UNIVERSITÀ
DI PARMA

introduzione ai sistemi operativi

Alberto Ferrari

classificazione del software

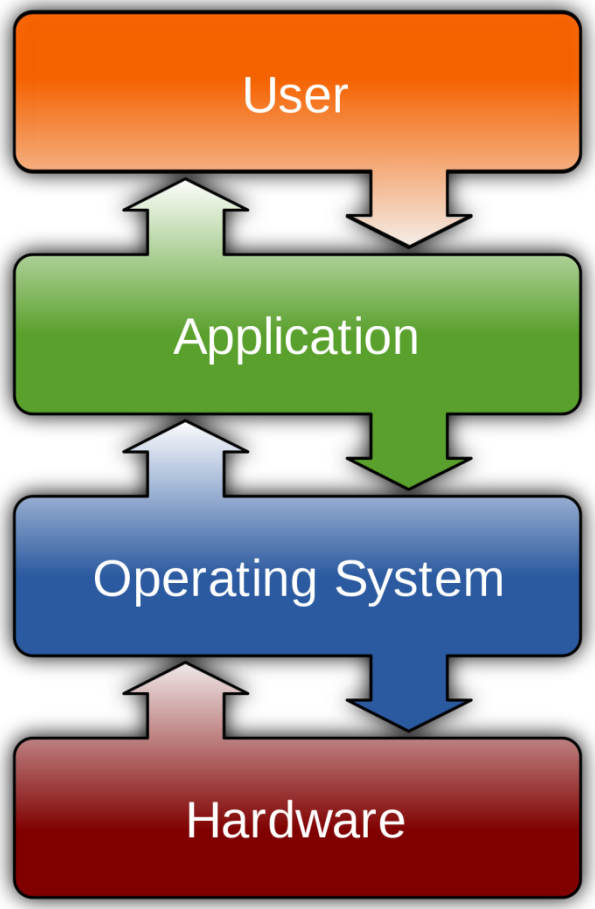
- il software può essere classificato in due categorie
  - ***programmi di sistema***
    - gestiscono le operazioni del sistema di elaborazione
  - ***programmi applicativi***
    - risolvono i problemi dei loro utilizzatori
- l'insieme dei programmi di sistema viene identificato con il nome di ***sistema operativo*** (***OS O****perating* ***S****ystem*)

caratteristiche

- il sistema operativo
  - è un insieme di ***moduli software***
  - ***controlla*** le risorse ***hardware*** del sistema
  - mette a disposizione dell'utente una ***macchina virtuale***, in grado di eseguire comandi dati dall'utente, utilizzando la macchina "reale"
    - la macchina virtuale ***nasconde*** tutti i dettagli hardware che sarebbero troppo complicati da gestire per la maggior parte degli utenti

scopo del sistema operativo

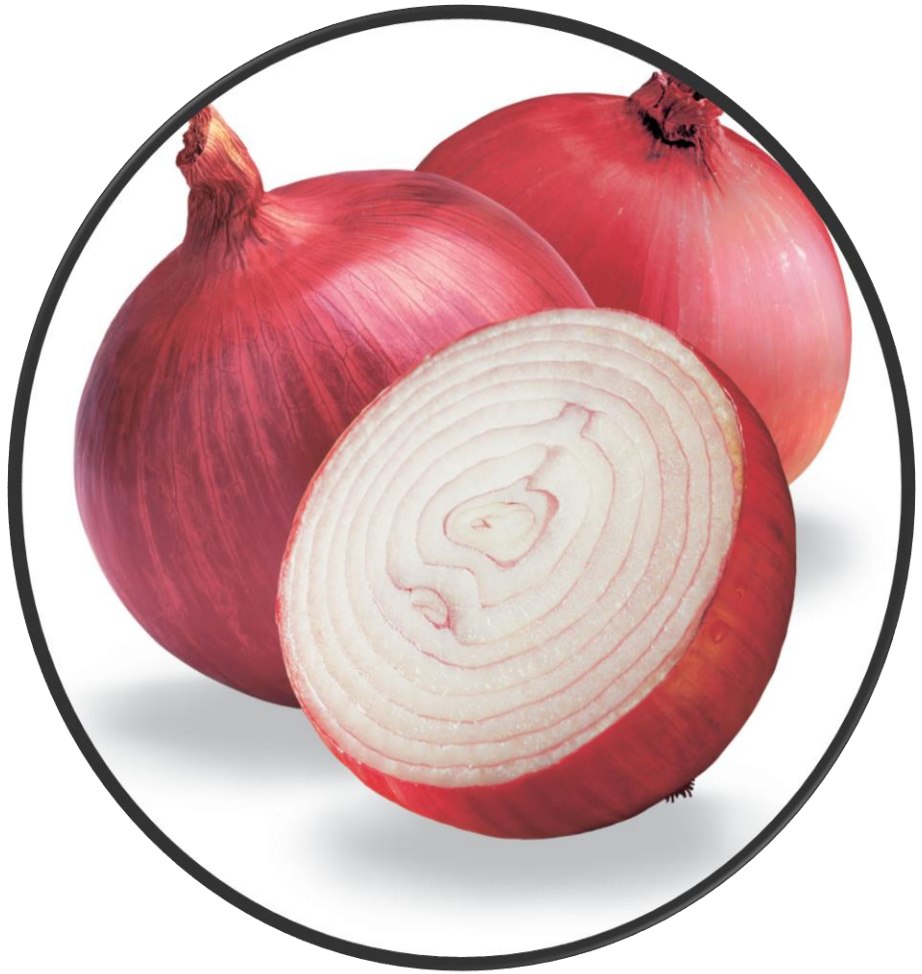
- ***gestione delle risorse*** del sistema di elaborazione
- semplificare l'***interfacciamento*** tra utente e sistema



strati

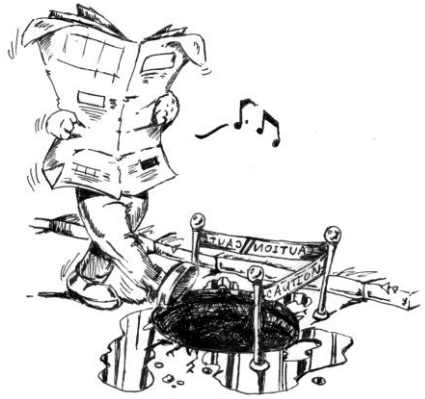

• Utente
• Applicazioni
• Sistema Operativo
• Firmware
• Hardware

struttura a “cipolla”



utenti e livelli

- ogni strato (livello) costituisce una ***macchina virtuale***
  - ***usa*** le funzionalità di quello ***sottostante***
  - ***fornisce servizi*** al livello che segue nella gerarchia
  - ***gestisce*** le risorse mediante politiche invisibili ai livelli superiori
- l'utente finale interagisce solo con il livello più esterno della gerarchia
  - è ignaro di tutti i dettagli delle operazioni svolte dai livelli inferiori



gerarchia di macchine virtuali

- chi scrive un sistema operativo vede il sistema come un insieme di ***risorse fisiche*** da comandare direttamente
- chi progetta un ambiente di programmazione vede la macchina come l'insieme delle ***funzioni*** messe a disposizione dal sistema operativo
- il programmatore usa un ***linguaggio*** ad alto livello per realizzare un programma applicativo e vede l'elaboratore come l'insieme delle funzionalità messe a disposizione dall'ambiente di programmazione
- per l'***utilizzatore*** di un programma applicativo il sistema appare virtualmente come l'insieme dei comandi che può fornire alla macchina per soddisfare le sue esigenze

bootstrap

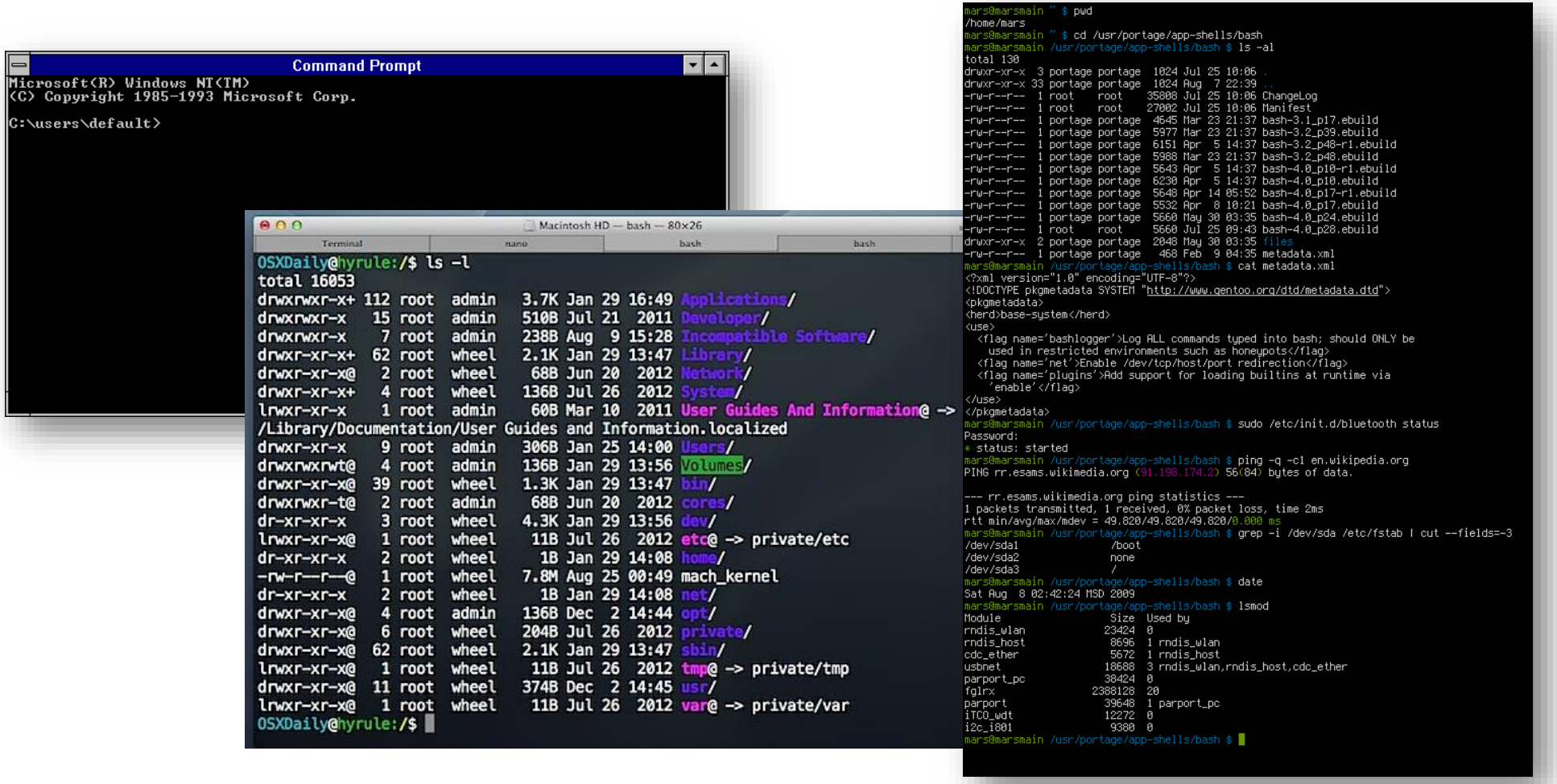
- il sistema operativo viene caricato nella memoria RAM all'accensione della macchina (programma di boot) e rimane attivo fino allo spegnimento

```
Kernel command line: block2mtd.block2mtd=/dev/hda2,131072,rootfs root=/dev/mtdbl
ock0 rootfstype=jffs2 init=/etc/preinit  noinitrd console=tty0 console=ttyS0,384
00n8 reboot=bios
Found and enabled local APIC!
Enabling fast FPU save and restore... done.
Enabling unmasked SIMD FPU exception support... done.
Initializing CPU#0
PID hash table entries: 32 (order: 5, 128 bytes)
Detected 1991.657 MHz processor.
Console: colour VGA+ 80x25
console [tty0] enabled
console [ttyS0] enabled
Dentry cache hash table entries: 1024 (order: 0, 4096 bytes)
Inode-cache hash table entries: 1024 (order: 0, 4096 bytes)
Memory: 5112k/8128k available (1497k kernel code, 2624k reserved, 597k data, 196
k init, 0k highmem)
virtual kernel memory layout:
    fixmap  : 0xfffb9000 - 0xfffff000   ( 280 kB)
    vmalloc : 0xc1000000 - 0xfffb7000   (1007 MB)
    lowmem  : 0xc0000000 - 0xc07f0000   (   7 MB)
      .init : 0xc0313000 - 0xc0344000   ( 196 kB)
      .data : 0xc027653c - 0xc030bcfc   ( 597 kB)
      .text : 0xc0100000 - 0xc027653c   (1497 kB)
Checking if this processor honours the WP bit even in supervisor mode...Ok.
Calibrating delay using timer specific routine.. 4047.64 BogoMIPS (lpj=20238210)
```

interfaccia utente

- tutti i sistemi operativi implementano dei meccanismi per rendere agevole l'***utilizzo del sistema*** da parte degli utente
  - ***interfaccia utente***
  - consentire all'utente di accedere al sistema tramite un meccanismo di autenticazione (***login***), o di interrompere l'attività del sistema (***logoff*** e/o ***shutdown***)
- ***interfaccia testuale***
  - interprete dei comandi (***shell***)
  - ***command line interface***
- ***interfaccia grafica*** (a finestre):
  - l'output dei vari programmi viene visualizzato in maniera grafica all'interno di finestre
  - l'utilizzo di immagini rende ***più intuitivo*** l'uso del calcolatore

interfaccia testuale



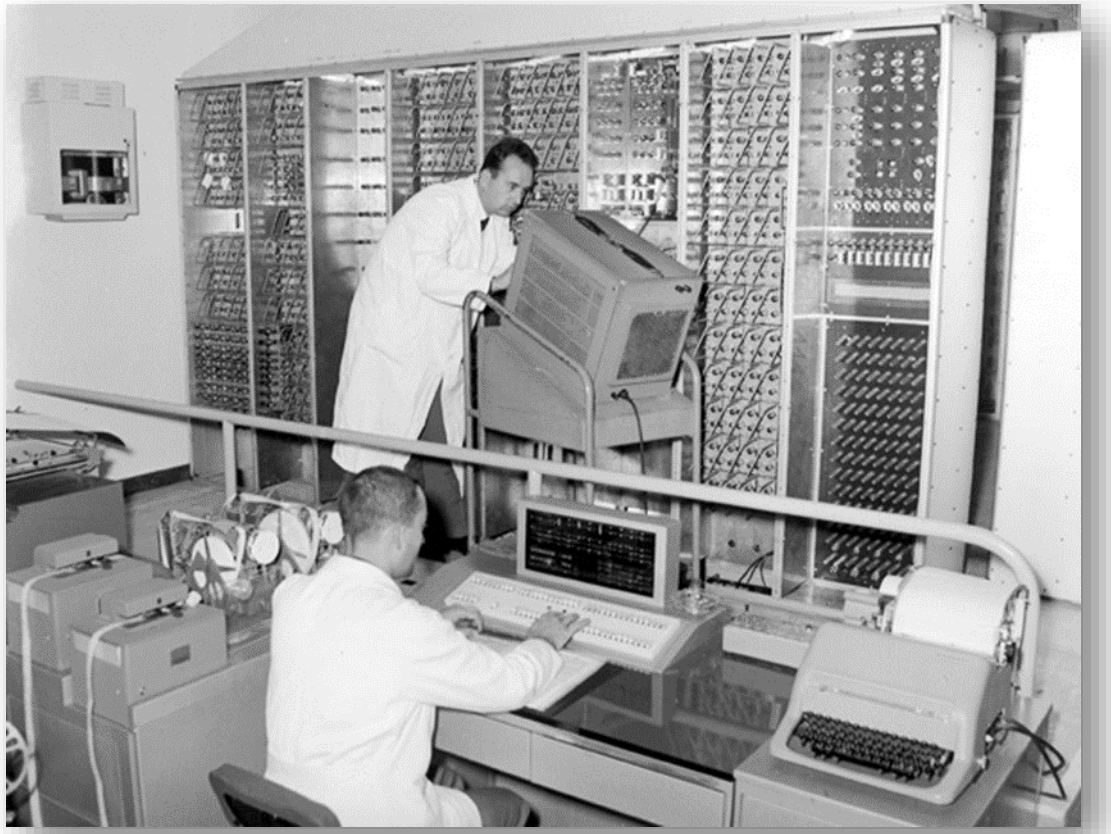
interfaccia grafica
Graphic User Interface

interprete dei comandi

- le richieste dell'utente al Sistema Operativo sono fatte
  - ***digitando*** i comandi nelle interfacce a carattere
    - interfacce ***a riga di comando*** (***c***ommand ***l***ine ***i***nterface)
  - selezionando oggetti con il mouse nelle interfacce grafiche
    - ***g***raphic ***u***ser ***i***nterface
    - ***WIMP*** (windows, icons, menus, pointer)
- le richieste sono intercettate dall'***interprete dei comandi*** (***shell***) che attiva le specifiche funzioni del nucleo del sistema (***kernel***)
  - i moduli software attivano poi i dispositivi hardware (processore, memoria, I/O controller …) che rispondono alla richiesta dell'utente

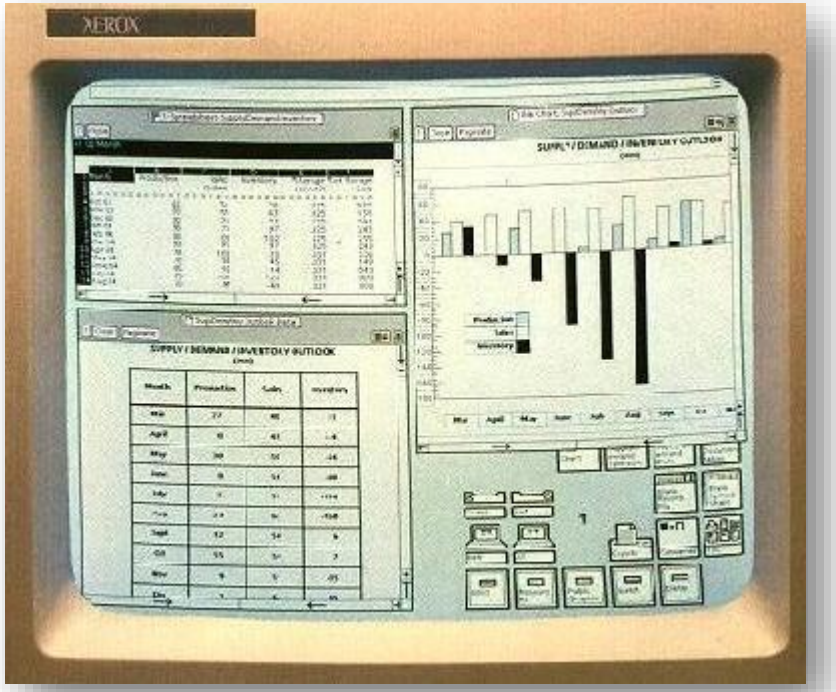
i primi sistemi operativi

- 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 formata da interruttori e spie luminose (pannello di controllo)
- i comandi venivano impartiti in codice binario



Xerox Star

- la workstation ***Xerox Star*** è stata il primo computer commercializzato ad essere dotato di interfaccia grafica di tipo WIMP
  - commercializzato a partire dal 27 aprile 1981
  - costituito da una serie di workstation collegate fra loro tramite LAN
- lo Xerox Star per la prima volta proponeva, in alternativa alla multiutenza del minicomputer, una serie di computer monoutenti collegati tramite LAN.



Application Program Interface (API)

- il Sistema Operativo espone una ***API*** che assume la forma di una libreria di funzioni
  - ***system-call*** (chiamate di sistema)
- l'API di Windows è nota come ***WINAPI***
- l'interfaccia di riferimento per il mondo Linux è denominata ***POSIX***
- molti programmatori non utilizzano direttamente le API, i linguaggi di programmazione "***nascondono***" l'interazione col SO
  - Es. ***printf*** del linguaggio C o ***cout*** del C++ vengono trasformate dal compilatore in chiamate alle API del SO

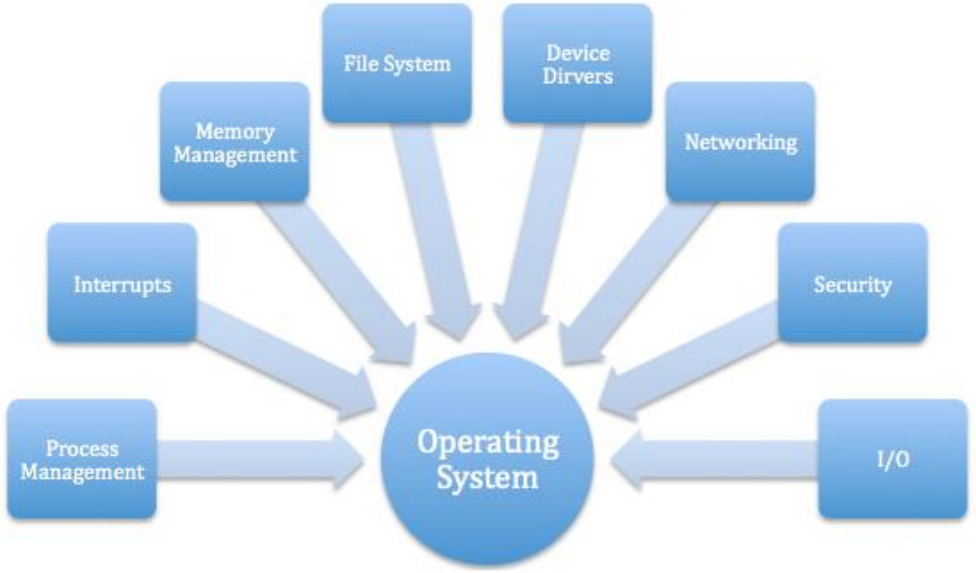
tipologie di sistemi operativi

- ***monoutente***
- ***multiutente***
  - il sistema operativo deve garantire che ogni utente avverta la macchina come dedicata

- ***monoprogrammati***
  - in grado di mandare in esecuzione un solo processo alla volta su un sistema in cui un solo utente può agire
  - sistemi ormai scomparsi
- ***multiprogrammati***
  - possono gestire più processi contemporaneamente, per i quali alternano un preciso quanto di tempo di esecuzione di CPU

OS - struttura

- i sistemi operativi sono generalmente costituiti da un insieme di ***moduli***
  - ogni modulo è dedicato a svolgere una determinata ***funzione***
- i moduli ***interagiscono*** secondo regole precise al fine di realizzare le funzionalità di base del sistema



OS - principali funzionalità

- gestione della ***memoria di massa***
  - file system
- gestione della ***memoria centrale***
- gestione dei ***processi***
- gestione dell' ***interfaccia utente***
- ***accesso simultaneo*** di più utenti al ***sistema***
- ***esecuzione simultaneamente*** di più ***processi*** sullo stesso sistema

Kernel (nucleo)

gestione dei processi

- nei sistemi multiprogrammati più programmi in esecuzione contemporaneamente (***processi***) consentono
  - all'utente di utilizzare ***più applicazioni*** nello stesso tempo
  - alla macchina di ***distribuire*** il ***carico computazionale*** con maggiore efficacia
  - per esempio sulle operazioni di Input/Output verso i dispositivi
- il SO alterna sulla CPU differenti processi (***schedulazione***) per tempi molto ridotti (***time-sharing***)
- questo rende l'esecuzione dei programmi ***contemporanea*** agli occhi degli utenti

gestione della memoria

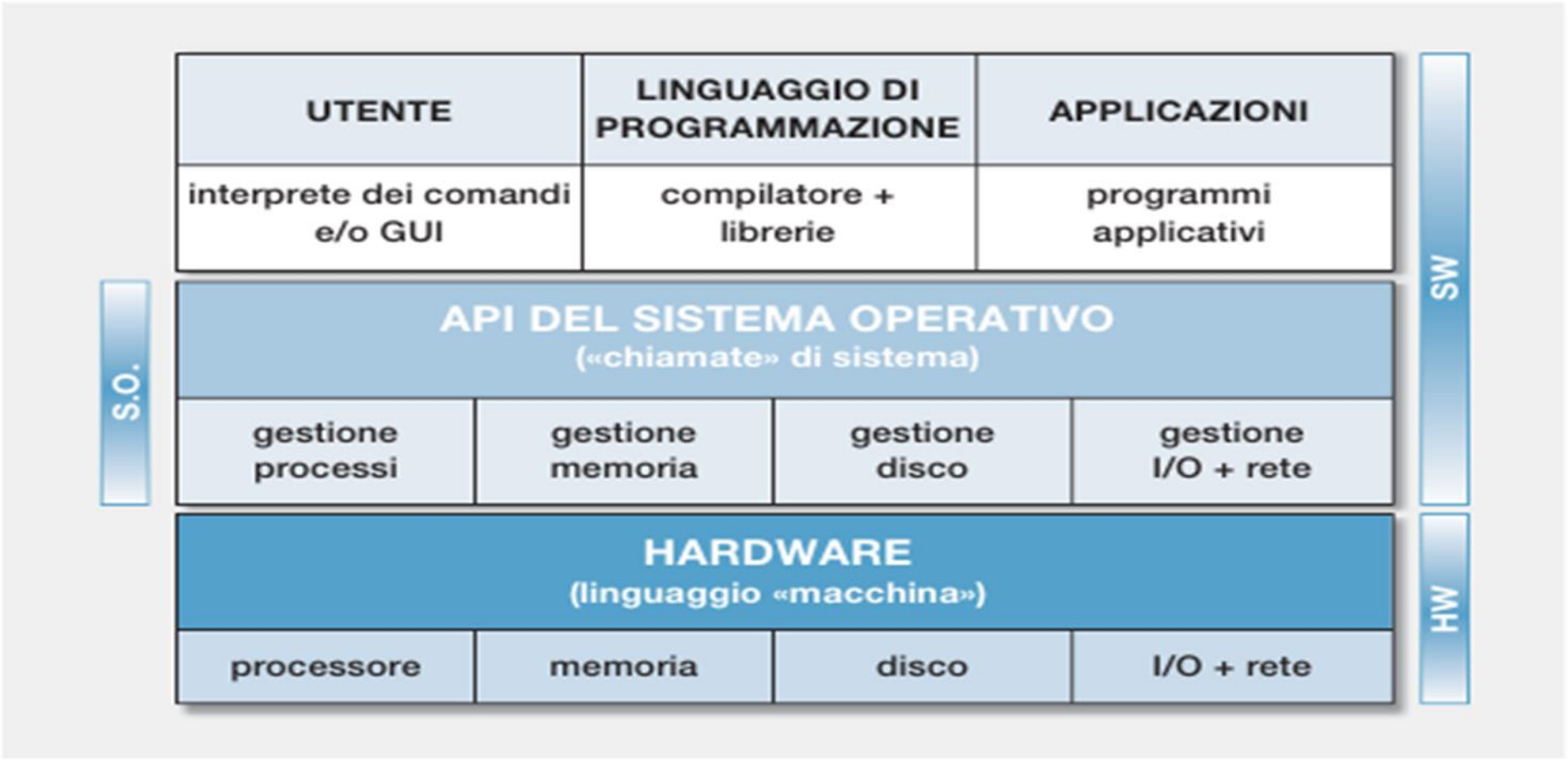
- più processi devono utilizzare la memoria in modo ***concorrente*** senza sovrapporre i loro dati
- la multiprogrammazione impone che la ***memoria*** sia sempre più ***grande***, per contenere più processi in time-sharing
- per risolvere il problema della dimensione della memoria i SO ***simulano*** la memoria mancante sulla memoria di massa
- l'insieme delle tecniche che consentono di simulare la memoria su disco viene detta ***memoria virtuale***

gestione dei dispositivi

- la gestione dei dispositivi o dell'Input/Output è una delle parti più problematiche di ogni sistema operativo
- i dispositivi sono spesso sviluppati da produttori di periferiche (terze parti) spesso diversi dai produttori di calcolatori e dai produttori del SO
  - parti consistenti del SO devono essere integrate con programmi scritti da terze parti (***driver***)

gestione della memoria di massa

- la gestione delle memorie di massa utilizza strutture dati denominate ***File System***
- i dati residenti fisicamente sulle memorie secondarie sono organizzati in ***settori***, a loro volta componenti di elementi logici denominati ***files***
- una seconda astrazione permette di organizzare i files tramite directory per costituire un cosiddetto ***file system gerarchico*** organizzato ad albero
  - ***root directory*** (directory radice)
  - nome completo del file (***pathname***, percorso più nome logico)


UTENTE
LINGUAGGIO DI PROGRAMMAZIONE
APPLICAZIONI
interprete dei comandi e/o GUI
compilatore + librerie
programmi applicativi
API DEL SISTEMA OPERATIVO
(«chiamate» di sistema)
gestione processi
gestione memoria
gestione disco
gestione I/O + rete
HARDWARE
(linguaggio «macchina»)
processore
memoria
disco
I/O + rete
S.O.
SW
HW

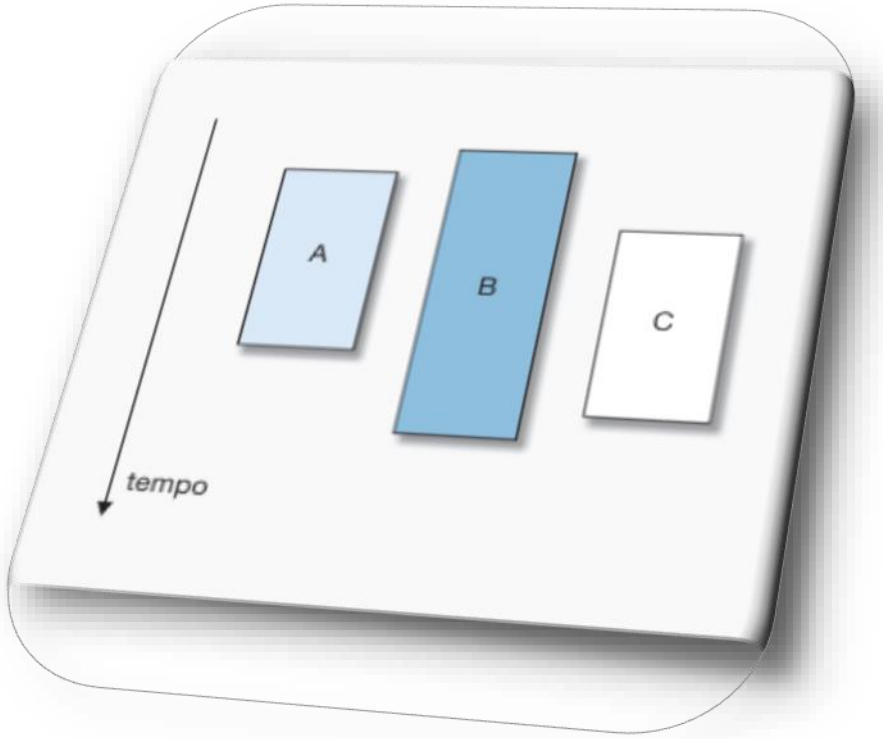
sistemi operativi

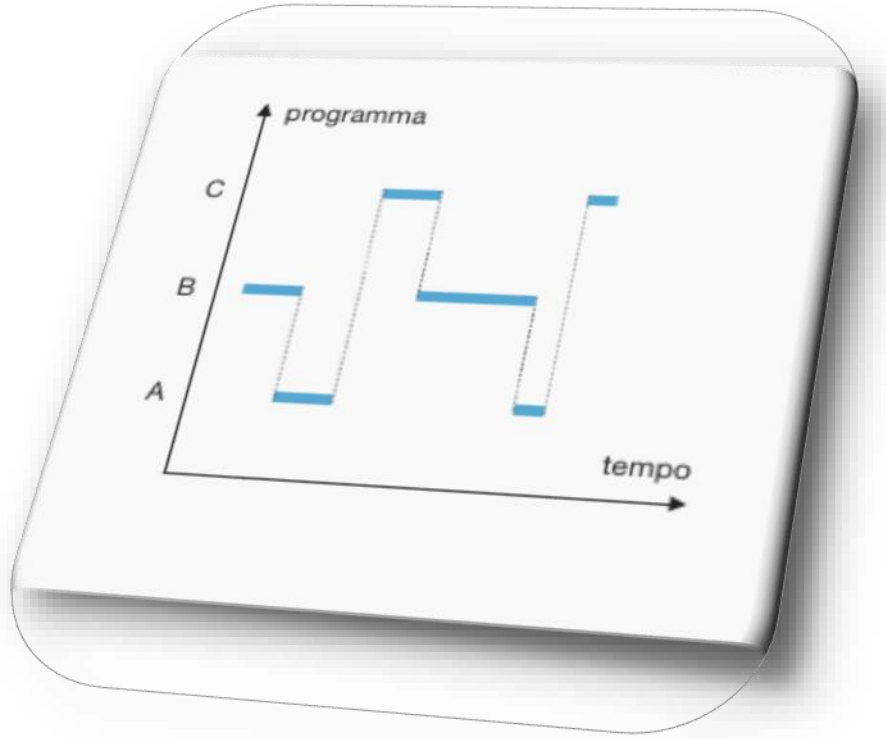
gestore dei processi

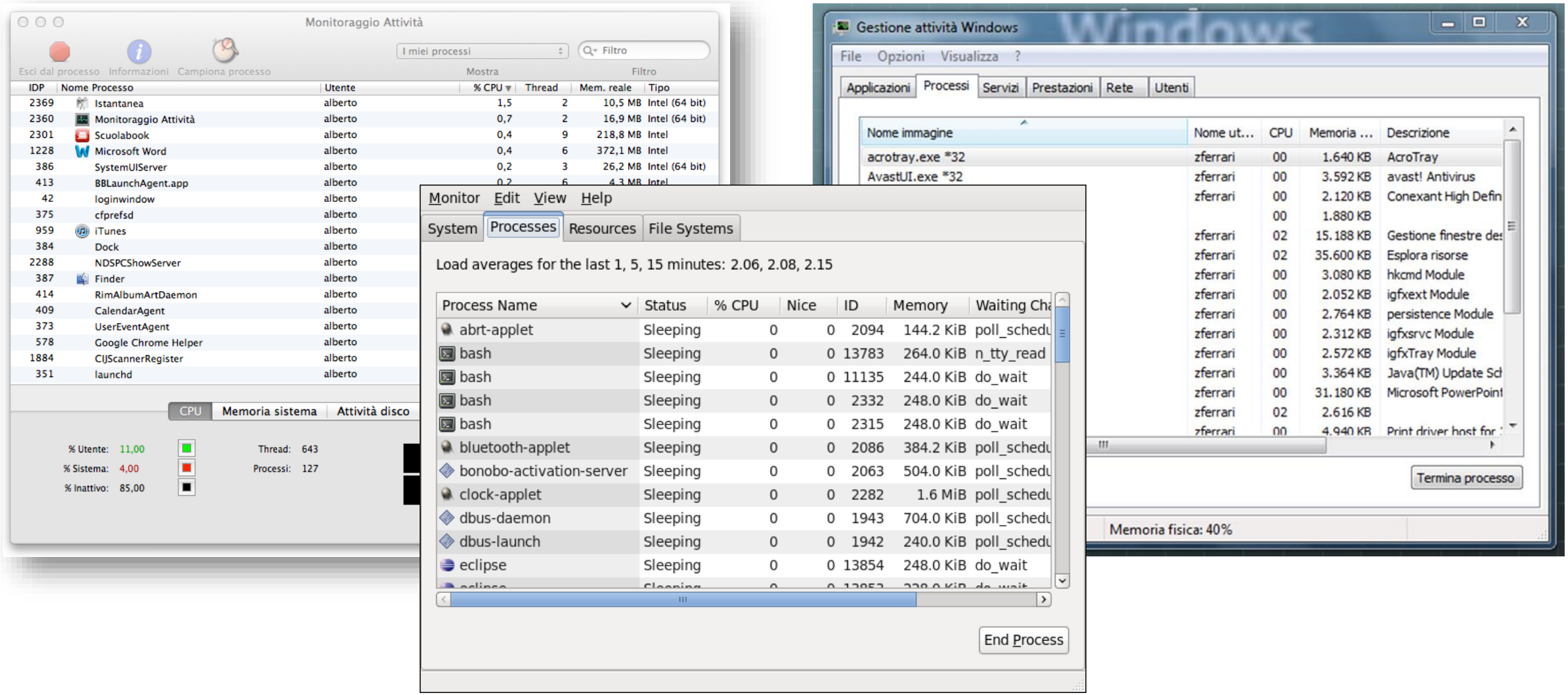
multiprogrammazione

- tutti i Sistemi Operativi moderni sono in grado di eseguire ***contemporaneamente*** più di un programma
- il ***numero*** di ***programmi*** in esecuzione è ***superiore*** al numero di ***processori*** del sistema
- l'utente deve avere la sensazione che le proprie applicazioni siano eseguite ***contemporaneamente*** dal computer
- in realtà la sensazione di contemporaneità è data dall'elevata ***velocità*** di esecuzione dei programmi da parte del computer

esecuzione «parallela»





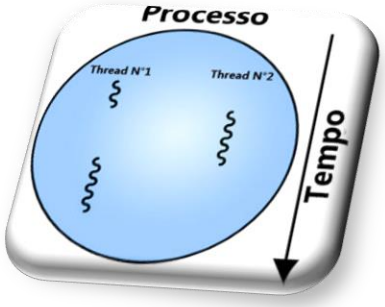

Monitoraggio Attività
Esci dal processo
Informazioni
Campiona processo
I miei processi
Filtro
Mostra
Filtro
IDP
Nome Processo
Utente
% CPU
Thread
Mem. reale
Tipo
2369 Istantanea alberto 1,5 2 10,5 MB Intel (64 bit)
2360 Monitoraggio Attività alberto 0,7 2 16,9 MB Intel (64 bit)
2301 Scuolabook alberto 0,4 9 218,8 MB Intel
1228 Microsoft Word alberto 0,4 6 372,1 MB Intel
386 SystemUIServer alberto 0,2 3 26,2 MB Intel (64 bit)
413 BBLaunchAgent.app alberto 0,2 6 4,3 MB Intel
42 loginwindow alberto
375 cfprefsd alberto
959 iTunes alberto
384 Dock alberto
2288 NDSPCShowServer alberto
387 Finder alberto
414 RimAlbumArtDaemon alberto
409 CalendarAgent alberto
373 UserEventAgent alberto
578 Google Chrome Helper alberto
1884 CIJScannerRegister alberto
351 launchd alberto
CPU
Memoria sistema
Attività disco
% Utente: 11,00
% Sistema: 4,00
% Inattivo: 85,00
Thread: 643
Processi: 127
Gestione attività Windows
File Opzioni Visualizza ?
Applicazioni
Processi
Servizi
Prestazioni
Rete
Utenti
Nome immagine
Nome ut...
CPU
Memoria ...
Descrizione
acrotray.exe *32 zferrari 00 1.640 KB AcroTray
AvastUI.exe *32 zferrari 00 3.592 KB avast! Antivirus
zferrari 00 2.120 KB Conexant High Defin
00 1.880 KB
zferrari 02 15.188 KB Gestione finestre des
zferrari 02 35.600 KB Esplora risorse
zferrari 00 3.080 KB hkcmd Module
zferrari 00 2.052 KB igfxext Module
zferrari 00 2.764 KB persistence Module
zferrari 00 2.312 KB igfxsrvc Module
zferrari 00 2.572 KB igfxTray Module
zferrari 00 3.364 KB Java(TM) Update Sch
zferrari 00 31.180 KB Microsoft PowerPoint
zferrari 02 2.616 KB
Termina processo
Memoria fisica: 40%
Monitor Edit View Help
System
Processes
Resources
File Systems
Load averages for the last 1, 5, 15 minutes: 2.06, 2.08, 2.15
Process Name
Status
% CPU
Nice
ID
Memory
Waiting Cha
abrt-applet Sleeping 0 0 2094 144.2 KiB poll_schedu
bash Sleeping 0 0 13783 264.0 KiB n_tty_read
bash Sleeping 0 0 11135 244.0 KiB do_wait
bash Sleeping 0 0 2332 248.0 KiB do_wait
bash Sleeping 0 0 2315 248.0 KiB do_wait
bluetooth-applet Sleeping 0 0 2086 384.2 KiB poll_schedu
bonobo-activation-server Sleeping 0 0 2063 504.0 KiB poll_schedu
clock-applet Sleeping 0 0 2282 1.6 MiB poll_schedu
dbus-daemon Sleeping 0 0 1943 704.0 KiB poll_schedu
dbus-launch Sleeping 0 0 1942 240.0 KiB poll_schedu
eclipse Sleeping 0 0 13854 248.0 KiB do_wait
End Process

programma vs processo

- un ***programma*** è costituito dal ***codice oggetto*** generato dalla compilazione del codice sorgente
- è salvato sotto forma di uno o più ***file***
- è un'entità ***statica***
  - rimane immutata durante l'esecuzione
- il programma esiste ***indipendentemente*** dal computer che è in grado di eseguirlo

- un ***processo*** è definito come un'***istanza*** di un programma in ***esecuzione***
- al momento dell'esecuzione del programma il ***SO crea il processo***
- è un'entità ***dinamica***, che dipende dai dati che vengono elaborati, e dalle operazioni eseguite su di essi
- è caratterizzato, oltre che dal codice eseguibile, dall'insieme di tutte le informazioni che ne definiscono lo ***stato***
- ***da un solo programma possono scaturire più processi distinti***

- un ***thread*** è una ***suddivisione*** di un processo in due o più filoni, che vengono eseguiti ***concorrentemente*** dal processore
- un thread è ***contenuto*** all'interno di un processo
- un processo ha sempre ***almeno*** un thread (se stesso), ma in alcuni casi un processo può avere *più thread* che vengono eseguiti in ***parallelo***
- i vari thread dello stesso processo ***condividono*** alcune risorse (*lo spazio d'indirizzamento del processo*), mentre processi differenti non condividono le loro risorse
- i thread sono spesso utilizzati per la ***parallelizzazione*** di un programma, per sfruttare i moderni processori ***multi core***



processo vs thread

- ***processo***
  - il processo è l'oggetto del sistema operativo a cui sono ***assegnate tutte*** le risorse di sistema per l'esecuzione di un programma, ***tranne la CPU***
- ***thread***
  - il thread è l'oggetto del sistema operativo o dell'applicazione a cui è ***assegnata*** la ***CPU*** per l'esecuzione

identificativo di processo

- il Sistema Operativo associa a ogni processo (al momento della creazione) un ***identificativo*** numerico ***univoco*** (***PID*** - ***p****rocess* ***id****entifier*)
- a un processo sono associate le strutture dati:
  - uno o più segmenti di ***codice***
  - uno o più segmenti di ***memoria dati***
  - i ***descrittori*** di eventuali ***risorse*** in uso (file, finestre, periferiche, ecc.)
  - uno o più ***thread***

process control block

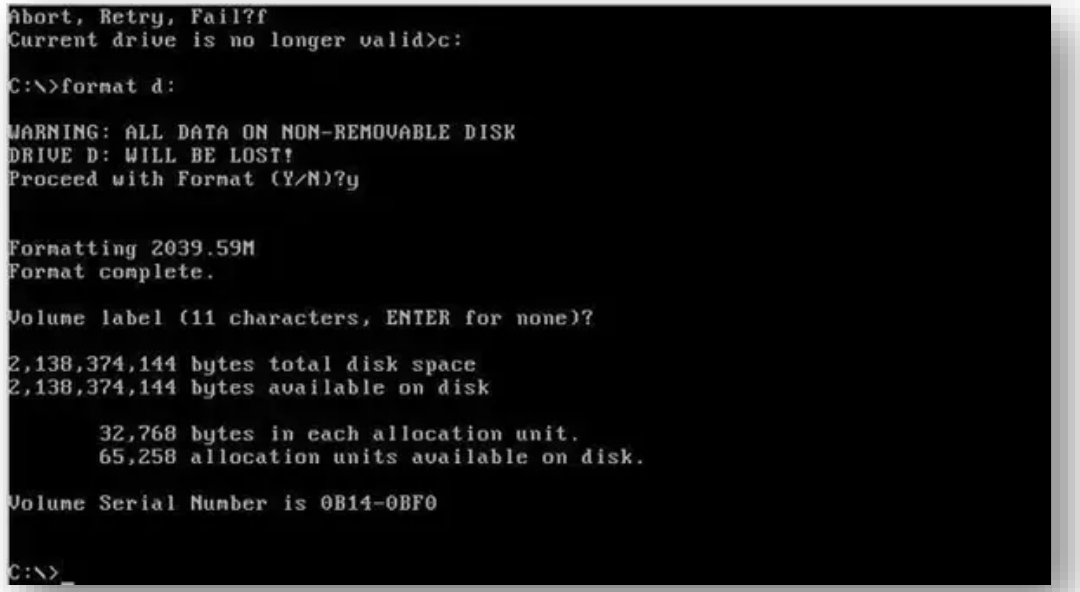
- il ***process control block*** (***PCB*** - *blocco di controllo del processo*) contiene le informazioni necessarie per la gestione del processo
  - ***program counter***
  - area per il salvataggio dei ***registri***
  - ***stato*** di avanzamento del processo (*ready, run, wait*)
  - ***PID*** (identificatore del processo)
  - livello di ***priorità***
  - ***risorse*** in uso
  - …

sistemi mono-tasking

- nei primi sistemi operativi ***mono-tasking*** si gestisce ***un solo programma in esecuzione*** (job) alla volta
  - DOS (*Disk Operating System*) è un sistema operativo monotasking: non si può fare niente altro mentre si formatta un floppy o si memorizzano dati su disco
- il computer è a disposizione del programma ***dall'inizio alla fine*** della sua esecuzione
- elaborazione "*a lotti*" (***batch***) vengono raccolti un insieme di programmi da eseguire uno dopo l'altro
- ***coda dei job***, gestita FIFO (first in, first out) e/o con priorità

svantaggi dei sistemi mono-tasking

- ***nessuna interazione*** utente-programma (l'utente può solo interrompere o sospendere l'esecuzione)
- ***lentezza***: la CPU non può essere usata da nessun processo mentre il programma in esecuzione svolge operazioni di I/O (molto più lente di letture/scritture in memoria)

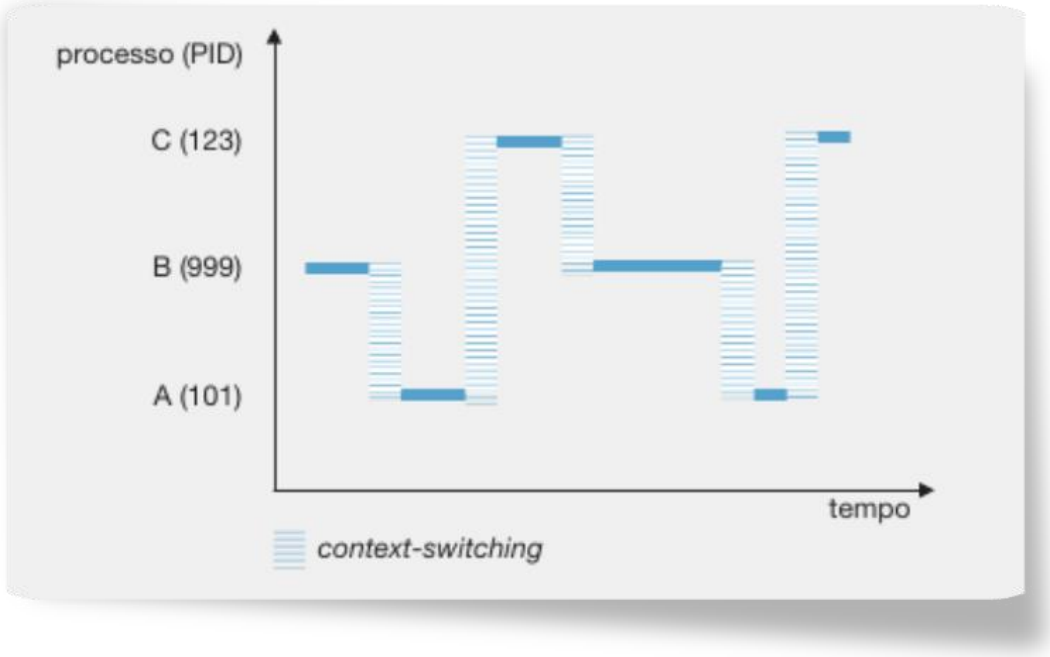


sistemi multi-tasking

- i sistemi ***multi-tasking*** permettono l'esecuzione ***contemporanea*** di più processi
- un processo può essere ***interrotto*** e la CPU può essere utilizzata da un altro processo
- l'obiettivo principale dei sistemi multi-tasking è quello ***ottimizzare l'utilizzo del processore***

context switching

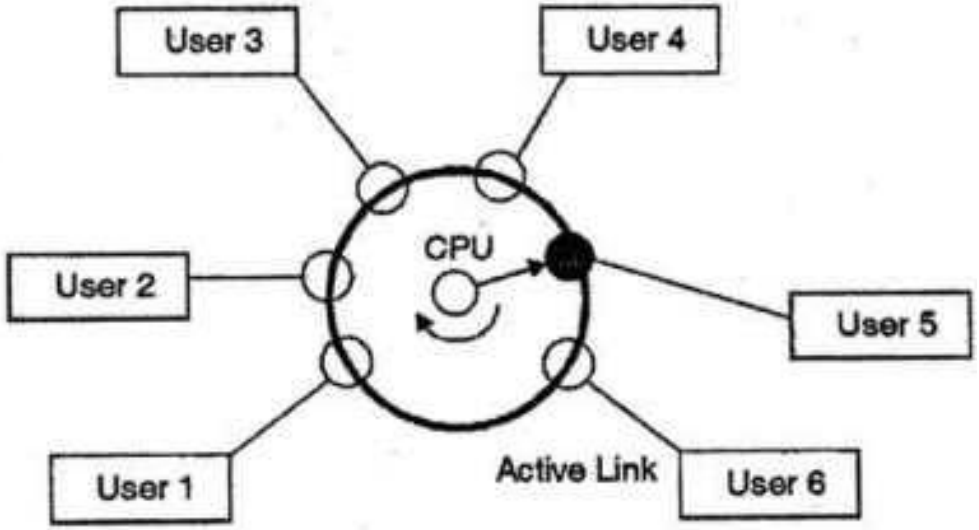
- nel di ***interruzione*** dell'esecuzione di un processo per assegnare la ***CPU*** a un altro processo (***context switching***) è necessario salvare le informazioni necessario al ripristino successivo dell'elaborazione
- nel context-switching viene salvato il ***PCB*** del primo processo e caricato il PCB del secondo

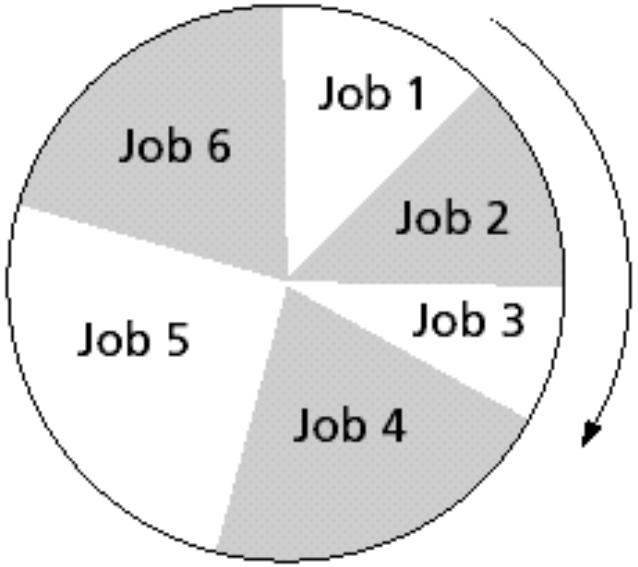


sistemi time-sharing

- i sistemi ***time-sharing*** sono una evoluzione dei sistemi multi-tasking
- l'esecuzione dell'attività della CPU viene suddivisa in ***quanti*** *(intervalli temporali)*
- ogni quanto è ***assegnato sequenzialmente*** a vari processi di uno stesso utente o a processi di più utenti
- ogni programma in esecuzione viene quindi eseguito ***ciclicamente*** per piccoli quanti di tempo
- se la velocità del processore è sufficientemente elevata si ha l'impressione di un'evoluzione ***parallela*** dei processi
- l'obiettivo principale dei sistemi time-sharing è quello di ***minimizzare il tempo di risposta***

time-sharing





sistemi dual core e multi core

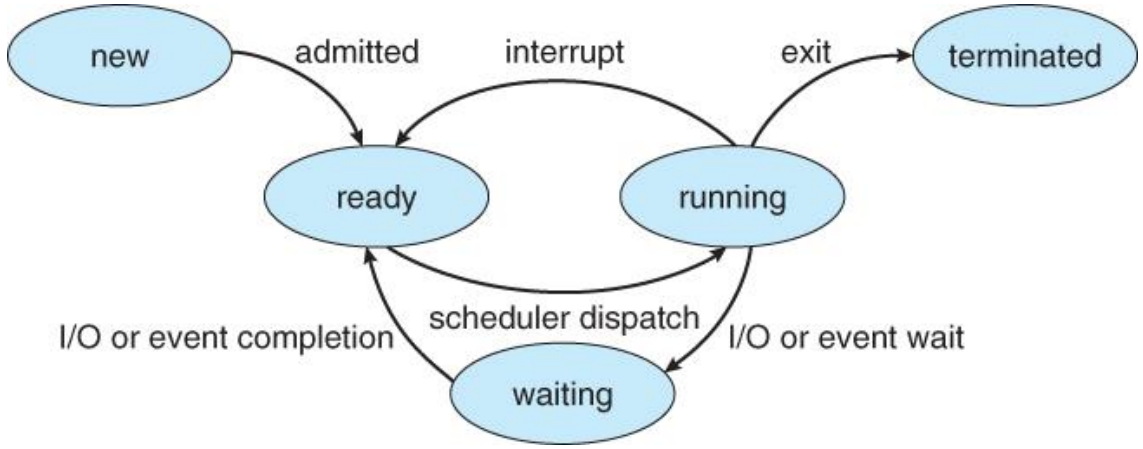
- una CPU ***dual core*** (*doppio nucleo*) unisce ***due processori*** indipendenti in un singolo package
- i primi dual core sono gli IBM PowerPC del ***2003***
- il termine ***multi core*** si usa per descrivere una CPU composta da tre o più core: più nuclei di processori fisici montati sullo stesso package

scheduler - gestore dei processi

- il gestore dei processi si occupa di controllare la ***sincronizzazione***, ***interruzione*** e ***riattivazione*** dei ***programmi in esecuzione*** cui viene assegnato un processore
- la gestione dei processi può seguire varie ***politiche*** in funzione del tipo di utilizzo cui il sistema è rivolto
- il modulo si occupa della distribuzione del ***tempo di CPU*** tra i vari ***processi attivi*** decidendone le modalità di avvicendamento
- in sistemi ***multi-processore*** si occupa anche di gestire la ***cooperazione*** tra le varie CPU

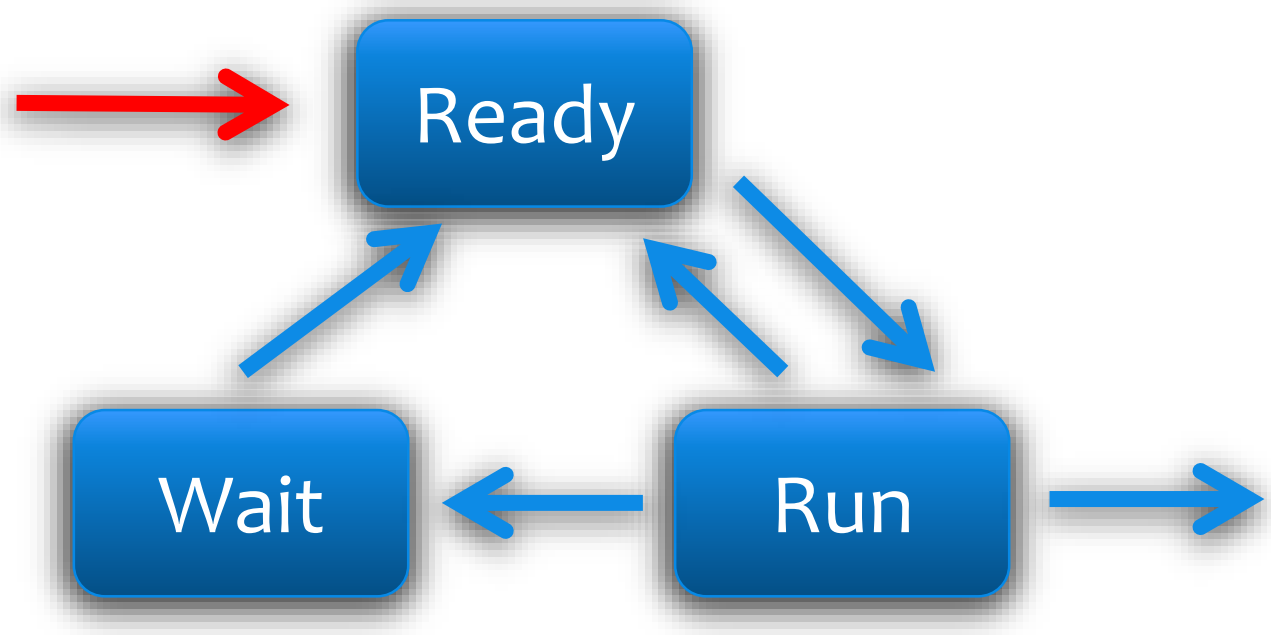
stati di un processo

- pronto (***ready***)
  - il processo non è in esecuzione, ha ***tutte le risorse*** necessarie ***tranne la CPU***
- attesa (***wait***) (sleep)
  - il processo non è in esecuzione, ha invocato un servizio (*es I/O*) ed è in ***attesa*** di un segnale da parte di un altro processo relativo al completamento del servizio richiesto
- esecuzione (***run***)
  - il processo dispone di tutte le risorse ed ***è in esecuzione sulla CPU***

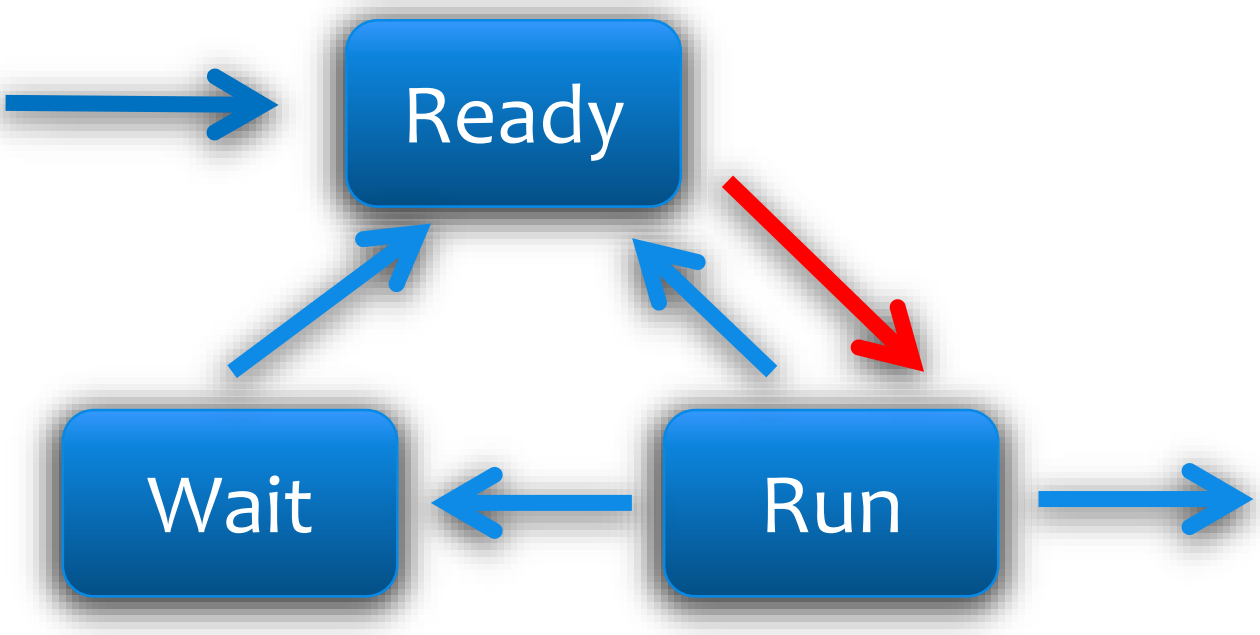


transazione 1

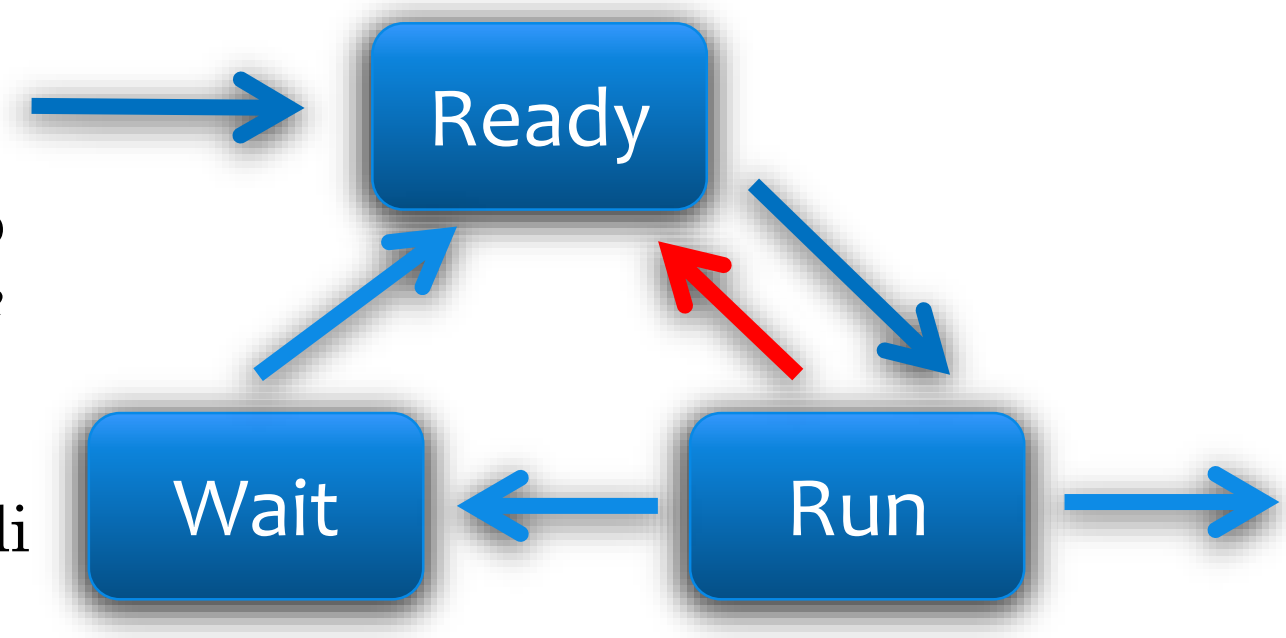
- il processo viene ***creato*** e posto nello stato di ***ready***
- più processi possono essere in stato di ready pronti per ottenere la risorsa CPU e passare in esecuzione
- i processi vengono messi in ***coda*** d'attesa



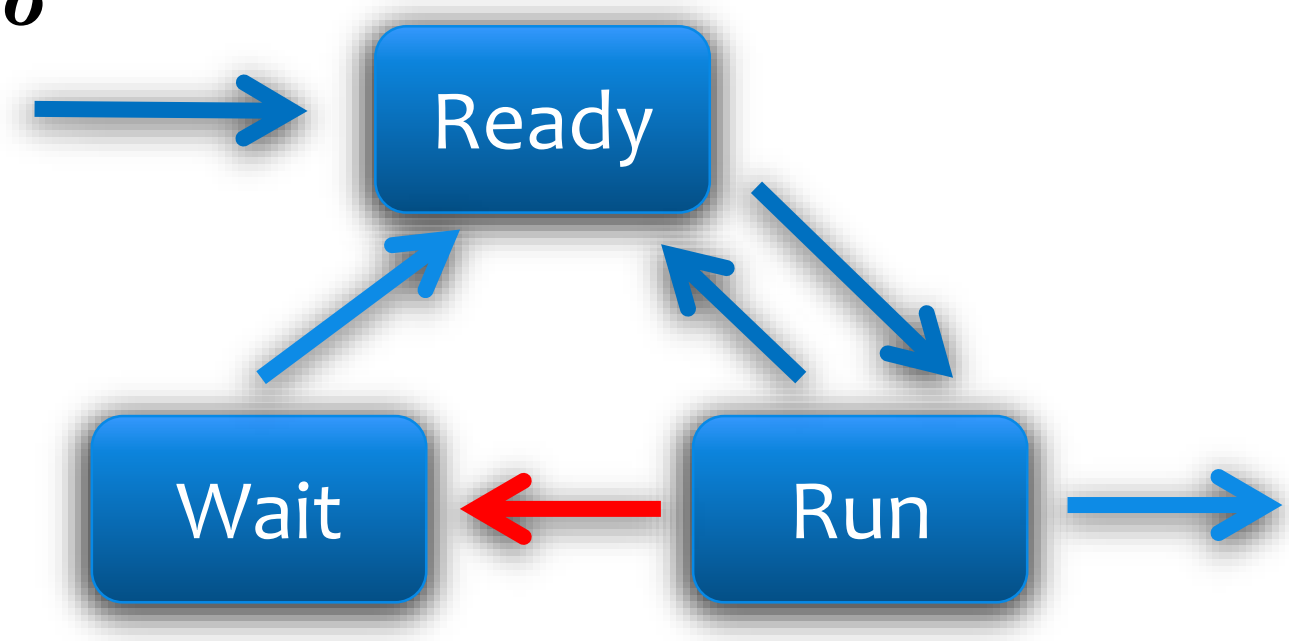
- il sistema operativo seleziona uno dei processi in stato ready e lo pone in esecuzione (***run***)



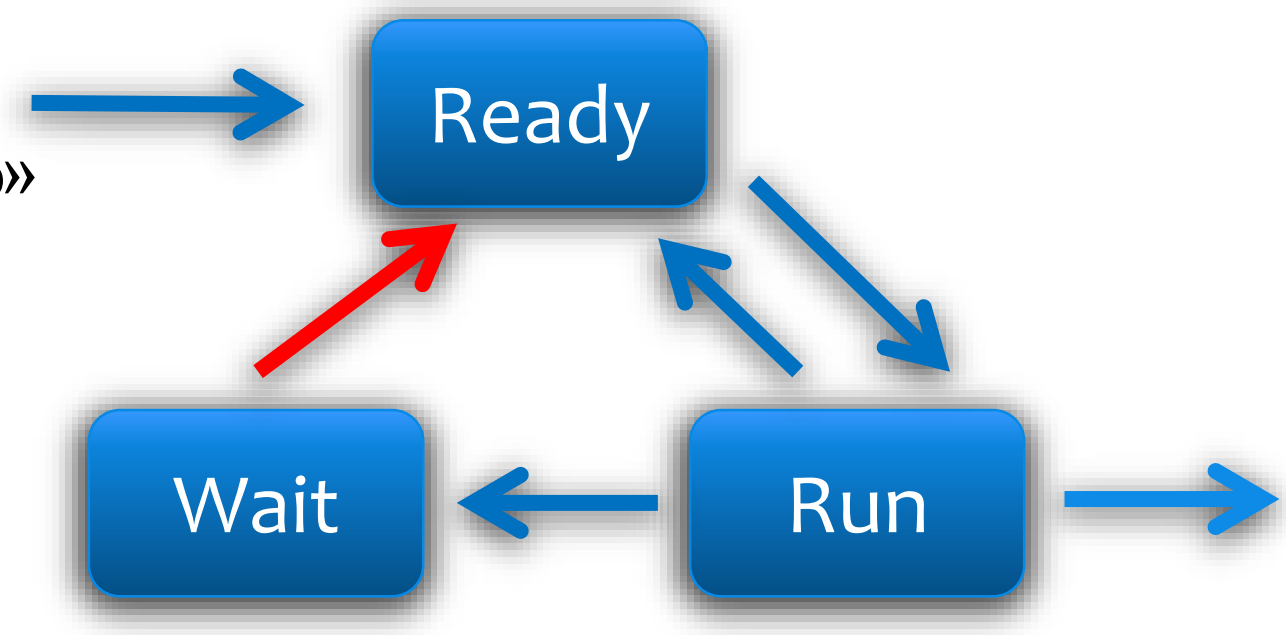
- il sistema operativo ***interrompe*** l'esecuzione di un processo e lo pone in stato di ***ready***
- Ad ogni processo viene assegnato un "quanto" di tempo (***time slice*** *dell'ordine dei millisecondi*) dopo il quale deve ***rilasciare*** il processore passando dallo stato di run a quello di ready



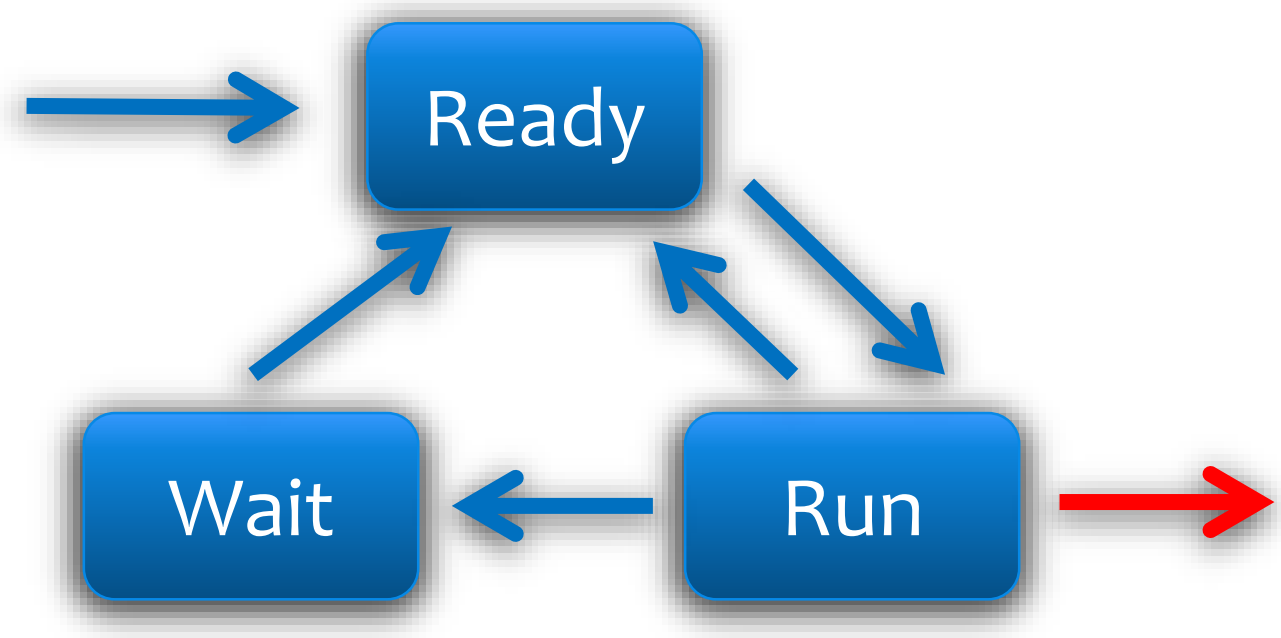
- se nel corso dell'esecuzione il processo richiede un ***servizio*** al sistema operativo viene posto nello stato di ***wait***



- se viene completata la richiesta di un processo o risulta disponibile la risorsa il processo viene «risvegliato» e passa allo stato di ready

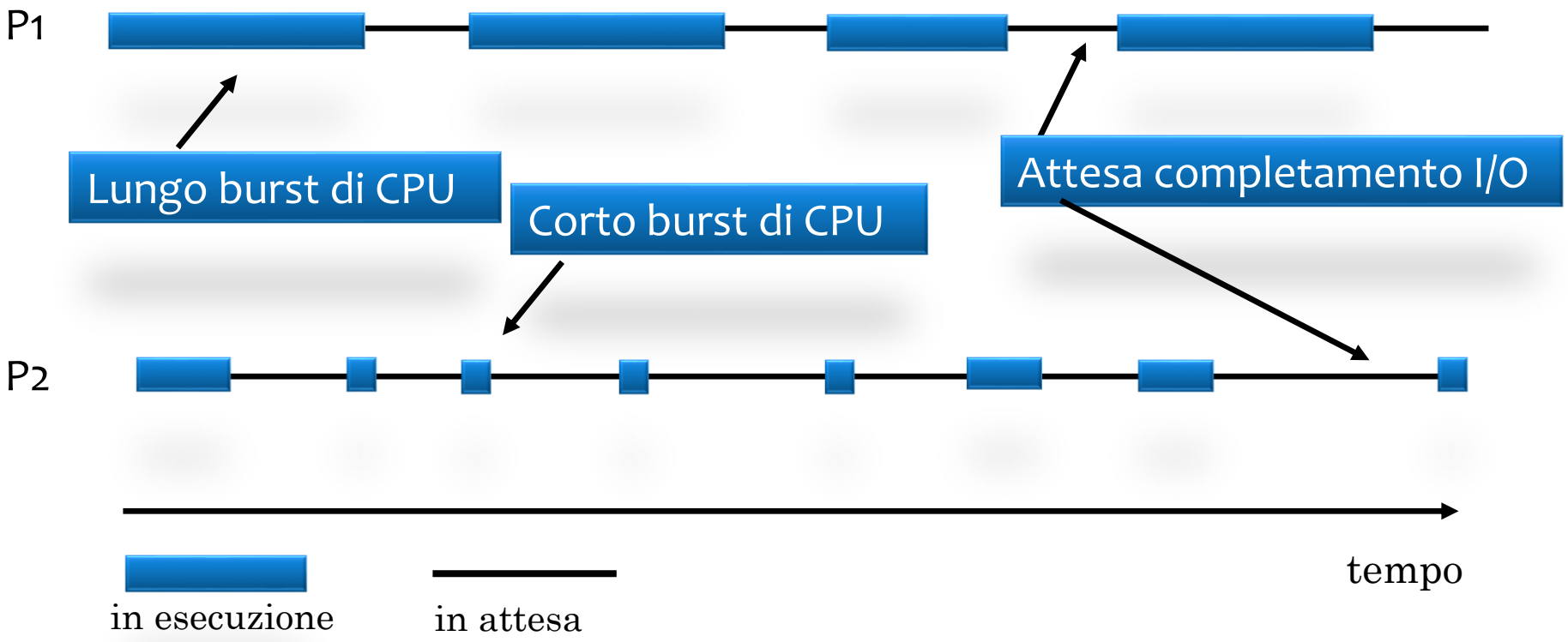


- se il processo è completato viene distrutto dal sistema operativo



tipi di processi

- ***CPU bound***
  - sfruttano pesantemente le risorse computazionali del processore, ma non richiedono servizi di ingresso/uscita dati al sistema operativo in quantità rilevanti
  - *Es. i programmi di calcolo matematico, i quali necessitano spesso di un'enorme potenza di calcolo, ma sfruttano l'I/O solo all'inizio della loro vita (per caricare gli input) ed alla fine di essa (per produrre gli output)*
- ***I/O bound***
  - effettuano molti accessi alle periferiche, il processo è spesso interrotto per attendere il completamento delle richieste di I/O


P1
Lungo burst di CPU
Corto burst di CPU
Attesa completamento I/O
P2
tempo
in esecuzione
in attesa

scheduler

- lo ***scheduler*** è una funzione del SO che gestisce l'insieme delle richieste di accesso ad una risorsa
  - stabilisce un ordinamento temporale per l'esecuzione di tali richieste
  - ha lo scopo di ottimizzare l'accesso alla risorsa
- lo ***short-time-scheduler*** (*scheduler della CPU*) deve garantire l'***esecuzione*** di ogni processo, in modo ordinato, concedendo e revocando la CPU alternativamente mediante la commutazione di contesto tra i processi

obiettivi dello scheduler

- ***fairness*** (*equità*)
  - processi della stesso tipo devono avere trattamenti simili
- ***massimizzazione*** del ***throughput*** (*produttività dell'intero sistema*)
  - minimizzazione dei tempi in cui la risorsa è inutilizzata
- ***minimizzazione*** del ***tempo di servizio*** della richiesta
  - da quando è generata a quando è soddisfatta
- evitare fenomeni indesiderati
  - esempio ***starvation*** (attesa eterna) di alcune richieste
    "*Corre voce che quando fu chiuso l'IBM 7094 al MIT, nel 1973, si scoprì che un processo con bassa priorità sottoposto nel 1967 non era ancora stato eseguito*"

politiche di scheduling

- esistono diverse ***politiche di scheduling***
  - ***FIFO*** (first in first out) lo scheduler può esegue le richieste in base al loro ordine di arrivo
  - dare precedenza alle richieste che impegnano per ***meno tempo*** la risorsa
  - ordinare le richieste in base a ***principi statistici*** per ottenere un obiettivo ottimale
- le ***politiche di scheduling*** sono raggruppabili in due macro categorie:
  - ***preemptive***
    - la CPU in uso da parte di un processo può essere tolta e passata a un altro in un qualsiasi momento
  - ***non preemptive***
    - una volta che un processo ha ottenuto l'uso della CPU non può essere interrotto fino a che lui stesso non la rilascia

esempi di algoritmi di scheduling

- ***round-robin***
  - time-sharing e politica FIFO di selezione dei processi in coda di ready
- ***scheduling a priorità***
  - un valore di ***priorità*** (*numero intero*) è associato a ciascun processo
  - la CPU viene allocata al processo con la priorità ***più alta***
  - il valore di priorità di un processo può variare nel corso della sua esecuzione

classi di scheduling

- il fattore tempo porta a identificare tre tipi di ***scheduling***:
  - scheduling a ***breve termine***
    - si occupa della selezione del nuovo processo da eseguire, quando il processo in esecuzione passa allo stato di attesa o ritorna allo stato di pronto
  - scheduling a ***medio termine***
    - coordina le operazioni di trasferimento temporaneo dei processi in memoria secondaria (swap out):
    - interviene nella gestione dei processi in attesa di eventi da lungo tempo
  - scheduling a ***lungo termine***
    - controlla il livello di multiprogrammazione
    - è impiegato nella scelta dei programmi da caricare in memoria centrale per la creazione dei corrispondenti processi
    - regola la presenza in memoria centrale di processi I/O bound e CPU bound

sistemi operativi

gestore della memoria

gestore della memoria

- tutti i ***processi*** in esecuzione necessitano della ***memoria centrale*** per memorizzare:
  - le istruzioni che fanno parte del ***codice***
  - i ***dati*** su cui operano
- il ***gestore della memoria*** è un modulo del sistema operativo incaricato di assegnare la memoria ai vari task
- la complessità del gestore della memoria dipende dal tipo di sistema
  - nei sistemi multi-tasking più programmi contemporaneamente possono essere caricati in memoria
- l'obiettivo è quello di ***allocare*** lo spazio in modo ***ottimale***

indirizzi

- indirizzi ***simbolici***
  - nel codice sorgente gli indirizzi sono espressi in modo simbolico
  - etichette simboliche (***label***) in assembler
  - ***identificatori*** di variabili nei linguaggi ad alto livello
- indirizzi ***logici***
  - l'assemblatore (o il compilatore) trasforma gli indirizzi simbolici in ***valori*** che non rappresentano ancora però un vero indirizzo di memoria
  - in genere gli indirizzi logici sono calcolati a partire da un ***indirizzo iniziale*** che vale in genere zero (*indirizzi binari logici relativi a zero*)
- indirizzi ***virtuali***
  - il linker completa i riferimenti ai vari moduli e genera indirizzi che non sono ancora però riferimenti fisici alla memoria

rilocazione

- indirizzi ***fisici***
  - gli indirizzi virtuali sono tradotti in indirizzi di memoria fisica
  - la traduzione degli indirizzi virtuali in indirizzi fisici è definita ***rilocazione***
- per evitare che un processo faccia riferimento a zone di memoria appartenenti ad alti processi molti sistemi utilizzano registri speciali di protezione (***registri limite***) per controllare ed eventualmente inibire riferimenti non autorizzati
- questi metodi sono ottenibili con sistemi dotati di ***MMU*** (***m***emory ***m***anagement ***u***nit)
  - i registri ***base*** e ***limite*** sono elementi della MMU del sistema

rilocazione statica

- il ***loader***, dopo aver letto l'intero programma e prima di porlo in esecuzione, rialloca in memoria tutto il codice ***adattando gli indirizzi*** virtuali alle posizioni attualmente disponibili nella memoria fisica
- il loader in questo caso viene detto ***caricatore rilocante***
  - dopo la rilocazione statica il programma in memoria ha riferimenti di memoria fisica che non potranno più cambiare per tutta la durata dell'esecuzione

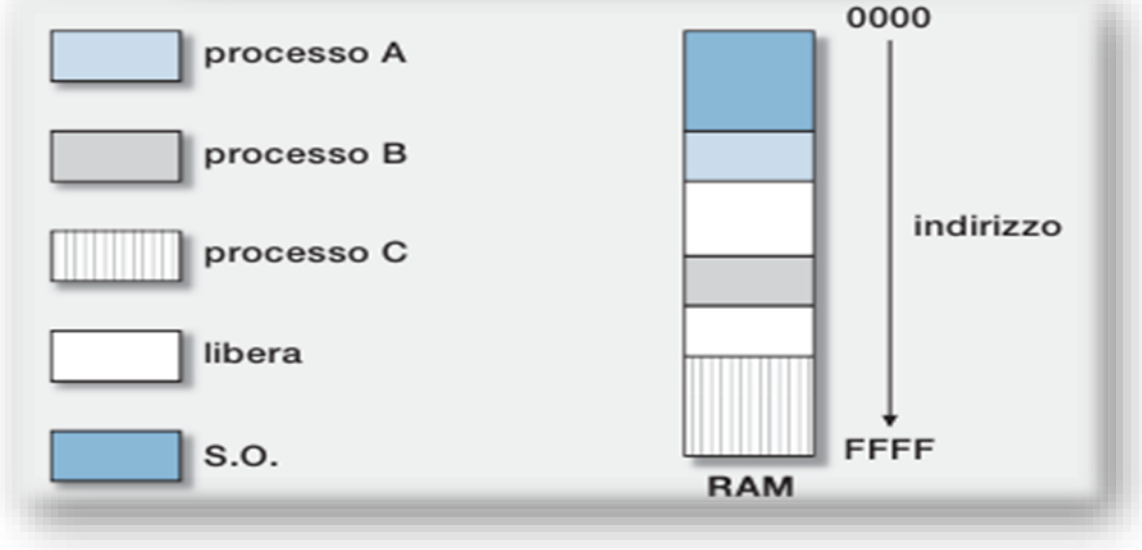
rilocazione dinamica

- il sistema operativo ***rialloca*** il codice durante l'***esecuzione*** del programma, utilizzando elementi di MMU
  - il caricatore non riloca gli indirizzi virtuali del linker ma li carica in memoria così come il linker li ha generati nel file eseguibile
- ***run-time*** ogni indirizzo viene ***tradotto*** nel corrispondente indirizzo fisico prima di accedere alla memoria
- la rilocazione dinamica permette ai programmi di essere caricati in ***aree*** di memoria ***differenti*** durante ***l'esecuzione***
  - è sufficiente modificare le informazioni contenute nel meccanismo hardware che realizza la funzione di rilocazione (*meccanismo di MMU*)

memoria suddivisa in settori

- a ogni processo viene assegnato un ***settore*** di memoria compreso fra un ***indirizzo iniziale*** e uno ***finale***
- Iil gestore della memoria gestisce una ***tabella di indirizzi*** in cui associa ad ogni processo il suo indirizzo iniziale e finale

Processo	Indirizzo iniziale	Indirizzo finale
S.O.	0000H	4FFFH
A	5000H	7FFFH
B	A000H	BFFFH
C	D000H	FFFFH



memoria suddivisa in settori: indirizzi

- indirizzo ***logico*** (*generato staticamente*)
  - il compilatore assume che l'indirizzo di partenza per i dati e le istruzioni sia l'indirizzo 0 (***zero***)
- indirizzo ***fisico*** (*generato dinamicamente*)
  - in fase di esecuzione gli indirizzi vengono ***traslati*** sommando l'indirizzo di partenza del settore associato al processo (*operazione effettuata run time dalla MMU*)
- un indirizzo di memoria contiene una parte che identifica un ***settore***, e una parte che specifica l'***offset*** entro il settore

memoria suddivisa in settori: problemi

- il problema principale è la ***frammentazione*** della memoria:
  - quando termina un processo viene ***rilasciato*** il suo ***settore*** di memoria che può essere associato a un nuovo processo che richiede una quantità di memoria ***minore o uguale*** a quella rilasciata
  - le parti ***inutilizzate*** dei settori portano a una progressiva frammentazione
    - potrebbe essere disponibile memoria sufficiente per allocare un processo ma non in un settore continuo

memoria suddivisa in settori: strategie di allocazione

- ***first fit***
  - individua la prima partizione che può contenere il processo
    - tra le partizioni disponibili viene scelta quella con indirizzi più bassi
  - è efficiente per mantenere compattate le zone rilasciate
- ***best fit***
  - ricerca nella tabella la partizione ***più piccola*** che può contenere il processo
  - si vengono a creare numerose partizioni libere molto piccole (si aumenta la frammentazione)
- ***worst fit***
  - tra le partizioni libere che possono contenere il processo si sceglie quella più ampia per attenuare l'effetto della frammentazione

memoria suddivisa in settori: compattazione

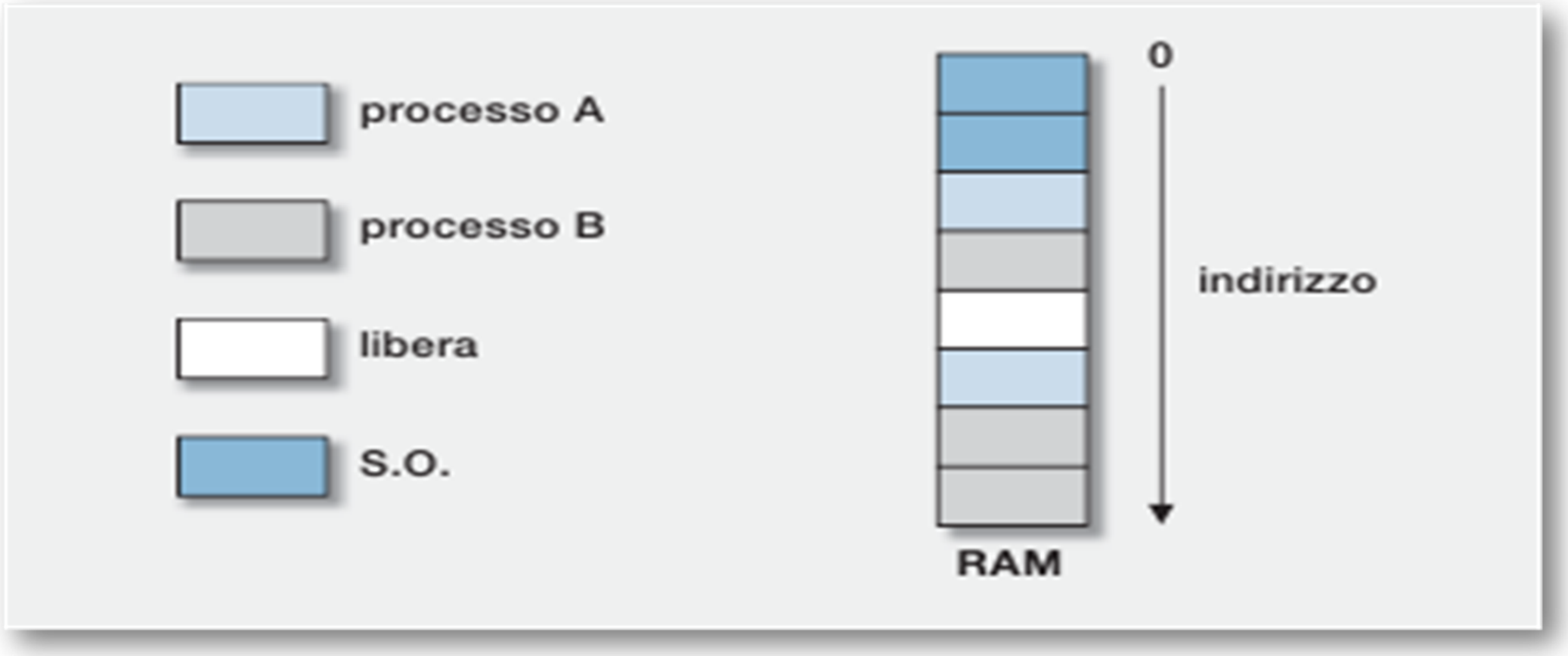
- in alcuni casi la riduzione della frammentazione si può ottenere con una tecnica detta di ***compattazione*** della memoria
- il gestore della memoria predispone un ***algoritmo*** che ***periodicamente*** controlla lo stato della memoria e quando necessario interrompe le esecuzioni per ***compattare*** in modo ***contiguo*** tutta la memoria allocata eliminando i buchi e aggiornando la tabella della memoria

paginazione

- la memoria viene «*vista*» dal processore come un array di ***settori*** aventi tutti la ***stessa dimensione*** predefinita
- il sistema operativo assegna a ogni processo in esecuzione un ***numero*** di ***pagine*** sufficiente per contenere il codice e i dati
- le pagine ***non*** sono necessariamente ***contigue***

Pagina	0	1	2	3	4	5	6	7
Processo/Stato	S.O.	S.O.	A	B	libera	A	B	B

paginazione: organizzazione della memoria



paginazione: traslazione

- per la ***traslazione degli indirizzi*** ogni processo è dotato di una ***tabella*** di corrispondenza fra pagina logica e pagina fisica
- il compito della ***MMU*** è più ***complesso*** per la traduzione da indirizzo logico a indirizzo fisico:
  - si individua la ***pagina logica***
  - poi si individua la corrispondente ***pagina fisica***

Processo A

Pagina logica	Indirizzo base logico	Pagina fisica	Indirizzo base fisico
0	0x0000	2	0x4000
1	0x2000	5	0xA000

Processo B

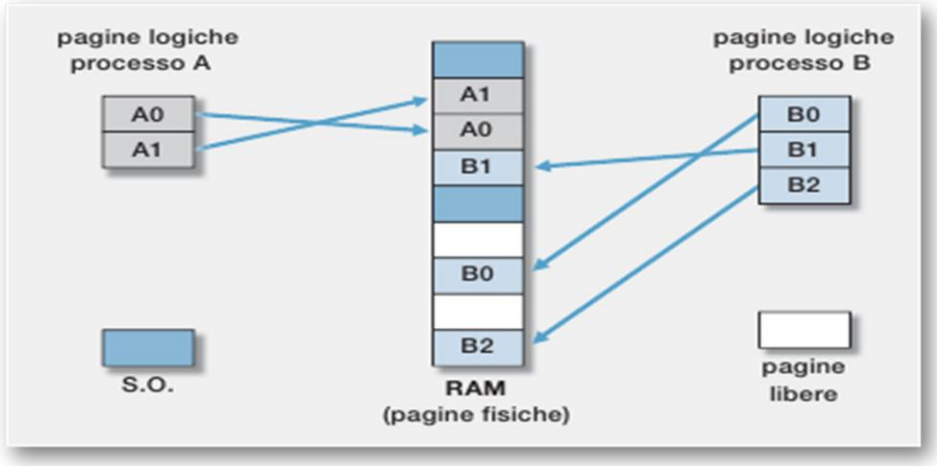
Pagina logica	Indirizzo base logico	Pagina fisica	Indirizzo base fisico
0	0x0000	3	0x6000
1	0x2000	6	0xC000
2	0x4000	7	0xE000

paginazione: dimensione delle pagine

- se la dimensione di una pagina è una ***potenza di 2*** gli indirizzi vengono di fatto spezzati in due:
  - i primi bit determinano la pagina
  - i successivi determinano l'indirizzo interno alla pagina (***offset***)
- la ***traslazione*** sostituisce i bit relativi alla pagina fisica con quelli relativi alla pagina logica e lascia inalterati i restanti bit

paginazione: frammentazione

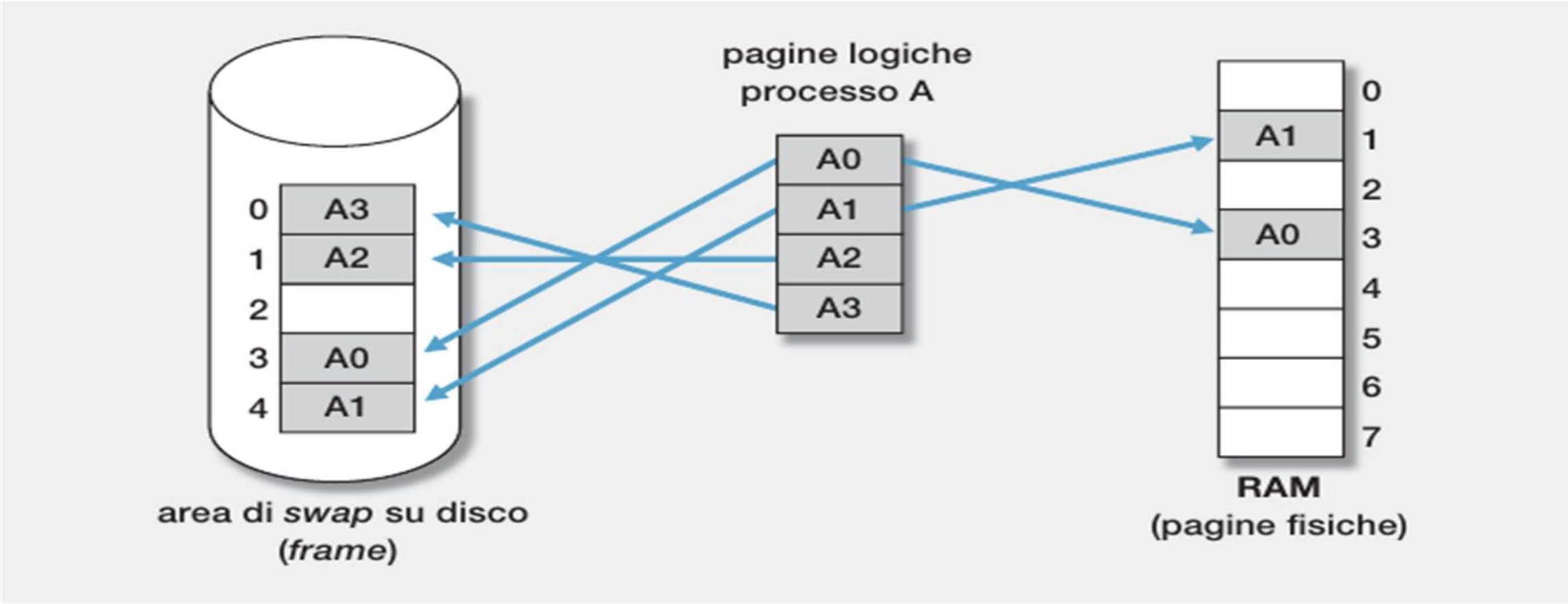
- il problema della ***frammentazione*** è ***risolto***
- al termine di un processo vengono "***liberate***" tutte le pagine utilizzate da questo
- un nuovo processo ha a disposizione tutte le ***pagine*** rimaste "***libere***"



memoria virtuale

- i processi attivi in un sistema operativo multitasking sono molti ed è probabile che il numero complessivo delle ***pagine richieste*** da tutti i processi sia ***superiore*** al numero di pagine di memoria effettivamente ***disponibili***
- tutti i moderni sistemi operativi implementano la tecnica della ***"memoria virtuale"***
  - il gestore della memoria mette a disposizione un ***numero*** di pagine ***superiore*** a quelle presenti nella memoria fisica
  - alcune ***pagine*** sono salvate ***temporaneamente*** nella ***memoria di massa***
- la memoria di massa utilizzata a questo scopo è comunemente chiamata, in ambiente Unix-Linux, ***swap*** o spazio di swap, mentre, in ambiente Windows, è chiamata file di ***paging***

pagine logiche e pagine fisiche



prestazioni

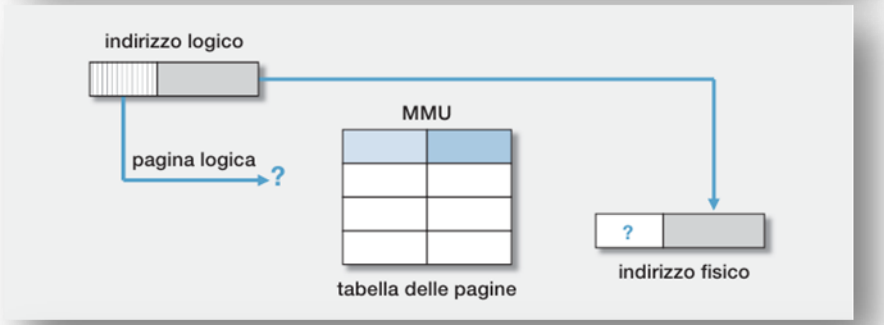
- la memoria di massa ha ***tempi*** di accesso estremamente ***più lenti*** della memoria centrale
  - è necessario quindi ***ridurre*** al minimo le operazioni di ***swapping*** (spostamento delle pagine dallo spazio di swap alla memoria fisica)
- normalmente un processo in ogni fase della sua esecuzione fa riferimento a ***istruzioni*** e ***dati*** contenuti in poche pagine di memoria ***contigue***
  - ***località del codice***
    - l'esempio classico è un ciclo che ripete più volte istruzioni consecutive
  - ***località dei dati***
    - la struttura più comunemente utilizzata è l'***array*** in cui i dati sono contigui in memoria

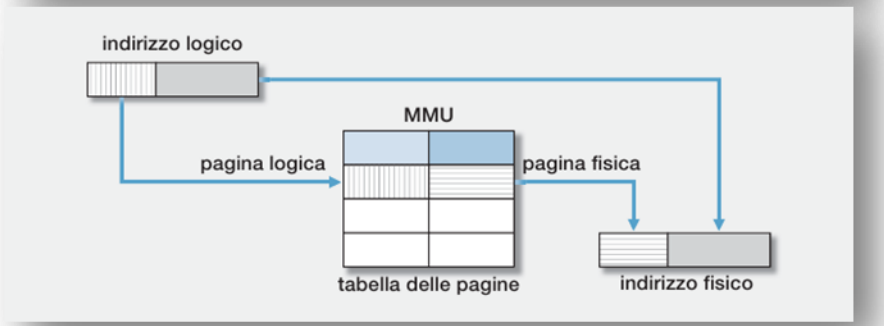
page fault

- la MMU trasla gli indirizzi
- se la pagina ***non*** è ***presente*** in memoria (***page fault***) il processo viene posto in stato di ***wait*** in attesa che la pagina venga caricata in memoria
- il gestore della memoria ***recupera*** la pagina, ***aggiorna*** la ***tabella*** delle pagine e riporta il processo in ready

Processo A

Pagina logica	Indirizzo base logico	Presente	Pagina fisica	Indirizzo base fisico	Posizione su disco
0	0x0000	SÌ	3	0x6000	3
1	0x2000	SÌ	1	0x2000	4
2	0x4000	NO			1
3	0x6000	NO			0





page fault

- ogni ***page fault rallenta*** drasticamente l'esecuzione di un processo che deve transitare dallo stato di wait e deve attendere il caricamento della pagina dalla memoria di massa
- se non sono disponibili pagine libere in memoria centrale è necessario ***sostituire*** una pagina dello stesso processo (***allocazione locale***) o di un altro processo (***allocazione globale***)

politiche per la gestione del page fault

- ***FIFO*** (First In First Out)
  - la pagina da ***rimuovere*** è la ***prima*** che è stata caricata
  - l'idea è che le pagine "***vecchie***" non vengano più utilizzate in futuro
- ***LRU*** (Least Recently Used)
  - la pagina da rimuovere è quella ***inutilizzata*** da ***più tempo***
  - l'idea è che se non è utilizzata da molto tempo non verrà più utilizzata
- ***LFU*** (Least Frequently Used)
  - la pagina da rimuovere è quella ***meno utilizzata***
  - l'idea è che se è stata poco utilizzata sarà poco utilizzata anche in futuro

strumenti per le politiche di page fault

- ***FIFO***
  - è sufficiente memorizzare per ogni pagina il ***momento*** del caricamento in memoria
- ***LRU***
  - deve essere memorizzato il ***tempo*** ad ogni ***accesso*** alla pagina
- ***LFU***
  - deve essere memorizzato un ***contatore*** incrementato ad ogni accesso alla pagina
  - per semplificare la gestione si utilizza un ***bit*** che viene settato se si fa accesso a una pagina (i bit vengono periodicamente azzerati)
  - la politica diventa quindi *NRU* (Not Recently Used)

swapping

- se una pagina viene eliminata dalla memoria centrale deve essere ***copiata*** sulla memoria di massa
- un "***dirty bit***" settato se la pagina viene modificata può essere utilizzato per evitare questa fase di riscrittura su disco
  - (molto spesso per le pagine di codice)

sistemi operativi

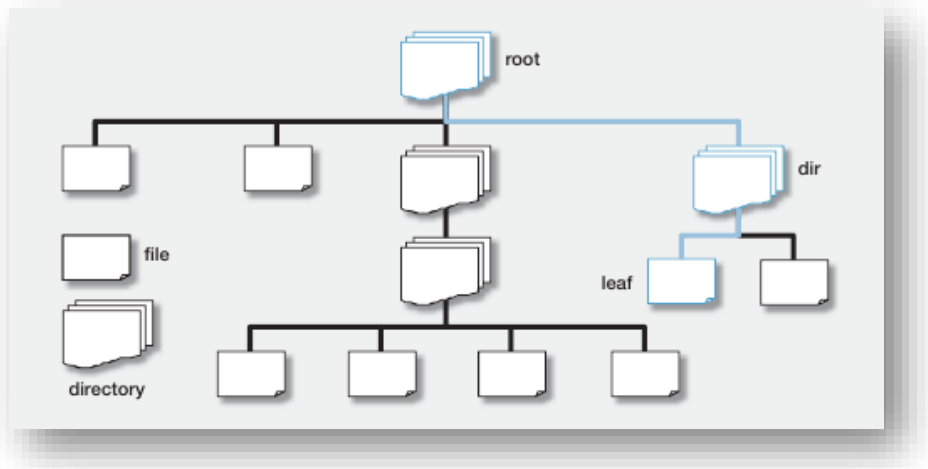
gestore del file system

file system

- il ***file system*** è la parte del sistema operativo che permette la gestione delle informazioni memorizzate in modo ***permanente*** (i file sui vari sistemi di memorizzazione di massa)
- deve garantire la ***correttezza*** e la ***coerenza*** delle informazioni
- nei sistemi multi-utente, deve mettere a disposizione dei meccanismi di ***protezione*** in modo tale da consentire agli utenti di proteggere i propri dati dall'accesso da parte di altri utenti non autorizzati
- i file system possono essere rappresentati sia ***graficamente*** tramite file browser sia ***testualmente*** tramite shell testuale
- nella rappresentazione grafica (GUI) è generalmente utilizzata la metafora delle ***cartelle*** che contengono documenti (i file) ed altre sottocartelle

- le funzioni tipiche che deve svolgere sono:
  - fornire un meccanismo per l'***identificazione*** dei file
  - fornire opportuni metodi per ***accedere*** ai dati
  - rendere ***trasparente*** la struttura fisica del supporto di memorizzazione
  - implementare meccanismi di ***protezione*** dei dati

file e directory



- organizzazione del file system come ***albero***: la directory di livello più alto è la ***radice***, i file sono le foglie
- ogni singolo file è univocamente individuato dal percorso (***pathname***) che lo connette alla radice

operare con i file

- ***operazioni***:
  - ***navigazione*** nell'albero delle directory
  - ***ricerca***
  - ***elencazione***
  - ***eliminazione***
  - ***ridenominazione***
  - ***copia*** o ***spostamento***
- per il SO i file sono semplici sequenze di byte (***byte stream***) che vengono poi opportunamente interpretate dai programmi applicativi

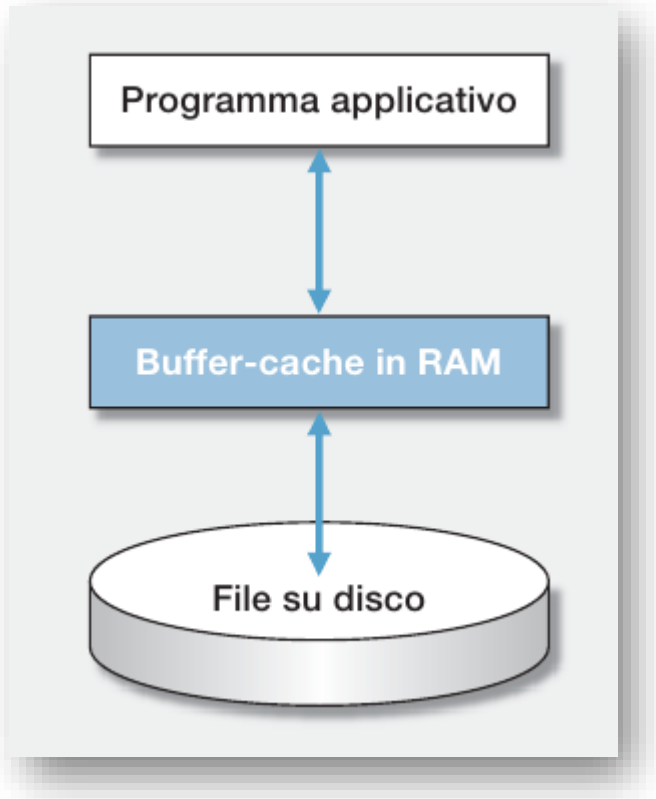
- il sistema operativo deve garantire la il ***controllo*** dell'***accesso*** ai file e alle directory
- gli utenti privi di specifici ***privilegi*** non hanno la possibilità di leggere, scrivere, creare, eliminare, rinominare i file

tipi di file

- ***estensione***
  - caratteri che seguono l'ultimo . nel nome del file
- ***formato***
  - codice identificativo memorizzato nei primi byte del file (***magic number***)
  - i magic number sono nati negli ambienti Unix per identificare il formato dei file
  - un magic number è costituito da un numero di byte variabile
    - *i file immagine GIF cominciano sempre con la stringa ASCII GIF87a o GIF89a*
    - *le classi Java compilate hanno il magic number CAFEBABE, espresso in notazione esadecimale. Probabilmente è un altro riferimento al caffè che di Java è simbolo e nome e all'ipotetica cameriera che lo serve*
    - *i file ZIP cominciano tutti per PK (in esadecimale 50 4B), dalle iniziali del nome dell'ideatore Phil Katz*

bufferizzazione

- per ***ridurre*** il numero di accessi al disco vengono letti più blocchi consecutivi e memorizzati in un ***buffer*** nella RAM
- le operazioni di ***lettura scrittura*** possono operare direttamente sul buffer ed essere "salvate" su disco solo al termine dell'esecuzione



integrità dei dati

- ***Journaling***
  - in un file di log vengono memorizzate le operazioni da effettuare sul file e quelle effettivamente concluse
- in caso di ***malfunzionamento*** è possibile sapere quali operazioni non hanno avuto successo e ***ripristinare*** uno stato consistente

sistemi operativi

gestore dispositivi I/O

connessioni HW

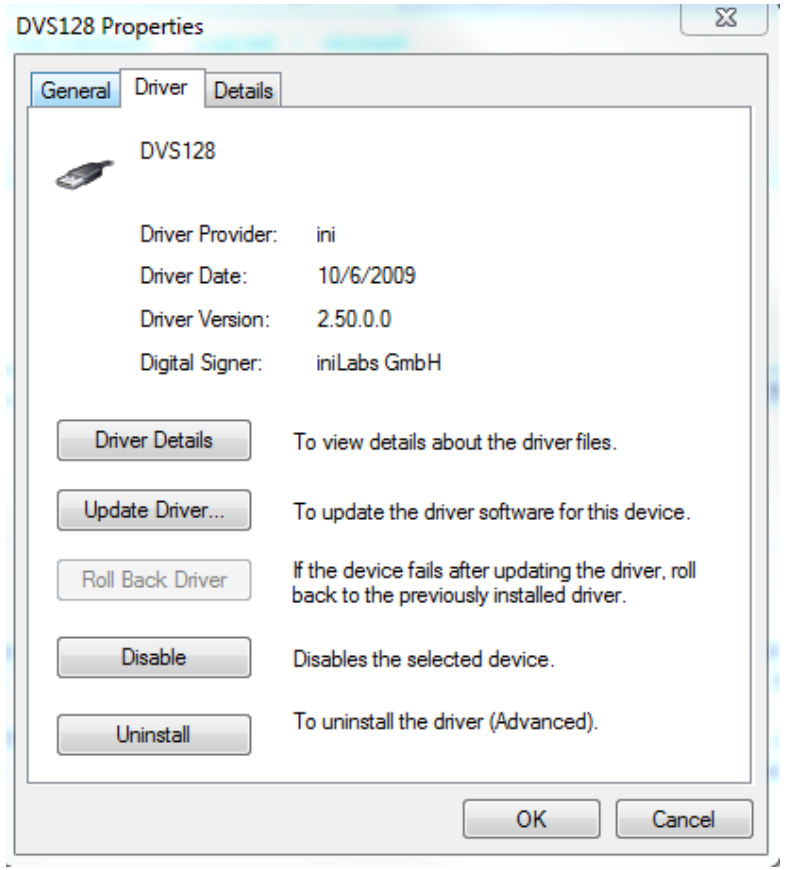
- il computer dispone di ***connessioni hardware*** per i dispositivi di input/output
  - tastiera
  - mouse
  - monitor
  - stampante
  - …

gestore dei dispositivi di I/O

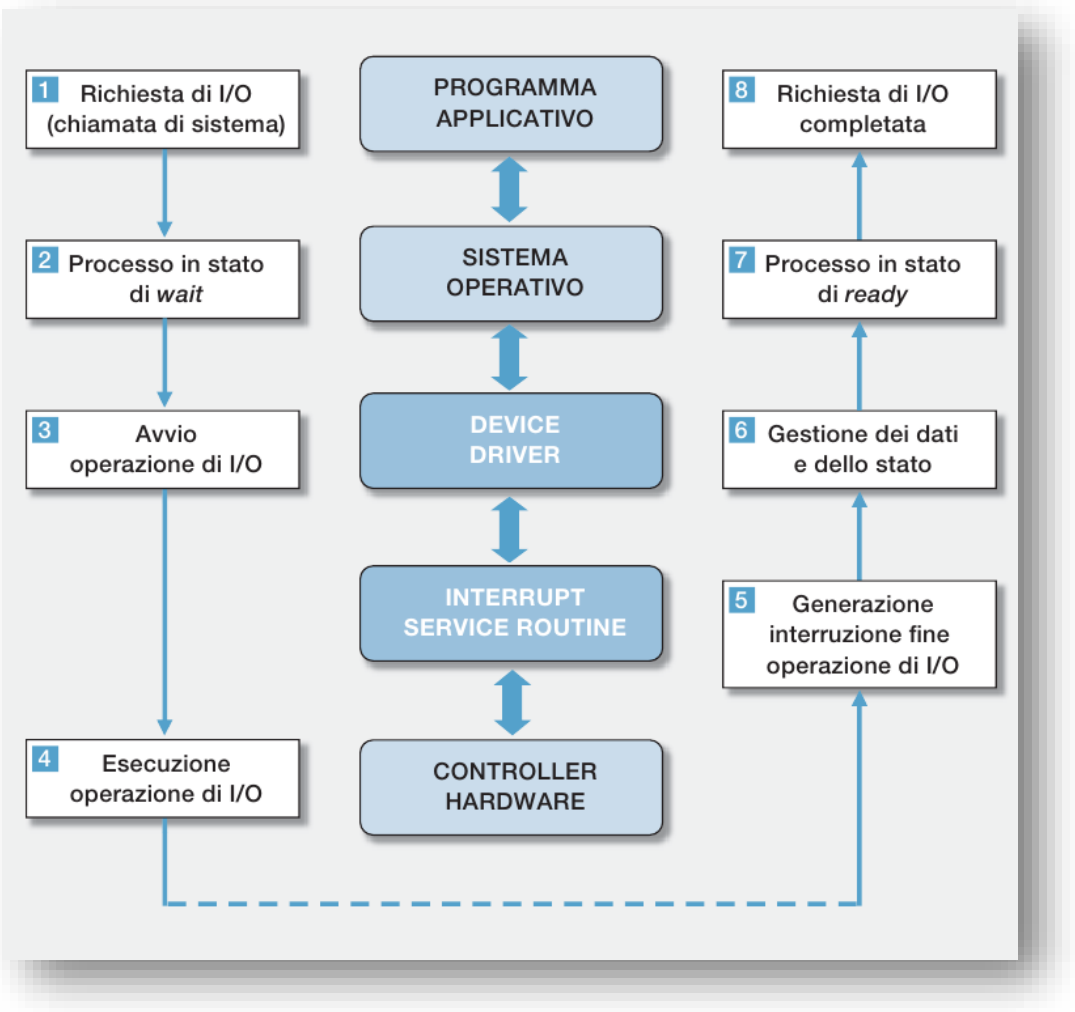
- per ogni dispositivo esiste un driver (***device driver***) che permette al sistema operativo di gestire il dispositivo stesso
- i driver devono essere ***installati*** manualmente o automaticamente nel sistema
- il driver ***integra*** il codice del sistema operativo con una serie di funzioni che permettono di operare con il dispositivo
- il ***gestore dei dispositivi di I/O*** è un modulo del sistema operativo incaricato di ***assegnare*** i ***dispositivi*** ai ***task*** che ne fanno richiesta e di controllare i dispositivi stessi

device driver

- i device driver sono realizzati dai ***produttori*** dei dispositivi
- rendono ***trasparenti*** le ***caratteristiche fisiche***
- gestiscono la ***comunicazione*** dei segnali verso i dispositivi
- gestiscono i ***conflitti***, nel caso di accesso contemporaneo di più task al dispositivo
- ***plug & play***: il SO "riconosce" la categoria del dispositivo e "ricerca" il driver fra quelli installati o cerca di scaricarlo dalla rete



gestione di una operazione di I/O



bufferizzazione

- la CPU e i dispositivi di I/O operano a ***velocità sensibilmente differenti***
- per ***evitare attese*** e rendere il più possibili ***asincrone*** le esecuzioni dei processi si utilizza la tecnica della bufferizzazione
- *esempio*
  - se la ***CPU*** (alta velocità) deve spedire dati alla ***stampante*** (velocità molto minore)
    - ***scrive*** i dati nel ***buffer*** di memoria e continua il suo lavoro
    - la stampante stampa i dati ***leggendo*** dal ***buffer*** e non interrompe la CPU

interrupt service routine