATIVI

alla base di tutto

Sistemi Operativi: avvio

All'avvio del computer, terminate le verifiche del BIOS, il controllo passa al **sistema operativo**.



Il Sistema Operativo opera come **intermediario** tra l'hardware del sistema e uno o più utenti.

Sistema Operativo: funzioni

Due sono le funzioni principali di un sistema operativo:

- gestione delle risorse hardware
- interfaccia verso l'utente

Come tali funzioni vengano svolte, e se sia interamente il sistema operativo ad occuparsene, dipende dal dispositivo in questione.

Dispositivi

Esistono diversi dispositivi elettronici la cui complessità ha reso necessario dotarli di un sistema operativo.



Nokia 7650



Palm m505



Sendo z100

Tra questi, agende elettroniche e telefoni cellulari sono esempi comuni.

Dispositivi

La gestione dell'hardware è un aspetto di secondaria importanza per gli utenti di questi dispositivi.

L'interfaccia ha un'importanza maggiore, ed è sia **fisica** che **virtuale** (software).

Sistemi operativi: un po' di storia



Ken Thompson,
UNIX



Gary Kidall, BIOS



Dennis Ritchie, C



Bill Gates,
Windows

Sistemi operativi: storia

I primi sistemi operativi sono stati progettati negli anni '50 per i calcolatori allora disponibili.

Consistevano in poche centinaia di istruzioni per il *caricamento* del programma in memoria centrale e per la produzione, su un dispositivo di output, dei *risultati* dell'elaborazione.

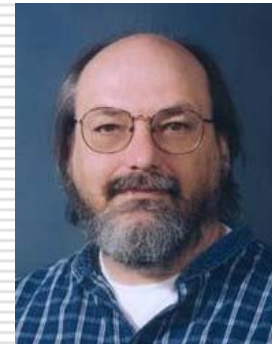
L'interfaccia era quella comune allora - interruttori e spie luminose.

I comandi venivano impartiti in **codice binario**.

Sistemi operativi: storia (UNIX)

Nella seconda metà degli anni '60, la AT&T abbandona il progetto MULTICS - un sistema a time-sharing - rivelatosi deludente.

Ken Thompson, allora ai Bell Labs della AT&T, scrive un sistema operativo in assembler e lo battezza, scherzosamente, UNICS (Uniplexed Information and Computing Services).



Nasceva così **UNIX**, capostipite di una numerosa e varia famiglia di sistemi operativi. Era il 1969.

Sistemi operativi: storia (UNIX)

Nel frattempo Dennis Ritchie, che collaborava con Thompson, sviluppa il linguaggio di programmazione "C".



Nel 1972 UNIX viene riscritto in C, aprendo la strada all'utilizzo su *altre piattaforme hardware*.

Dal 1975 in poi, UNIX inizia a diffondersi nelle università - negli Stati Uniti, in Giappone, Australia, Europa...

Sistemi operativi: storia (DOS)

Negli anni '70, iniziano a diffondersi i primi personal computer. **Gary Kildall** sviluppa, nel 1972, il CP/M (Control Program for Microprocessors), un sistema operativo funzionante su uno dei primi processori Intel (il 4004).



Il primo BIOS fu scritto da Kildall come modulo per il proprio CP/M, in modo da rendere il sistema meno dipendente dall'hardware.

Sistemi operativi: storia (DOS)

Nel 1980 Tim Patterson, della Seattle Computer Products, sviluppò un proprio sistema operativo basato sul CP/M. Impiegò due mesi, e lo battezzò QDOS (Quick and Dirty Operating System).

Dopo quattro mesi, un'altra ditta di Seattle, la Microsoft di *Bill Gates*, acquista dalla SCP i diritti per rivendere il DOS ad un cliente non menzionato.

Il cliente è l'**IBM**, che nel 1981 lancerà il primo PC, dando il via alla rivoluzione dei personal computer.

Sistemi operativi: storia (Win & Mac)

La *Apple Computer*, negli stessi anni, sta portando avanti diversi progetti, e sperimenta diversi sistemi operativi. Nel 1984, con il lancio (e il successo) dell'**Apple Macintosh**, si focalizzerà sul suo sistema operativo, il **System** - allora alla versione 1.0.

Al contrario del DOS, nelle sue varie incarnazioni, il System ha un'interfaccia grafica.

L'anno successivo, il 1985, la Microsoft lancia la prima versione di **Windows**.

Sistemi operativi: storia (Win & Mac)

Dal 1984-85 ad oggi, sia la Apple che la Microsoft hanno aggiornato più volte i propri sistemi operativi.

L'interfaccia Apple è rimasta simile (perché molto funzionale), mentre la Microsoft ha via via abbandonato il DOS, e nel 1995 ha introdotto, con Windows 95, un'interfaccia grafica più curata ed efficiente.

Sistemi operativi: storia (Linux)

Nel 1991 uno studente finlandese, **Linus Torvalds**, sviluppò il kernel per un sistema operativo basato su una variante di UNIX.

Lo distribuì in rete secondo la GNU *General Public License*, una licenza che ne consentiva l'uso, la redistribuzione e la modifica (a certe condizioni).

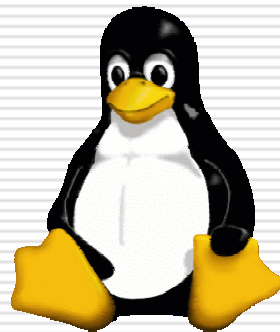


Iniziò a ricevere da subito contributi da altri sviluppatori.

Sistemi operativi: storia (Linux)

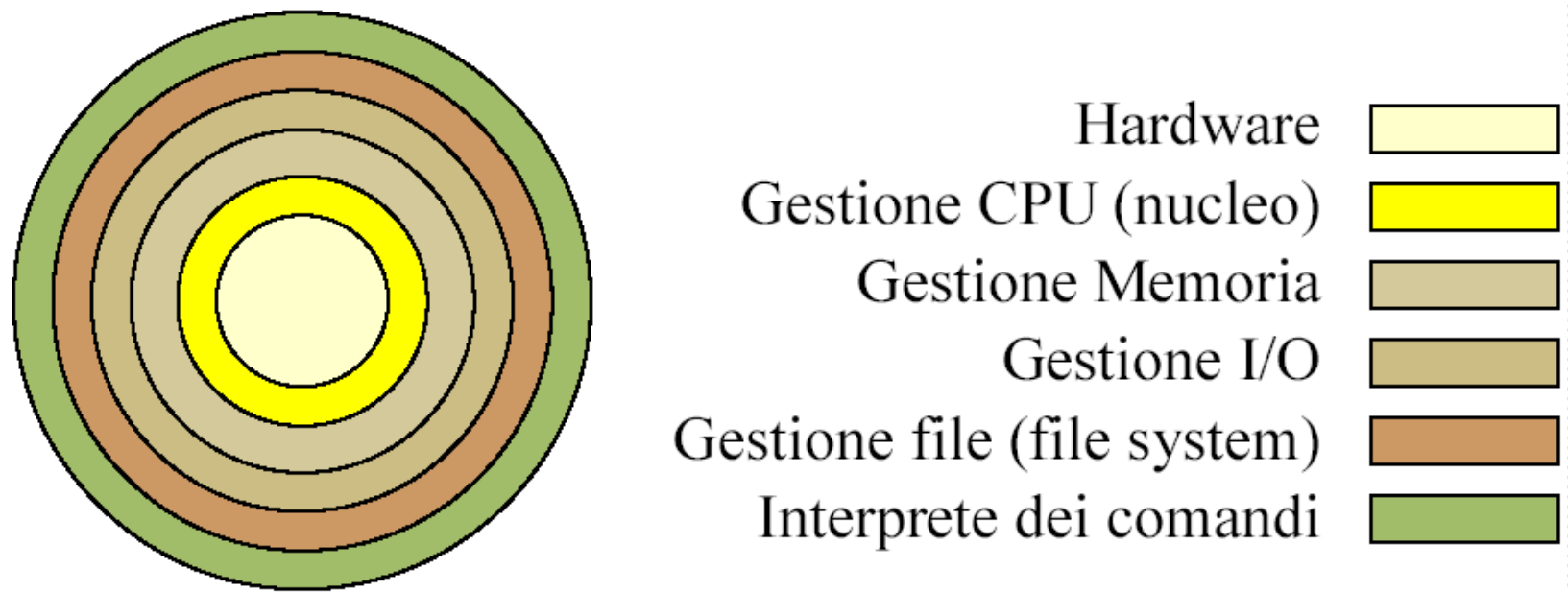
Nasceva così **Linux**, una delle varianti UNIX oggi più diffuse. Il kernel di Linux è continuamente aggiornato, e disponibile anche gratuitamente.

Uno dei punti di forza di questo sistema è la **comunità** che lo supporta, e la filosofia su cui si basa, quella del **software libero**.



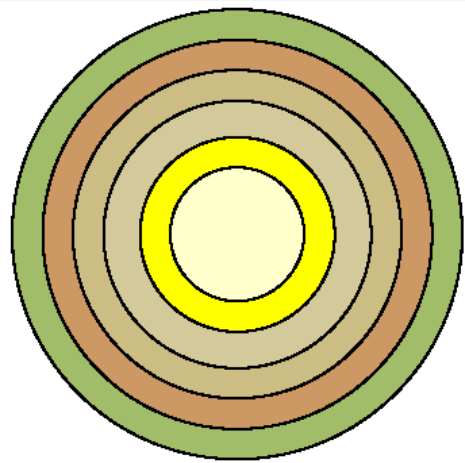
livelli

La struttura di un sistema operativo, tipicamente, è quella *a cipolla*:



livelli

L'hardware è dunque "ricoperto" da una serie di strati di software.

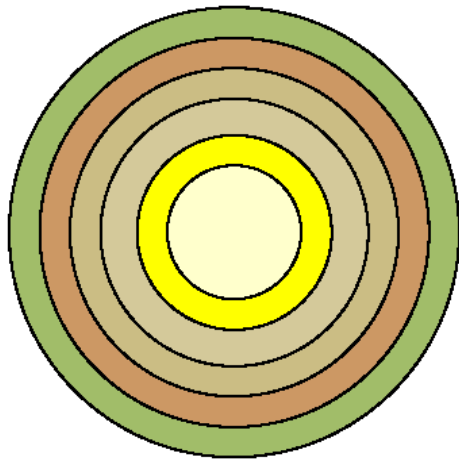


Ciascun livello:

- usa le **funzionalità** di quello sottostante
- fornisce **servizi** al livello che segue nella gerarchia
- gestisce delle **risorse** mediante politiche invisibili ai livelli superiori

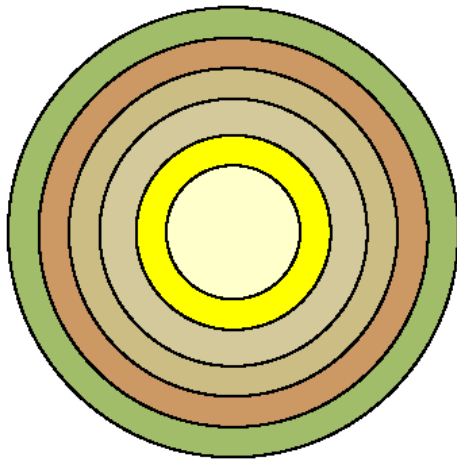
“macchine virtuali”

Si crea, in questo modo, una gerarchia di “macchine virtuali”.



- l'esperto che scrive un sistema operativo vede il sistema come un insieme di risorse fisiche da comandare direttamente;
- colui che *progetta* un ambiente di programmazione vede la macchina come l'insieme delle funzioni messe a disposizione dal *sistema operativo*;

“macchine virtuali”



- l'utente che *usa* un linguaggio di alto livello per progettare un programma applicativo vede l'elaboratore come l'insieme delle funzionalità messe a disposizione dall'*ambiente di programmazione*;
- per l'utilizzatore di un programma applicativo, infine, il sistema appare virtualmente come l'insieme dei comandi che può fornire alla macchina per soddisfare le sue esigenze.

“macchine virtuali”

Riassumendo: l'utente finale del sistema interagisce solo con il livello più esterno della gerarchia.

Idealmente, l'utente finale è *ignaro* di tutti i dettagli delle operazioni svolte dai livelli inferiori.

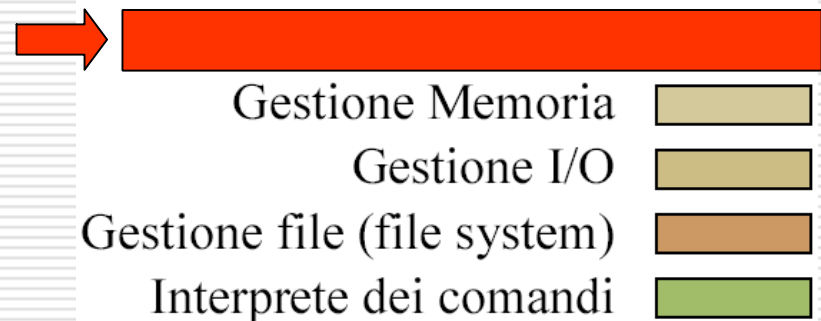
Potrebbe (sempre idealmente) essere a conoscenza *solo delle operazioni che è interessato ad effettuare.*

gestione CPU

Il livello più basso è quello del **kernel** (nucleo). Questa parte del sistema operativo si occupa di gestire l'esecuzione dei programmi.

Un programma in esecuzione è detto **processo**.

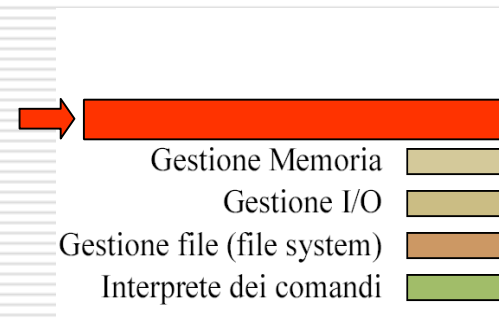
Il kernel **distribuisce** le risorse di calcolo tra i vari processi attivi.



gestione CPU

Una prima distinzione è dunque tra quei sistemi che eseguono un processo per volta e quelli in grado di gestirne diversi “contemporaneamente”.

Questi ultimi sono detti **multitasking**.

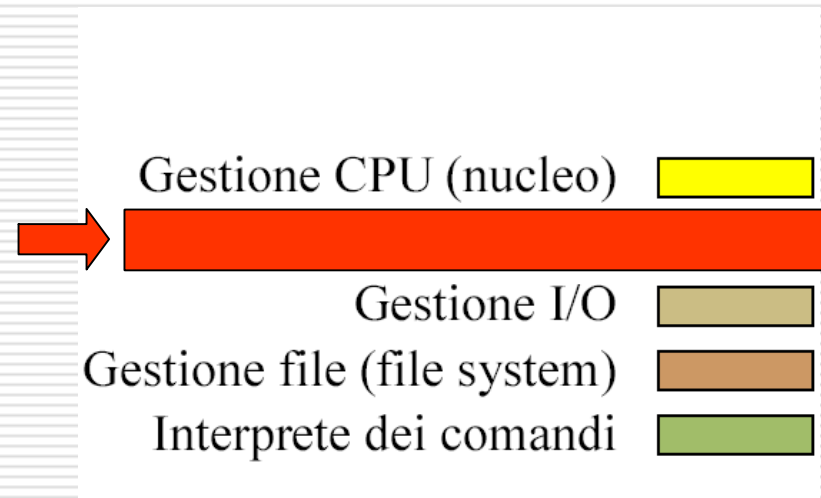


gestione memoria

La memoria è una risorsa **essenziale** e **limitata**.

Essenziale, perché ogni programma in esecuzione (processo) deve essere “caricato” in memoria, e così i dati su cui opera.

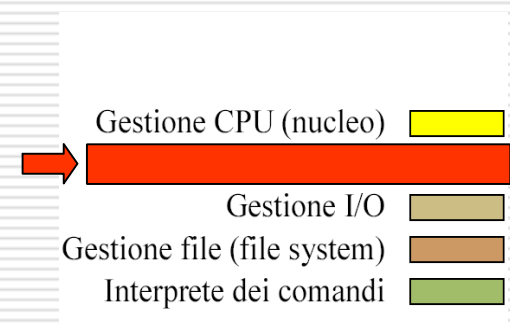
Limitata, perché nei sistemi moderni possono essere attivi più processi nello stesso tempo.



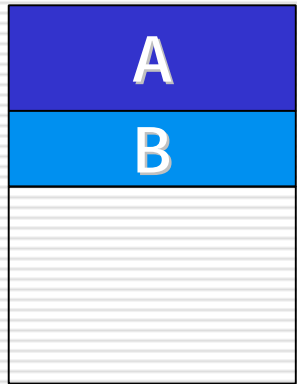
gestione memoria

Dal momento che la memoria di sistema (RAM) è una risorsa finita, nell'allocarla ai vari processi il sistema operativo deve risolvere vari problemi:

- trovare spazio per i vari processi;
- “rilocare” il codice caricato in memoria;
- ridurre la frammentazione.



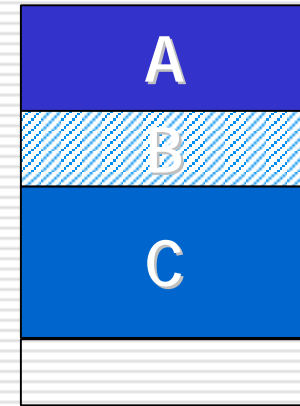
gestione memoria



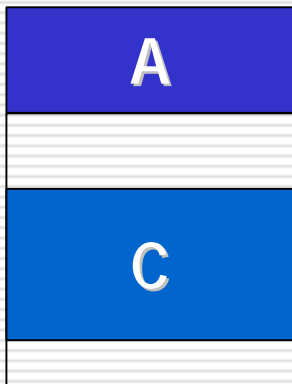
In memoria sono caricati i processi A e B.



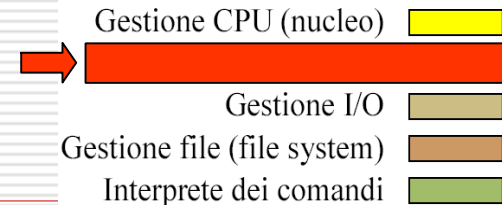
Anche C viene caricato in memoria.



Il processo B termina, e libera la memoria.

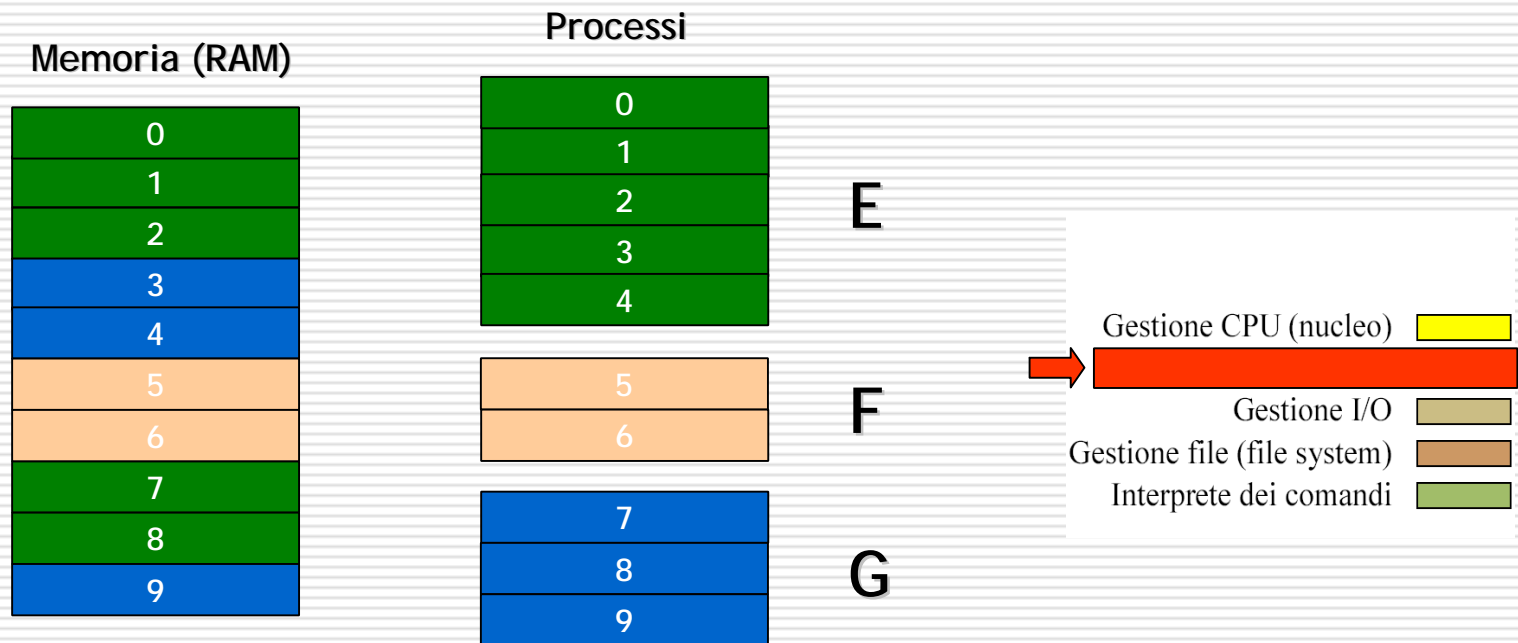


Il processo D non trova spazio in cui inserirsi, anche se la memoria libera complessiva sarebbe sufficiente.



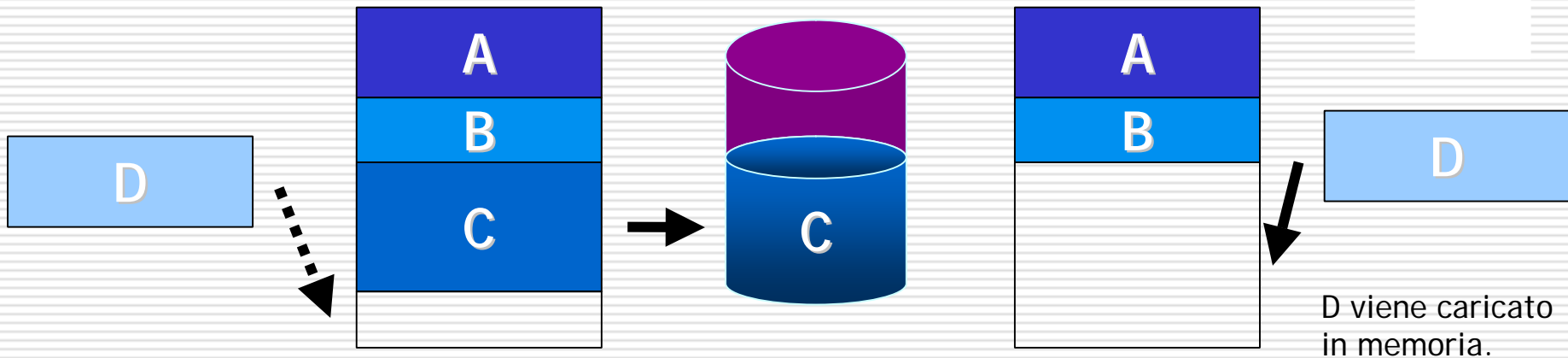
gestione memoria: paginazione

E' anche possibile suddividere la memoria in blocchi (pagine fisiche). I programmi da caricare dovranno essere divisi in pagine logiche.



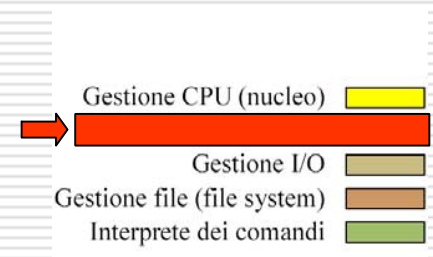
gestione memoria: swapping

Il sistema operativo può riservare un'area di un disco per lo swapping.



In memoria sono presenti A, B, e C. D non trova spazio.

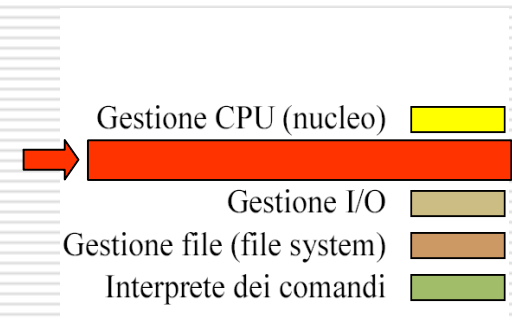
C viene copiato su un disco dal gestore della memoria.



memoria virtuale

Le tecniche appena esposte possono coesistere tranquillamente. Il risultato della loro interazione è che il sistema può disporre, in apparenza, di una quantità di memoria *maggiore* di quella fisica installata.

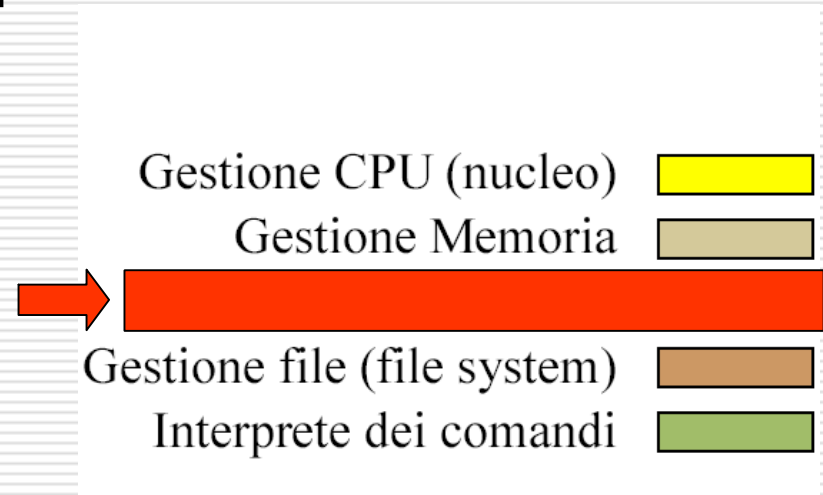
L'utilizzo di una memoria secondaria (su disco) per *estendere* la memoria di sistema consente di parlare di **memoria virtuale**.



Gestione input/output (I/O)

L'accesso alle periferiche di I/O viene gestito dal sistema operativo insieme ai *driver di periferica*.

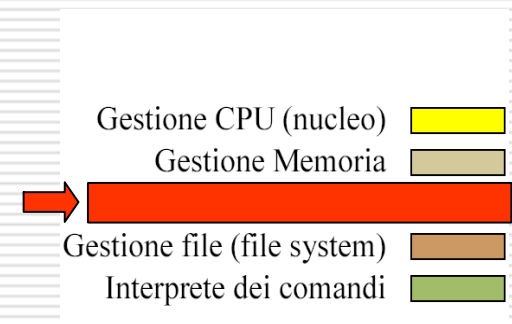
Questi sono programmi specifici per ciascun dispositivo che si colleghi all'elaboratore (stampanti, scanner dischi...).



Gestione input/output (I/O)

L'interazione tra un programma e una periferica è standardizzata. Un programma di elaborazione testi, ad esempio, può inviare un comando di stampa senza curarsi del tipo di stampante collegata al computer.

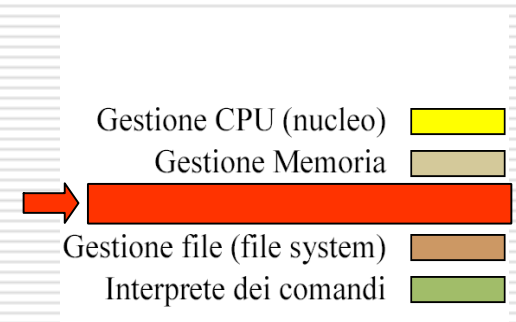
Spetta al sistema operativo smistare la richiesta al driver della stampante.



Gestione input/output (I/O)

A questo livello è implementato anche un sistema di gestione degli **errori di I/O** (ad es. dischetto mancante o danneggiato, carta esaurita, ecc.).

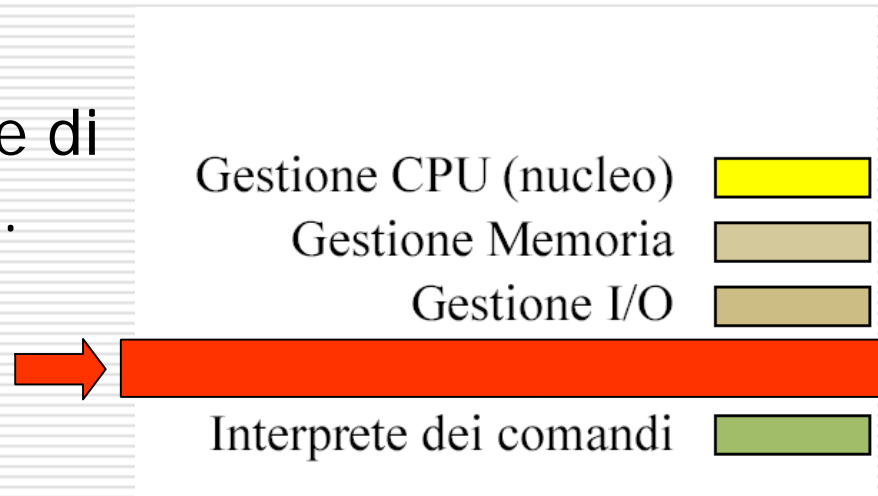
Anche il controllo dell'ordine di accesso ai dispositivi è cruciale. Il sistema operativo deve prevenire, o risolvere, eventuali conflitti.



Gestione files

Il **file system** è il modo in cui il sistema operativo organizza i file (documenti) sulle unità di memorizzazione.

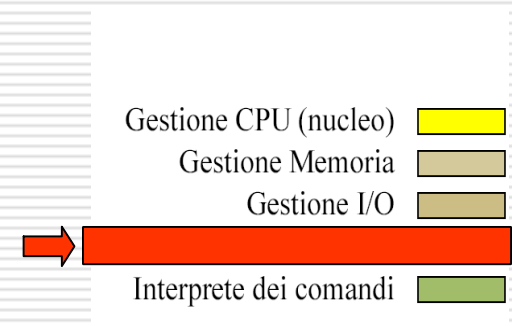
Un **file** è un'astrazione che rappresenta un insieme di byte logicamente collegati.



Gestione files: funzioni

Il **file system** deve mettere a disposizione diverse funzioni per la manipolazione dei file:

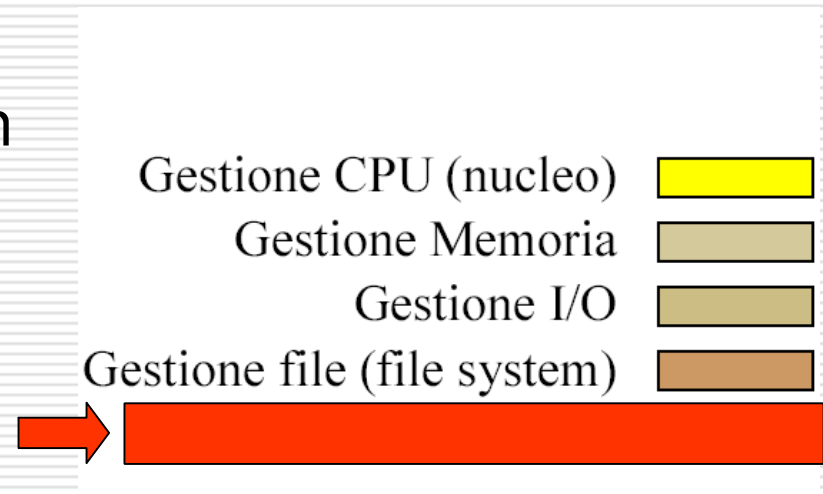
- creazione/eliminazione
- lettura/scrittura/esecuzione
- coordinamento accessi contemporanei
- controllo degli accessi (nei sistemi multiutente)



Interprete dei comandi (shell)

L'**interprete dei comandi** è quella parte del sistema operativo che riceve ed elabora le istruzioni impartite da un utente.

E' possibile utilizzare lo stesso sistema operativo con **shell** differenti. Questo può rendere molto diverso il modo di impartire comandi.

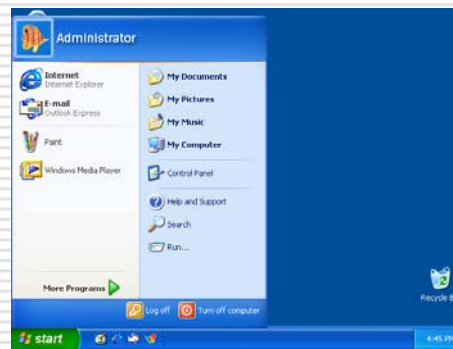


Interprete dei comandi (shell)

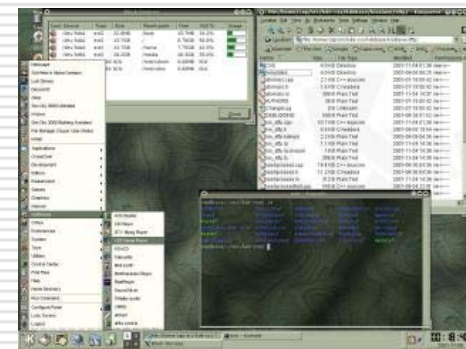
La shell è dunque lo strato più esterno di un sistema operativo. Di fatto, rappresenta l'interfaccia tra utente e sistema.



MacOS X



Windows XP



Linux - KDE 3.0

- Gestione CPU (nucleo)
- Gestione Memoria
- Gestione I/O
- Gestione file (file system)



Inteprete dei comandi (shell) e interfaccia

Che differenza c'è tra "interprete dei comandi",
"shell", e "interfaccia utente"?

I primi due termini sono sinonimi.

"Shell" vuol dire "guscio", in riferimento al fatto che si tratta dello strato più *esterno* di un sistema operativo.

E' un programma.

Inteprete dei comandi (shell) e interfaccia

L' *interfaccia* è il **mezzo** attraverso il quale l'utente invia comandi alla shell.

Esistono diverse interfacce possibili per l'interazione uomo-macchina. Il loro successo dipende, oltre che da un'accurata **progettazione**, dal **tipo di utente** cui sono rivolte.

Interfaccia

Due sono i tipi di interfaccia più diffusi per interagire con un computer:

- Interfacce a **caratteri**, anche dette a **riga di comando** (*CLI, command line interfaces*)
- Interfacce **grafiche** (GUI, graphical user interfaces), in particolare di tipo WIMP (Windows, Icons, Menus, Pointing device)