# Le infrastrutture SoftWare

# Funzioni del sistema operativo

- **Rendere utilizzabili le risorse fisiche presenti nel sistema informatico:**
  - **correttezza** e **precision**;
  - **anywhere**, **anytime**;
  - **affidabilità**, **disponibilità** e **sicurezza** dei sistemi;
  - **privatezza** dei dati;
  - **interoperabilità** fra dispositivi forniti da diversi produttori;
  - superare i problemi legati alla **limitazione del numero di risorse**.

- **Il sistema operativo può essere inteso come uno strumento che virtualizza le caratteristiche dell'hardware sottostante, offrendo di esso la visione di una macchina astratta più potente e più semplice da utilizzare di quella fisicamente disponibile.**

# SO: funzionalità

- SO come GESTORE DELLE RISORSE
- SO come MACCHINA ESTESA

# Funzioni di servizio del SO

- **Esecuzione di applicazioni**
- **Accesso ai dispositivi di ingresso/uscita**
- **Archiviazione di dati e programmi**
- **Controllo di accesso**
- **Contabilizzazione**
- **Gestione dei malfunzionamenti**

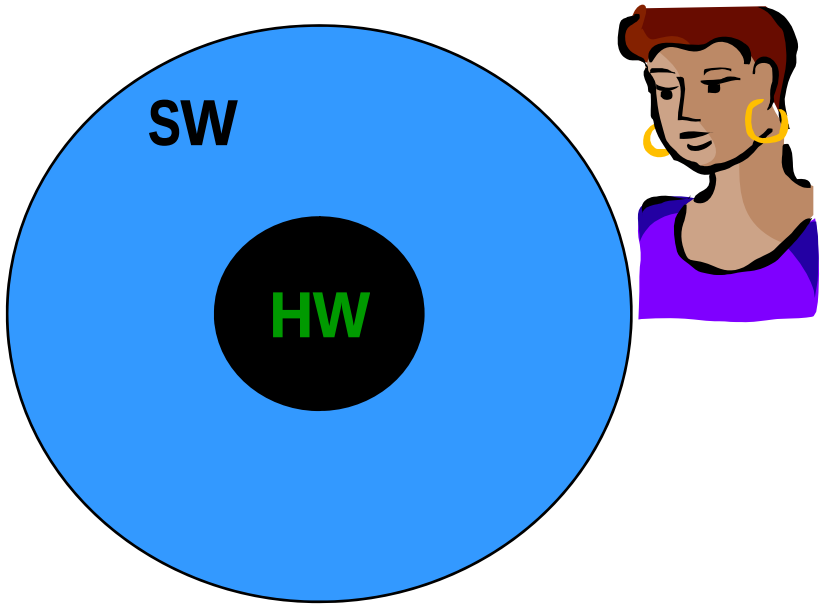
# Vantaggi di un SO

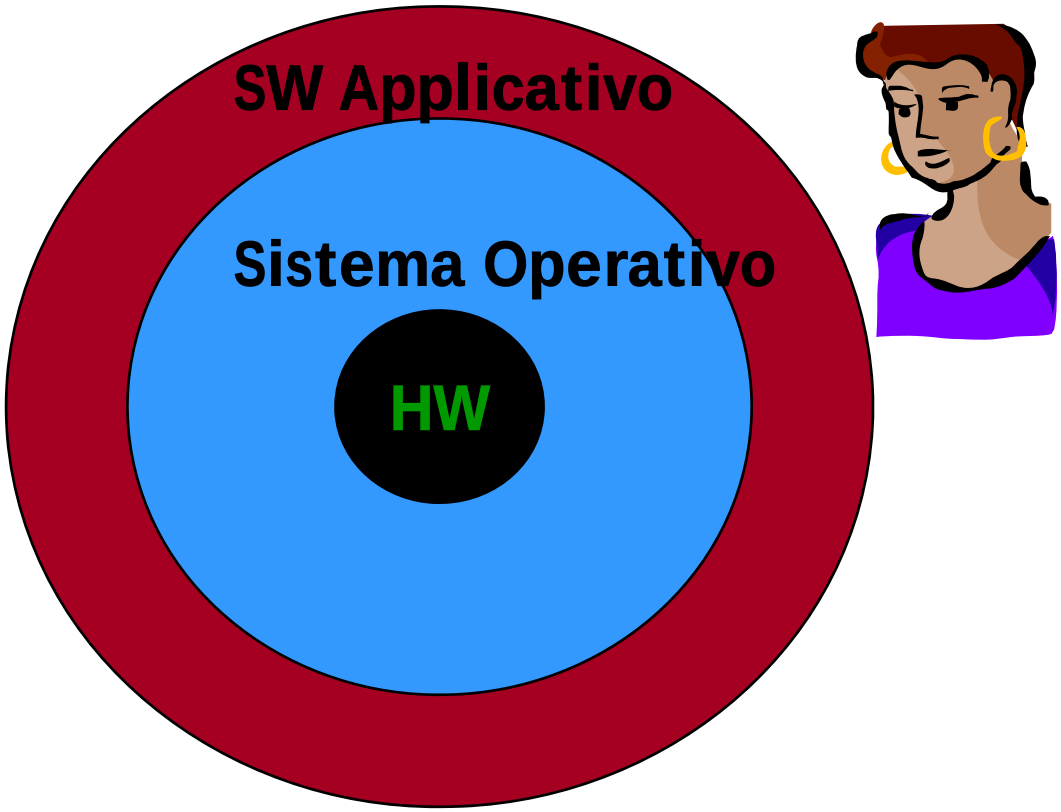
- **Sono legati alla possibilità di definire modalità standard di interfaccia con i dispositivi fisici, cosicché sia possibile:**
  - sviluppare programmi in modo semplice, modulare ed indipendente dallo specifico calcolatore su cui viene fatto funzionare il sistema operativo;
  - aggiornare il software di base e l'hardware in modo trasparente ai programmi applicativi e all'utente, ossia senza che vengano influenzati dall'operazione.

# Visioni fornite da un SO

- **Dall'alto:** il sistema operativo fornisce all'utente un'interfaccia conveniente.
- **Dal basso:** gestisce tutti le parti di un sistema complesso, allocando in modo ordinato le diverse risorse della macchina: processori, memorie, dischi, interfacce di rete, stampanti e altre periferiche.

# Il software



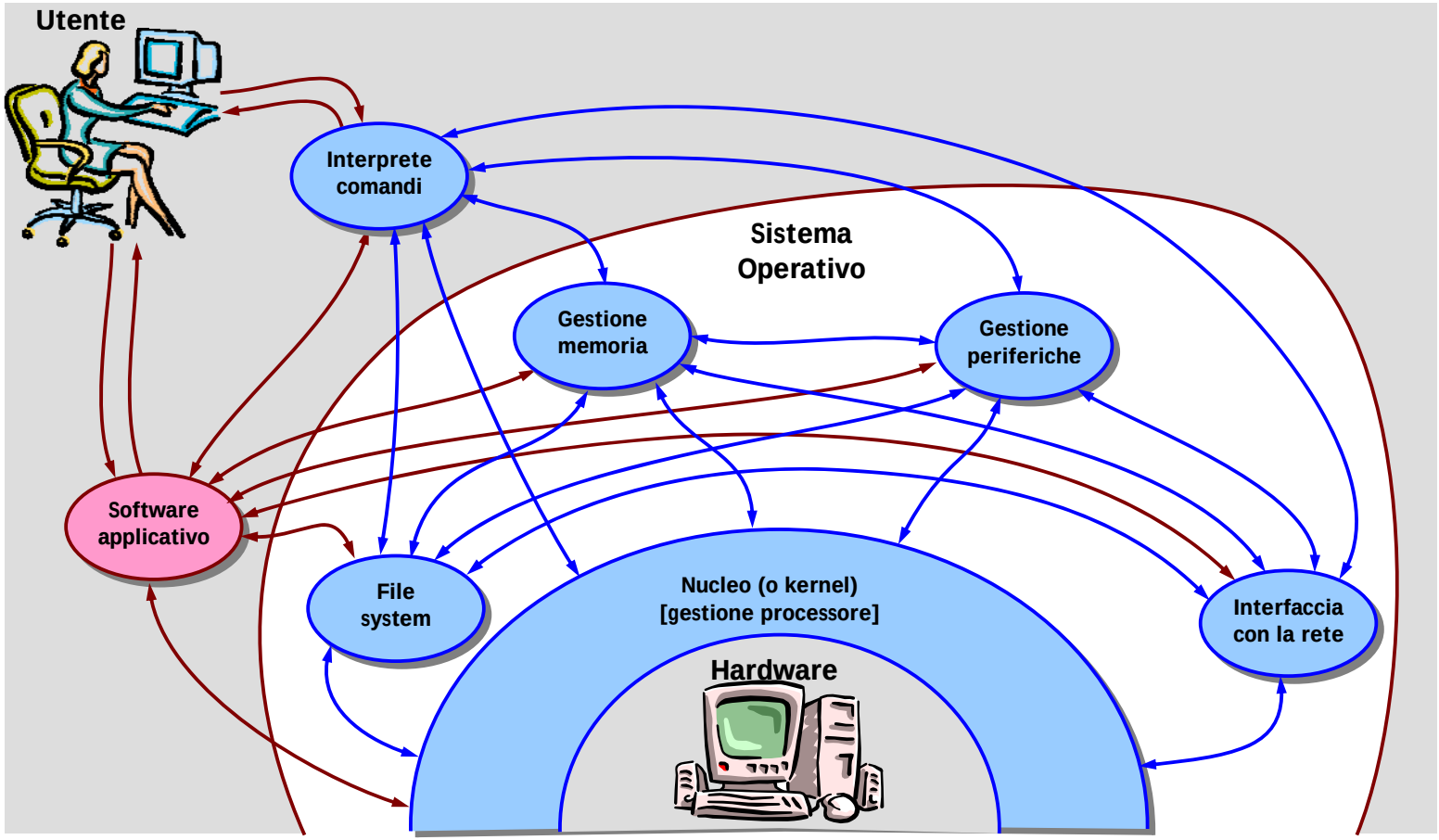

- **SW= Sistema Operativo + SW applicativo**

- **Il S.O. come necessario** intermediario

# Elementi di un SO

- **Sistema di** *gestione del processore*,
- **Sistema di** *gestione della memoria*,
- **Sistema di** *gestione delle periferiche*,
- **Sistema di** *gestione dei file* **(file system)**
- **Sistema di** *gestione degli utenti* **e dei relativi comandi (interprete comandi),**
- **Sistema di** *gestione della rete.*

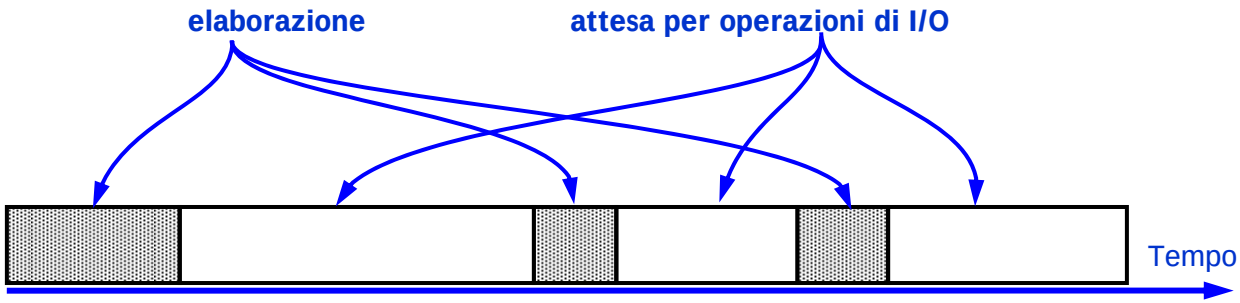
# Elementi di un SO

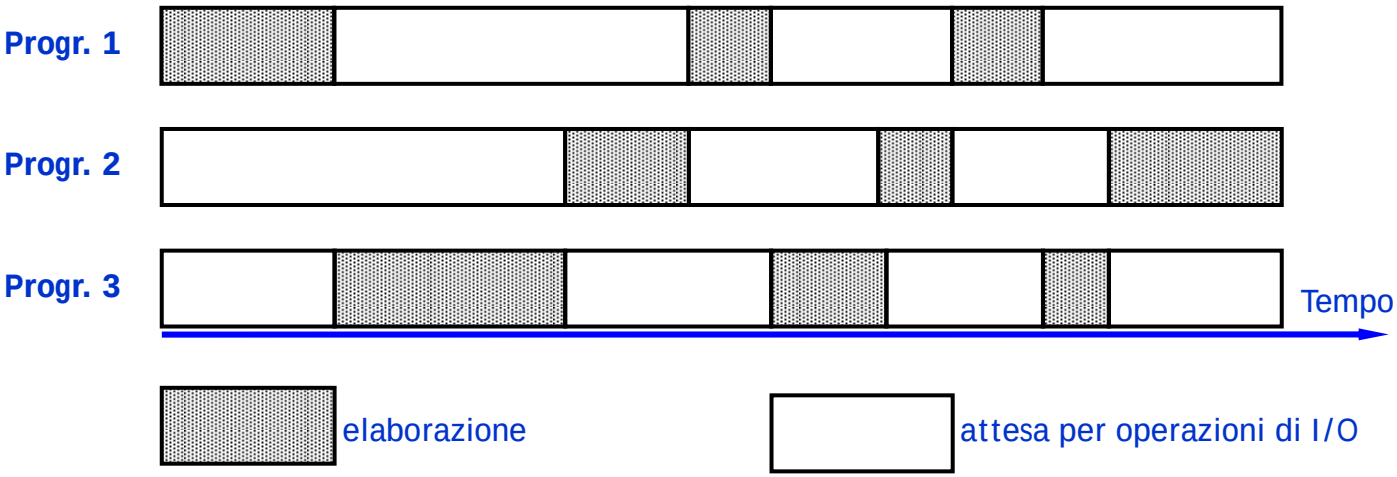


# SO vs applicazioni

- **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;
  - …

# Multiprogrammazione



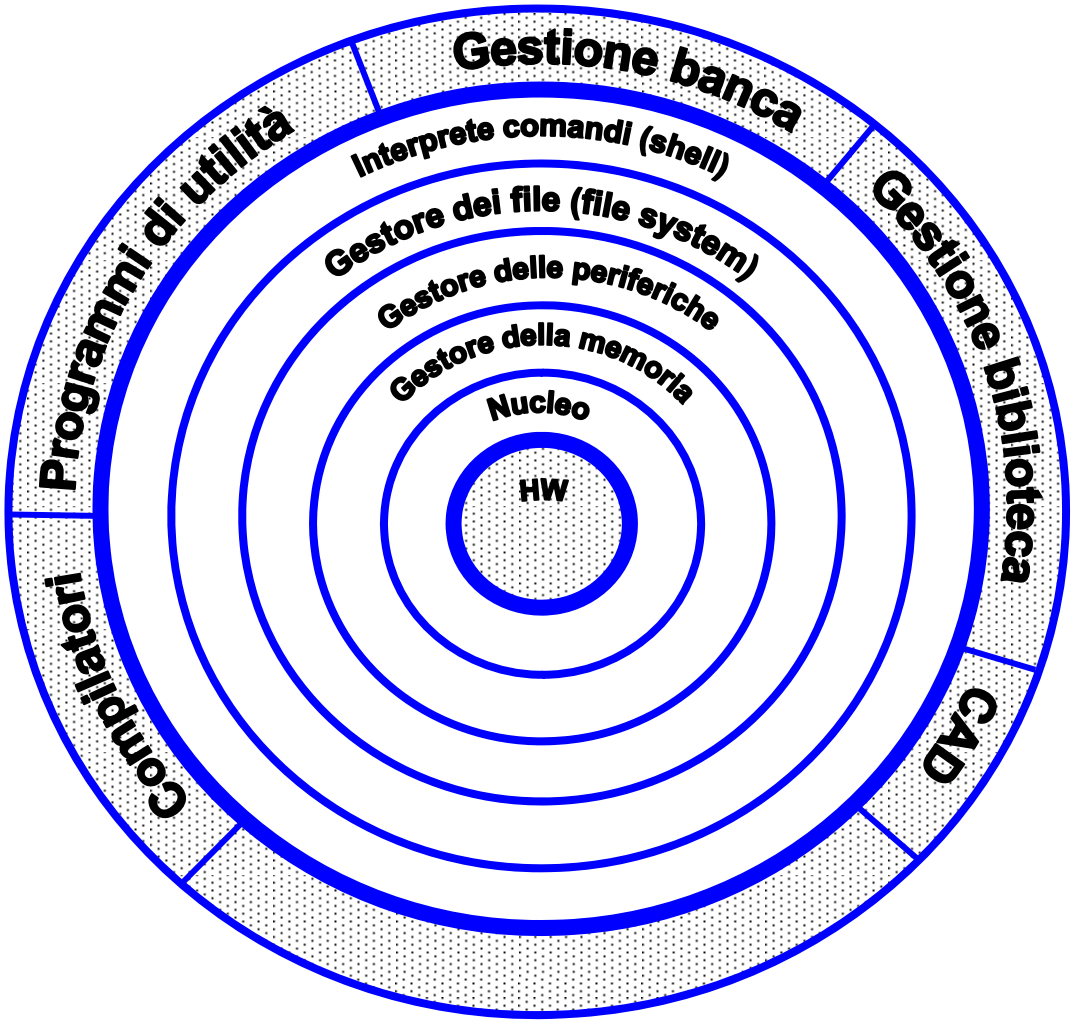



# Processo vs programma

- **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, …).

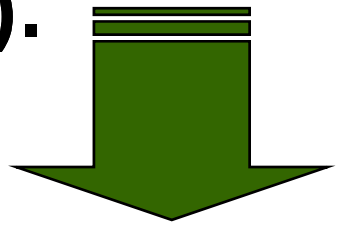
# Organizzazione di un SO

- Gerarchia di "macchine virtuali"
- La visione della macchina virtuale a livello n è quella fornita dall'HW e dagli strati del SO fino all'ennesimo (incluso)



# Organizzazione a “strati”

- **Ogni macchina virtuale è un insieme di programmi che realizza delle funzionalità che utilizzano i servizi forniti a livello inferiore.**
- **Ogni macchina virtuale ha il compito di gestire risorse specifiche di sistema regolandone l’uso e mascherandone i limiti.**
- **I meccanismi che garantiscono la correttezza logica sono separati dalle politiche di gestione (maggiore flessibilità).**



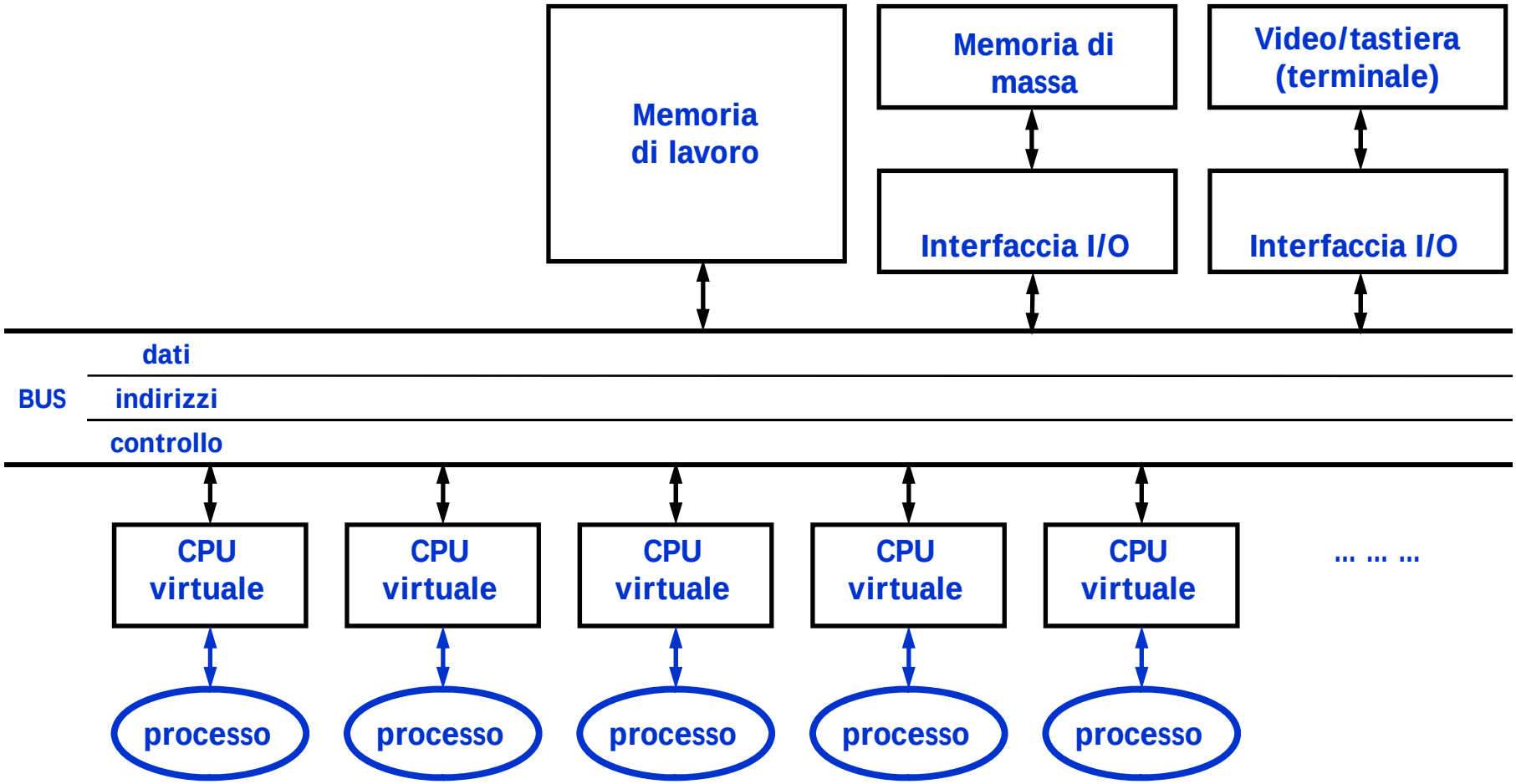
**ogni “strato” risolve un problema specifico**

# Nucleo

- **Interagisce direttamente con l'hardware**
- **Si occupa dell'esecuzione dei programmi e della risposta agli eventi esterni generati dalle unità periferiche.**
- **Scopo principale: gestire i processi corrispondenti ai programmi che sono contemporaneamente attivi.**

- **Fornisce alle macchine virtuali di livello superiore la visione di un insieme di unità di elaborazione virtuali ciascuna dedicata a un processo presente in memoria**
- **Gestisce il contesto di esecuzione dei vari processi**
- **Attua una politica di alternanza (*scheduling*) nell'accesso alla CPU da parte dei processi in esecuzione.**

# Nucleo: macchina astratta

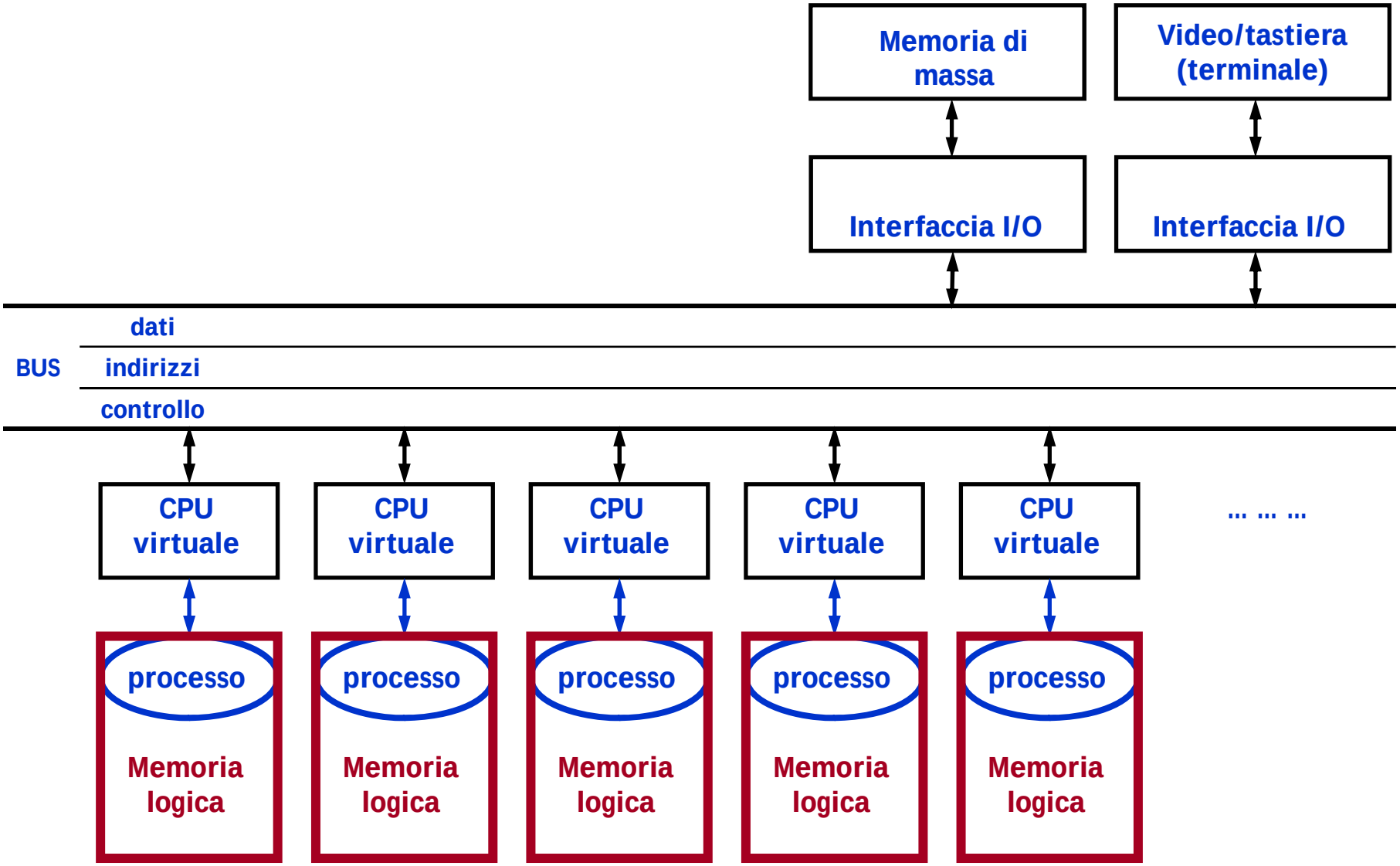


# Gestore della memoria

- **Controlla la memoria centrale, al fine di risolvere le relative esigenze dei vari processi in modo trasparente ed efficiente.**
- **Consente ai programmi di lavorare in un proprio** *spazio di indirizzamento virtuale* **e di ignorare quindi le effettive zone di memoria fisica occupata.**
- **Si occupa di:**
  - proteggere programmi e relativi dati caricati nella memoria di lavoro;
  - mascherare la collocazione fisica dei dati;
  - permettere, in modo controllato, la parziale sovrapposizione degli spazi di memoria associati ai vari programmi.

- **Fornisce alle macchine di livello superiore la possibilità di lavorare come se esse avessero a disposizione una memoria dedicata, di capacità anche maggiore di quella fisicamente disponibile.**

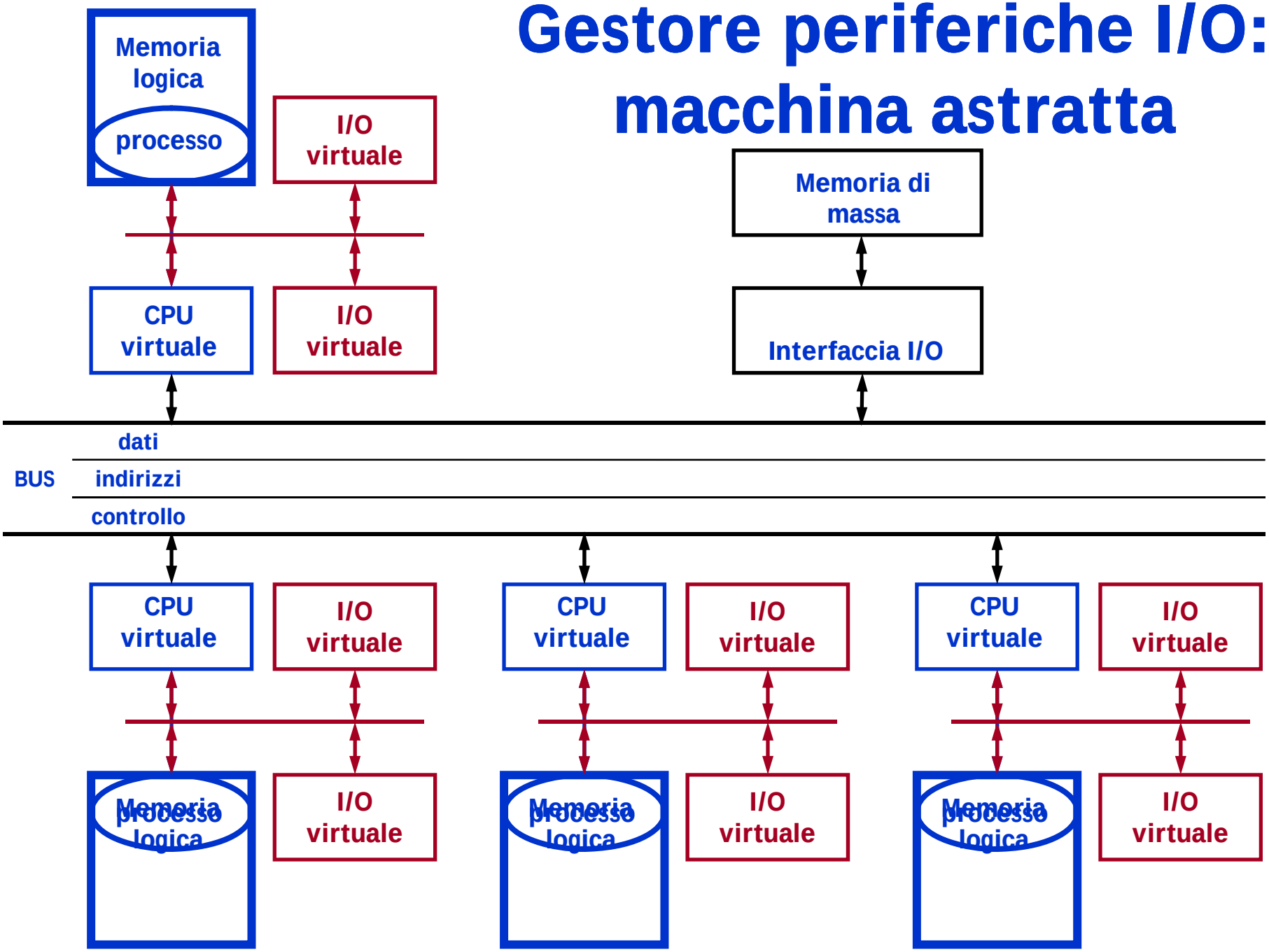
# Gestore memoria: macchina astratta



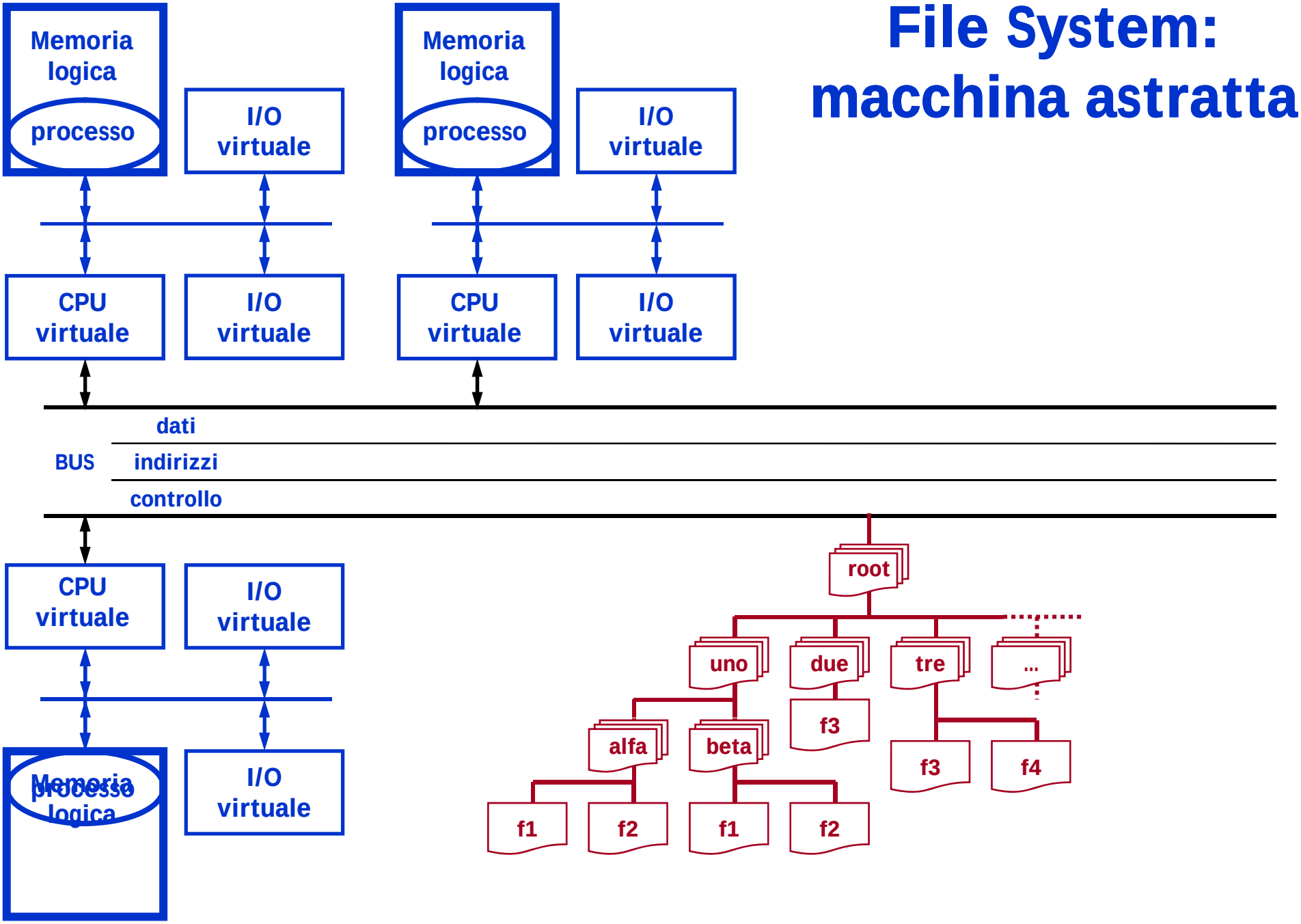
# Gestore delle periferiche

- **Fornisce una visione del sistema in cui i processi possono operare mediante** *periferiche astratte*.
- **Maschera le caratteristiche fisiche delle periferiche e le specifiche operazioni di ingresso/uscita**
- **Ogni processo ha a disposizione delle periferiche virtuali**

# Gestore periferiche I/O: macchina astratta



# File System: macchina astratta



# File System (gestore dei file)

- **Gestisce la memoria di massa**
- **Gestisce i file**

# Interprete dei comandi

- **Modulo del SO direttamente accessibile dall'utente**
- **Ha la funzione di interpretare i comandi che gli giungono (da tastiera e/o point&click) e di attivare i programmi corrispondenti.**
- **Le operazioni che svolge sono:**
  - *lettura* dalla memoria di massa del programma da eseguire;
  - *allocazione* della memoria centrale;
  - *caricamento* del programma e dei relativi dati nella memoria allocata;
  - *creazione* e *attivazione* del processo corrispondente.

# Il middleware

- **Insieme di librerie di utilità che viene "standardizzato" fino a poter essere considerato uno strato del SO**
- **Facilita lo sviluppo di software applicativo**
- **Vere e proprie macchine virtuali**
  - possibilità di eseguire software indipendentemente dalla piattaforma HW/SW
- **Astrazione estrema: middleware per sistemi distribuiti**
  - travalica il limite fisico della macchina
- **Esempi:**
  - Sun Java Runtime Environment, Microsoft .NET, CORBA...

# La gestione dei processi

# Elaborazione parallela

- **Il concetto di elaborazione parallela si riferisce specificamente:**
  - **ai dati**;
  - **alle istruzioni**;
  - **ai programmi**.

- **Il parallelismo a livello di dati e di istruzioni è possibile solo con l'impiego di architetture di elaborazione parallela, basate sulla presenza di più unità di elaborazione in grado di eseguire istruzioni in modo concorrente ma anche, per esempio, di adeguati linguaggi di programmazione.**

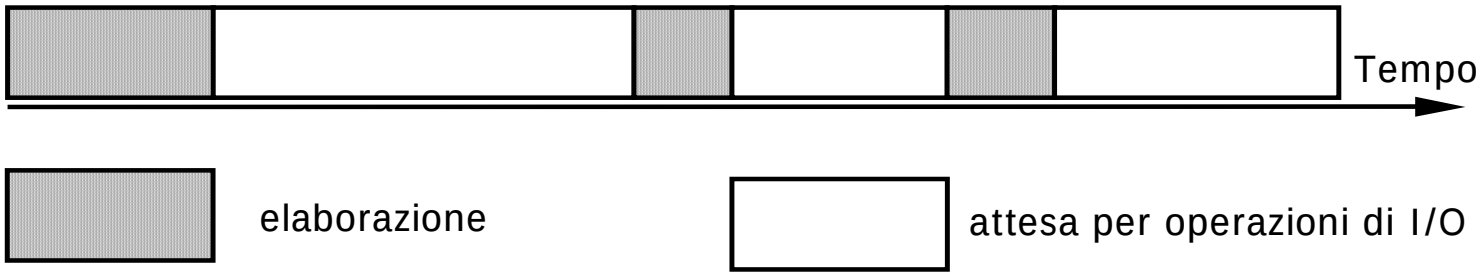
- **Il parallelismo a livello di programma ricade nell'ambito dei sistemi operativi.**

- **Le condizioni che un sistema operativo deve soddisfare sono:**
  - **efficienza**;
  - **interattività**;
  - **sincronizzazione/cooperazione**.

# SO in time sharing

- **Permette la condivisione della CPU tra più processi interattivi**
- **Il tempo di esecuzione del processore è condiviso tra più utenti**
- **Ogni processo in esecuzione ha a disposizione un quanto di tempo di utilizzo della CPU, al termine del quale viene sospeso per lasciare il posto ad un altro processo in attesa di esecuzione**

# Esecuzione di un processo



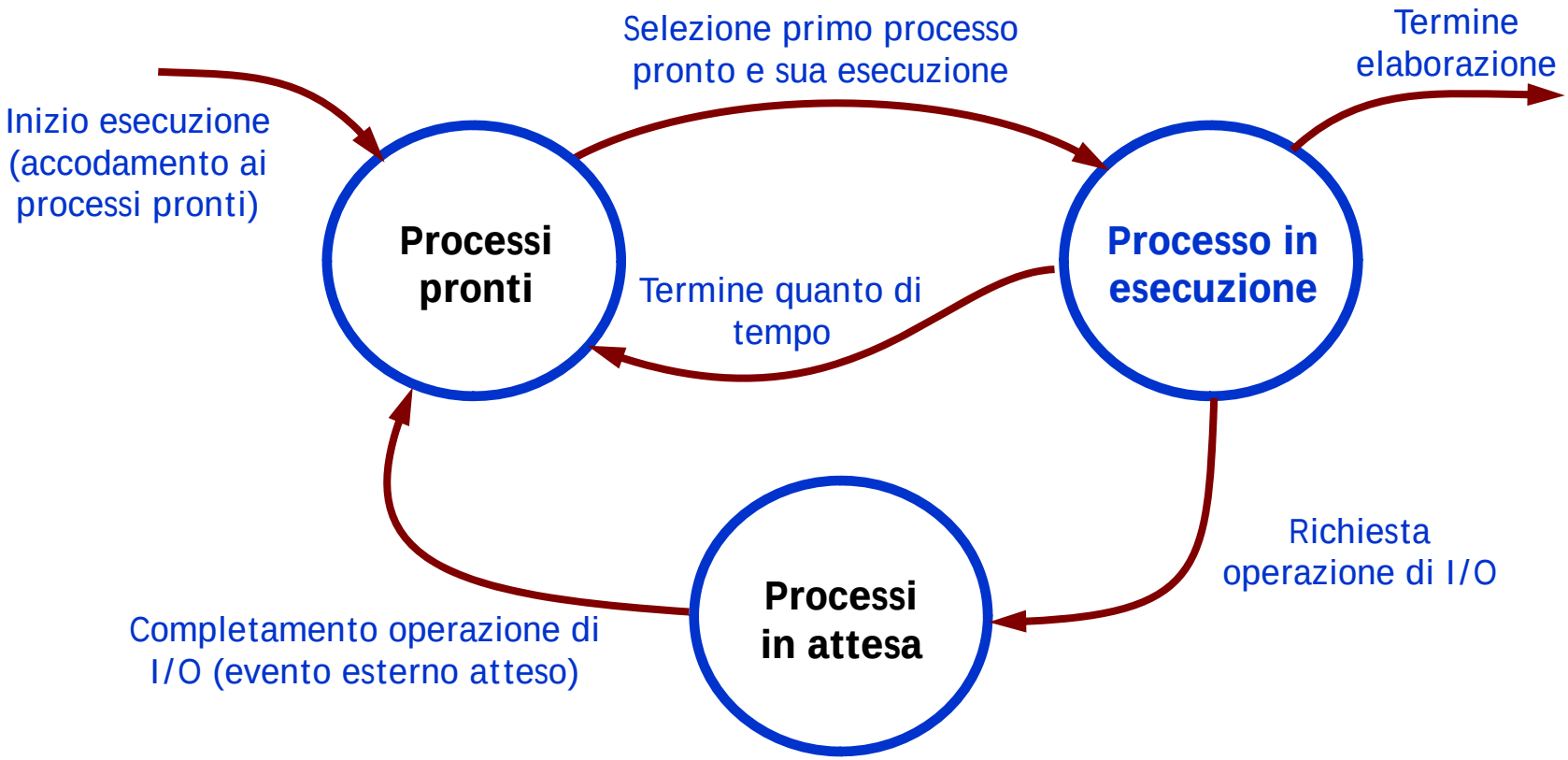

- Un processo utente può effettivamente essere in esecuzione sulla CPU
- Ogni operazione di I/O consiste in una chiamata al sistema operativo e quindi in una sospensione del processo utente per l'esecuzione dell'operazione di I/O da parte del kernel

# Stati di un processo

# Processi non in esecuzione

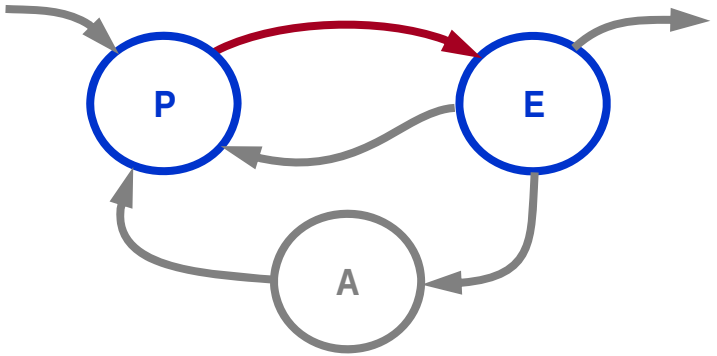
- **Si possono distinguere due casi**
  - Processi in attesa di un evento esterno (ad esempio I/O)
  - Processi pronti ad essere eseguiti in attesa della CPU
- **Si tratta di due stati diversi: PRONTO e ATTESA realizzati con due code diverse**

# Diagramma a tre stati

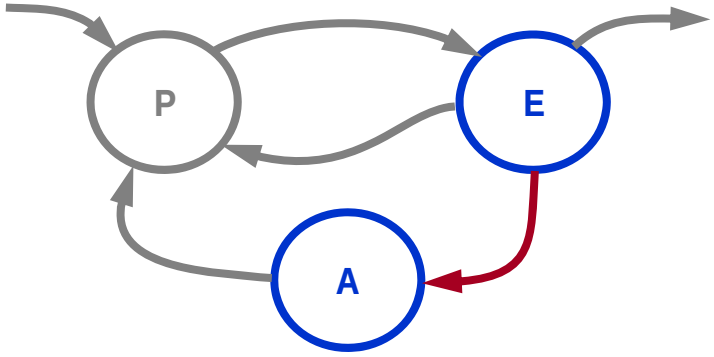


# Transizioni di stato

- Pronto → Esecuzione
  - Il SO stabilisce quale dei processi "pronti" debba essere mandato in "esecuzione".
  - La scelta è fatta dall'algoritmo di scheduling che deve bilanciare **efficienza** e **fairness.**

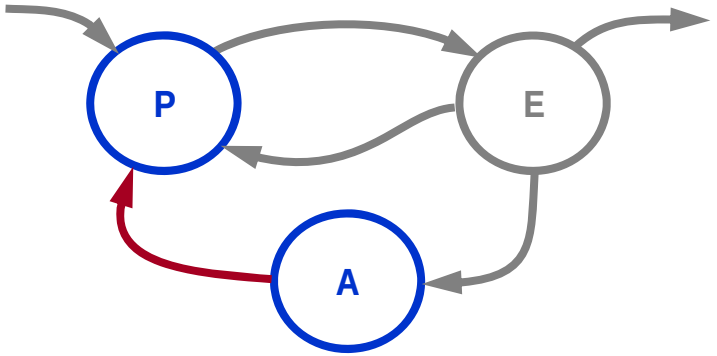



- Esecuzione → Attesa
  - il processo chiede delle risorse che non sono disponibili o attende un evento
  - il SO salva tutte le informazioni necessarie a riprendere l'esecuzione e l'informazione relativa all'evento atteso nella tabella dei processi



# Transizioni di stato

- 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".