Laboratorio di Informatica

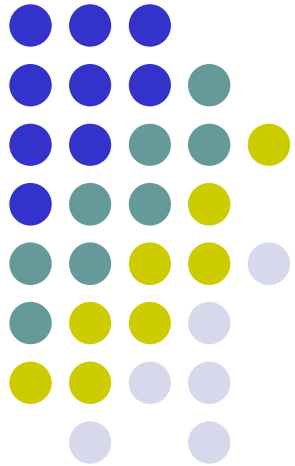
Corso di Laurea in Matematica

A.A. 2007/2008

Dott.Davide Di Ruscio

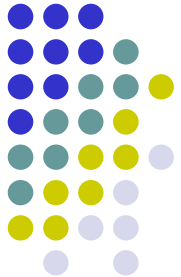
Dipartimento di Informatica

Università degli Studi di L'Aquila



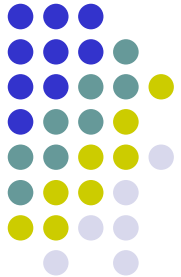
Lezione del 16/01/08

Nota



Questi lucidi sono tratti dal materiale distribuito dalla McGraw-Hill e basati su del materiale fornito dal Prof. Flammini Michele

Avvisi

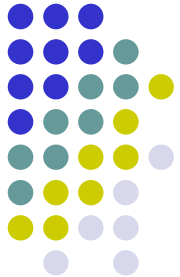


- Il giorno **Giovedì 24 Gennaio 2008** alle ore **11:00** presso l'**aula 2.4**, **Coppito 1** si terrà il primo appello della prova d'esame

- Il giorno **Mercoledì 13 Febbraio 2008** alle **ore 11:00** si terrà il secondo appello della prova d'esame

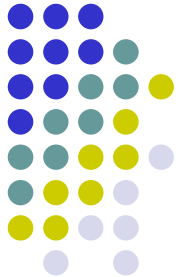
Gli studenti che devono sostenere la prova sono pregati di contattare il docente per posta elettronica

Sommario (I parte)



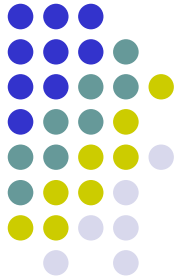
- Concetti fondamentali
- Aspetti architetturali di un sistema di calcolo
  - hardware
  - software
    - software di base
    - software applicativo
- Codifica dell'informazione
  - numeri naturali, interi, reali
  - caratteri
  - immagini
- Macchina di Von Neumann
  - CPU (UC, ALU, registri, clock)
  - memoria centrale
  - bus di bistema
  - periferiche
- Linguaggio macchina
- Linguaggio assembler
- **Sistema operativo**
- Ambiente di programmazione

Sistema Operativo (1/2)



- E’ uno strato software che opera direttamente sull’hardware, isolando gli utenti dai dettagli dell’architettura hardware e fornendo loro un insieme di funzionalità richiamabili tramite comandi (espliciti ed impliciti):
  - copia dei file
  - esecuzione di un programma
  - caricare i programmi in memoria centrale, eseguirli, leggere e scrivere dati da/su memoria di massa
  - …
- È il cuore del software di base
- Permette un utilizzo versatile ed efficiente delle risorse dell’elaboratore

Sistema Operativo (2/2)



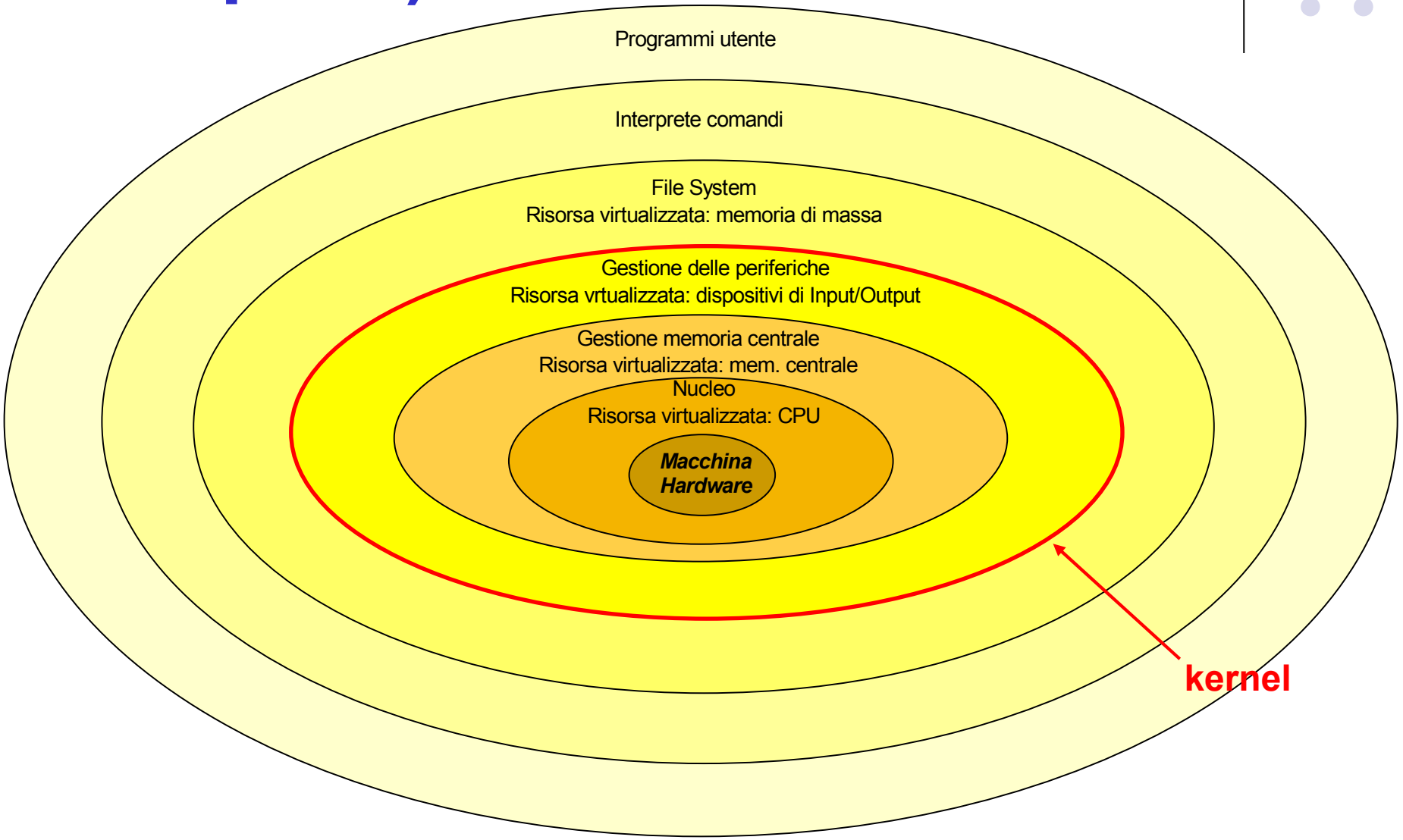
- Cerca di ottimizzare le prestazioni del sistema informatico, determinando le politiche migliori di gestione delle risorse sotto il suo controllo

- Può essere:
  - mono-utente: l'intero sistema è dedicato a un singolo utente
  - multi-utente: diversi utenti condividono lo stesso sistema (il SO nasconde a ciascun utente la presenza degli altri, dando l'impressione che l'intero sistema gli sia dedicato)

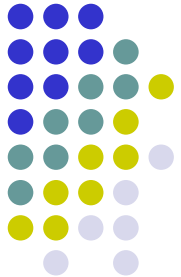
- Per facililtare la sua concezione ed isolare le varie componenti del sistema, il SO è tipicamente organizzato per strati funzionali

- Ogni strato, sfruttando quelli inferiori, "virtualizza" una particolare risorsa, ossia simula la presenza di una risorsa più potente ed accessibile ad un livello logico superiore, mascherandone le peculiarità fisiche e le limitazioni imposte dal numero e dalla condivisione

Architettura del S.O. (a buccia di cipolla)



Nucleo (1/2)

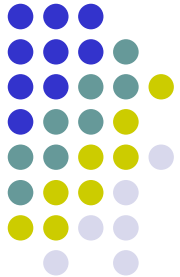


- E' responsabile dell'esecuzione dei programmi da parte dell'unità di elaborazione

- Processo: entità *dinamica* associata all'esecuzione di un programma e caratterizzato da
  1. codice (lista di istruzioni = nozione *statica*)
  2. stato, ossia istruzione corrente, contenuto delle variabili o celle di memoria e dei registri della C.P.U., …

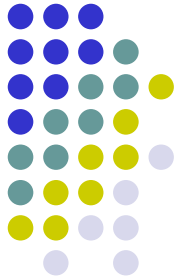
- La corrispondenza tra programma e processi non è necessariamente biunivoca:
  - Ad ogni programma possono corrispondere contemporaneamente più processi, ognuno relativo ad un'esecuzione indipendente con un input ed uno stato differente
  - Al contrario, un programma può essere suddiviso in più parti, ognuna eseguita da un processo differente

Nucleo (2/2)



- In caso siano presenti molti utenti, in nucleo deve garantire l'esecuzione quasi contemporanea di molti processi
- Sfrutta il meccanismo delle interruzioni, mediante le quali viene segnalato alla C.P.U. il verificarsi di eventi asincroni, ossia indipendenti dall'esecuzione del processo, quali lo stato di pronto di una periferica, l'avvenuta esecuzione di un'operazione di I/O, …
- Il nucleo virtualizza la C.P.U. simulando la presenza di una C.P.U. dedicata ad ogni processo in esecuzione
- Anche quando un sistema ha più di una CPU, questi vengono normalmente condivisi fra molti utenti

Gestore della memoria

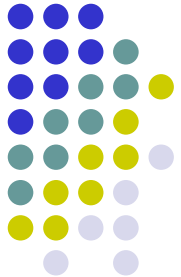


- Ha la funzione di allocare la memoria e partizionarla tra i vari programmi che la richiedono

- In un sistema multi-utente è opportuno che molti programmi siano contemporaneamente presenti in memoria in modo che possa aver luogo la loro pseudo-simultanea esecuzione

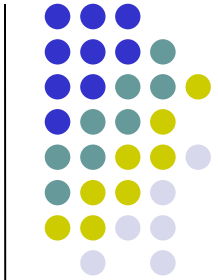
- Offre agli strati superiori una macchina virtuale in cui ciascun programma opera come se avesse disponibile una memoria dedicata

Gestione delle periferiche (driver)



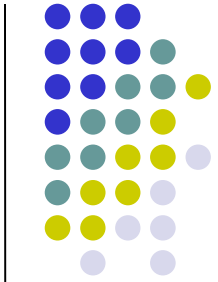
- Sono responsabili delle operazioni di ingresso/uscita che coinvolgono le periferiche
- Sono programmi che eseguono operazioni di ingresso/uscita per uno specifico componente
- Questo strato offre all'utente una visione astratta in cui le caratteristiche hardware delle periferiche vengono mascherate
- L'utente ha a disposizione un insieme di procedure standard di alto livello, che leggono dati di ingresso e scrivono dati in uscita
  - ad esempio per la lettura, scrittura di dati su memorie secondarie
  - scrittura su stampanti, ecc
- Anche in questo caso l'utente ha l'impressione che la periferica sia dedicata

File system



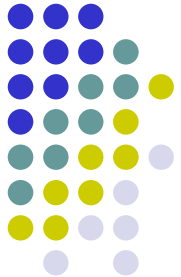
- Responsabile della gestione dei file in memoria di massa
- Struttura i dati in file, li organizza in directory e fornisce all'utente un insieme di funzioni ad alto livello per operare su di essi
- Tramite il file system, ciascun utente può organizzarsi una zona della memoria di massa e garantire che i suoi file siano protetti da accessi esterni
- Consente che alcuni file vengano condivisi fra più utenti

Interprete comandi



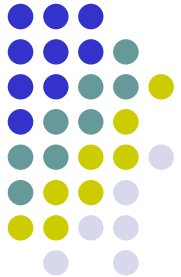
- Consente all'utente di attivare programmi
- Per eseguire un programma, l'interprete comandi svolge in modo trasparente all'utente un insieme di operazioni, tra cui:
  1. accedere al programma, tipicamente residente in memoria di massa, tramite il *file system*
  2. allocare memoria e caricarvi il programma, tramite il *gestore della memoria*
  3. attivare un processo, tramite il *nucleo*

Evoluzione dei Sistemi Operativi



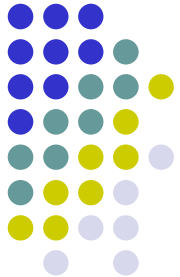
- Sistemi batch con schede (50’s-60’s)
- System/360 IBM compatibili (65-70’s)
- Sistemi operativi UNIX e DOS (80’s)
- WINDOWS (90’s)

Più in dettaglio



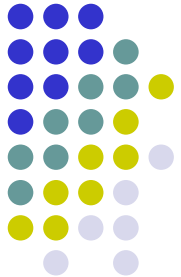
- Gestione dei Processi
- Gestione della Memoria Centrale
- File System
- Gestione delle Periferiche

Gestione dei Processi



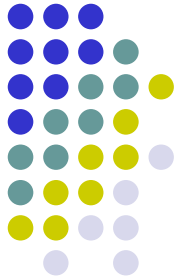
- L'esecuzione di un programma può comportare l'alternarsi di processi utente e di sistema all'interno della CPU
- Processo utente deriva da un programma applicativo
- Processo di sistema deriva da un programma del sistema operativo
  - Processi del nucleo
  - Gestori interruzioni

Gestione dei Processi



- I processi possono essere classificati come:
  - in esecuzione
  - pronti
  - in attesa

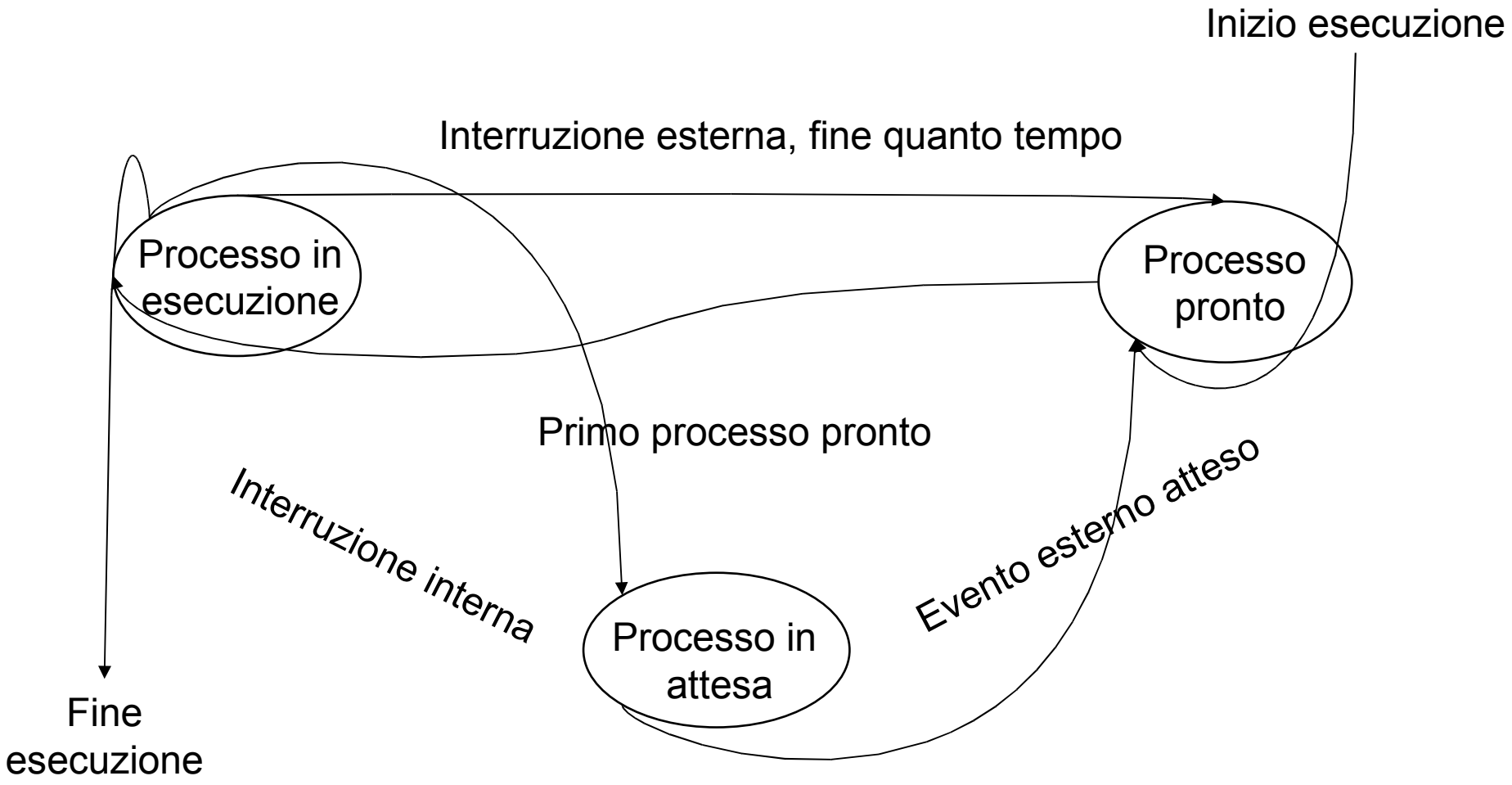
- Supponiamo che un programma in esecuzione da parte di un dato utente sia associato a un processo e che il sistema sia dotato di un unico processore



- Uno solo dei processi può essere in esecuzione in un certo istante (cioè essere eseguito dal processore)

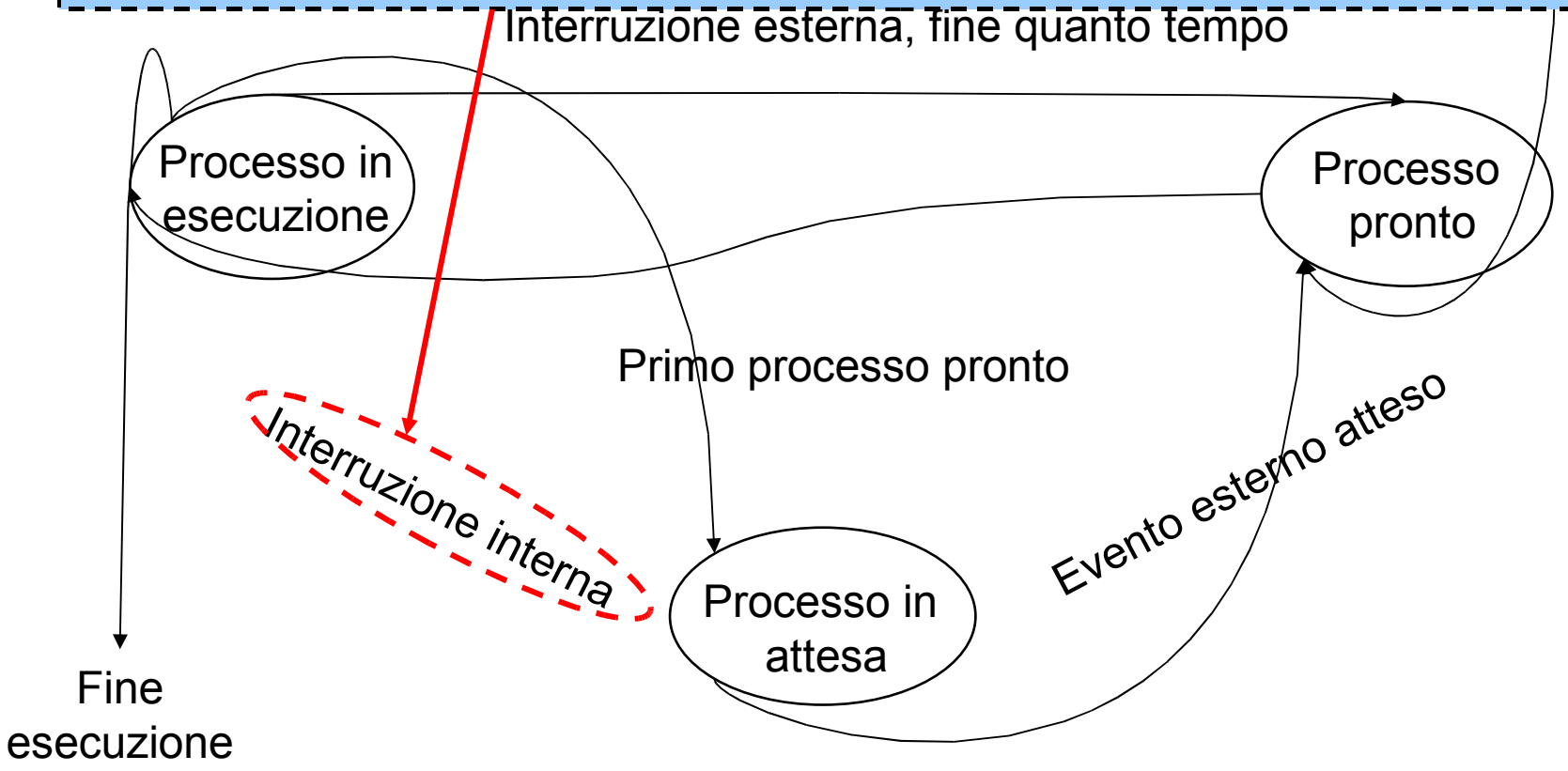
- Gli altri processi sono pronti oppure in attesa
  - I primi possono andare in esecuzione immediatamente (se il gestore dei processi lo decide), mentre i secondi attendono il verificarsi di un evento esterno per passare in stato di pronto

- Diagramma dell'evoluzione dei processi:

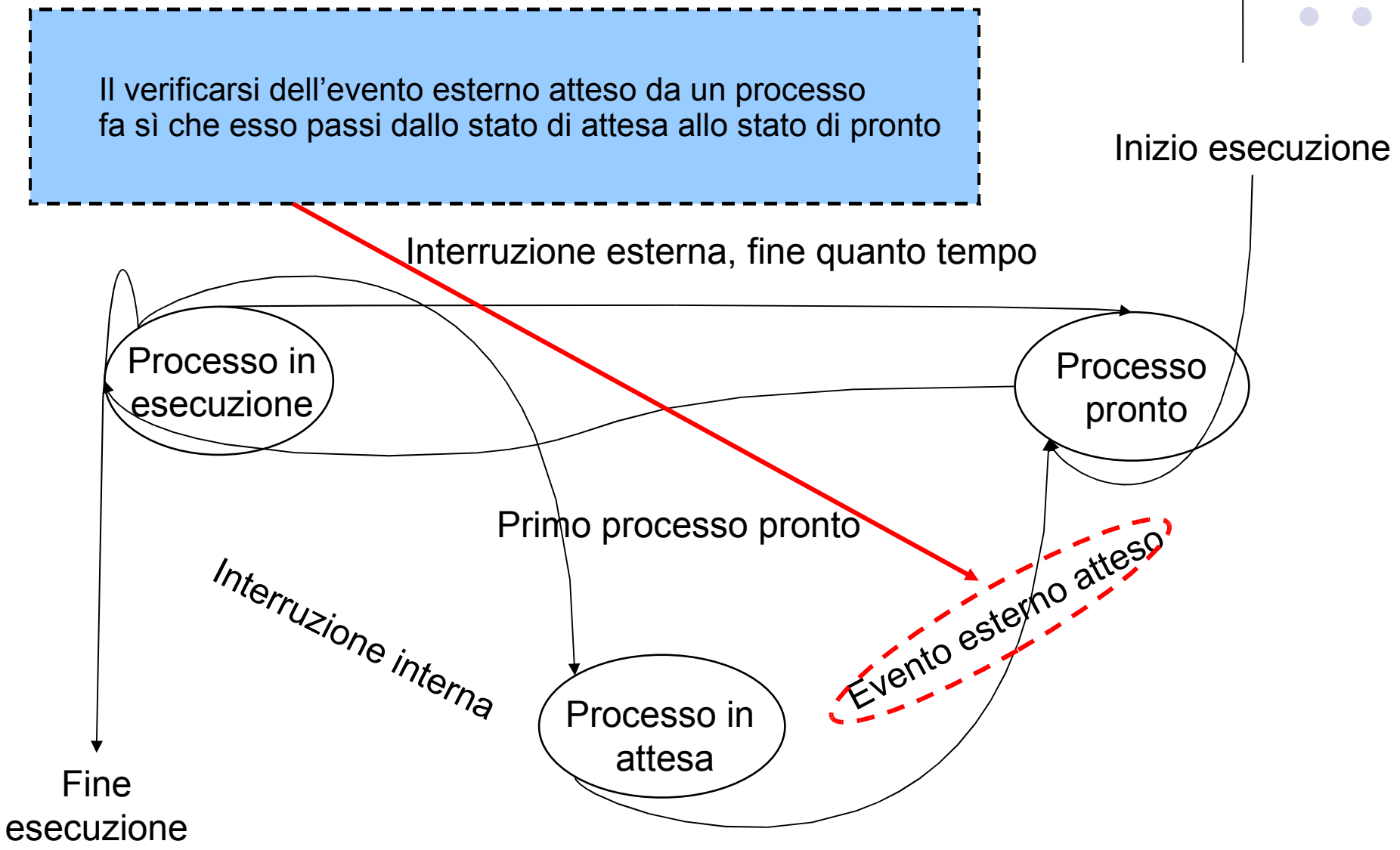


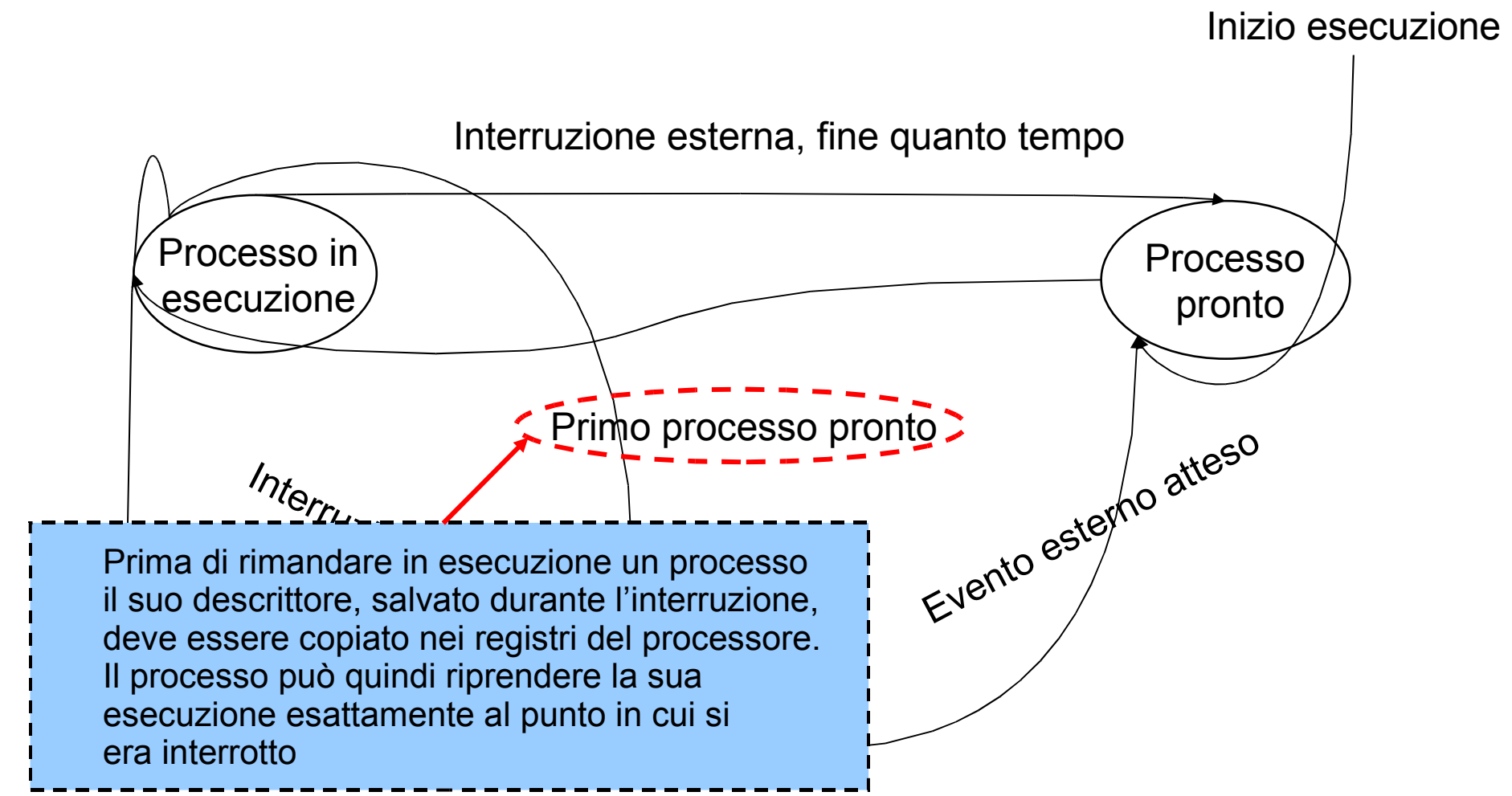
- Diagramma dell'evoluzione dei processi:

Sospensione del processo in esecuzione generata internamente al processo:
- L'esecuzione di un processo ***attivo*** si interrompe ad es. per operazioni di *input/output*
- Lo stato corrente (contenuto registri ecc) del processo interrotto viene salvato in memoria
- Il processo passa allo stato ***in attesa***
- Il controllo passa ad un processo di sistema che assegna la CPU ad un altro processo (per poter ottimizzare l'utilizzo della CPU)

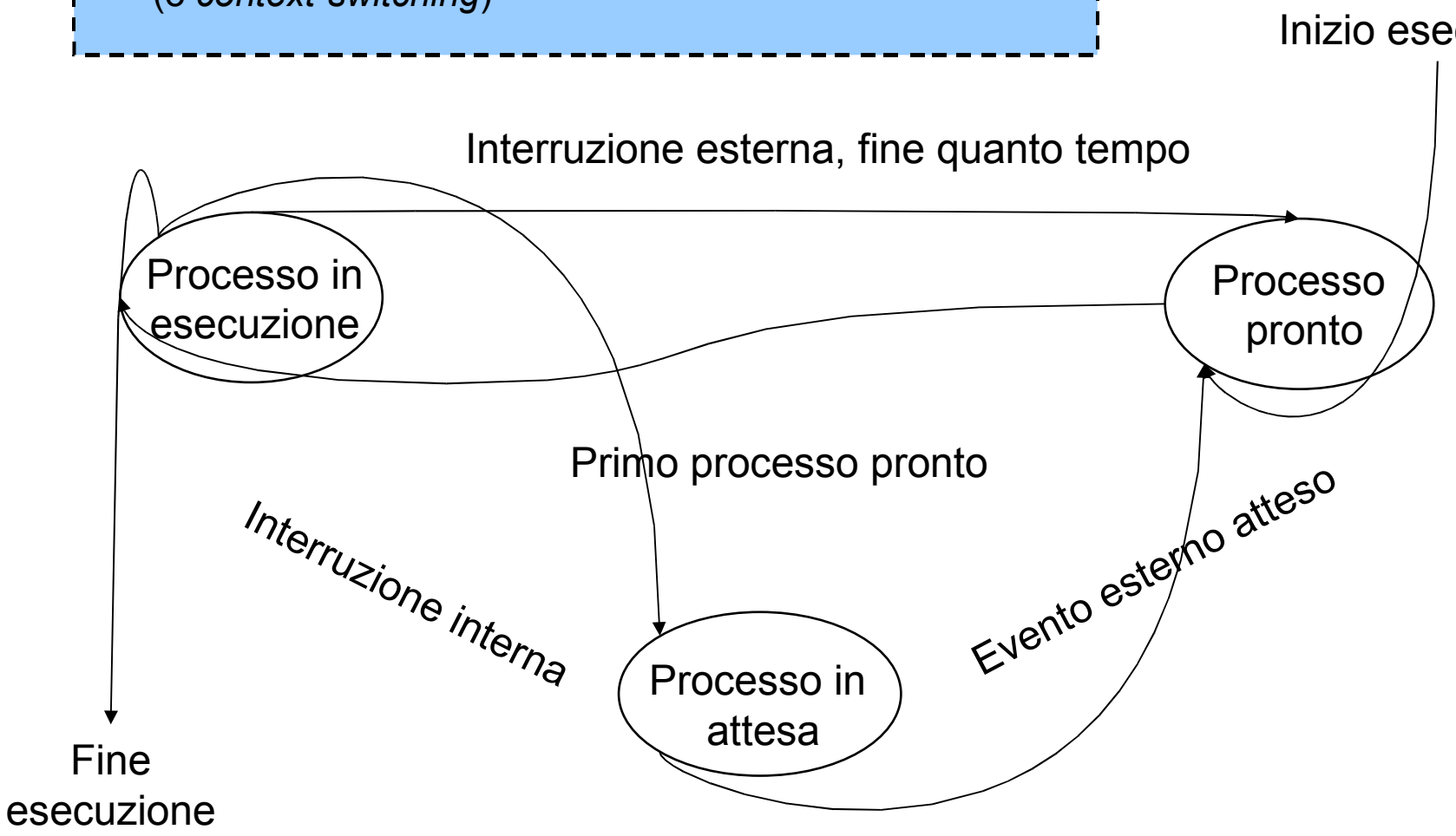


- Diagramma dell'evoluzione dei processi:



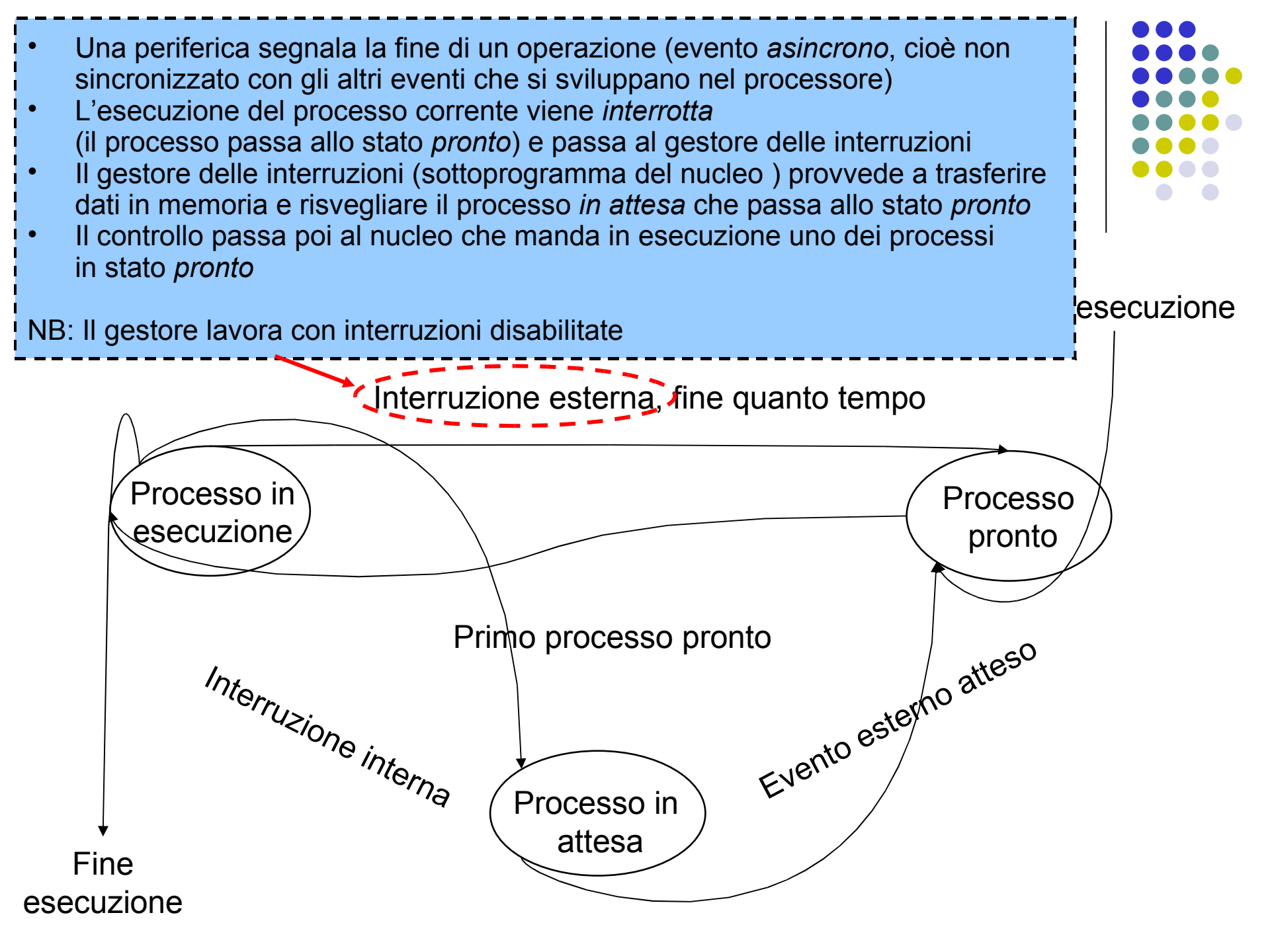

Inizio esecuzione
Interruzione esterna, fine quanto tempo
Processo in esecuzione
Processo pronto
Primo processo pronto
Interru
Evento esterno atteso
Prima di rimandare in esecuzione un processo il suo descrittore, salvato durante l'interruzione, deve essere copiato nei registri del processore. Il processo può quindi riprendere la sua esecuzione esattamente al punto in cui si era interrotto

L'operazione complessiva di sospendere un processo, salvarne lo stato, scegliere un altro processo pronto, ripristinarne lo stato e renderlo attivo prende il nome di *cambiamento di contesto* (o *context-switching*)

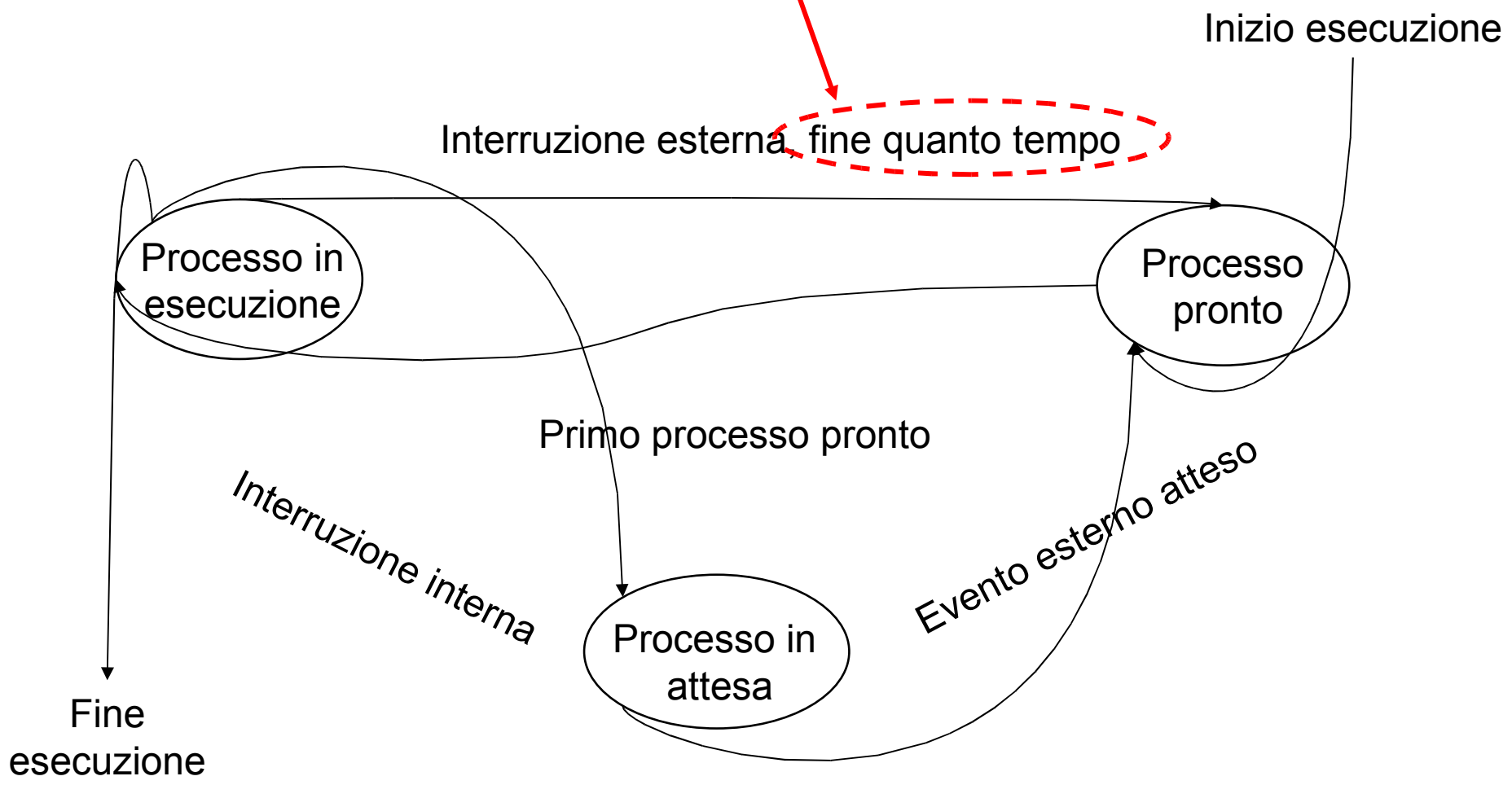


- Una periferica segnala la fine di un operazione (evento *asincrono*, cioè non sincronizzato con gli altri eventi che si sviluppano nel processore)
- L'esecuzione del processo corrente viene *interrotta* (il processo passa allo stato *pronto*) e passa al gestore delle interruzioni
- Il gestore delle interruzioni (sottoprogramma del nucleo) provvede a trasferire dati in memoria e risvegliare il processo *in attesa* che passa allo stato *pronto*
- Il controllo passa poi al nucleo che manda in esecuzione uno dei processi in stato *pronto*

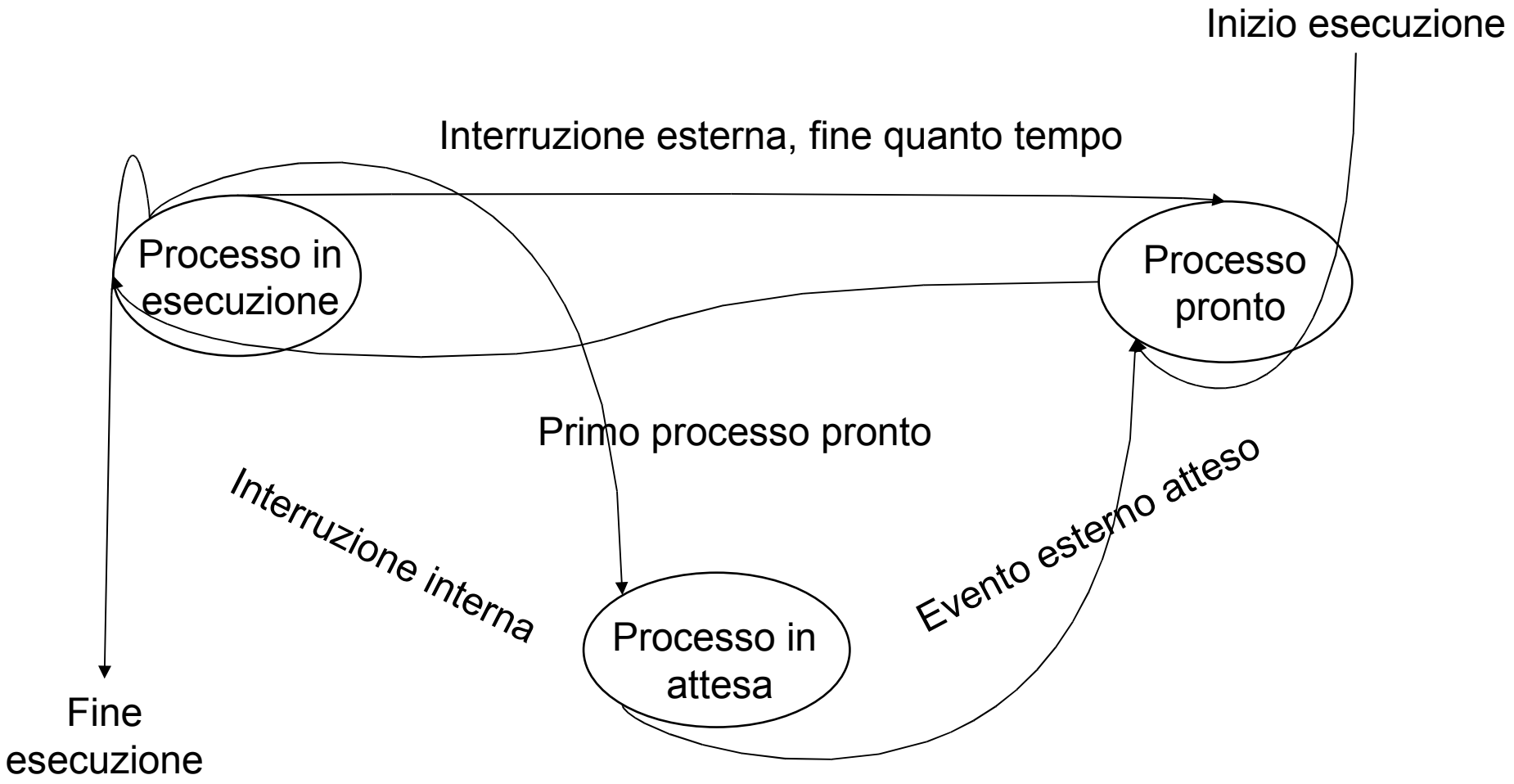
NB: Il gestore lavora con interruzioni disabilitate



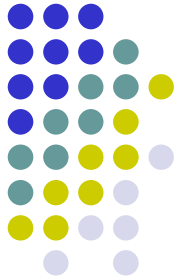
Un processo può anche essere sospeso dal nucleo dopo un certo intervallo temporale in modo da garantire a tutti i programmi un uso paritario della CPU



Un processo può anche essere interrotto e terminato forzatamente dal nucleo dopo il verificarsi di un errore (tentativo di utilizzare celle di memoria centrale al di fuori dello spazio allocato al programma di processo, etc)

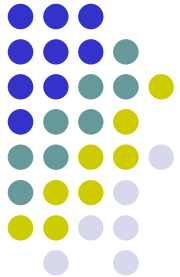


Scheduling dei processi (1/2)



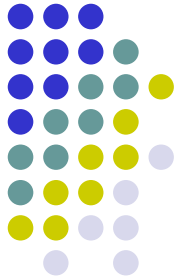
- Il sistema operativo può interrompere i processi per assicurare una politica fair di esecuzione
- Lo scheduler è quella parte del sistema operativo che seleziona il processo da mandare in esecuzione
- La politica di scheduling più semplice consiste nel garantire la rotazione dei processi (round robin)

Scheduling dei processi (2/2)



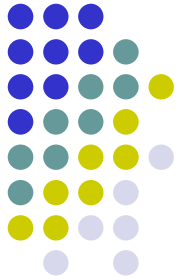
- La politica round robin è realizzata mediante una coda di processi pronti
- Ogni processo ha un quanto di tempo di esecuzione dopo il quale torna in attesa
  - Quanto >> Tempo per salvare/ripristinare stato (*context switching*)
  - Quanto << Tempo di esecuzione del programma (*per assicurare fairness*)

Interazione tra processi



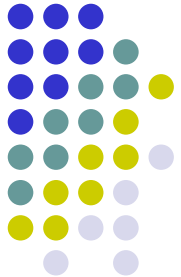
- L'esecuzione di un processo in generale non è indipendente da quella degli altri processi
- Ciò si verifica ad esempio in caso di
  - Competizione (interazione indesiderata e imprevista):
    - Principalmente per l'utilizzo di una o più risorse
    - In questo caso è necessario un utilizzo in mutua esclusione
  - Cooperazione (interazione desiderata e prevista):
    - Si verifica principalmente quando due processi P1 e P2 si ripartiscono lo svolgimento di un compito, ad esempio quando un processo P1 genera dati e un altro processo P2 deve (attenderli per) elaborarli
    - In questo caso di parla di rapporto di tipo produttore-consumatore
  - …
- In tutti questi casi i processi, sebbene logicamente corretti, possono dar luogo ad errori di esecuzione
- Sono quindi necessari meccanismi di sincronizzazione esplicita tra i processi

Gestione Memoria Centrale (1/3)



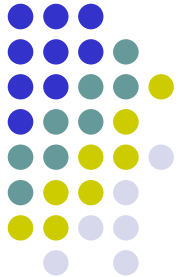
- I processi si alternano durante l'esecuzione nella CPU
- I programmi che vengono eseguiti dai processi pronti devono, almeno in parte, risiedere in memoria centrale
- Per ragioni di efficienza dobbiamo mantenere più programmi in memoria centrale
- Ciò comporta il partizionamento della memoria centrale e del suo spazio di indirizzi
- Possibili tecniche:
  - segmentazione
  - paginazione

Gestione Memoria Centrale (2/3)



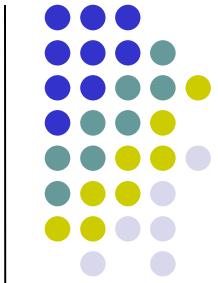
- Segmentazione
  - Durante la compilazione, un programma può essere frazionato in parti che svolgono differenti funzioni (es. si possono separare i dati dalle istruzioni)
  - La memoria centrale viene suddivisa in segmenti di lunghezza variabile contenenti i programmi
- Paginazione
  - La memoria centrale e i programmi vengono suddivisi in pagine di lunghezza fissa
  - I programmi vengono caricati in memoria anche in pagine non contigue

Gestione Memoria Centrale (3/3)



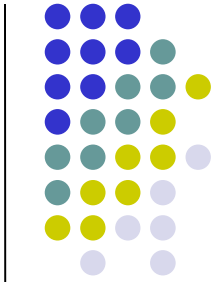
- In entrambi i casi il gestore della memoria offre al programma applicativo la visione di una memoria virtuale che può essere maggiore di quella fisica
- Le pagine o i segmenti che non sono al momento caricate in memoria rimangono disponibili nella memoria di massa
- Un sistema informatico può avere una memoria virtuale maggiore di quella fisica consentendo l'esecuzione di quei programmi la cui dimensione è maggiore di quella della memoria fisica
- Con la suddivisone del programma in pagine non è necessario mantenere tutto il programma in memoria centrale
  - Se la pagina del programma che contiene la prossima esecuzione da eseguire non è in memoria si carica da disco (page swapping)

File System (1/5)



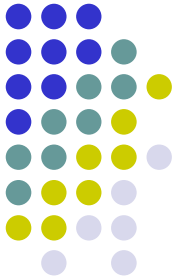
- Virtualizza la memoria di massa
- Permette di accedere agli archivi su disco (file) facendo riferimento ad un nome logico e mascherando i dettagli di memorizzazione
- Garantisce la riservatezza, l'integrità e la possibilità di condivisione di file permettendo di specificare i diritti di accesso
- Tali diritti negano o rendono possibile la lettura, scrittura ed esecuzione di file da parte dell'utente proprietario e degli altri utenti, anche al di fuori dell'elaboratore perché collegati in rete

File System (2/5)



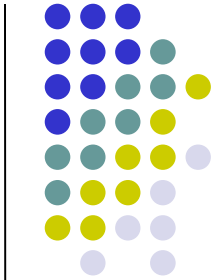
- Suddivide i file in
  1. direttori o cartelle: contenitori di file
  2. file standard: contenitori di informazioni
- I file sono caratterizzati da:
  - Nome:
    - Identifica il file spesso con una estensione che indica il tipo di file
    - es. Tesi.doc oppure somma.exe
  - Struttura:
    - Sequenza di byte
    - Sequenza di blocchi (record) di byte
  - Tipo:
    - File di caratteri e binari (eseguibili)
    - Directory
  - Attributi:
    - nome, diritti di accesso,proprietario

File System (3/5)



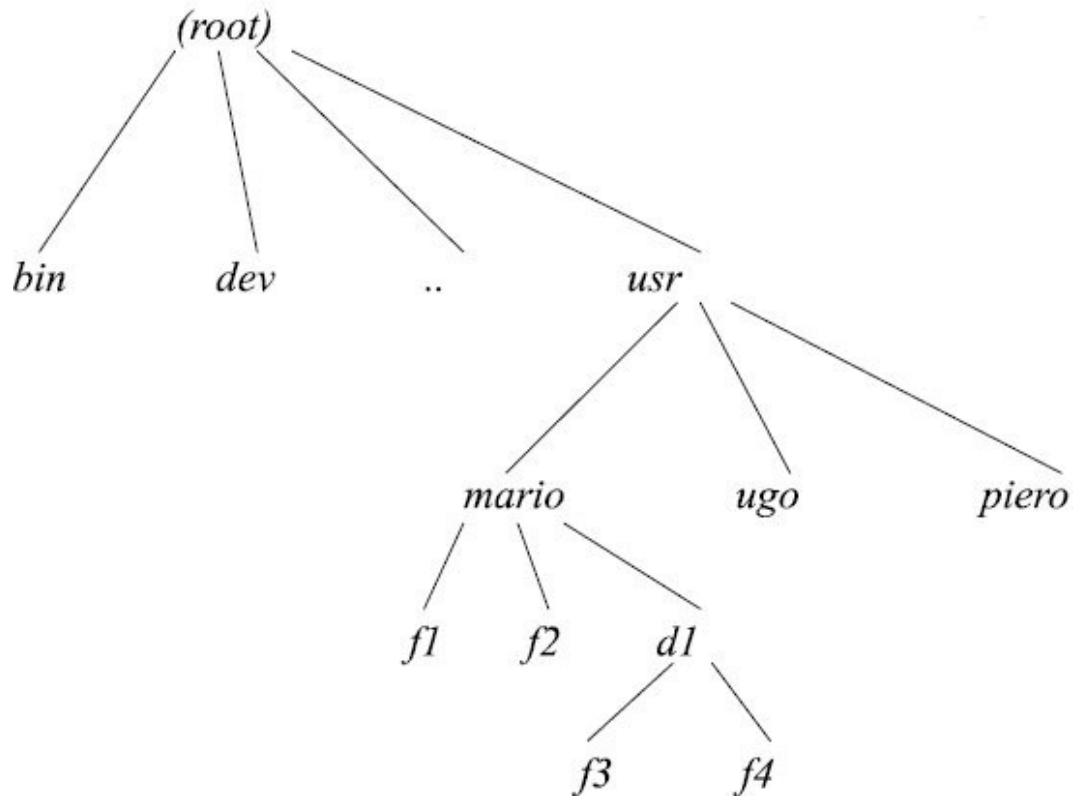
- Consente di effettuare le seguenti operazioni:
  - creare, cancellare, spostare, recuperare, modificare documenti in memoria di massa (memoria persistente)
  - Modificare gli attributi di un file
  - Ridenominare i file

File System (4/5)



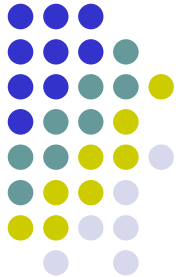
- A partire dal direttorio principale, anche detto radice, il contenimento di file e direttori in altri direttori genera una struttura ad albero
- Ogni file è identificato tramite un pathname, ossia dal suo nome preceduto dalla lista dei direttori che è necessario attraversare per accedere ad esso
- I pathname possono essere di due tipi:
  1. assoluti, ossia a partire dal direttorio radice
  2. relativi, ossia a partire dal direttorio corrente
- Solitamente ad ogni utente viene assegnato un direttorio personale (cartella documenti di Windows o direttorio home in Unix e Linux) nel quale memorizzare i suoi dati

File System (5/5)



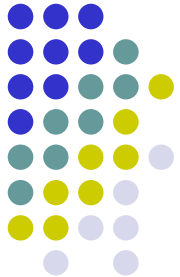
Organizzazione dei file nelle directory

Gestione delle Periferiche (1/4)



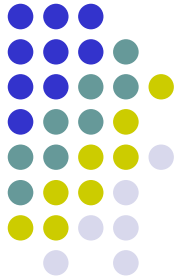
- I driver sono entità software cui è affidato il compito di comunicare dati da e verso le periferiche
- Essi garantiscono ai programmi che li usano una visione di alto livello (è possibile leggere o scrivere tramite primitive indipendenti dalla struttura hardware delle periferiche)
- Ad ogni operazione di alto livello corrisponde una serie di operazione di basso livello, gestite dal sistema operativo

Gestione delle Periferiche (2/4)



- I Driver si distinguono in
  - fisici (hardware)
    - per trasferire e manipolare
  - logici (software)
    - parte del sistema operativo che fornisce funzionalità ad alto livello che riguardano le periferiche

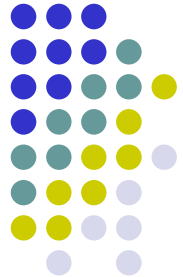
Gestione delle Periferiche (3/4)



Driver fisico

- Il device controller controlla i meccanismi fisici dell'apparecchiatura
  - (es. unità di lettura di floppy disk)
- Il device controller dialoga con la CPU attraverso registri e attraverso una memoria dedicata alle operazioni I/O chiamata DMA (Memoria ad accesso diretto)
- La DMA memorizza informazioni che il device controller può usare per scrivere in memoria direttamente senza passare attraverso la CPU

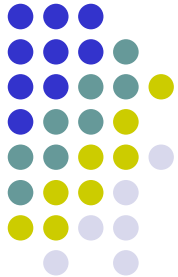
Gestione delle Periferiche (4/4)



Driver logico

- Software che maschera i device fisici
  - Gestisce gli errori in lettura/scrittura
  - Gestisce i nomi del device driver
  - …

Ambiente di Programmazione



- Ogni linguaggio di programmazione è caratterizzato da un *insieme di strumenti che facilitano la scrittura dei programmi e la verifica della loro correttezza*
- Ogni ambiente di programmazione comprende:
  - Editor serve per scrivere programmi sorgente
  - Compilatore traduce un programma sorgente in un linguaggio direttamente eseguibile dall'elaboratore
  - Interprete esegue direttamente il codice sorgente senza tradurlo in linguaggio macchina
  - Linker permette di "collegare" insieme vari programmi (o moduli prodotti usando il compilatore) in un unico programma eseguibile
  - Debugger permette di controllare il programma durante la sua esecuzione

Il Processo di Programmazione