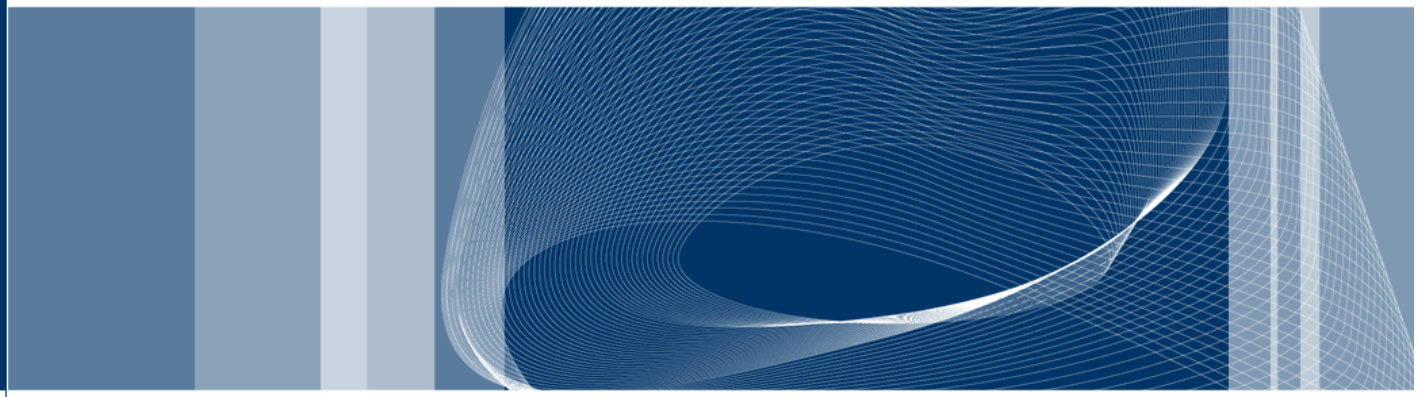
POLITECNICO DI MILANO



Introduzione al Sistema Operativo e catena di sviluppo in C

Fondamenti di Informatica

Daniele Loiacono

Il sistema Operativo

- Il sistema operativo (SO) è uno strato software che nasconde agli utenti i dettagli dell'architettura hardware del calcolatore
- Fornisce diverse funzionalità ad alto livello che facilitano l'accesso alle risorse del calcolatore
- Supporta l'esecuzione dei programmi applicativi definendo una macchina virtuale, cioè un modello ideale del calcolatore, sollevando il software applicativo dal compito di gestire i limiti delle risorse disponibili

Architettura del sistema operativo

- Il SO è tipicamente organizzato a strati
- Ciascun strato costituisce una *macchina virtuale* che gestisce una risorsa del calcolatore
- Le principali funzionalità offerte sono:
  - La gestione dei processi
  - La gestione della memoria
  - La gestione delle periferiche
  - La gestione del file system
  - La gestione della rete
  - La gestione dell'interfaccia utente
- Le prime tre funzionalità sono indispensabili per il funzionamento del sistema e pertanto costituiscono il nucleo del SO (Kernel)

Tipi di Sistema Operativo

- Esistono diversi tipi di sistema operativo, ma in generale si possono dividere in:
  - Monoutente e monoprogrammato
    - Esecuzione un solo programma applicativo alla volta
    - Viene utilizzato da un solo utente per volta
    - Esempio: DOS
  - Monoutente e multiprogrammato (multitasking)
    - Consente di eseguire contemporaneamente più programmi applicativi
    - Esempio: Windows 95
  - Multiutente
    - Consente l'utilizzo contemporaneo da parte di più utenti
    - E' inerentemente multiprogrammato
    - Esempio: Linux

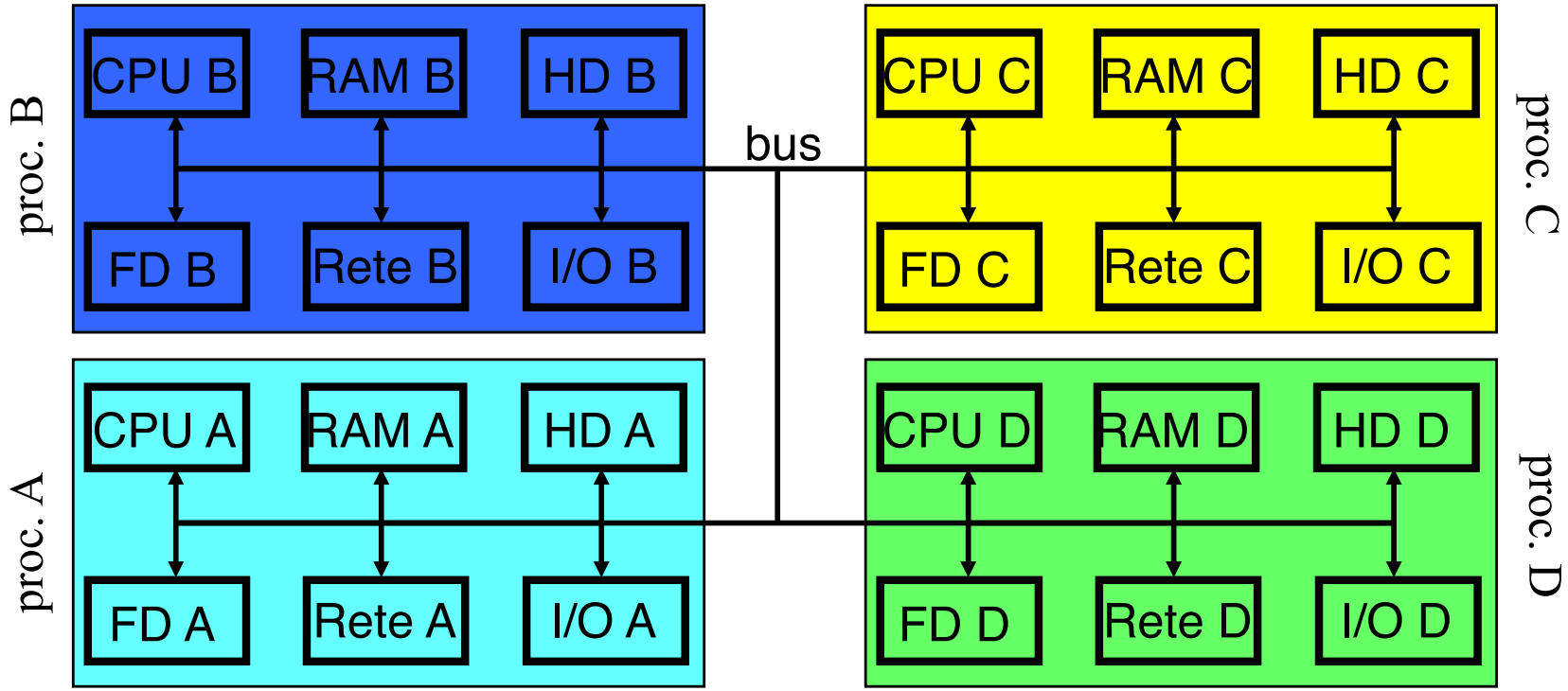
Gestione dei processi nel sistema operativo

Gestione dei processi

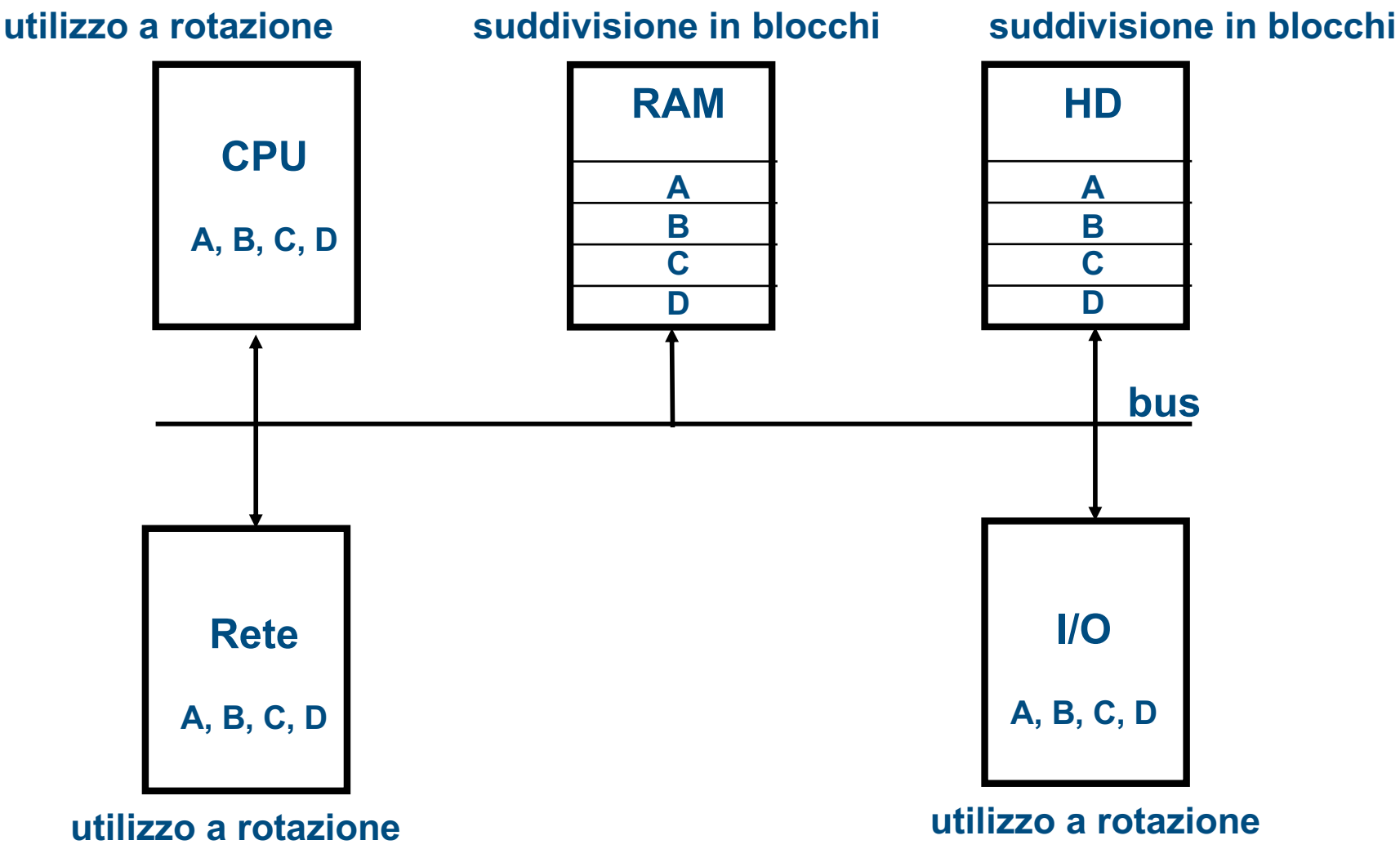
- Il SO si occupa di gestire l'esecuzione concorrente di più programmi utente
- La CPU del calcolatore (o le CPU nei sistemi multiprocessore) deve essere distribuita in maniera opportuna fra i programmi da eseguire
- Ogni programma eseguito ha a disposizione una macchina virtuale realizzata dal SO che ne consente l'esecuzione come se la CPU del calcolatore fosse interamente dedicata ad esso

Il sistema operativo e le macchine virtuali

- Il sistema operativo esegue più processi contemporaneamente
- Rende quindi visibile ad ogni processo una macchina virtuale ad esso interamente dedicata e quindi con risorse proprie



Il sistema operativo e la macchina reale



Che cosa è un processo per il SO?

- Processo ≠ programma !
- Rappresenta un'istanza di un programma composta da:
  - codice eseguibile (il programma stesso)
  - dati del programma
  - informazioni relative al suo funzionamento (stato)
- Lo stesso programma può essere associato a più processi:
  - Un programma può essere scomposto in varie parti e ognuna di esse può essere associata ad un diverso processo
  - Lo stesso programma può essere associato a diversi processi quando diverse copie del medesimo processo sono mandate in esecuzione

Lo stato di un processo

- Lo stato del processo può essere distinto fra stato interno e stato esterno.
- Lo stato interno indica:
  - la prossima istruzione del programma che deve essere eseguita;
  - i valori delle variabili e dei registri utilizzati dal processo.
- Lo stato esterno indica se il processo è:
  - in attesa di un evento, ad es. la lettura da disco o l'inserimento di dati da tastiera;
  - in esecuzione;
  - pronto per l'esecuzione, e quindi attende di accedere all'utilizzo della CPU.

I processi ed il sistema operativo

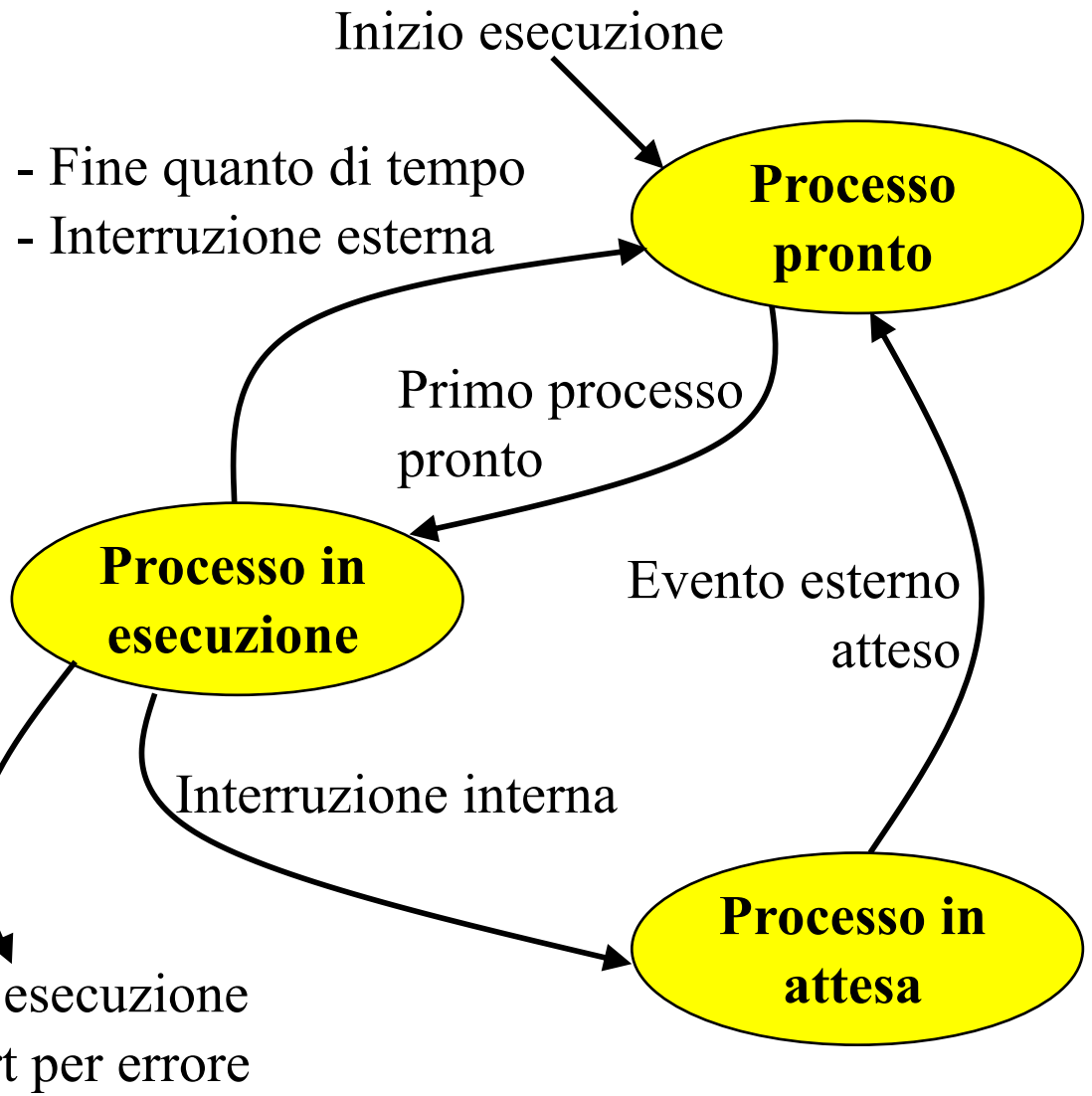
- Anche il sistema operativo è implementato tramite processi;
- Il sistema operativo è garante che i conflitti tra i processi siano controllati e gestiti correttamente;
- Il sistema operativo viene eseguito in modalità privilegiata (kernel mode o supervisor), così da poter controllare gli altri processi eseguiti in modalità user.

Chiamate al supervisor

- I processi utente per eseguire operazioni privilegiate (accesso a file, accesso ad altre risorse, operazioni di I/O, ecc.) invocano il supervisor tramite chiamate di sistema
- Perché usare la modalità privilegiata (supervisor)?
  - Le operazioni di I/O sono operazioni riservate:
    - un processo A non deve poter andare a scrivere messaggi su un terminale non associato allo stesso processo A;
    - un processo A non deve poter leggere caratteri immessi da un terminale non associato allo stesso processo A.
  - Un processo non deve poter sconfinare al di fuori del proprio spazio di memoria:
    - per non accedere allo spazio di memoria associato ad un altro processo, modificando codice e dati di quest'ultimo;
    - per non occupare tutta la memoria disponibile nel sistema, bloccandolo e rendendolo così inutilizzabile da altri processi.
  - La condivisione di risorse (dischi, CPU, ecc.) deve essere tale da cautelare i dati di ogni utente;

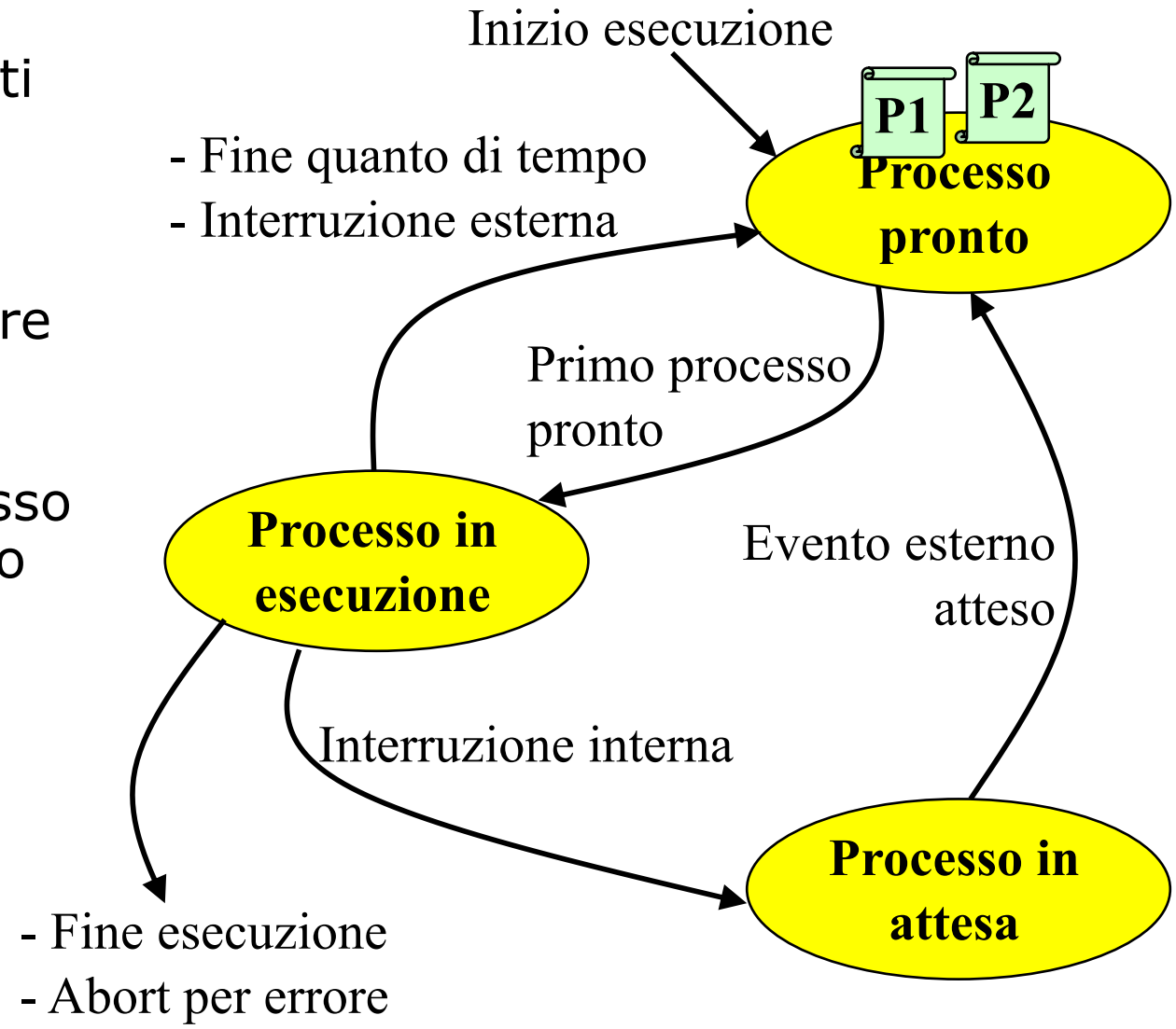
Stato di un processo (1)

- *In esecuzione*: assegnato al processore ed eseguito da esso
- *Pronto*: può andare in esecuzione, se il gestore dei processi lo decide
- *In attesa*: attende il verificarsi di un evento esterno per andare in stato di *pronto*



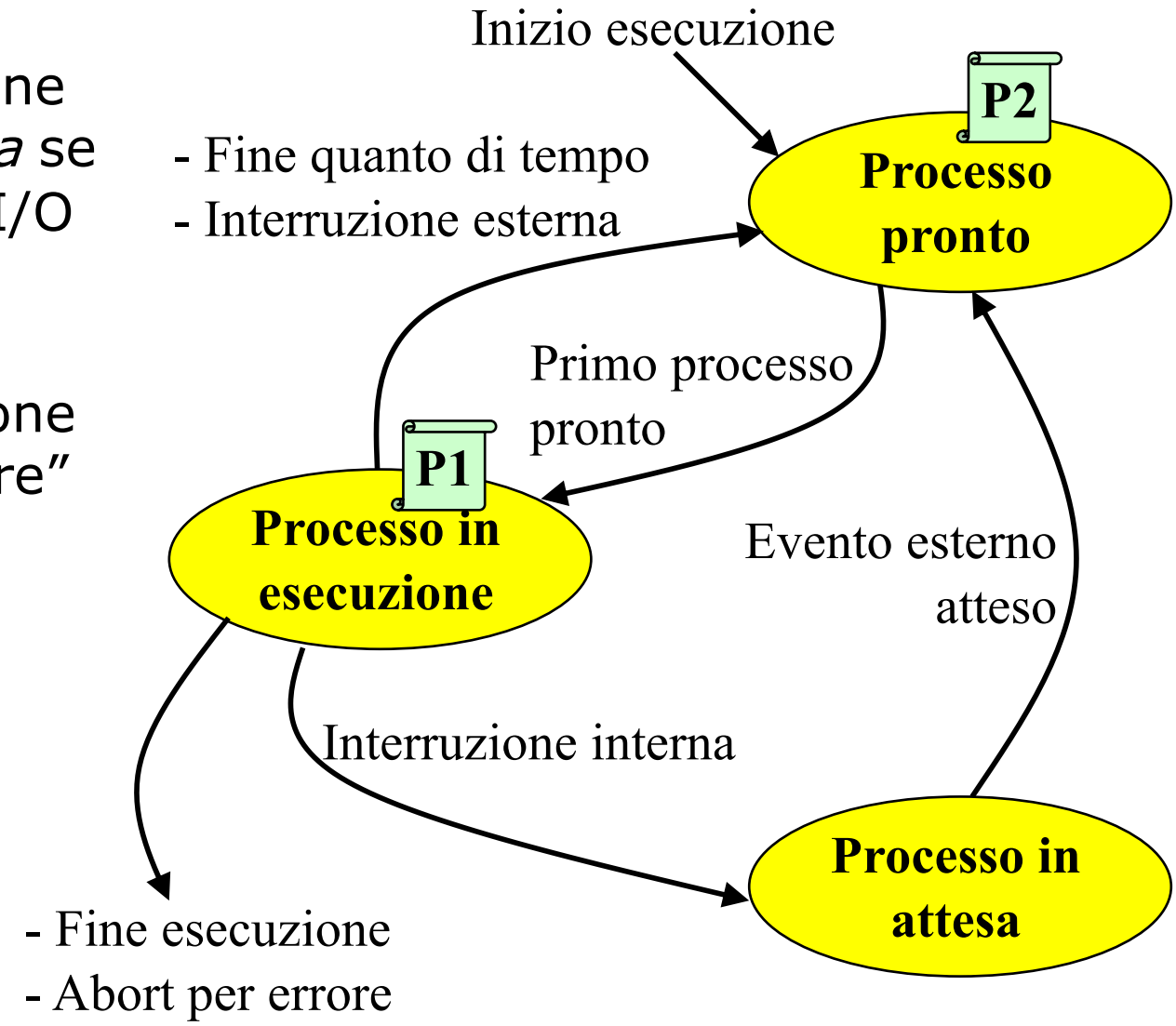
Stato di un processo (2)

- I processi appena creati sono messi in stato di *pronto*
- Il nucleo decide quale processo pronto mettere in stato di esecuzione
- Il nucleo assegna il processore a un processo per un quanto di tempo
  - Coda dei processi pronti
  - Round-robin
  - Priorità dei processi



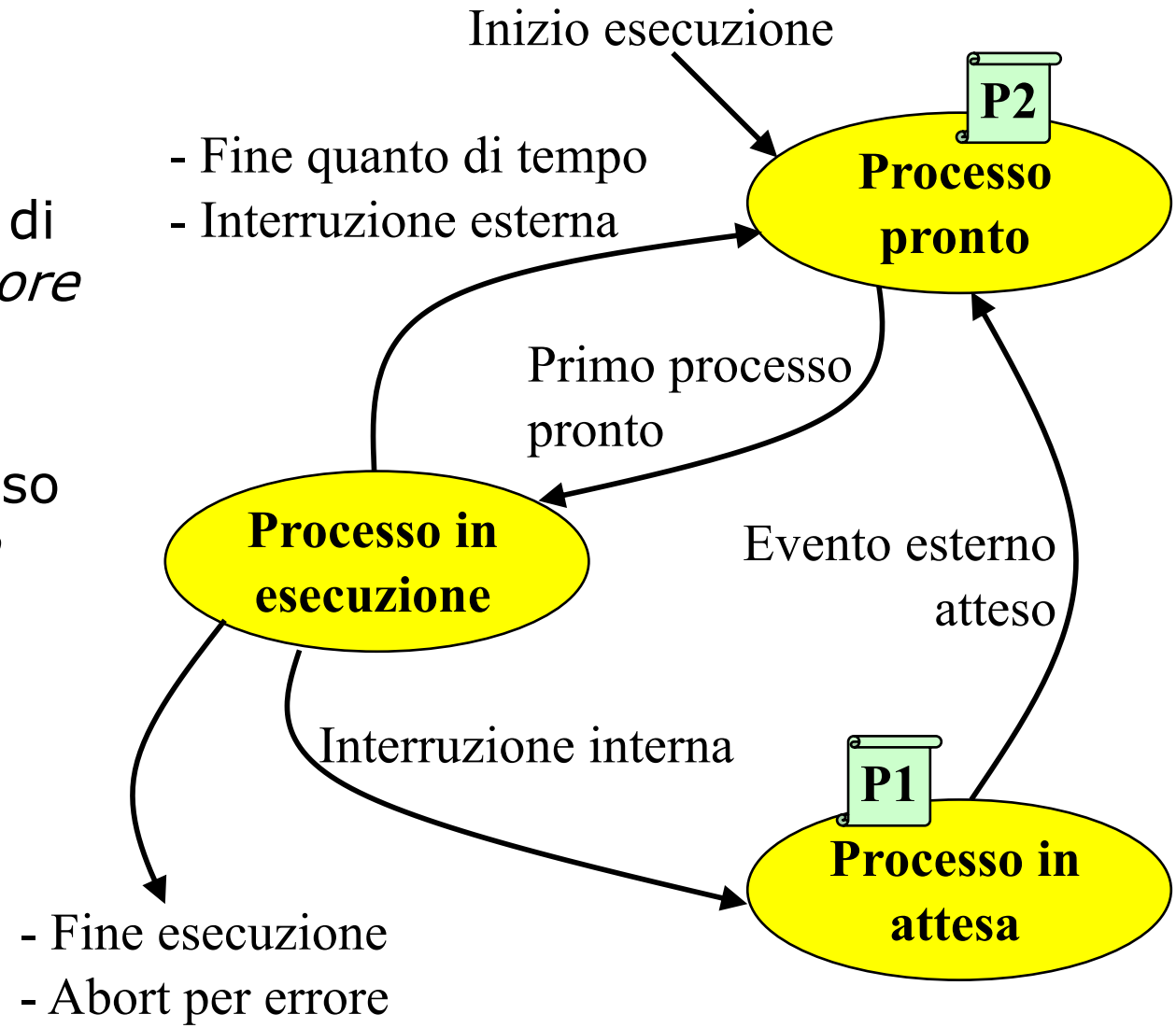
Stato di un processo (3)

- Il processo in esecuzione passa in stato di *attesa* se richiede operazioni di I/O (*interruzione interna*)
- Corrisponde alla esecuzione dell'istruzione "chiamata a supervisore" (*SuperVisor Call*, SVC)



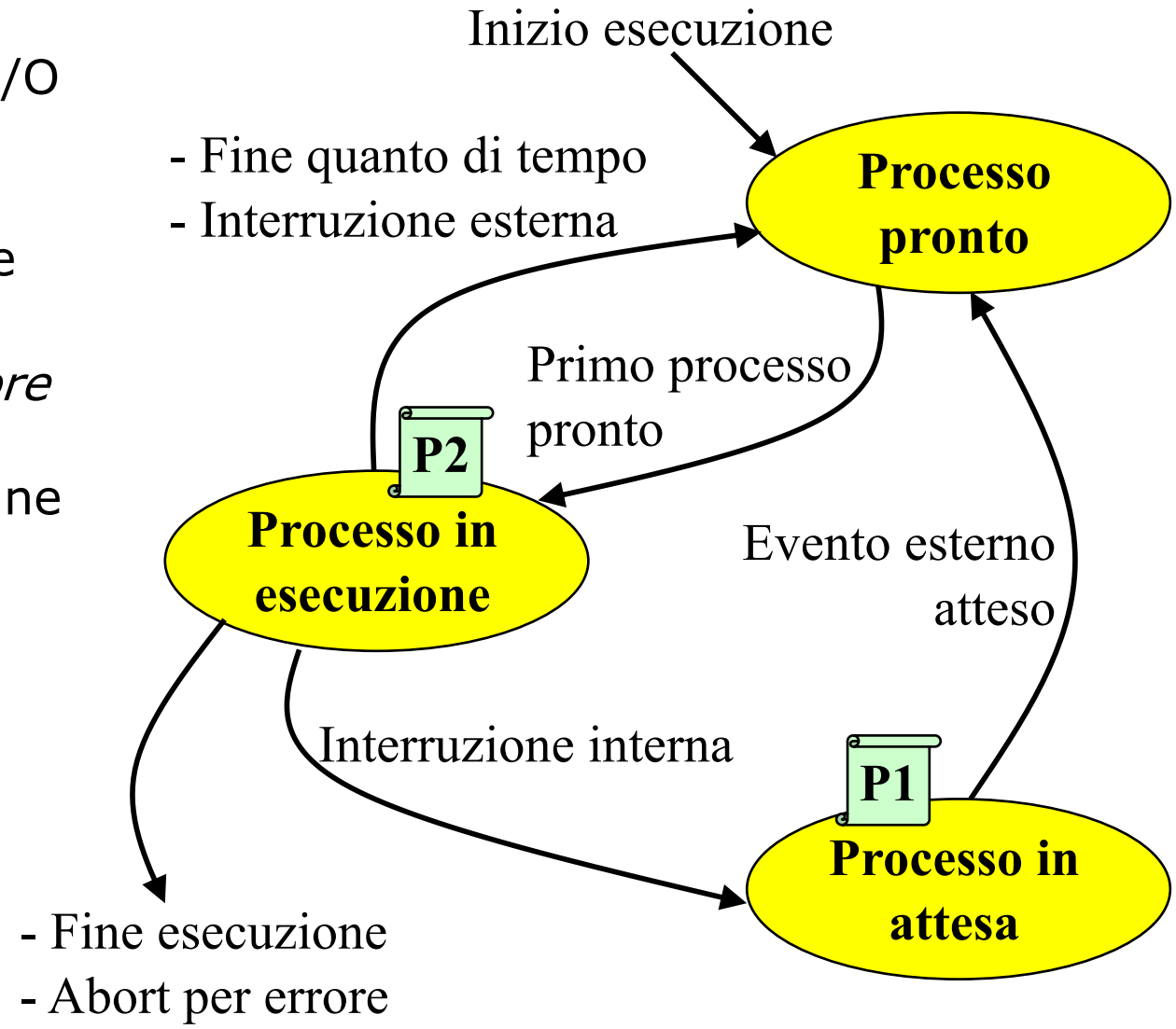
Stato di un processo (4)

- *Cambiamento di contesto*:
  - Salvare il *contesto* di P1 nel suo *descrittore di processo*
- Il processore è ora libero, un altro processo passerà in *esecuzione*



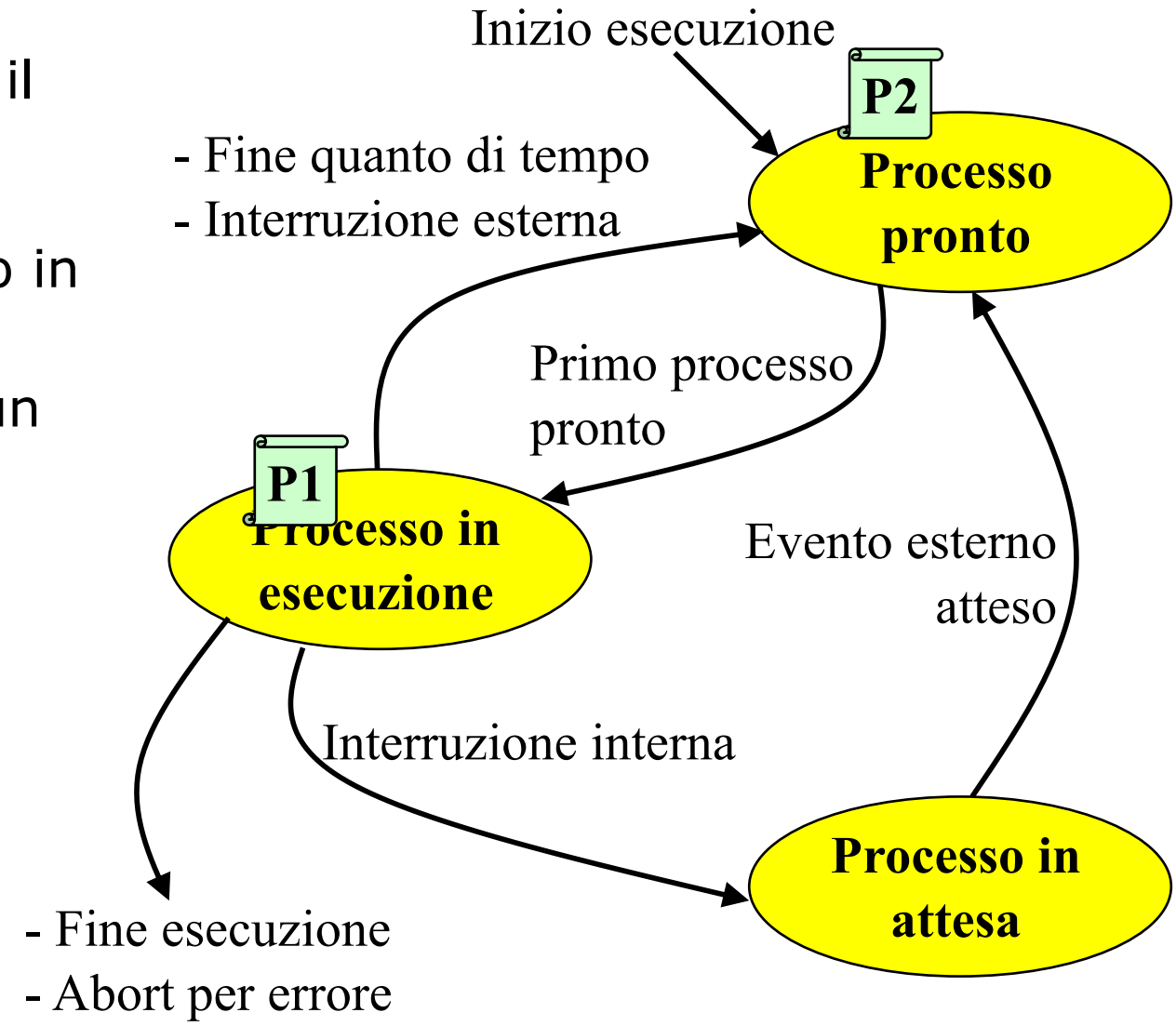
Stati di un processo (5)

- Quando l'operazione di I/O è finita viene generata un'*interruzione esterna*
- Il processo in esecuzione viene interrotto
- Il nucleo esegue il *gestore delle interruzioni* che esegue le azioni opportune
- P1 può tornare *pronto*
- Il nucleo sceglie quale processo mandare in esecuzione



Stato di un processo (6)

- *Pre-emption*: quando il quanto di tempo è scaduto, il nucleo interrompe il processo in esecuzione
- Si cerca di garantire un uso equo della CPU a tutti i processi



La gestione del quanto di tempo

- Il quanto di tempo è gestito da una particolare interruzione, generata dall'orologio di sistema:
  - a una frequenza definita, il dispositivo che realizza l'orologio di sistema genera un'interruzione. La routine di risposta relativa incrementa una variabile opportuna che contiene il tempo di esecuzione del processo corrente
  - se il quanto di tempo non è scaduto la routine termina, se non ci sono interruzioni annidate, il processo prosegue nell'esecuzione
  - se invece il quanto di tempo è scaduto viene invocata una particolare funzione del nucleo (preempt) che cambia lo stato del processo da esecuzione a pronto, salva il contesto del processo e attiva una particolare funzione del nucleo (change) che esegue una commutazione di contesto e manda in esecuzione un processo pronto.

Esercizio

- Siano P e Q due processi lanciati su un sistema monoprocessore. P contiene una scanf, mentre Q non comporta alcuna chiamata al supervisor. Dire se ciascuna delle seguenti affermazioni é vera o falsa. Giustificare le risposte.
  - Il processo P potrebbe terminare senza mai essere mai essere nello stato “in attesa”

  Falso.

  Dal momento che contiene una scanf dovrà necessariamente effettuata una supervisor call e il suo stato diverrà “in attesa”

Esercizio

- Siano P e Q due processi lanciati su un sistema monoprocessore. P contiene una scanf, mentre Q non comporta alcuna chiamata al supervisor. Dire se ciascuna delle seguenti affermazioni é vera o falsa. Giustificare le risposte.
  - Se il processo Q viene lanciato prima di P allora Q termina sicuramente prima di P

  Falso.

  Non è possibile sapere quale processo terminerà prima a priori dal momento che ad ogni processo è garantito un quanto di tempo alla volta.

Esercizio

- Siano P e Q due processi lanciati su un sistema monoprocessore. P contiene una scanf, mentre Q non comporta alcuna chiamata al supervisor. Dire se ciascuna delle seguenti affermazioni é vera o falsa. Giustificare le risposte.
  - Una volta lanciato Q rimarrà sempre nello stato “in esecuzione”

  Falso.

  Non è possibile affermarlo con certezza: se Q dovesse terminare prima dello scadere del quanto di tempo allora rimmarrà sempre nello stato “in esecuzione”, viceversa sarà posto nello stato di “pronto”

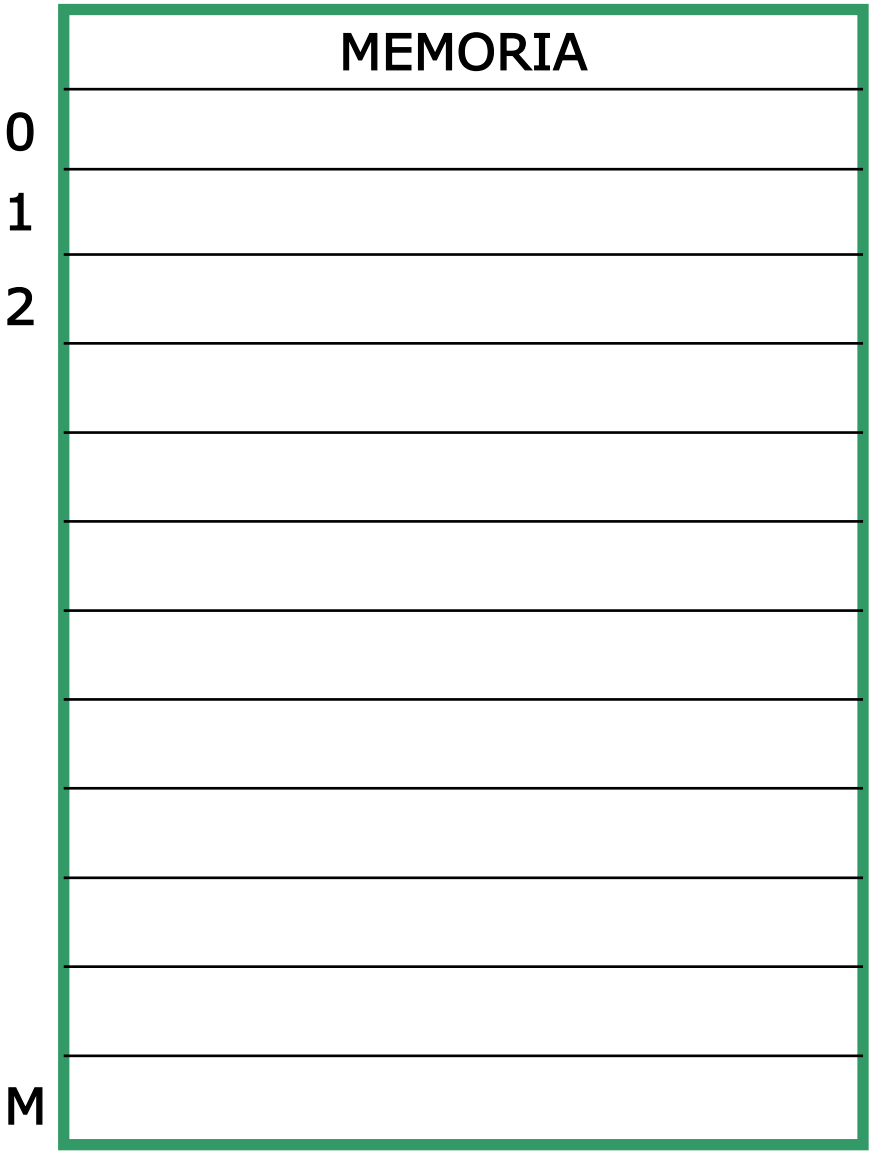
Gestione della memoria nel sistema operativo

Gestione della memoria

- La gestione concorrente di molti programmi appllicativi comporta la presenza di molti programmi in memoria centrale
- Il SO offre ad ogni programma applicativo la visione di una memoria virtuale, che può avere dimensioni maggiori di quella fisica
- Per gestire la memoria virtuale il SO dispone di diversi meccanismi:
  - Rilocazione
  - Paginazione
  - Segmentazione

Il modello della memoria

- E' un modello lineare
- La memoria è una sequenza di celle numerate da 0 fino a un valore massimo M
- Il numero che identifica ogni cella è detto indirizzo
- La dimensione della cella dipenda dal tipo di calcolatore (per noi sarà di 8 bit, ossia un byte)

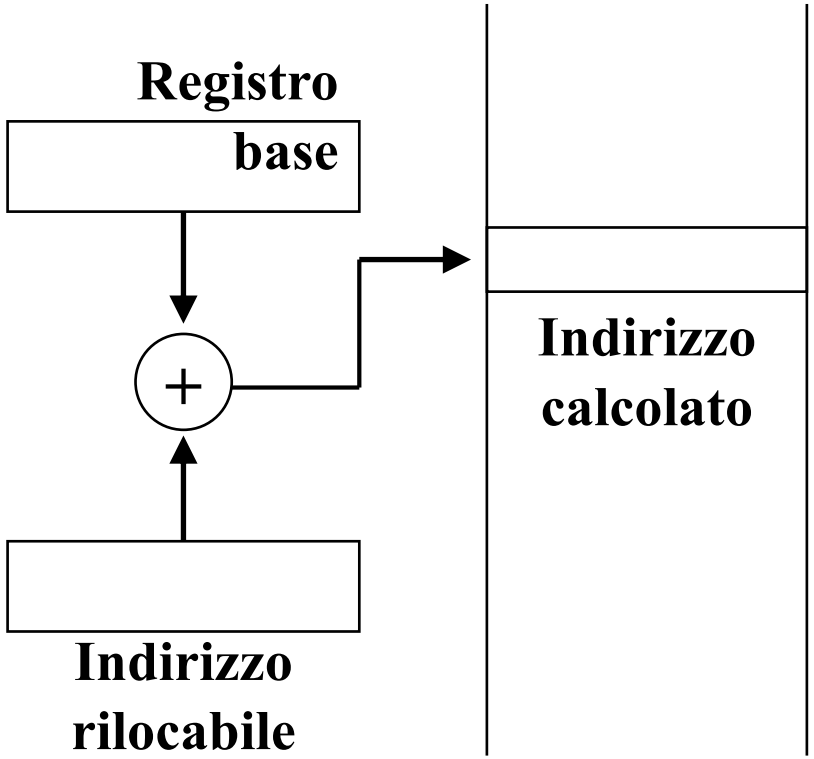


Spazio di indirizzamento

- Lo spazio di indirizzamento è il numero massimo di indirizzi possibili della memoria
- Dipende dalla lunghezza in bit degli indirizzi
- Se gli indirizzi sono lunghi N bit, lo spazio di indirizzamento è di 2^N celle
- Tutte le celle devono essere indirizzabili (cioè devono avere un indirizzo), quindi Dimensione memoria $\leq$ Spazio indirizzamento
- Le dimensioni della memoria sono generalmente espresse in:
  - KB (Kilobyte) = 2^{10} byte
  - MB (Megabyte) = 2^{20} byte
  - GB (Gigabyte) = 2^{30} byte

Memoria virtuale vs. fisica

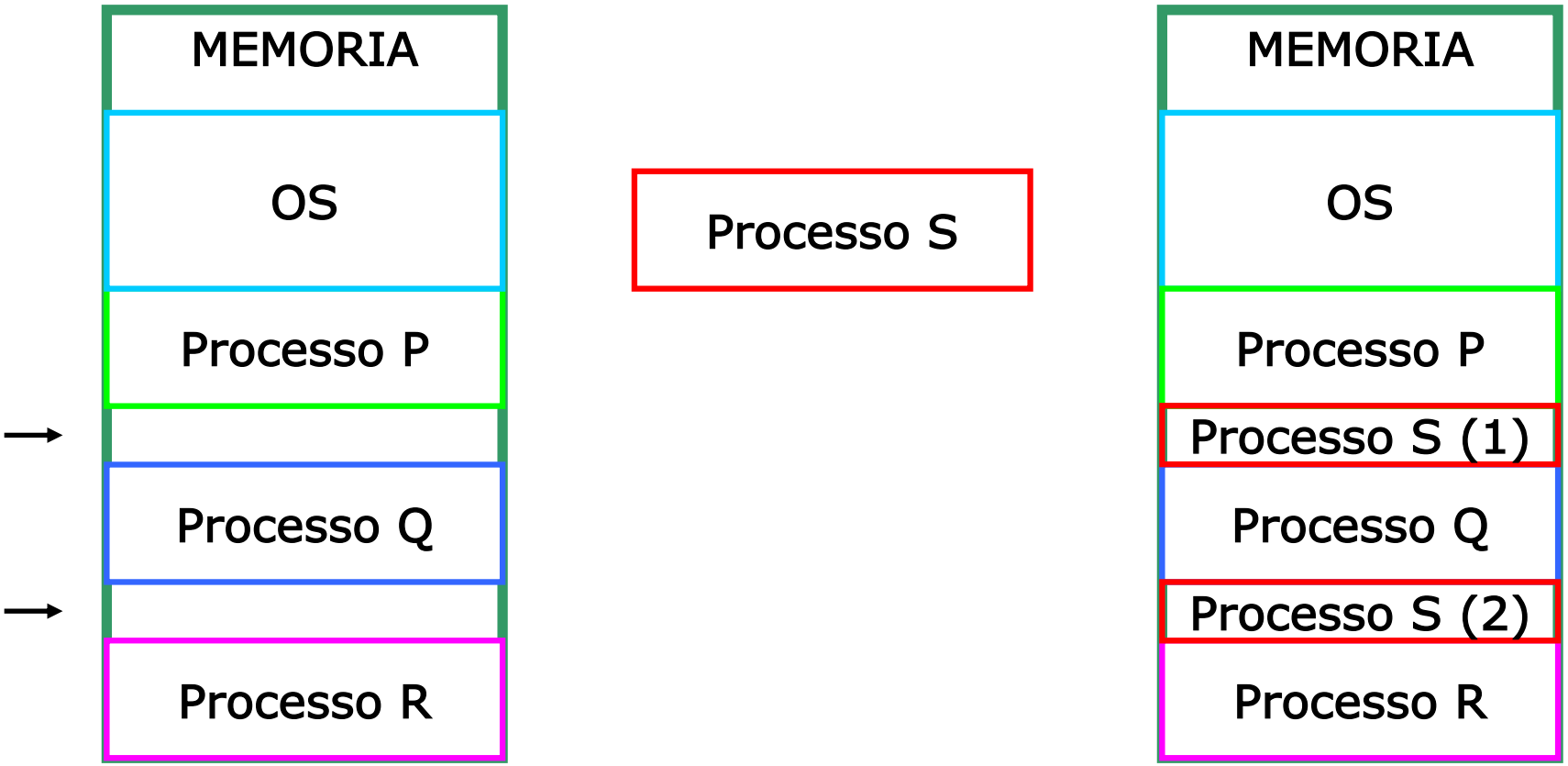
- Gli indirizzi contenuti in un programma eseguibile sono indirizzi *virtuali* e danno riferimento alla memoria *virtuale*
- La memoria effettivamente presente nel calcolatore è la memoria *fisica* e i suoi indirizzi sono detti *indirizzi fisici*
- La rilocazione dinamica è uno dei meccanismi di trasformazione tra virtuale e fisico



Memoria virtuale vs. fisica

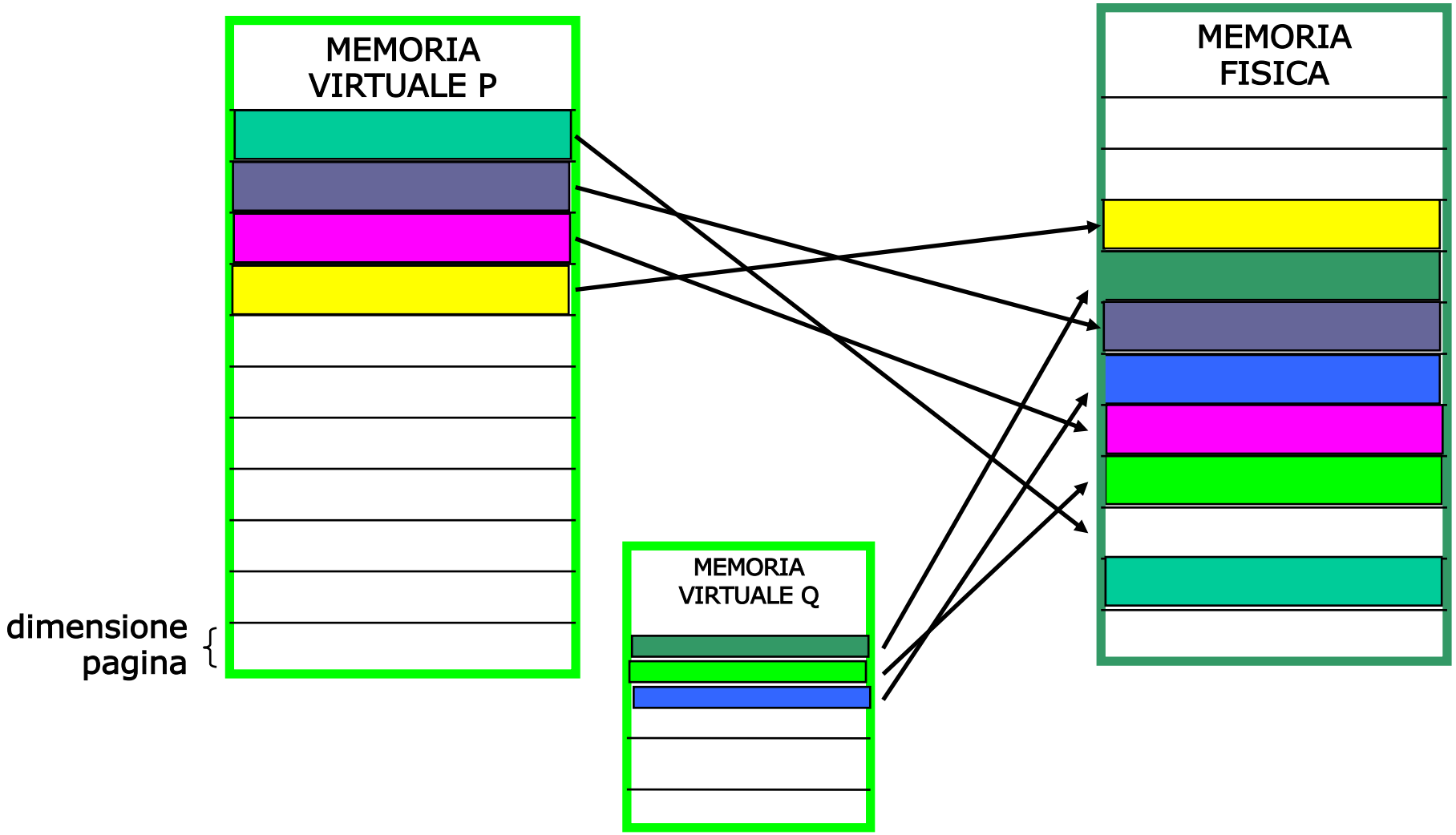
- La memoria virtuale e quella fisica non coincidono per i seguenti motivi:
  - nella memoria fisica risiedono contemporaneamente il s.o. e i diversi processi
  - conviene mantenere nella memoria fisica una sola copia di parti di programmi che sono uguali in diversi processi (memoria condivisa)
- per evitare la frammentazione della memoria (spazi vuoti in memoria inutilizzabili) è utile allocare i programmi suddividendoli in pezzi
- la memoria fisica può essere insufficiente a contenere la memoria virtuale di tutti processi

Soluzione al problema della frammentazione



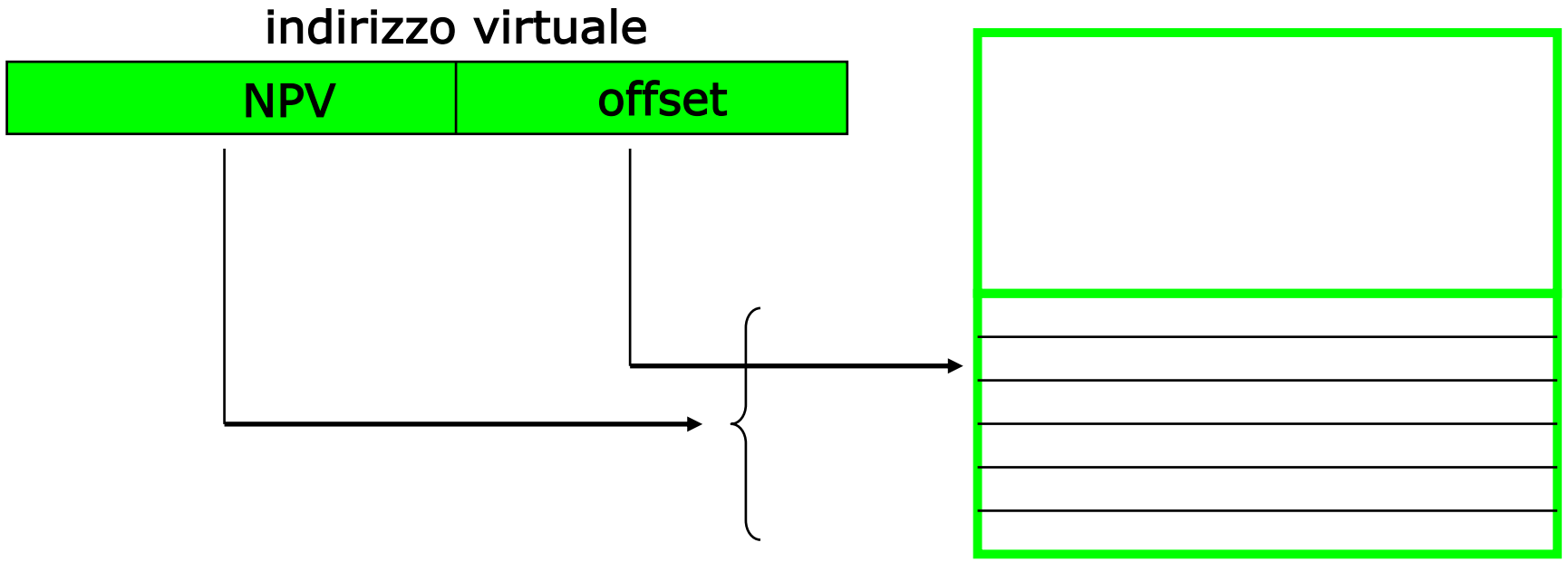
Paginazione

- Si rinuncia ad avere una zona contigua della memoria fisica per ciascun processo
- La memoria virtuale del programma viene suddivisa in porzioni (pagine virtuali) di lunghezza fissa (potenza di 2, es: 4K)
- La memoria fisica viene divisa in pagine fisiche della stessa dimensione
- Le pagine virtuali di un programma vengono caricate in altrettante pagine fisiche, non necessariamente contigue


MEMORIA
VIRTUALE P
MEMORIA
FISICA
MEMORIA
VIRTUALE Q
dimensione
pagina

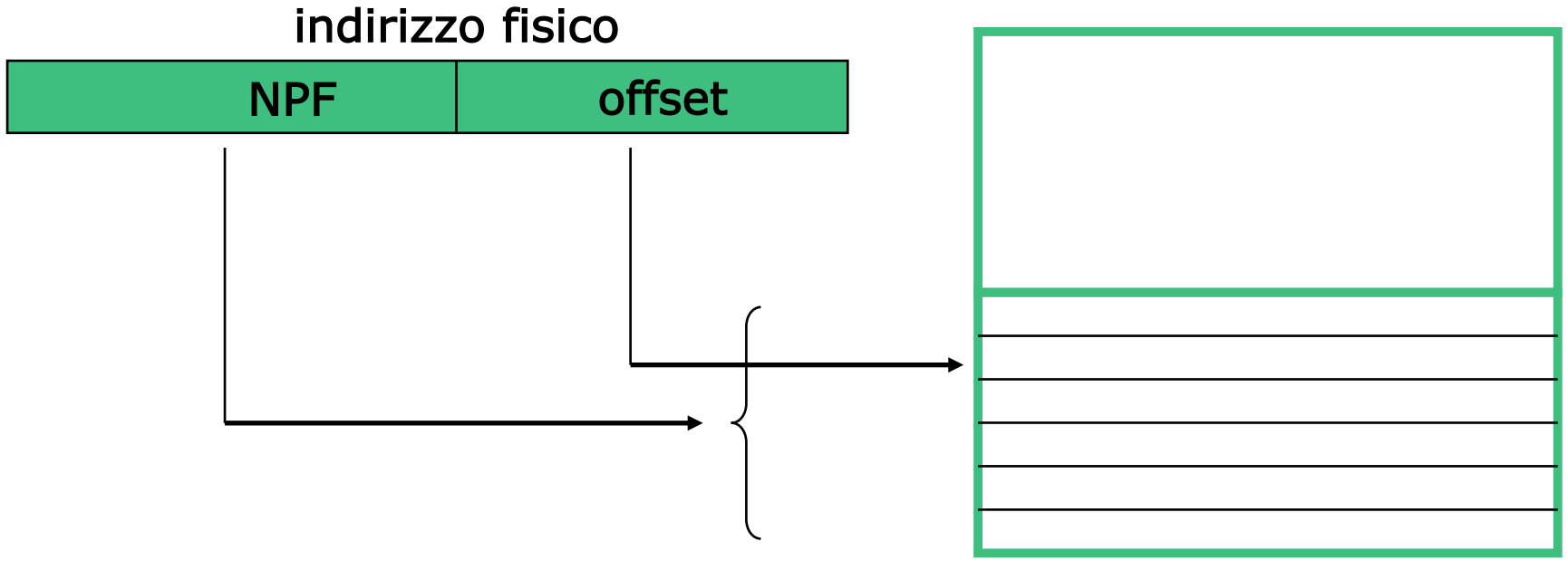
Struttura degli indirizzi virtuali

- Un indirizzo virtuale è costituito da un numero di pagina virtuale (NPV) e da uno spiazzamento (offset) all'interno della pagina

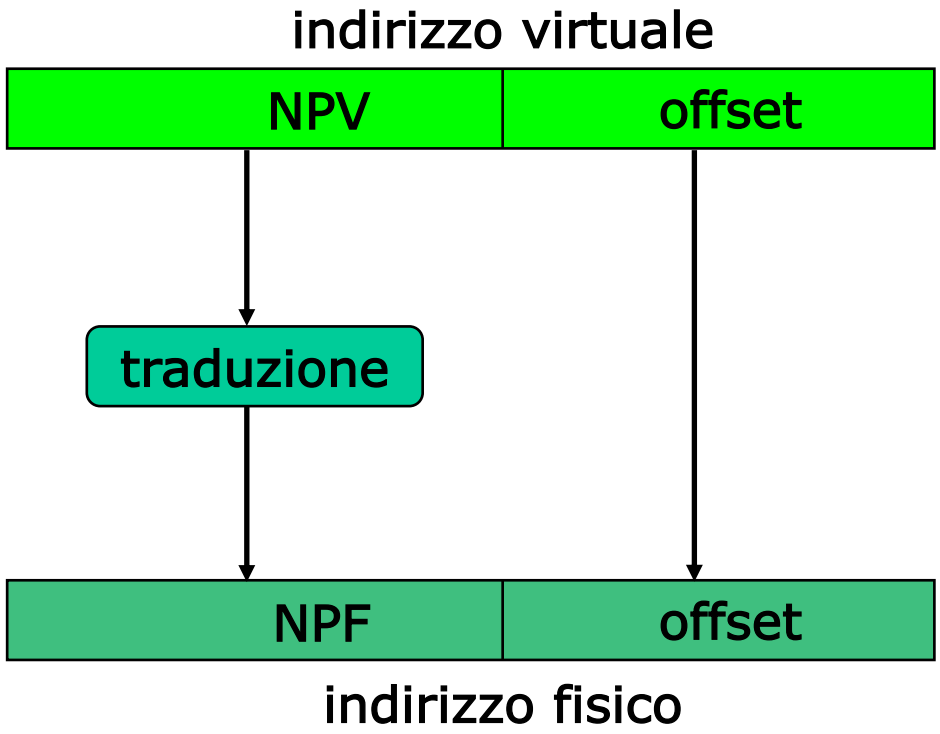


Struttura degli indirizzi fisici

- E' del tutto analoga: si hanno un numero di pagina fisica (NPF) e da uno spiazzamento (offset) all'interno della pagina



Traduzione dal virtuale al fisico

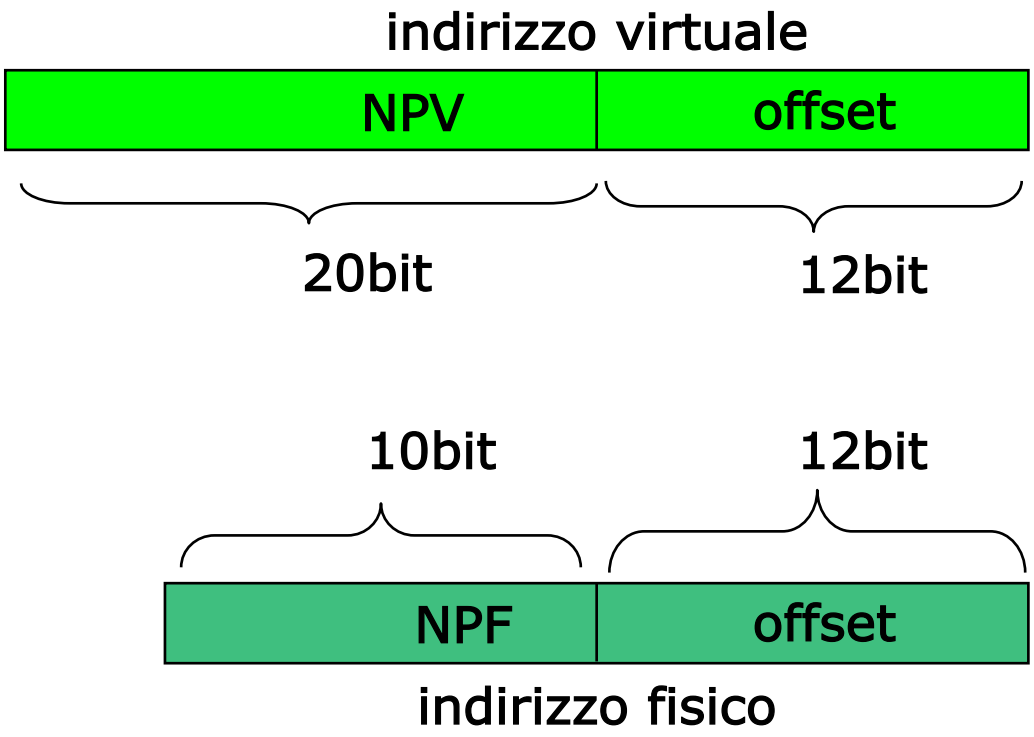


le pagine virtuali e quelle fisiche hanno la stessa dimensione, quindi l'offset è lo stesso

Esempio

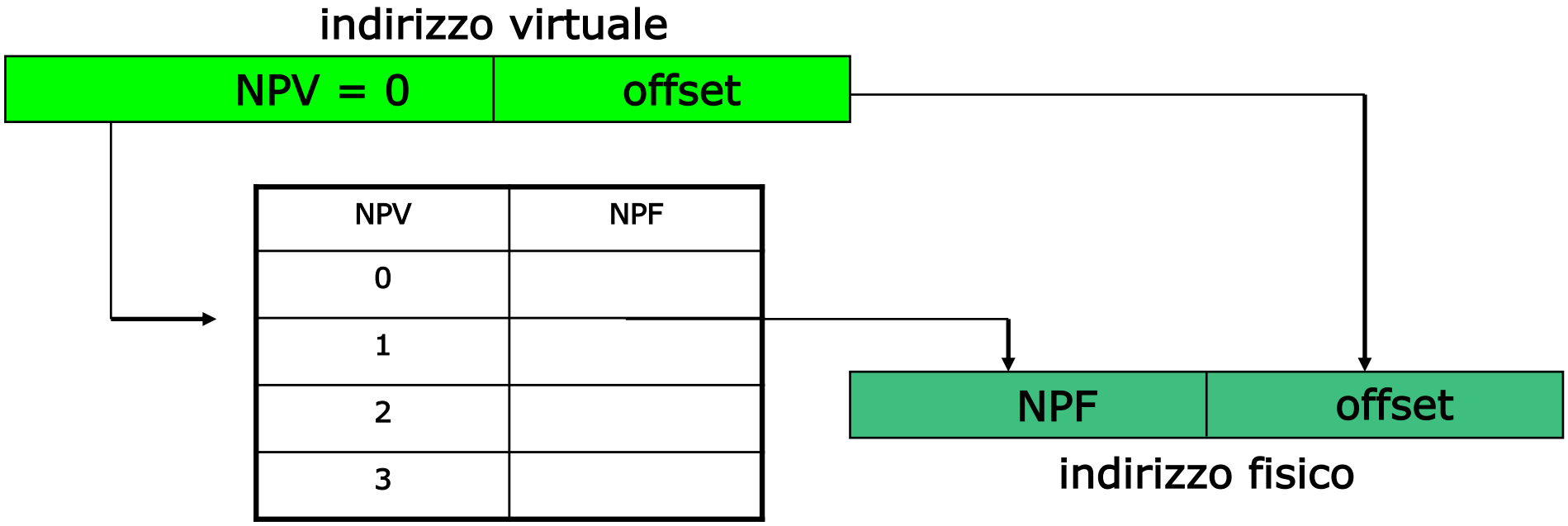
- Spazio di indirizzamento virtuale:
  indirizzi da 32 bit ⇨ 2^{32} indirizzi
- Dimensione di pagina:
  4K parole (o celle) ⇨ 2^{12} byte (1 cella occupa 1 byte)
- Numero di pagine dello spazio di indirizzamento virtuale = $2^{32} / 2^{12} = 2^{20}$ pagine
- Spazio di indirizzamento fisico: 4M parole (o celle) ⇨ 2^{22} indirizzi
- Numero di pagine dello spazio di indirizzamento fisico = $2^{22} / 2^{12} = 2^{10}$ pagine

Esempio


indirizzo virtuale
NPV
offset
20bit
12bit
10bit
12bit
NPF
offset
indirizzo fisico

La tabella delle pagine

- E’ il meccanismo più semplice per la traduzione da virtuale a fisico



C’è una tabella delle pagine per ciascun processo

Memory Management Unit

- Per accelerare la traduzione da NPV a NPF si ricorre allora alla MMU
- La MMU è una memoria particolarmente veloce (memoria associativa) dalle dimensioni ridotte, contenente solo le informazioni sulle pagine più utilizzate
- Visto che gli NPV e gli NPF si riferiscono alle pagine di un processo, ogni volta che il processo in esecuzione cambia la MMU dovrebbe essere tutta riscritta
- Per evitare ciò si aggiunge una colonna che dice a quale processo appartengono le pagine e un registro che dice qual è il processo attualmente in esecuzione

Pagine residenti e non

- Durante l'esecuzione di un programma solo un certo numero delle sue pagine virtuali è caricato in altrettante pagine fisiche
- Tali pagine sono dette pagine residenti
- Ad ogni accesso alla memoria si controlla che all'indirizzo virtuale corrisponda una pagina residente, altrimenti si produce un interrupt di segnalazione di errore detto page-fault
- Il processo viene sospeso in attesa che la pagina richiesta venga caricata in memoria, eventualmente scaricando su disco una pagina già residente per liberare lo spazio necessario

Esercizio 1

- Un sistema dotato di memoria virtuale con paginazione è caratterizzato dai seguenti parametri: l'indirizzo logico è di 13 bit e l'indirizzo fisico è di 12 bit; la dimensione delle pagine è di 512 byte.

  Definire la struttura dell'indirizzo logico e di quello fisico indicando la lunghezza dei campi che li costituiscono.

  - Indirizzo logico: NPV: 4 bit offset logico: 9 bit
  - Indirizzo fisico: NPF: 3 bit offset fisico: 9 bit

Esercizio 2

- Un sistema dispone di 8 Kbyte di memoria fisica indirizzabile; inoltre è dotato di memoria virtuale con paginazione caratterizzata dai seguenti parametri: l'indirizzo logico è di 15 bit e le pagine sono di 256 byte.

  - Qual è la dimensione della memoria virtuale indirizzabile?

  15bit → 2^15 byte → 32Kbyte

  - Definire la struttura dell'indirizzo logico e di quello fisico indicando la lunghezza dei campi che li costituiscono

  256 byte → 2^8 byte → offset = 8bit

  8kbyte → 2^13 byte → indirizzo fisico = 13 bit

  Indirizzo fisico: NPF 5bit, offset 8bit

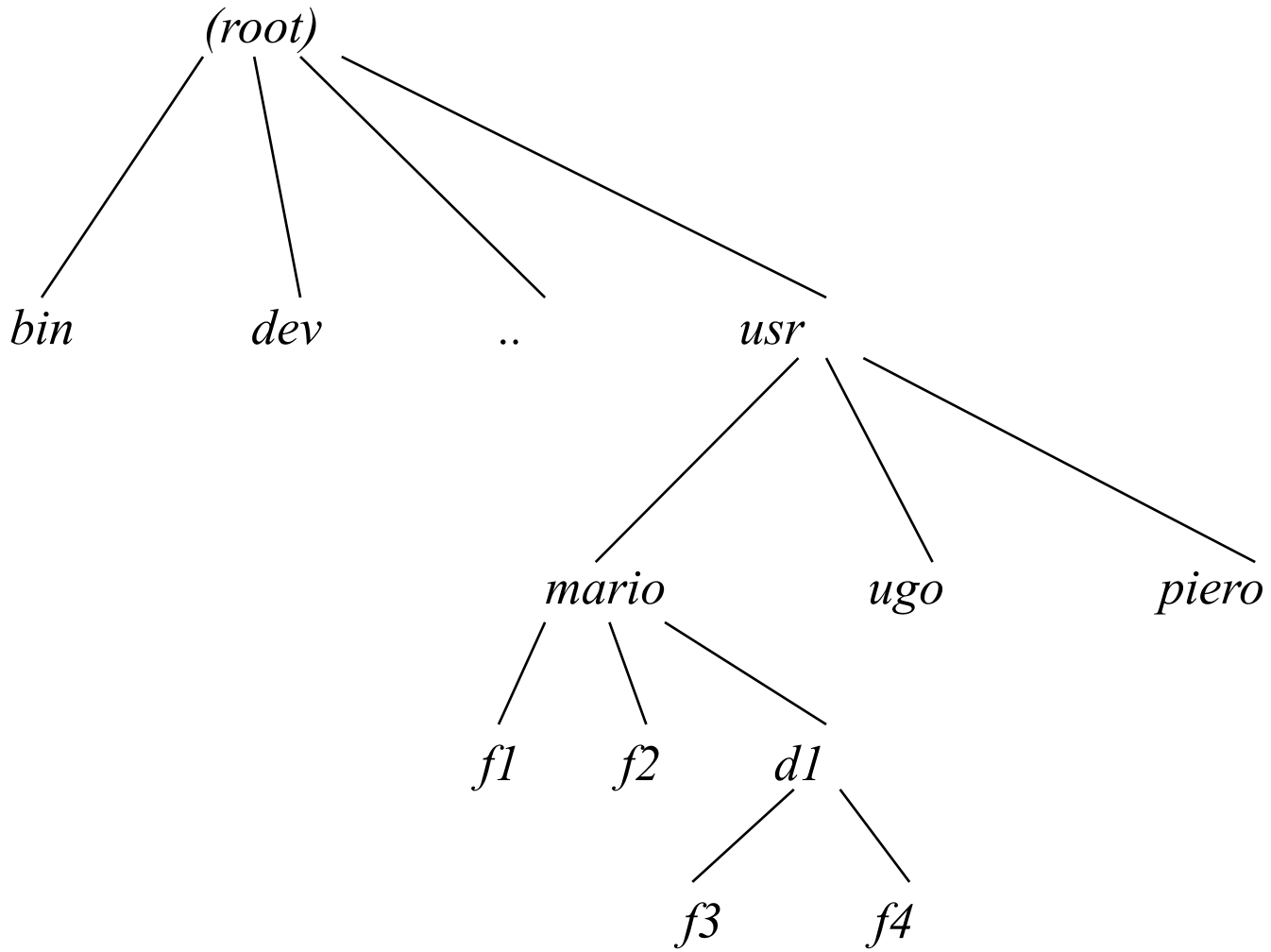
  Indirizzo logico: NPV 7bit, offset 8bit

Gestione del file system e altri sotto-sistemi

Gestione del file system

- Il SO si occupa di gestire i *file* sulla memoria di massa:
  - Creare un file
  - Dargli un nome
  - Collocarlo in un opportuno spazio nella memoria di massa
  - Accedervi in lettura e scrittura
- Gestione dei file indipendente dalle caratteristiche fisiche della memoria di massa
- I file vengono inclusi all'interno di *directory* (o *cataloghi*):
  - Hanno una tipica organizzazione ad albero
  - Alcuni sistemi operativi permettono una struttura a grafo

La struttura ad albero



Organizzazione dei file

- A ciascun utente è normalmente associata una directory specifica, detta *home directory*
- Il livello di *protezione* di un file indica quali operazioni possono essere eseguite da ciascun utente
- Ciascun file ha un *pathname* (o nome completo) che include l'intero cammino dalla radice dell'albero
- Il *contesto* di un utente all'interno del file system è la directory in cui correntemente si trova

Gestione delle periferiche

- Sono meccanismi software a cui è affidato il compito di trasferire dati da e verso le periferiche
- Consentono ai programmi applicativi di leggere o scrivere i dati con primitive di alto livello che nascondono la struttura fisica delle periferiche (e.g., nel sistema Unix le periferiche sono viste come file speciali)
- Si distingue generalmente fra:
  - Driver fisici, che vengono utilizzati dal gestore delle interruzioni per il trasferimento dei dati
  - Driver logici, che fanno parte del sistema operativo e forniscono una gerarchia di memorie

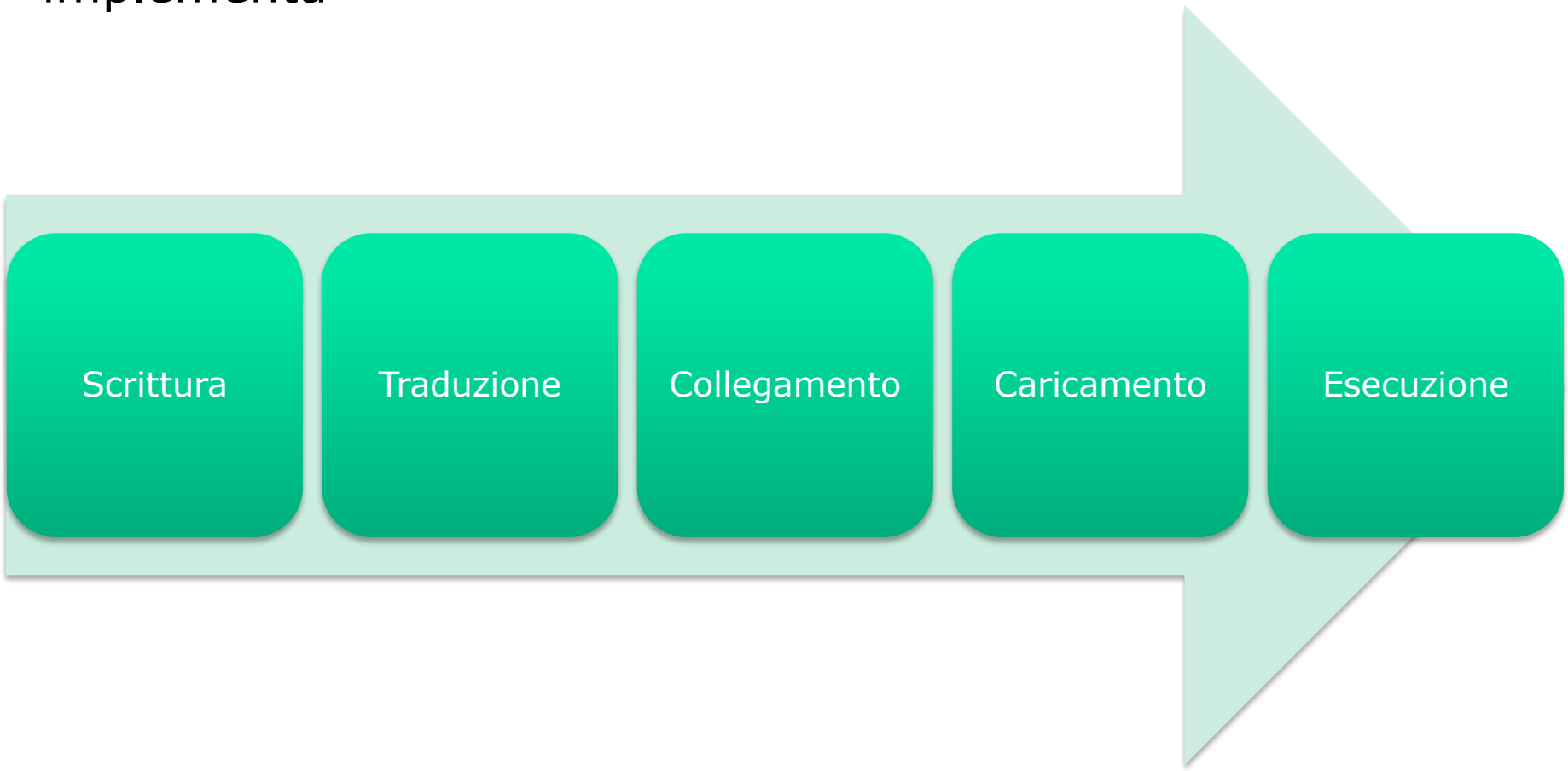
Gestione dell'interfaccia utente

- Il SO fornisce un interprete dei comandi inseriti dall'utente attraverso la tastiera o il mouse
- L'interfaccia utente può essere
  - Testuale (esempio: DOS)
  - Grafica (esempio: Windows)
- Consente l'inserimento di diversi comandi:
  - Esecuzione di programmi applicativi
  - Operazioni sulle periferiche
  - Configurazione dei servizi del SO
  - Operazioni sul file system (creazione, rimozione, copia, ricerca, ecc.)

Catena di sviluppo in C

La catena di sviluppo in C

- Il C è un linguaggio compilato
- Possiamo individuare 5 passi per passare dalla definizione di un algoritmo ad un programma in esecuzione che lo implementa



1. Scrittura

- Il programma, costituito da una sequenza di caratteri, viene **composto e modificato** usando un qualsiasi editor
- Così otteniamo un **codice sorgente** memorizzato in memoria di massa in un file di testo (es. XYZ.c)

2. Traduzione

- Il compilatore si occupa della traduzione dal linguaggio di alto livello al linguaggio macchina
- Durante questa fase si riconoscono i simboli, le parole e i costrutti del linguaggio:
  - eventuali messaggi diagnostici segnalano errori di sintassi
- Viene generato il codice macchina in forma binaria : a partire dal **codice sorgente** si genera il **codice oggetto**, cioè in un file **binario**

3. Collegamento (linking)

- Il collegatore (linker) deve collegare fra loro il file oggetto ed altre librerie utilizzate (es. librerie di I/O)
- Si rendono globalmente coerenti i riferimenti agli indirizzi dei vari elementi collegati
- Si genera un **programma eseguibile**, un file binario che contiene il codice macchina del programma eseguibile completo, di nome XYZ.exe
- Messaggi di errore possono essere dovuti ad errori nel citare i nomi delle funzionalità di librerie esterne da collegare
- Il programma sarà effettivamente eseguibile solo dopo che il contenuto del file sarà stato caricato nella memoria di lavoro (centrale) del calcolatore

4. Caricamento (loading)

- Il caricatore (*loader)* individua una porzione libera della memoria di lavoro e vi copia il contenuto del programma eseguibile
  - Eventuali messaggi rivolti all'utente possono segnalare che non c'è abbastanza spazio in memoria

5. Esecuzione

- Per eseguire il programma occorre fornire in ingresso i dati richiesti e in uscita riceveremo i risultati (su video o file o stampante)
- Durante l'esecuzione possono verificarsi degli errori (detti "errori di run-time"), quali:
  - calcoli con risultati scorretti (per esempio un overflow)
  - calcoli impossibili (divisioni per zero, logaritmo di un numero negativo, radice quadrata di un numero negativo,….)
  - errori nella concezione dell'algoritmo (l'algoritmo non risolve il problema dato)
- Tutti gli esempi citati si riferiscono ai cosiddetti *errori semantici*