

Introduzione all'Informatica

Riccardo Ortale



Sistema Operativo

FIRMWARE: IL BIOS

- ◆ BIOS = Basic Input-Output System
 - gestisce direttamente le risorse hardware e offre delle funzionalità standard di accesso
- ◆ risiede su un chip di memoria permanente
 - ROM, RAM + batteria di alimentazione
- ◆ gestisce la procedura di avvio del calcolatore, che consiste delle seguenti fasi
 - diagnostica
 - inizializzazione delle risorse hardware (setup)
 - caricamento (da disco rigido in RAM) ed esecuzione della routine di bootstrap, che provvede quindi a caricare il sistema operativo

SISTEMA OPERATIVO (S.O.)

- ◆ Strato di programmi che opera al di sopra di hardware e firmware e gestisce l'elaboratore
- ◆ Come **Gestore delle Risorse**
 - controlla e gestisce tutte le funzioni del calcolatore in modo efficiente
 - accetta e soddisfa le richieste dell'utente
 - funziona come mediatore tra risorse in conflitto
 - tiene traccia di chi utilizza le risorse
- ◆ Come **Macchina estesa**
 - costituisce una base sulla quale è possibile scrivere programmi applicativi
 - rappresenta all'utente una macchina estesa più facile da programmare

FUNZIONI DI UN S.O.

- ◆ Esecuzione di applicazioni
 - Caricamento dei programmi (istruzioni e dati) nella memoria centrale
- ◆ Accesso ai dispositivi di I/O
 - Gestione dei segnali per il trasferimento dei dati
 - Operazioni astratte di lettura/scrittura
- ◆ Archiviazione di dati e di programmi
 - Organizzazione logica dei dati (directory, file)
- ◆ Controllo di accesso
 - Condivisione di risorse da parte di più utenti o applicazioni
 - Meccanismi di protezione
- ◆ Contabilizzazione
 - Monitoraggio dell'uso delle risorse da parte di utenti e/o applicazioni (ottimizzazione/fatturazione)
- ◆ Gestione dei malfunzionamenti
 - rilevare e risolvere guasti hardware e operazioni scorrette del software

PROGRAMMI APPLICATIVI VS S.O.

➤ Programmi applicativi

- hanno accesso a un insieme ridotto di risorse;
- possono utilizzare solo un sottoinsieme delle istruzioni del processore (esecuzione in **modalità utente**);
- non possono decidere autonomamente quando e come avere accesso alle risorse del sistema (richiedono al sistema operativo l'esecuzione di alcuni servizi);

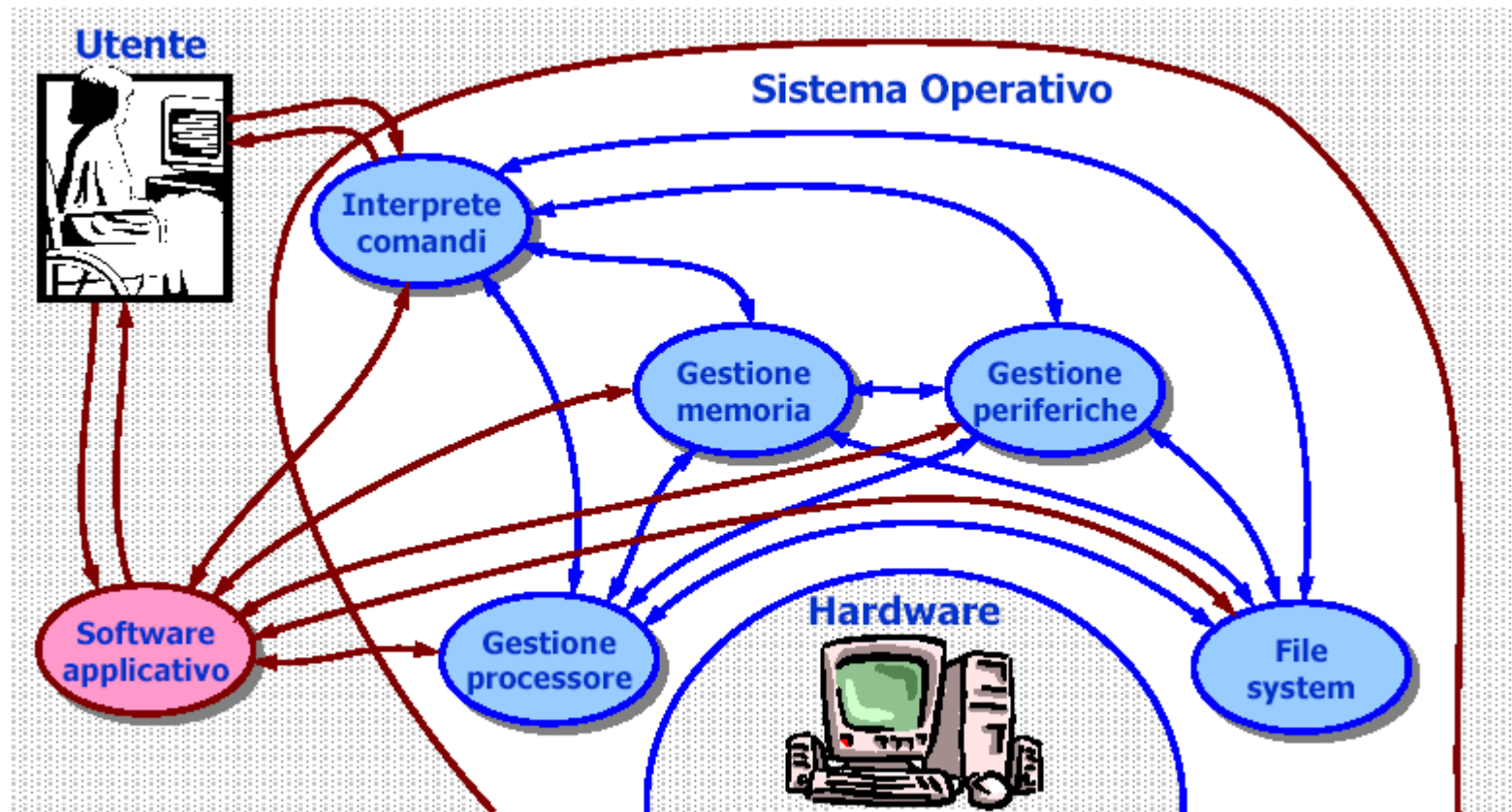
➤ Sistema operativo

- ha accesso a tutte le risorse;
- può utilizzare tutte le istruzioni del processore (esecuzione in **modalità supervisore**);
- stabilisce in che ordine e come le richieste che riceve devono essere soddisfatte;
- ...

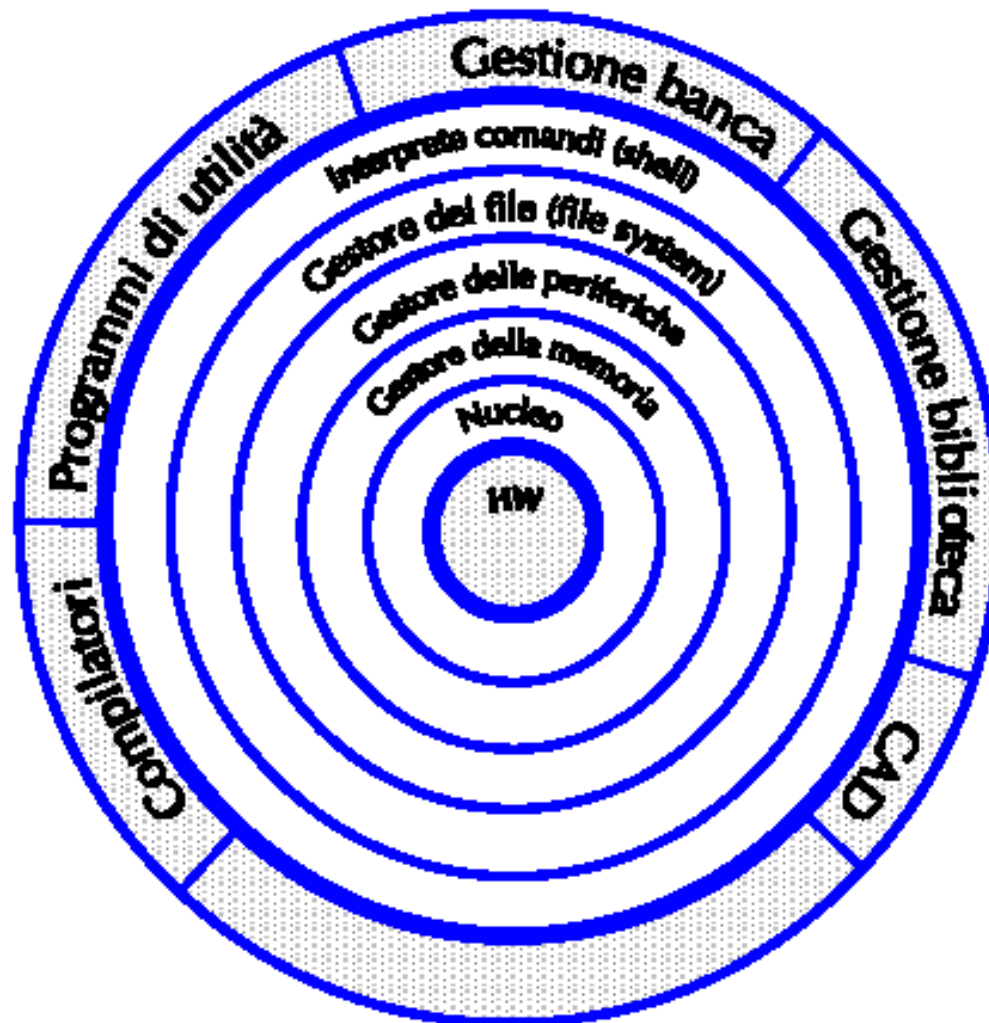
ELEMENTI DI UN S.O.

- Sistema di **gestione del processore**,
 - controlla l'unità centrale di elaborazione (CPU);
 - definisce quali programmi sono da eseguire e quali compiti sono da assegnare alla CPU;
- Sistema di **gestione della memoria**,
 - controlla l'allocazione della memoria di lavoro ai diversi programmi che possono essere contemporaneamente in esecuzione;
- Sistema di **gestione delle periferiche**,
 - garantisce l'accesso ai dispositivi di ingresso/uscita,
 - maschera i dettagli di basso livello e gli eventuali conflitti che possono insorgere nel caso che diverse richieste arrivino contemporaneamente a uno stesso dispositivo;
- Sistema di **gestione dei file (file system)**
 - consente l'archiviazione e il reperimento dei dati sfruttando le periferiche che costituiscono la memoria di massa;
- Sistema di **gestione degli utenti e dei relativi comandi (interprete comandi)**,
 - interfaccia diretta con gli utenti,
 - permette agli utenti di accedere in maniera semplice e intuitiva alle funzionalità disponibili.

ELEMENTI DI UN S.O.



MODELLO A STRATI DI UN S.O.



CLASSIFICAZIONE DEI S.O.

◆ In base al numero di utenti:

- **mono-utente (mono-user)**
 - un solo utente alla volta può utilizzare il sistema
- **multi-utente (multi-user)**
 - più utenti in contemporanea interagiscono con la macchina
 - il S.O. fornisce a ciascuno l'astrazione di un sistema “dedicato”

◆ In base al numero di processi:

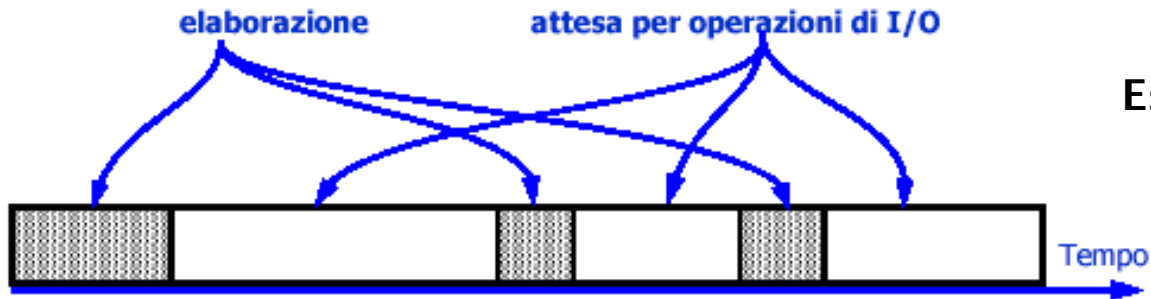
- **Mono-programmato (mono-task)**
 - si può eseguire un solo programma per volta
- **Multi-programmato (multi-task)**
 - il SO permette di eseguire più programmi in contemporanea
 - il SO gestisce la suddivisione del tempo della CPU fra i vari processi (*time-sharing*)

LIMITI DEI S.O. MONOPROGRAMMATI

- ◆ Qualunque programma alterna fasi di esecuzione a fasi in cui è bloccato in attesa di qualche evento esterno
 - attesa che sia terminata un'operazione di input
 - attesa per usare una risorsa al momento occupata
- ◆ Sotto-utilizzo del processore
 - mentre il programma è bloccato in attesa di eventi esterni, il processore rimane inattivo (*idle*)
 - i tempi di lavoro delle periferiche di input/output, o addirittura i tempi di reazione umani sono maggiori di molti ordini di grandezza della velocità del processore

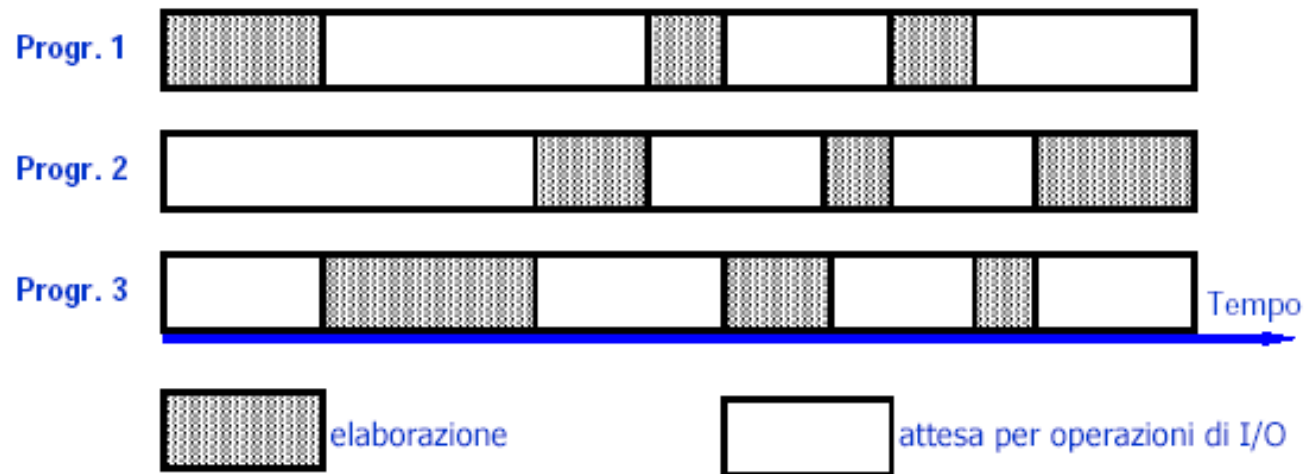
MULTIPROGRAMMAZIONE

Sistema monoprogrammato: mono-tasking



Es. Programma videoscrittura
Utilizzo CPU 2-5%

Sistema multiprogrammato: multi-tasking e time-sharing



MULTIPROGRAMMAZIONE

- ◆ Il tempo di lavoro della CPU è diviso tra i vari programmi
 - Ad ogni istante vi è un solo programma attivo
 - Il processore alterna l'esecuzione dei vari programmi
- ◆ Se l'alternanza tra i programmi è frequente (es. 10/100 ms), si ha l'impressione di un'esecuzione simultanea
 - a livello macroscopico si ha quindi l'impressione della contemporaneità, mentre a livello microscopico si ha una semplice alternanza sequenziale molto veloce
- ◆ Il tempo totale di esecuzione di un singolo programma aumenta rispetto al caso mono-tasking
 - a causa dell'alternanza con gli altri programmi

MULTIPROGRAMMAZIONE

- Nel sistema sono presenti diversi programmi, ognuno con un proprio tempo di elaborazione e propri tempi di attesa per le operazioni di ingresso/uscita.
- Per evitare che la CPU venga utilizzata in modo esclusivo (o per troppo tempo) da parte di un solo programma, il tempo viene idealmente suddiviso in unità elementari, dette **quanti**, da assegnare secondo opportune politiche a tutti i programmi.
- **Round-robin**: assegnare a rotazione la disponibilità di un quanto di tempo della CPU ai vari programmi presenti contemporaneamente in memoria.
- La durata del quanto di tempo incide significativamente sia sulle prestazioni del sistema che sull'efficacia del quasi parallelismo, che tende a scomparire se la durata diviene eccessiva e degrada nella sequenzializzazione dei programmi. D'altra parte, pur migliorando in generale le proprietà di parallelismo la scelta di un valore molto piccolo può comportare un degrado delle prestazioni complessive del sistema, qualora il tempo di commutazione fra programmi sia dello stesso ordine della durata del quanto di tempo (un valore tipico per il sistema operativo Unix è 100 ms).

MULTIPROGRAMMAZIONE

➤ **Programma:**

entità statica composta dal codice eseguibile dal processore.

➤ **Processo:**

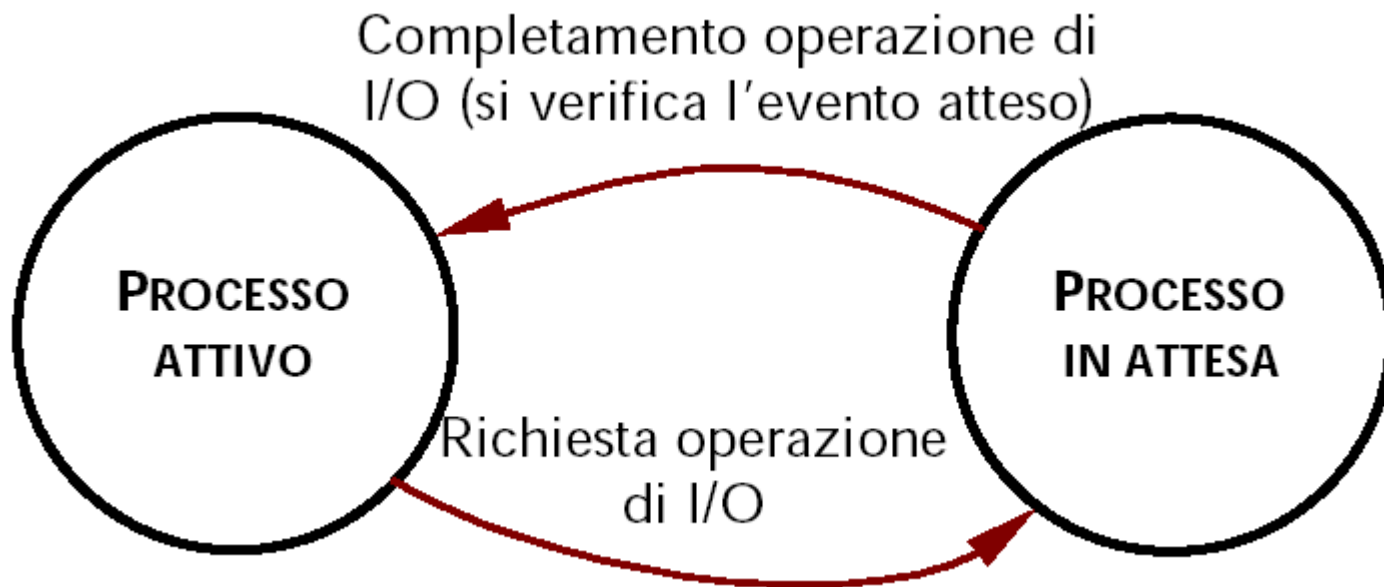
entità dinamica che corrisponde al programma in esecuzione, composto da:

- codice (il programma);
- dati (quelli che servono per l'esecuzione del programma);
- stato (a che punto dell'esecuzione ci si trova, cosa c'è nei registri, ...).

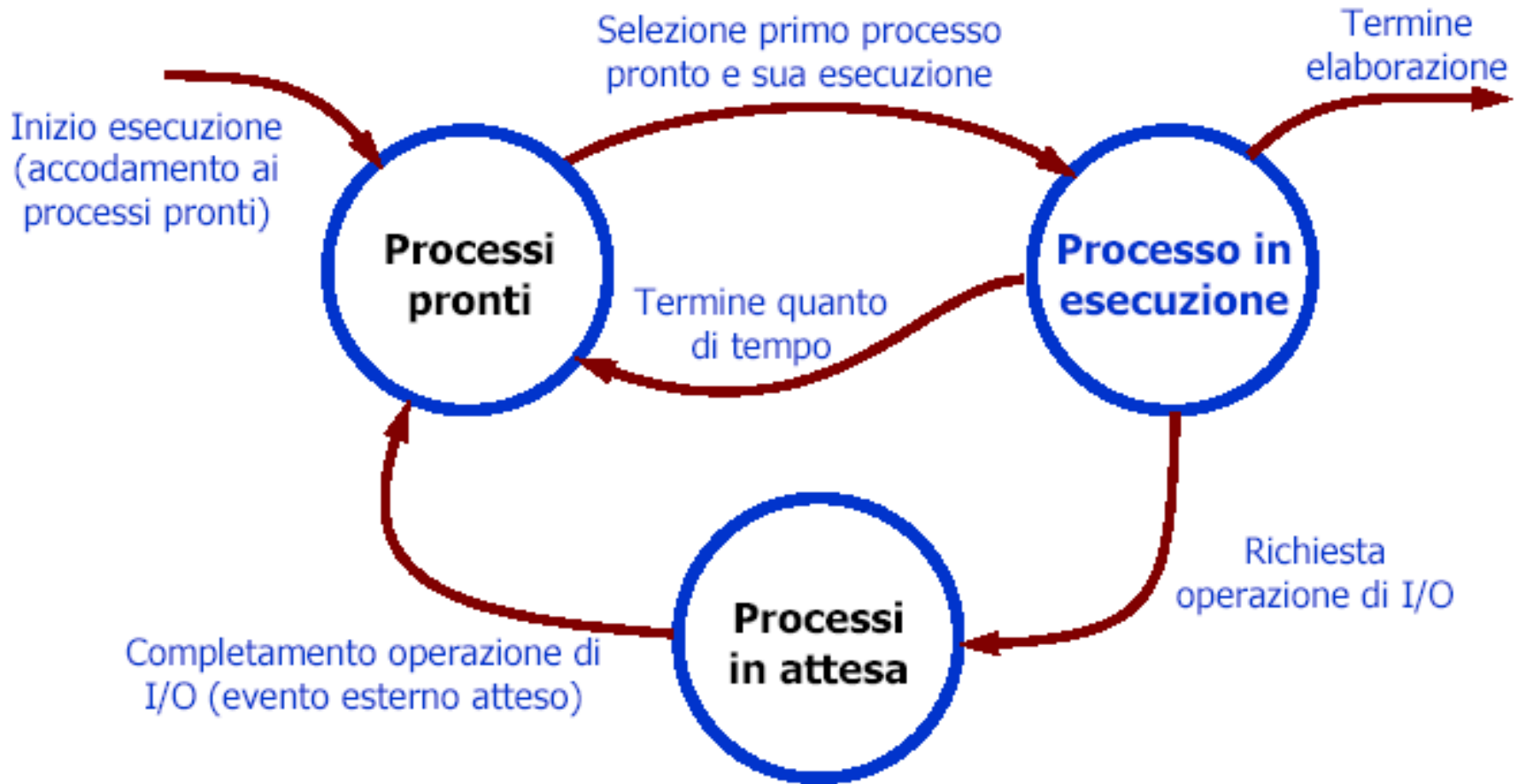
CONTEXT SWAPPING

- Il processo non si rende conto delle interruzioni:
 - il nucleo maschera al processo come effettivamente la sua elaborazione evolve nel tempo;
 - il nucleo rende trasparente la presenza delle operazioni di interruzione e di riassegnamento del processore a un processo.
- Contesto di un processo
 - insieme dei dati che rappresentano lo "stato" del processo: situazione della memoria, contenuto dei registri, livello di priorità, ...
 - quando un processo viene interrotto (esce dallo stato di esecuzione) il nucleo provvede a **salvare** del suo contesto (in una struttura dati chiamata **descrittore del processo**);
 - quando un processo torna nello stato di esecuzione il nucleo provvede a **ripristinare** il suo contesto (recuperando i dati precedentemente salvati nel descrittore).
- Cambio di contesto (context swapping)
 - si verifica quando un processo (e.g. P1) in esecuzione viene sostituito da un altro processo P2
 - il nucleo provvede a
 1. salvare il contesto di P1 e gestirne l'evoluzione (pronto vs attesa);
 2. ripristinare il contesto di P2 per consentirgli una corretta evoluzione.

STATI DI UN PROCESSO



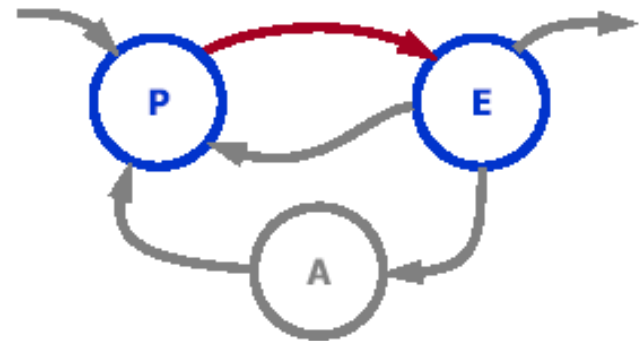
STATI DI UN PROCESSO



STATI DI UN PROCESSO

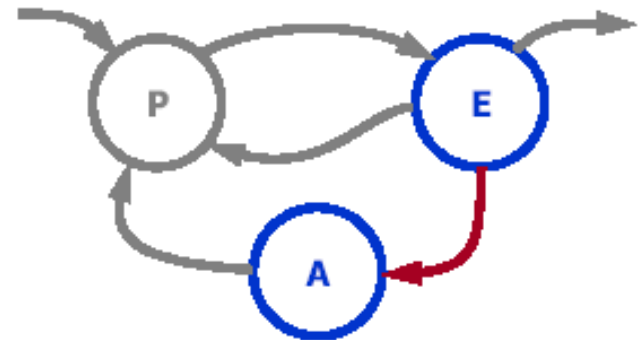
➤ Pronto → Esecuzione

- Il SO stabilisce quale dei processi "pronti" debba essere mandato in "esecuzione".
- La scelta è fatta dall'algoritmo di scheduling



➤ Esecuzione → Attesa

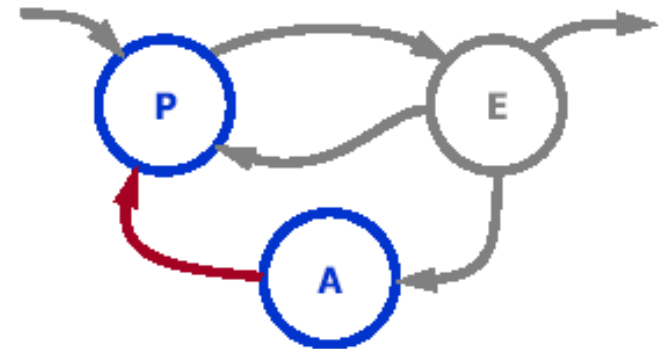
- il processo chiede delle risorse che non sono disponibili o attende un evento
- il SO salva tutte le informazioni necessarie a riprendere l'esecuzione e l'informazione relativa all'evento atteso nella tabella dei processi



STATI DI UN PROCESSO

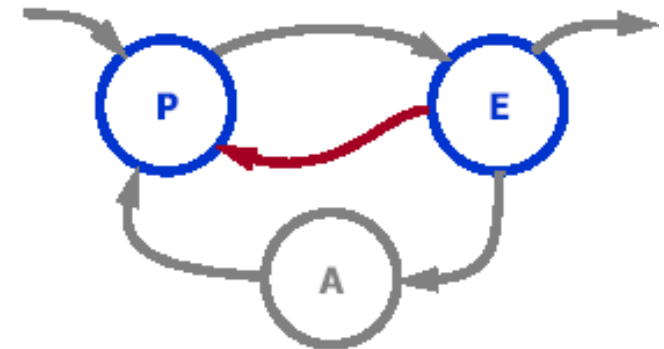
➤ **Attesa → Pronto**

- Si verifica l'evento atteso dal processo e il SO sposta quel processo nella coda dei processi pronti.

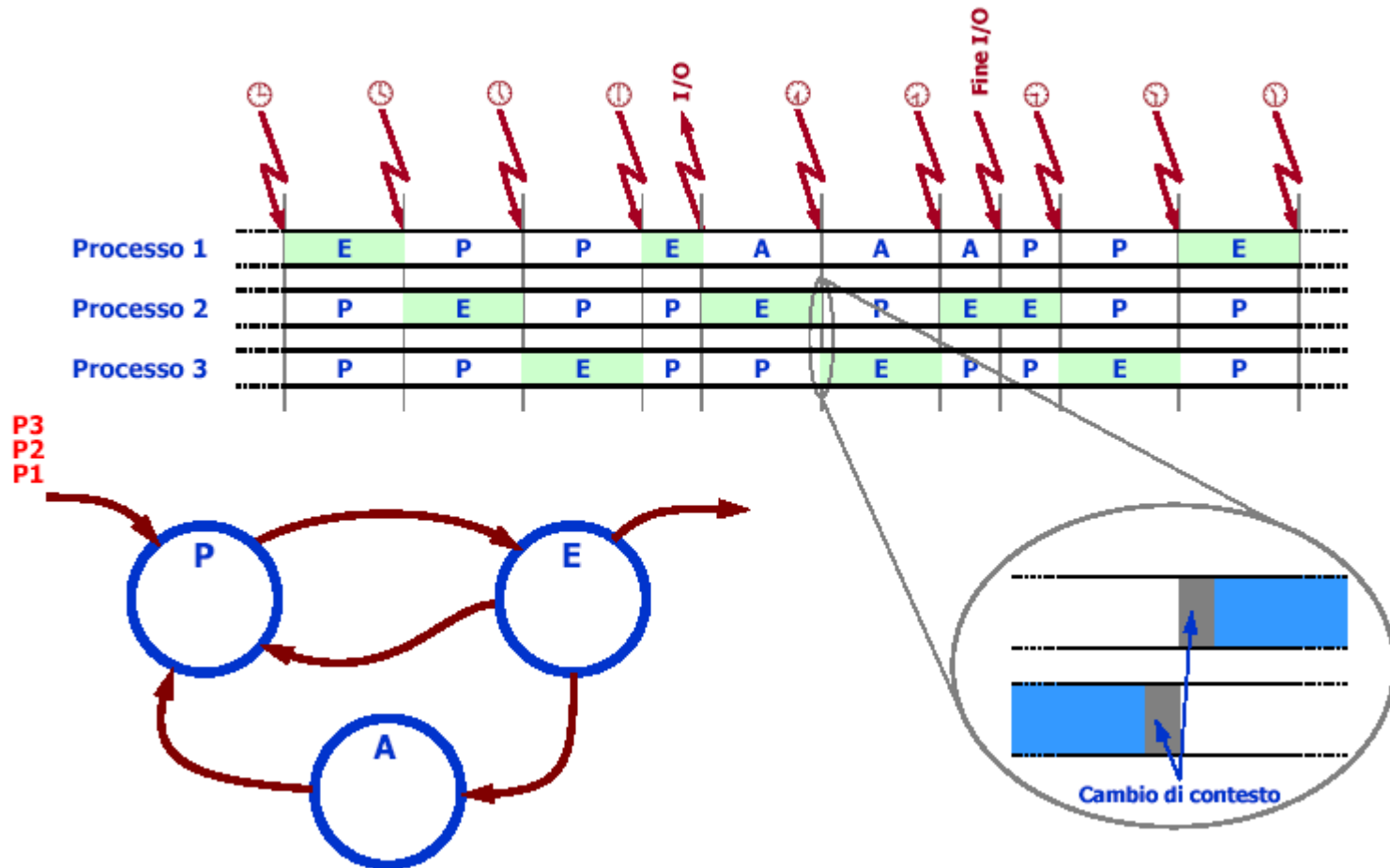


➤ **Esecuzione → Pronto**

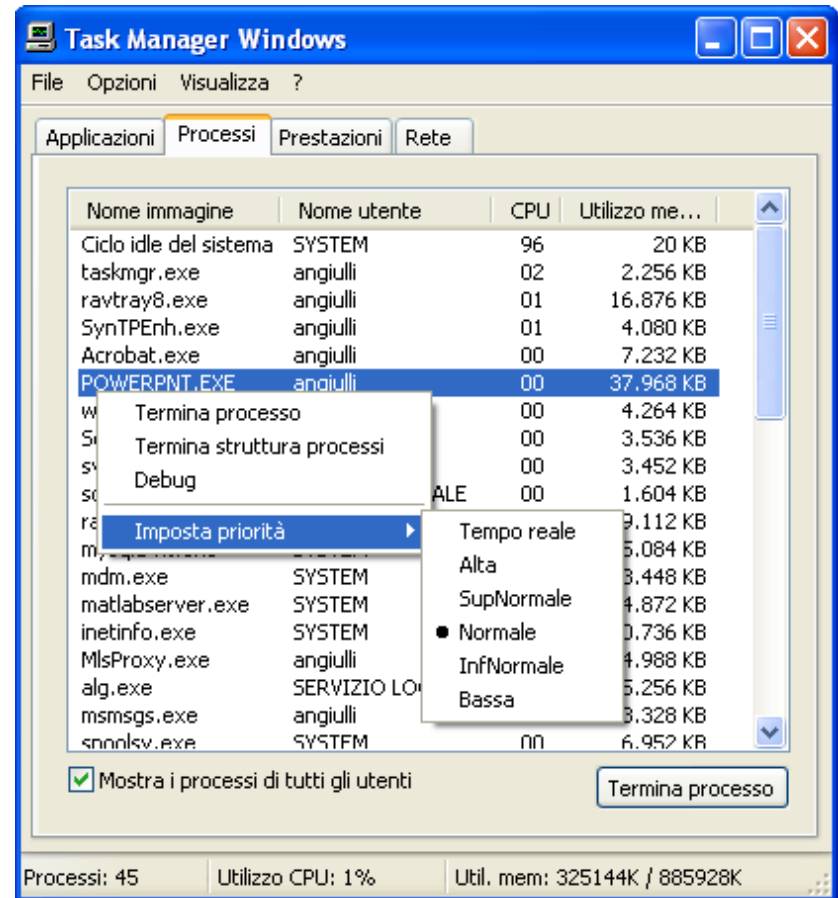
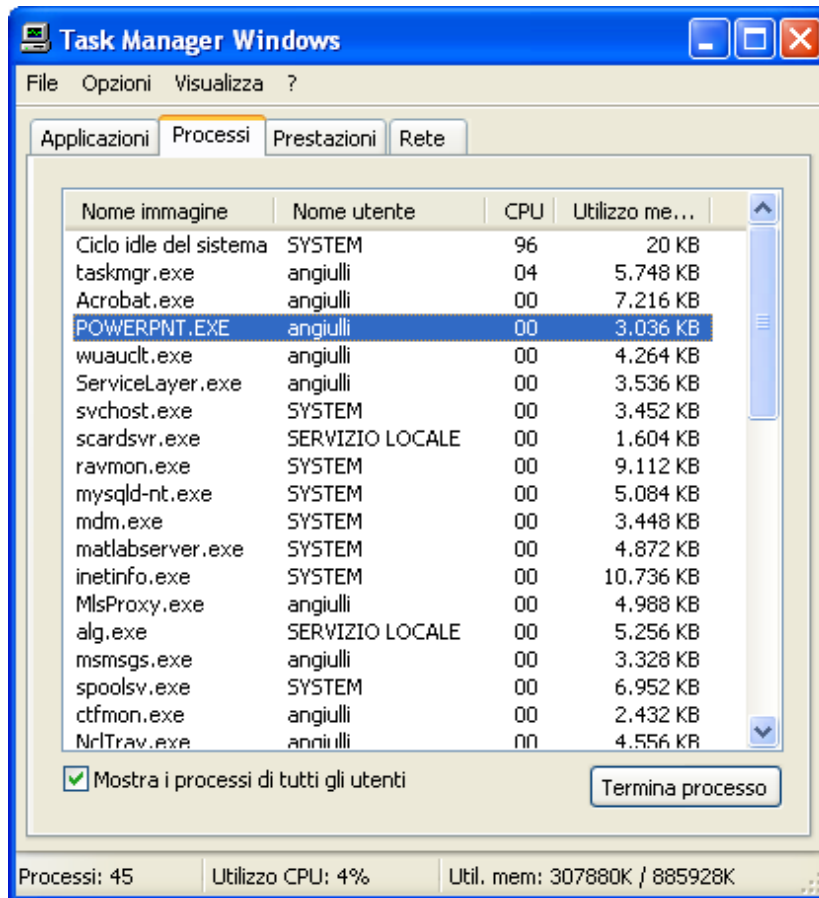
- Termina il quanto di tempo e il processo in "esecuzione" lascia spazio a un altro processo "pronto".
- Il SO salva (nella **tabella dei processi**) tutte le informazioni per riprendere l'esecuzione del processo dal punto in cui viene interrotta.
- Contemporaneamente un altro processo passa da "pronto" a "esecuzione".



ROUND ROBIN



PROCESSI (TASK MANAGER DI WINDOWS)



PROBLEMATICHE DI GESTIONE DEI PROCESSI

➤ Problemi

- **starvation**: un processo non riesce ad accedere ad una risorsa perché la trova sempre occupata da altri processi (che per esempio possono avere un livello di priorità maggiore);
- **blocco critico**: un insieme di processi rimane bloccato perché ciascuno di essi aspetta delle risorse che sono occupate da un altro processo compreso in questo stesso insieme (**vincolo circolare**).
- Evitare (prevenzione) o risolvere (eliminazione) situazioni di blocco critico o di starvation riduce le prestazioni complessive del sistema.

➤ Le **interazioni** fra processi sono classificabili in:

- **indesiderate e (spesso) impreviste**:
i processi **competono** per le risorse che servono per completare l'esecuzione (causando eventualmente problemi quali blocco critico e starvation);
- **desiderate e previste**:
legate a meccanismi di **cooperazione** fra processi che hanno l'obiettivo di giungere alla soluzione di un problema complesso.

➤ La **gestione delle interazioni** fra i processi implica

- la **sincronizzazione** fra le varie attività che ogni singolo processo deve svolgere in modo parallelo rispetto agli altri
- la **comunicazione**, ovvero una modalità per lo scambio dei dati fra i processi

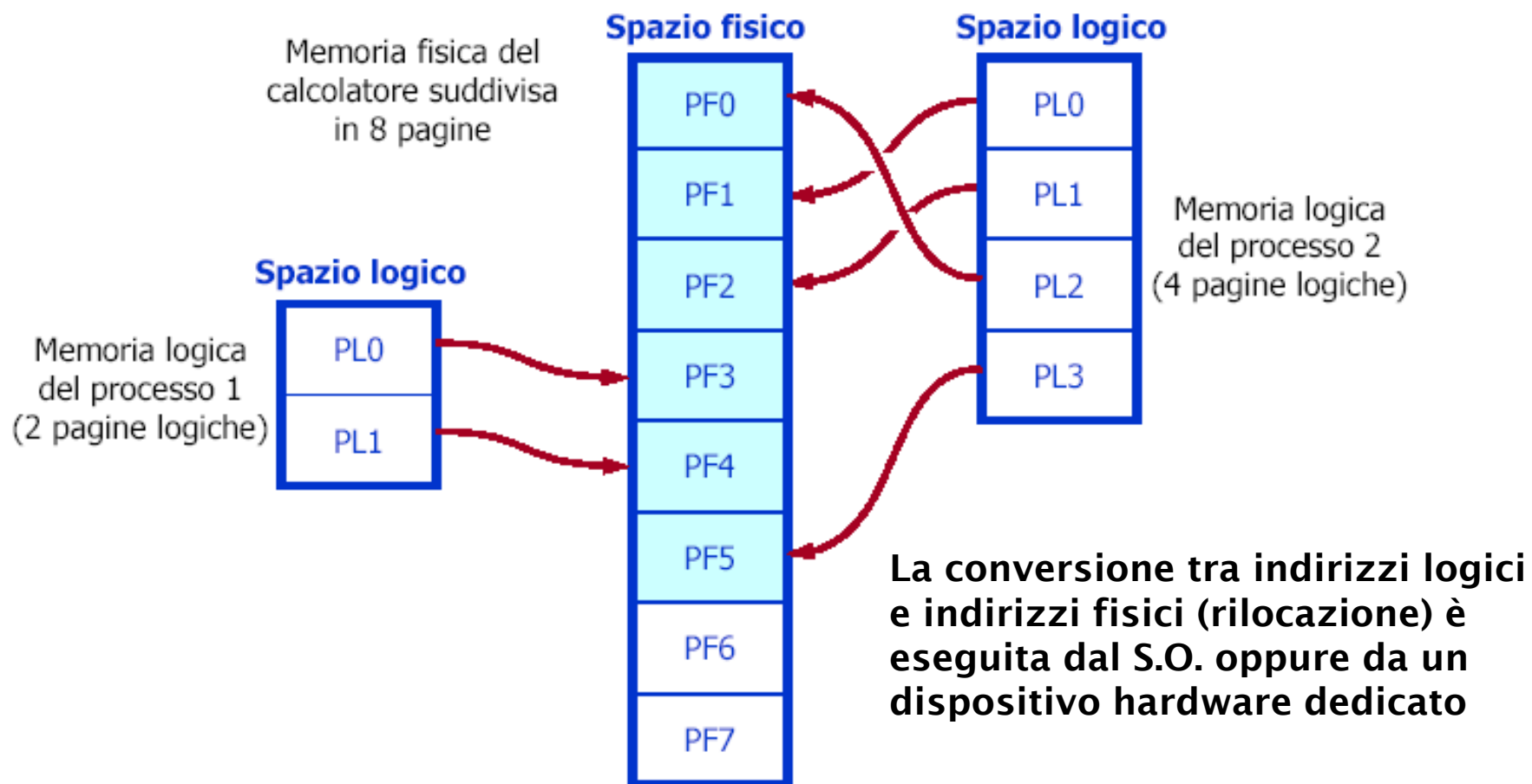
GESTIONE DELLA MEMORIA

- Applica tecniche per gestire il conflitto fra dimensione della memoria fisica e spazio complessivo richiesto dai programmi che devono essere eseguiti in modo concorrente e dai relativi dati.
- Combina le seguenti strategie:
 - consentire il caricamento di un programma a partire da un indirizzo qualunque della memoria;
 - ridurre la necessità di spazio tenendo in memoria solo una porzione dei programmi e dei dati;
 - condividere parte delle istruzioni (codice eseguibile) fra diversi processi corrispondenti a uno stesso programma.
- Il gestore della memoria
 - garantisce ai vari processi uno **spazio di indirizzamento virtuale** in cui lavorare, che può essere superiore alla memoria fisica presente nel calcolatore
 - mette in atto dei meccanismi di protezione che tutelano la privacy dello spazio di lavoro assegnato a ogni processo.

PAGINAZIONE

- Frammentazione della memoria (logica e fisica) in blocchi di dimensioni prefissate: **le pagine**.
- Lo spazio logico di indirizzamento del processo è suddiviso in sezioni, di dimensioni fisse e uguali fra loro, dette **pagine logiche**
- Lo spazio fisico di indirizzamento disponibile nel calcolatore è anch'esso suddiviso in **pagine fisiche**, della stessa dimensione delle pagine logiche.
- È possibile così:
 - estendere la dimensione di un processo utilizzando delle zone di memoria non necessariamente contigue;
 - tenere in memoria solo la porzione ridotta del programma che si sta utilizzando.

PAGINAZIONE



SWAPPING

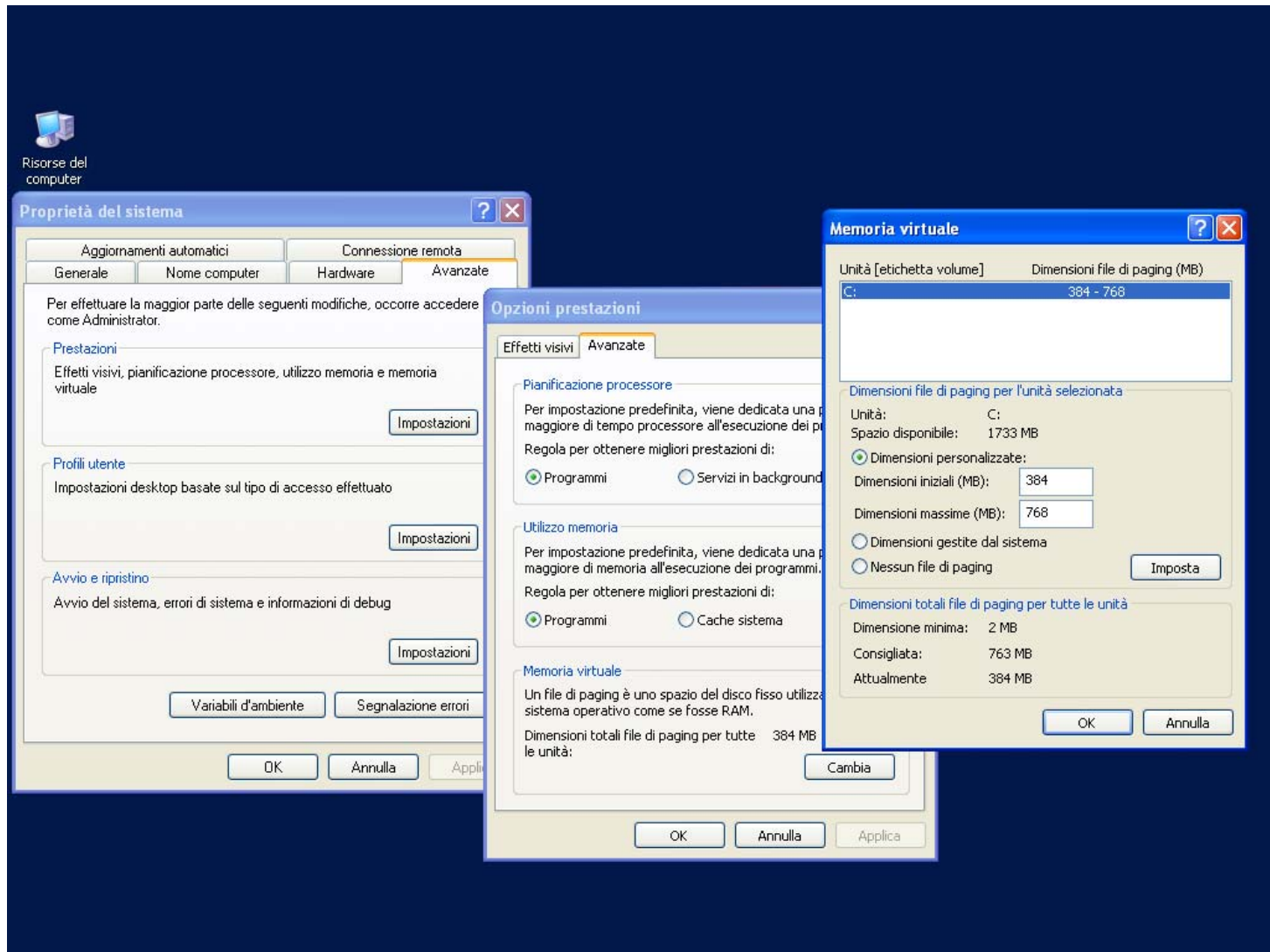
- Spesso la memoria centrale non ha dimensioni tali da contenere tutti i programmi che occorre eseguire in modo concorrente:
 - manca spazio per attivare nuovi processi
 - manca spazio per consentire l'evoluzione di qualche processo già in esecuzione
- Soluzione: trasferire il contenuto di un'area della memoria centrale in un'area della memoria di massa (**area di swap**)

MEMORIA VIRTUALE

- Si possono tenere in memoria solo le pagine logiche di un programma a cui si sta accedendo più frequentemente.
 - quando un processo cerca di accedere a una pagina non in memoria si verifica un evento di **page fault**;
 - il sistema operativo libera dello spazio in memoria (per esempio scaricando delle pagine non utilizzate) e a caricare la pagina richiesta.
- Un opportuno dimensionamento su base statistica del numero di pagine logiche da tenere in memoria consente in genere di ridurre il numero di page fault, che comportano un rilevante sovraccarico per il sistema operativo.

- La paginazione risolve contemporaneamente tre problemi:
 1. Dove mettere il processo in memoria
 - non ha importanza quale sia il posto dove è allocato il processo;
 - la conversione delle pagine logiche in pagine fisiche maschera l'allocazione.
 2. Superare il numero di processi che posso gestire contemporaneamente
 - Se i processi fossero messi interamente in memoria lo spazio sarebbe sufficiente solo per pochi processi, non per tanti.
 3. Superare la dimensione fisica della memoria di lavoro
 - Un processo può avere molte più pagine logiche di quante siano le pagine fisiche disponibili in RAM.

MEMORIA VIRTUALE DI WINDOWS



TRADUTTORI: COMPILATORI E INTERPRETI

- I **traduttori** sono programmi particolari che provvedono a convertire il codice di programmi scritti in un dato linguaggio di programmazione (sorgenti), nella corrispondente rappresentazione in linguaggio macchina (eseguibili)

➤ **Compilatori**

- Accettano in ingresso l'intero programma e producono in uscita la rappresentazione dell'intero programma in linguaggio macchina.

➤ **Interpreti**

- Traducono ed eseguono direttamente ciascuna istruzione del programma sorgente, istruzione per istruzione.

TRADUTTORI: COMPILATORI E INTERPRETI

➤ **Compilazione**

- applicazioni più veloci
- maggior lavoro nel processo di messa a punto e manutenzione
- OK per i prodotti commerciali a larga diffusione.

➤ **Interpretazione**

- consente tempi di sviluppo più contenuti,
- produce programmi meno efficienti;
- OK in fase di prototipazione dei programmi che, una volta ultimati, venivano compilati prima del rilascio commerciale.

➤ Considerare le applicazioni che debbono essere eseguite:

• **I/O bound:**

- programmi che eseguono molte operazioni di ingresso/uscita intervallate da brevi periodi di elaborazione;
- convenientemente eseguiti da un interprete senza penalizzazioni nell'efficienza (il tempo di ingresso/uscita è indipendente dalle modalità di esecuzione del programma),
- appartengono a questa categoria la maggioranza dei programmi gestionali.

• **CPU bound:**

- programmi che eseguono poche operazioni di ingresso/uscita rispetto alla mole delle elaborazioni effettuate.
- sono programmi prevalentemente di tipo scientifico o ingegneristico,
- sono scarsamente efficienti se eseguiti da un interprete e la loro compilazione risulta pressoché indispensabile.

RILOCABILITA' DEL CODICE

- Durante la compilazione i nomi simbolici e i riferimenti a celle di memoria sono stati risolti:
 - le istruzioni sono in formato macchina
 - tutti i riferimenti a istruzioni e dati sono espressi nella forma di indirizzi.
- Due spazi di memoria
 - **spazio logico**: intervallo di celle contigue che partono dall'indirizzo 0 in cui si immagina siano collocate le istruzioni durante la compilazione;
 - **spazio fisico**: lo spazio di memoria in cui risiede effettivamente il codice.
- Per far funzionare il programma caricandolo a partire da una posizione arbitraria della memoria bisogna effettuare una **rilocalizzazione**: sommare a tutti gli indirizzi presenti nel programma un valore (**spiazzamento**) corrispondente alla differenza fra l'indirizzo a partire dal quale verrà effettivamente caricato il programma e il valore a partire dal quale sono stati calcolati gli indirizzi.



FINE

