Sommario

- Concetto di processo
- Stati di un processo
- Operazioni e relazioni tra processi
- Concetto di thread
- Gestione dei processi del S.O.

Concetto di processo

Programma e processo

Processo = istanza di programma in esecuzione

- programma = concetto statico
- processo = concetto dinamico

Processo eseguito in modo sequenziale

- Un'istruzione alla volta ma…

… in un sistema multiprogrammato i processi evolvono in modo concorrente

- Risorse (fisiche e logiche) limitate
- Il S.O. stesso consiste di più processi

Immagine in memoria

- Processo consiste di:
  - Istruzioni (sezione Codice o Testo)
    - Parte statica del codice
  - Dati (sezione Dati)
    - Variabili globali
  - Stack
    - Chiamate a procedura e parametri
    - Variabili locali
  - Heap
    - Memoria allocata dinamica
  - Attributi (id, stato, controllo)

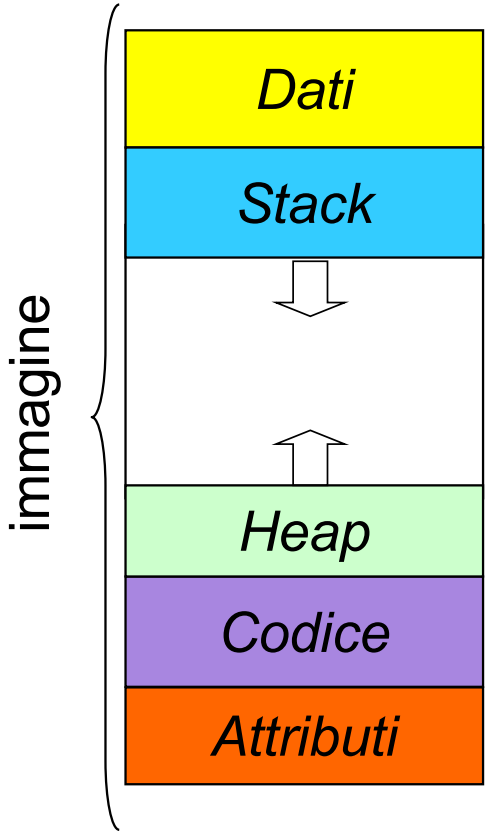
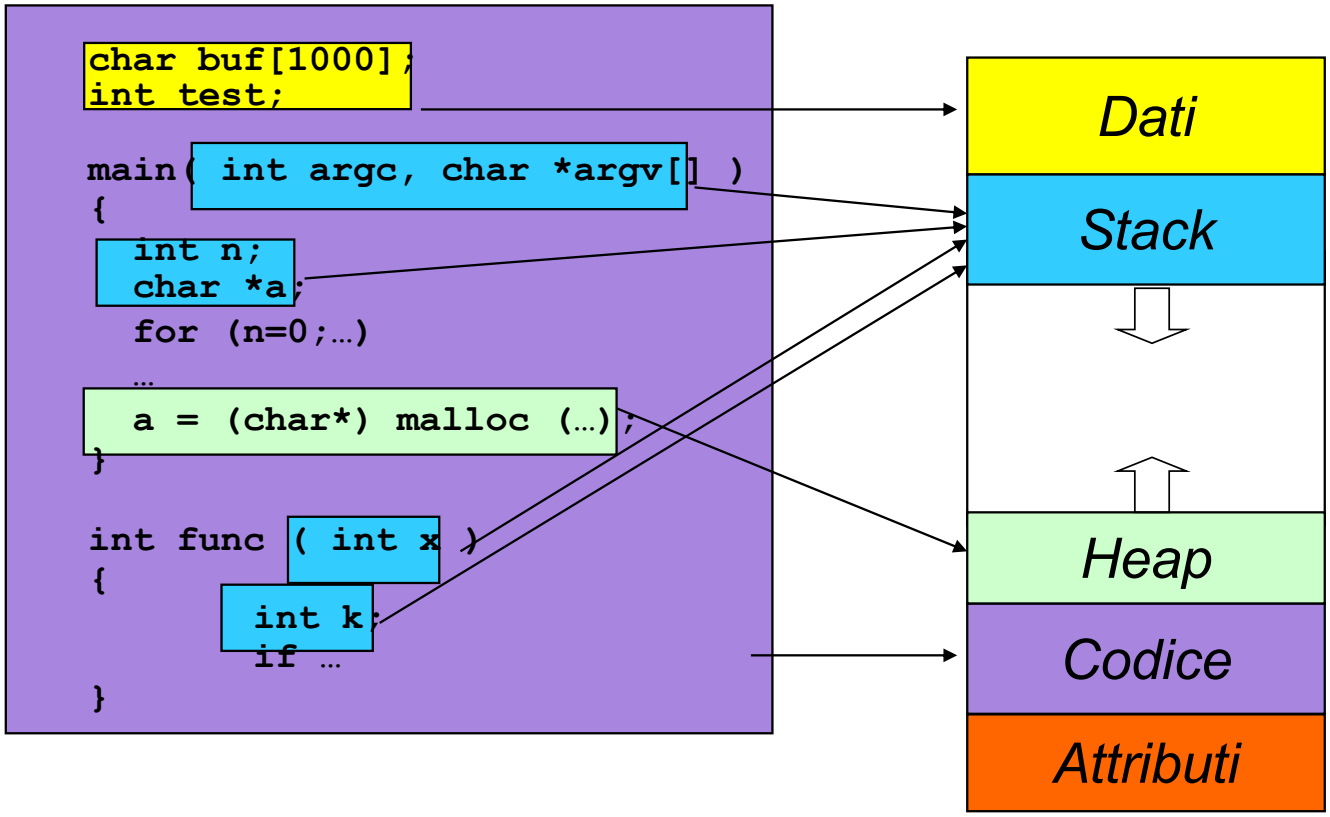
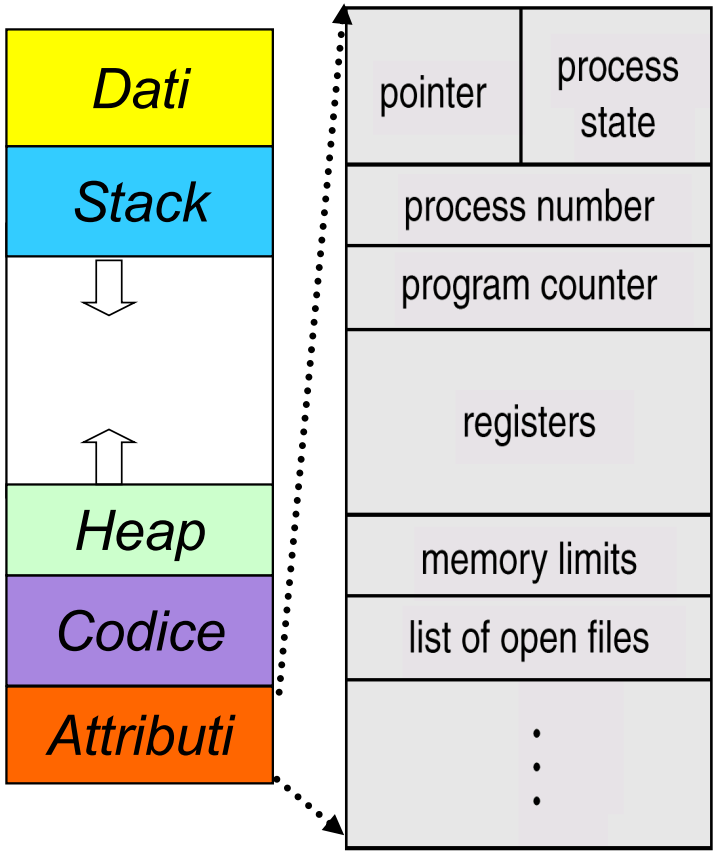


Immagine in memoria



Attributi (Process Control Block)

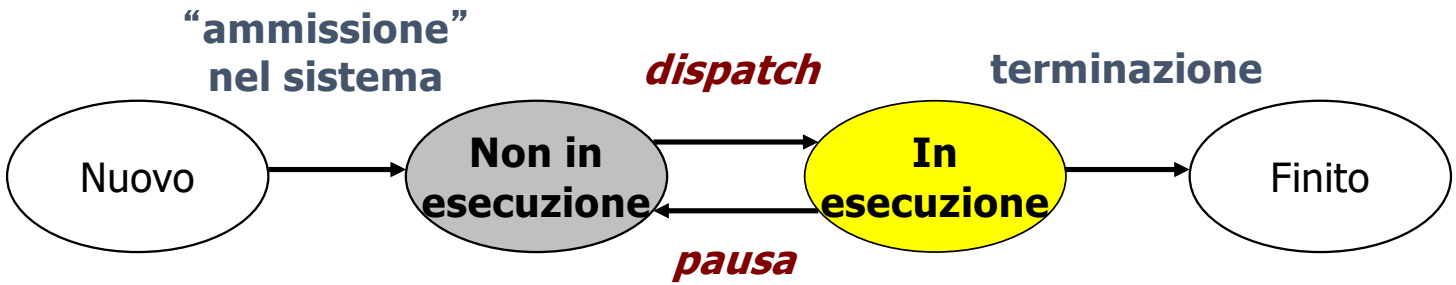
- All'interno del S.O. ogni processo è rappresentato dal process control block (PCB)
  - stato del processo
  - program counter
  - valori dei registri
  - informazioni sulla memoria (es:. registri limite, tabella pagine)
  - informazioni sullo stato dell'I/O (es:. richieste pendenti, file)
  - informazioni sull'utilizzo del sistema (CPU)
  - informazioni di scheduling (es. priorità)



Stati di un processo

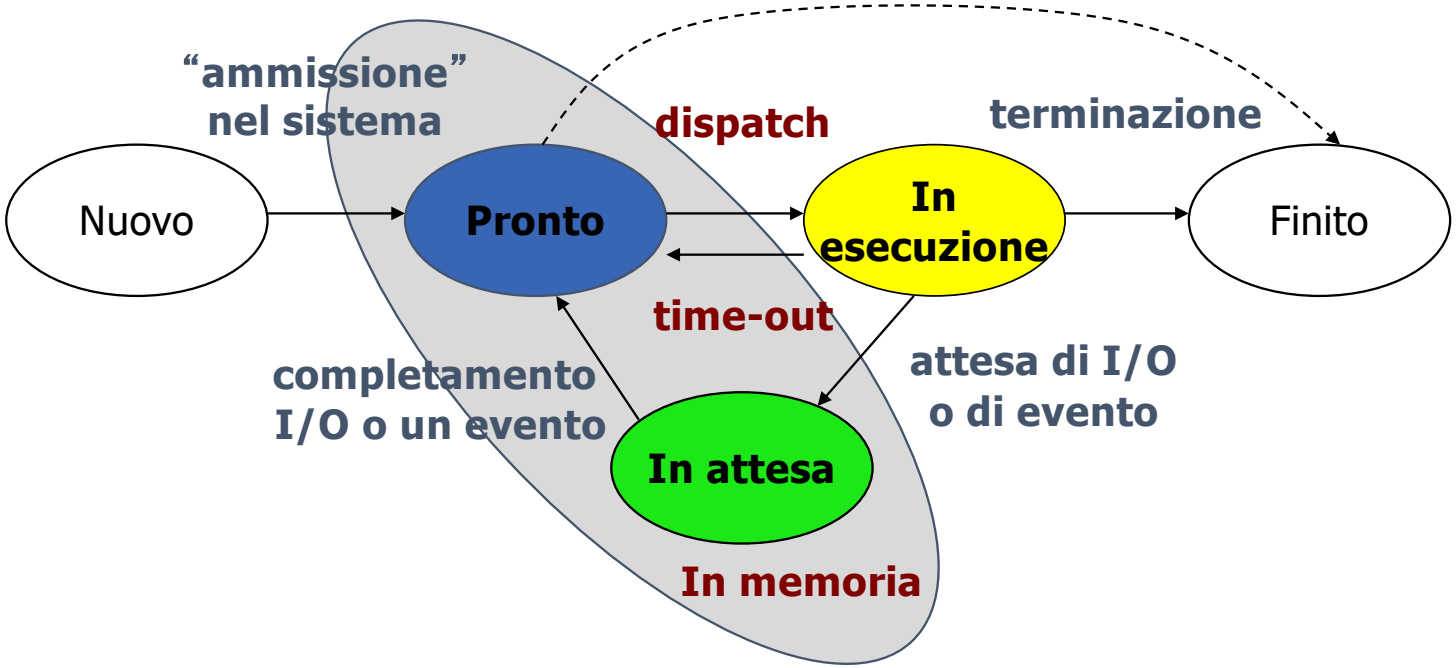
Stati di un processo

- Durante la sua esecuzione, un processo evolve attraverso diversi stati
  - Diagramma degli stati diverso per S.O. diversi
- Lo schema base è il seguente:



Stati di un processo

- Evoluzione di un processo (schema con stato di attesa)



Scheduling

- Selezione del processo da eseguire nella CPU al fine di garantire:
  - Multiprogrammazione
    - Obiettivo:
      - massimizzare uso della CPU → più di un processo in memoria
  - Time-sharing
    - Obiettivo:
      - commutare frequentemente la CPU tra processi in modo che ognuno creda di avere la CPU tutta per sè

Code di scheduling

- Ogni processo è inserito in una serie di code
  - Ready queue
    - Coda dei processi pronti per l'esecuzione
  - Coda di un dispositivo
    - Coda dei processi in attesa che il dispositivo si liberi

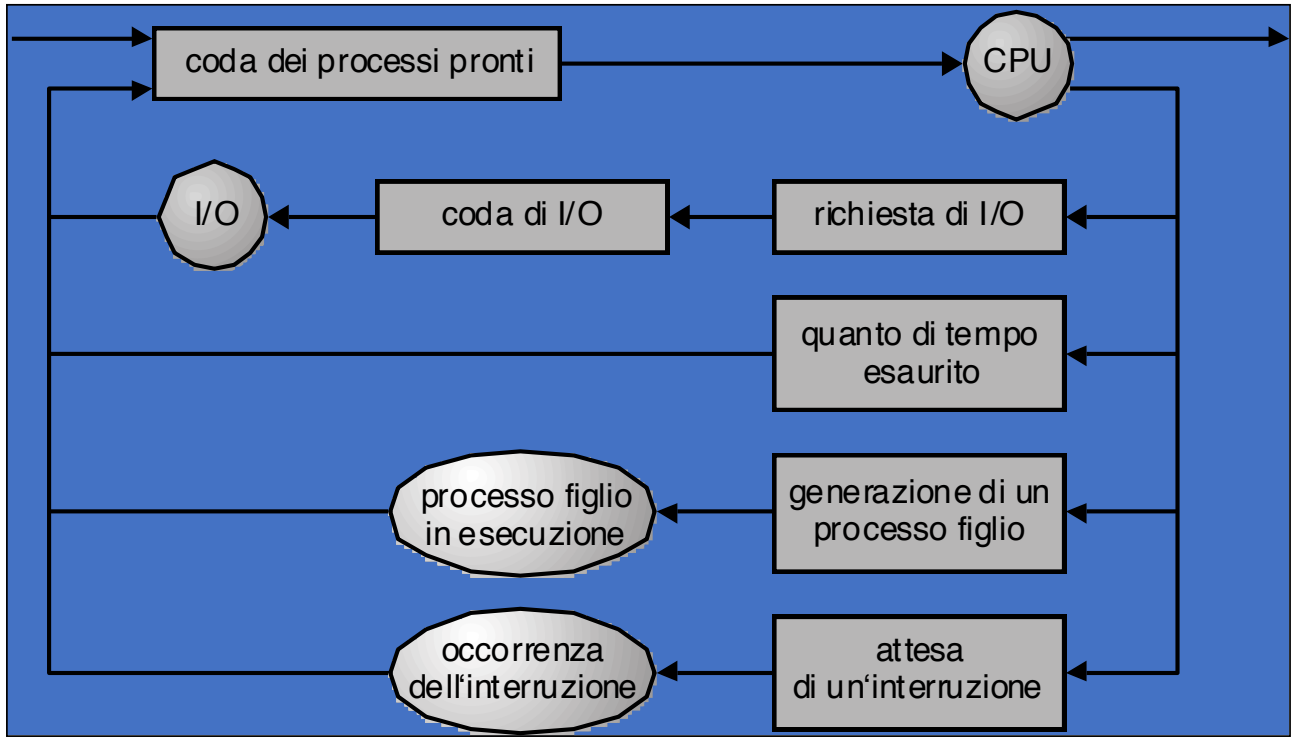
Code di scheduling

All'inizio il processo è nella *ready queue* fino a quando non viene selezionato per essere eseguito (*dispatched*)

Durante l'esecuzione può succedere che:

- Il processo necessita di I/O e viene inserito in una coda di un dispositivo
- Il processo termina il *quanto di tempo,* viene rimosso forzatamente dalla CPU e re-inserito nella ready queue
- Il processo crea un figlio e ne attende la terminazione
- Il processo si mette in attesa di un evento

Diagramma di accodamento



Operazione di dispatch

Cambio di contesto

salvataggio PCB del processo che esce e caricamento del PCB del processo che entra

Passaggio alla modalità utente

(all'inizio della fase di dispatch il sistema si trova in modalità kernel)

Salto all'istruzione da eseguire del processo appena arrivato nella CPU

Cambio di contesto (context switch)

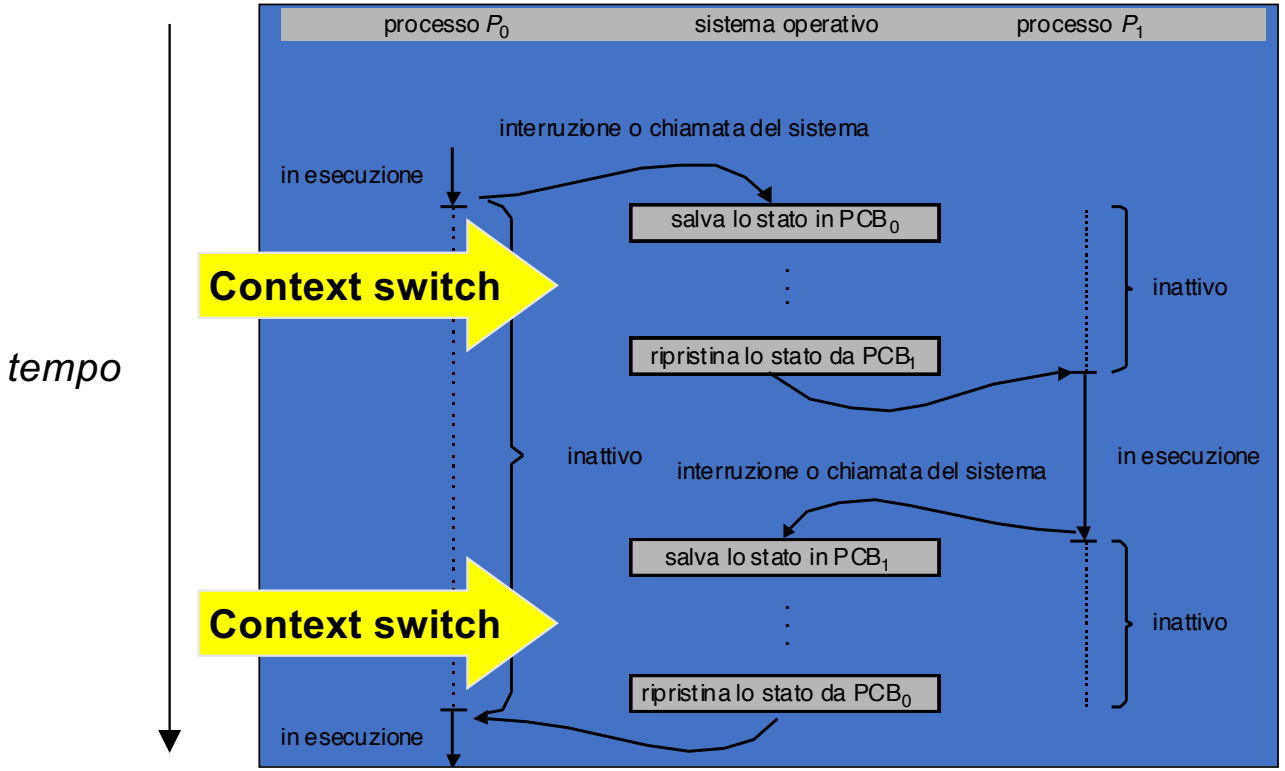
Passaggio della CPU a un nuovo processo

- Registrazione dello stato del processo vecchio
- Caricamento dello stato (precedentemente registrato) del nuovo processo

Il tempo necessario al cambio di contesto è puro sovraccarico

- Il sistema non compie alcun lavoro utile durante la commutazione
- La durata del cambio di contesto dipende molto dall'architettura

Commutazione della CPU



Operazioni sui processi

Creazione di un processo

Un processo può creare un figlio

- Figlio ottiene risorse dal S.O. o dal padre per
- Spartizione
- Condivisione

Tipi di esecuzione

- Sincrona
  - Padre attende la terminazione dei figli
- Asincrona
  - Evoluzione “parallela” di padre e figli

Terminazione di un processo

Processo finisce la sua esecuzione

Processo terminato forzatamente dal padre

- Per eccesso nell'uso delle risorse
- Il compito richiesto al figlio non è più necessario
- Il padre termina e il S.O. non permette ai figli di sopravvivere al padre

Processo terminato forzatamente dal S.O.

- Utente chiude applicazione
- Errori (aritmetici, di protezione, di memoria, …)

Relazioni tra processi

Processi indipendenti

- Esecuzione deterministica (dipende solo dal proprio input) e riproducibile
- Non influenza, né viene influenzato da altri processi
- Nessuna condivisione dei dati con altri processi

Processi cooperanti

- Influenza e può essere influenzato da altri processi
- Esecuzione non deterministica e non riproducibile

Perché processi cooperanti?

Condivisione informazioni

Accelerazione del calcolo

- Esecuzione parallela di "*subtask*" su multiprocessore

Modularità

- Funzioni distinte su vari processi

Convenienza