

Processi

Capitolo 3 -- Silberschatz

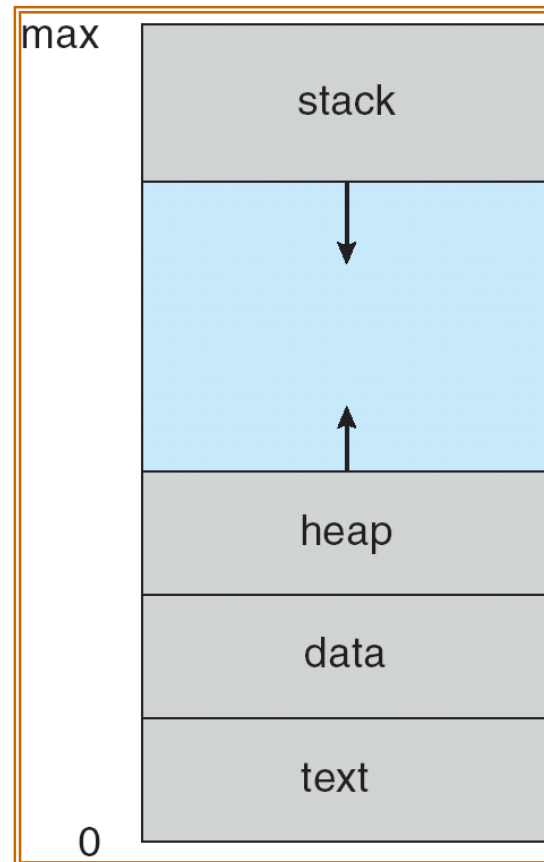
Concetto di processo

- Un **programma** può corrispondere a **diversi processi**
 - Si pensi a un insieme di utenti che utilizzano uno stesso editor in relazione a file diversi
- Quindi, un programma descrive non un processo, ma un insieme di processi □ *istanze del programma* □ ognuno dei quali è relativo all'esecuzione del programma da parte dell'elaboratore per un particolare insieme di dati in ingresso
- Un processo può rappresentare un ambiente di esecuzione di altri processi
 - Java Virtual Machine e programmi scritti in Java

Concetto di processo

- Un SO esegue una varietà di attività:
 - Sistemi batch – job
 - Sistemi time-sharing – programmi utenti o task
- Processo – un programma in esecuzione; l' esecuzione di un processo avviene in modo sequenziale
- Un processo include:
 - program counter
 - contenuto registri CPU
 - sezione testo
 - sezione dati
 - heap
 - stack

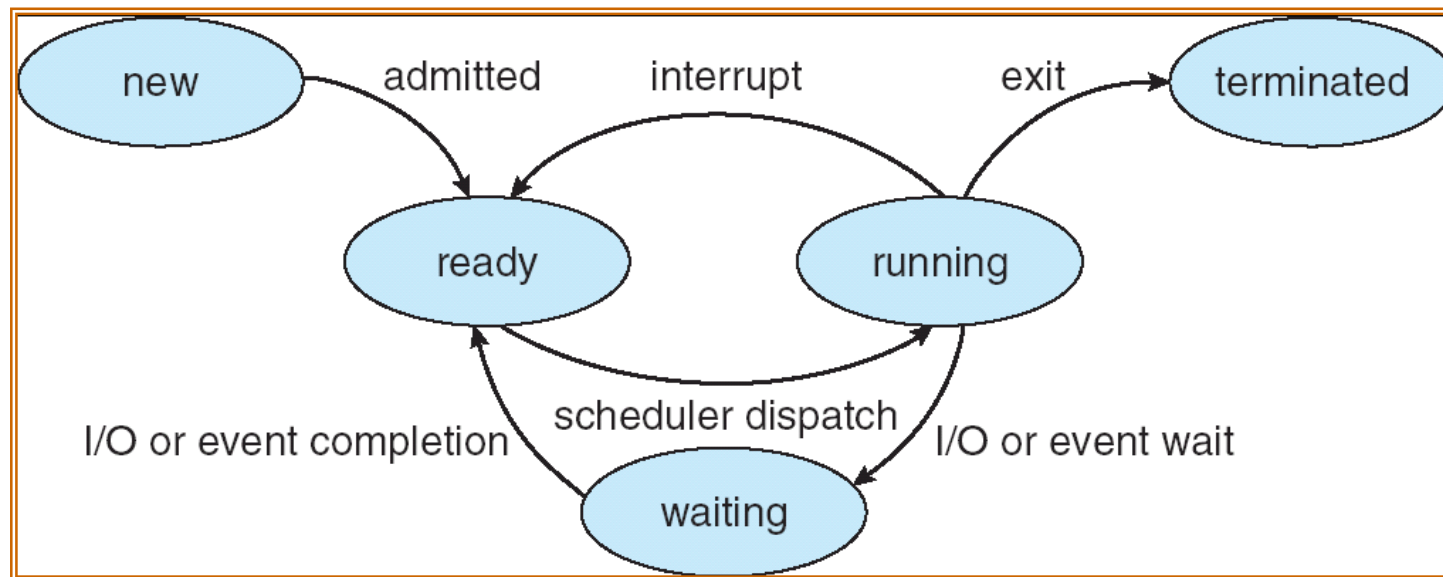
Immagine in memoria di un processo



Stati di un processo

- Durante l'esecuzione un processo cambia stato
 - **new**: Il processo è stato creato
 - **running**: Le sue istruzioni vengono eseguite
 - **waiting**: Il processo è in attesa di qualche evento (es. fine di un'operazione di I/O)
 - **ready**: Il processo è in attesa di essere assegnato ad un processore
 - **terminated**: Il processo ha terminato l'esecuzione

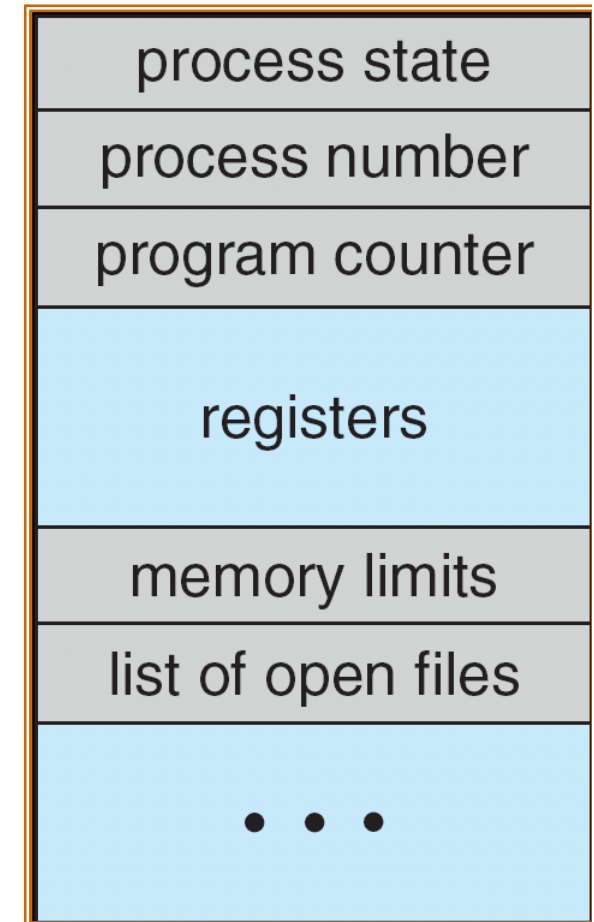
Diagramma di Stato dei Processi



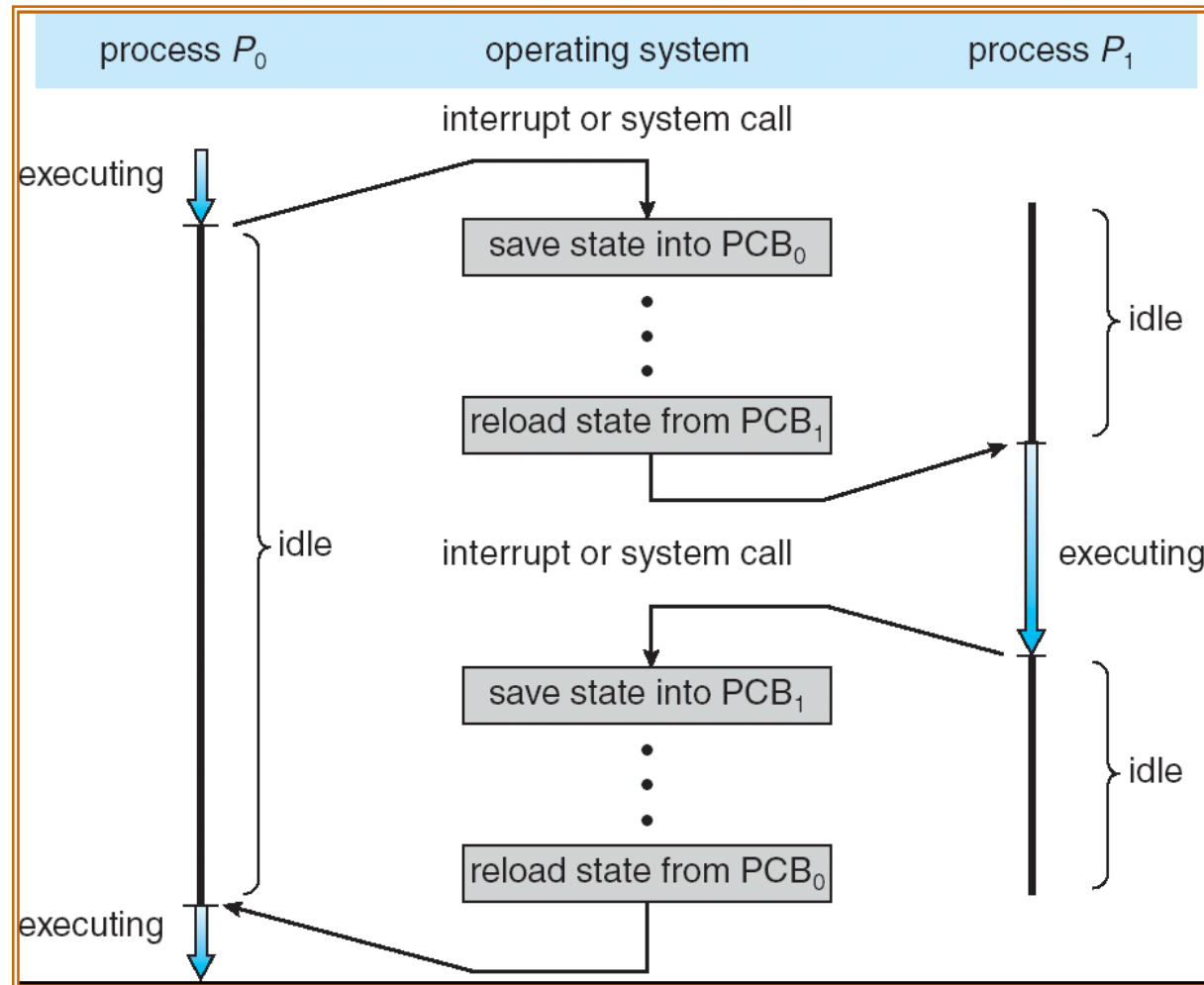
Process Control Block

Contiene informazioni per la gestione del processo

- Stato del processo
- Identificatore
- Program counter
- Registri della CPU (accumulatori, registri indice, ...)
- Informazioni per lo scheduling CPU (priorità del processo, puntatori alle code di sched., ...)
- Informazioni per la gestione della memoria (tabelle delle pagine)
- Informazioni di contabilizzazione (tempo di utilizzo della CPU)
- Informazioni sullo stato dell' I/O (lista dei dispositivi di I/O, ...)



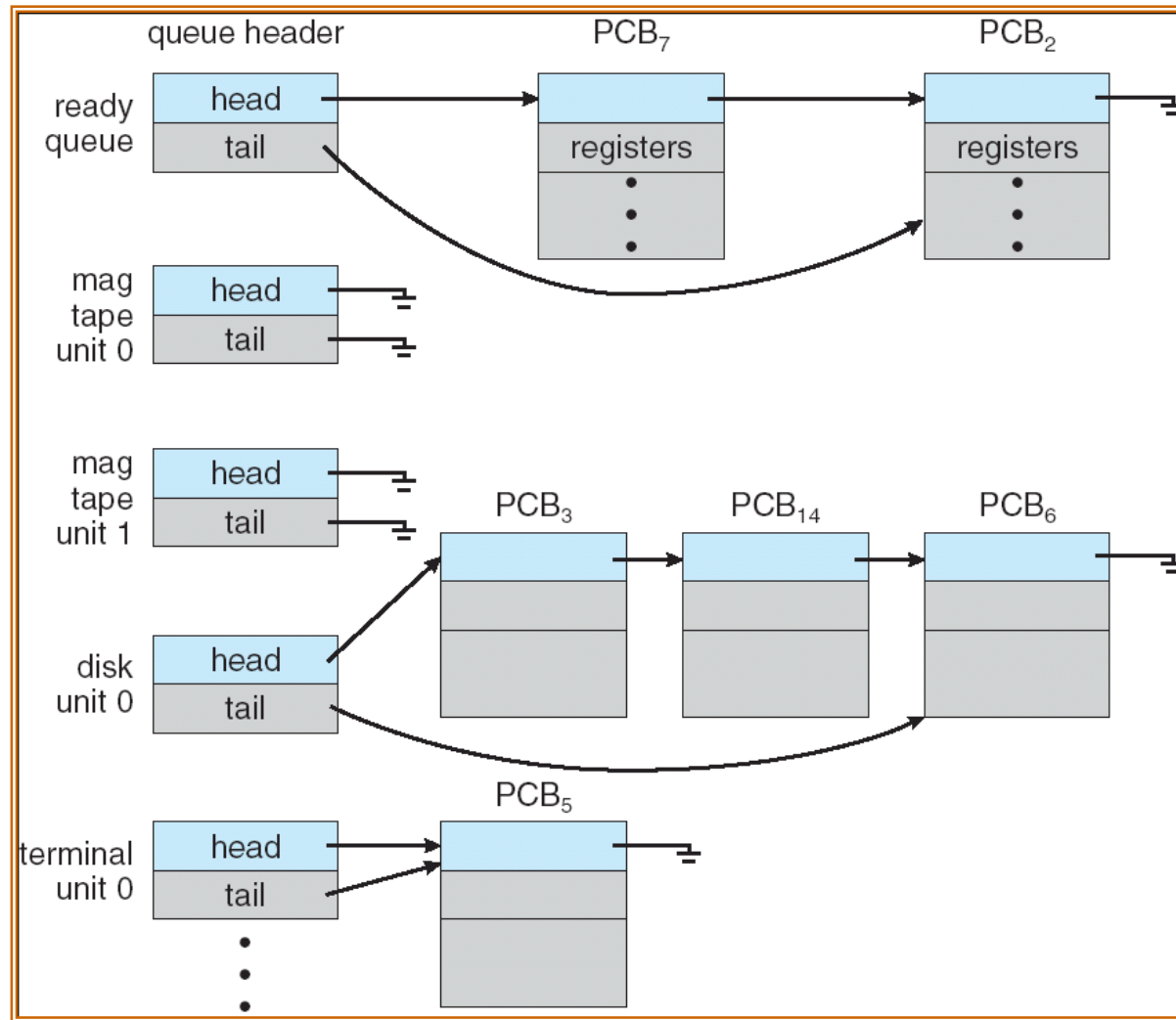
La CPU commuta da Processo a Processo



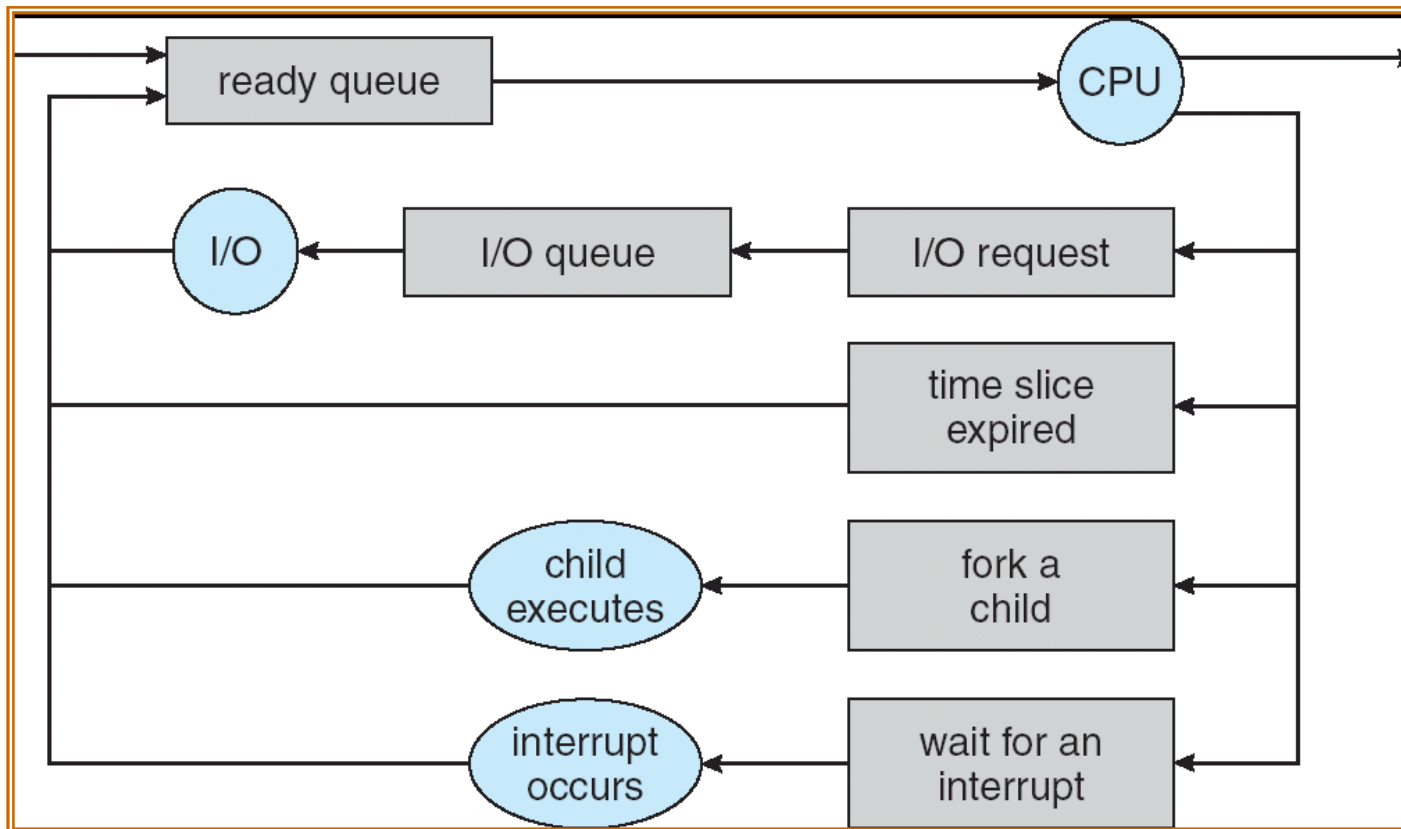
Code di scheduling per i processi

- **Job queue** – insieme di tutti i processi nel sistema
- **Ready queue** – insieme di tutti i processi in memoria centrale pronti per l'esecuzione
- **Code dei dispositivi** – insieme dei processi in attesa di qualche dispositivo di I/O
- I processi, durante la loro vita, migrano tra varie code

Ready Queue e varie code di I/O



Ciclo di vita di un processo



Schedulatori

Il sistema si serve di uno schedulatore per selezionare un processo

- **Schedulatore a lungo termine** (o job scheduler) – seleziona i processi che devono essere caricati in memoria centrale (ready queue)
- **Schedulatore a breve termine** (o CPU scheduler) – seleziona il prossimo processo che la CPU dovrebbe eseguire
 - In un sistema batch si sottopongono più processi di quanti se ne possano servire perciò parte di essi vengono trasferiti in memoria secondaria e poi prelevati dal job scheduler
 - Sistemi time shering, come Unix e windows, sono privi di scheduler a lungo termine e si limitano a caricare in memoria tutti i nuovi processi ed a gestirli con lo scheduler a breve termine. E' l'utente che, se vede che le prestazioni della macchina diminuiscono decide di chiudere alcune applicazioni

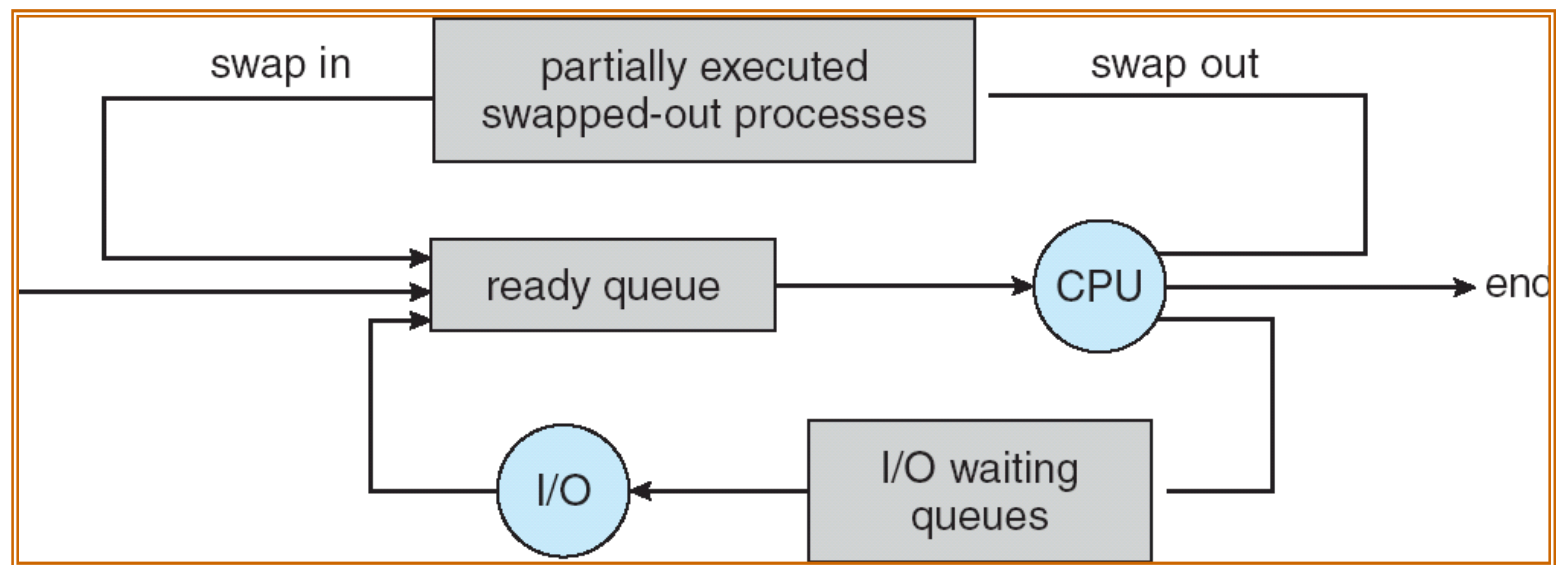
Schedulatori

- Lo schedulatore a breve termine viene invocato molto *frequentemente* (millisecondi) deve essere veloce
- Lo schedulatore a lungo termine viene invocato *raramente* (secondi, minuti) può essere più lento
- Lo schedulatore a lungo termine controlla il *grado di multiprogrammazione* ed interviene quando un processo termina

Schedulatori

- I processi possono essere descritti come:
 - **Processi I/O-bound** – consumano più tempo facendo I/O che computazione, contengono molti e brevi CPU burst
 - **Processi CPU-bound** – consumano più tempo facendo computazione; contengono pochi e lunghi CPU burst
- Compito dello scheduler a lungo termine è quello di scegliere una giusta combinazione tra i processi I/O bound e quelli CPU-bound che sono presenti in mem contrale.

Schedulatore a medio termine



Cambio di contesto (Context switching)

- Quando la CPU viene allocata ad un altro processo, il SO deve salvare lo stato del processo in esecuzione e caricare lo stato del nuovo processo
- Il tempo per un context switch è tempo sprecato (overhead); il sistema non fa nulla di utile mentre commuta
- Il tempo dipende dal supporto hardware
- Il context switch viene realizzato dal **dispatcher**

Creazione di processi

- Il sistema operativo fornisce meccanismi per creare e terminare processi
- Un processo padre crea processi figli che, a loro volta, creano altri processi, formando un albero di processi
- Ogni processo ha un identificatore (process identifier) PID

Multitasking per sistemi mobili

iOS

- Alcuni SO per mobile (per es., le prime versioni di **iOS**) prevedevano un solo processo utente in esecuzione (sospendendo tutti gli altri)
- Anche attualmente, per le limitate dimensioni dello schermo, l'interfaccia utente iOS permette l'esecuzione...
 - ...di un unico processo in foreground, controllabile mediante GUI
 - ...più processi in background, in memoria centrale, in esecuzione, ma impossibilitati ad utilizzare il display (quindi con “capacità” limitate)

Multitasking per sistemi mobili

iOS

- Applicazioni eseguibili in background se...
 - ...realizzano un unico task di lunghezza finita (completamento di un download dalla rete)
 - ...ricevono notifiche sul verificarsi di un evento (ricezione di email)
 - ...impongono attività di lunga durata (lettore audio)

Multitasking per sistemi mobili

Android

- **Android** non pone particolari limiti alle applicazioni eseguite in background
- Un'applicazione in elaborazione in background deve utilizzare un **servizio** (un componente applicativo separato) che viene eseguito per conto della app
 - Esempio: app per streaming audio □ se l'app va in background, il servizio continua comunque ad inviare il file audio al driver

Multitasking per sistemi mobili

Android

- Il servizio continua a funzionare anche se l'app in background viene sospesa
- I servizi non hanno un'interfaccia utente e hanno un ingombro di memoria moderato

Tecnica efficace per mobile multitasking

Un albero di processi di Linux

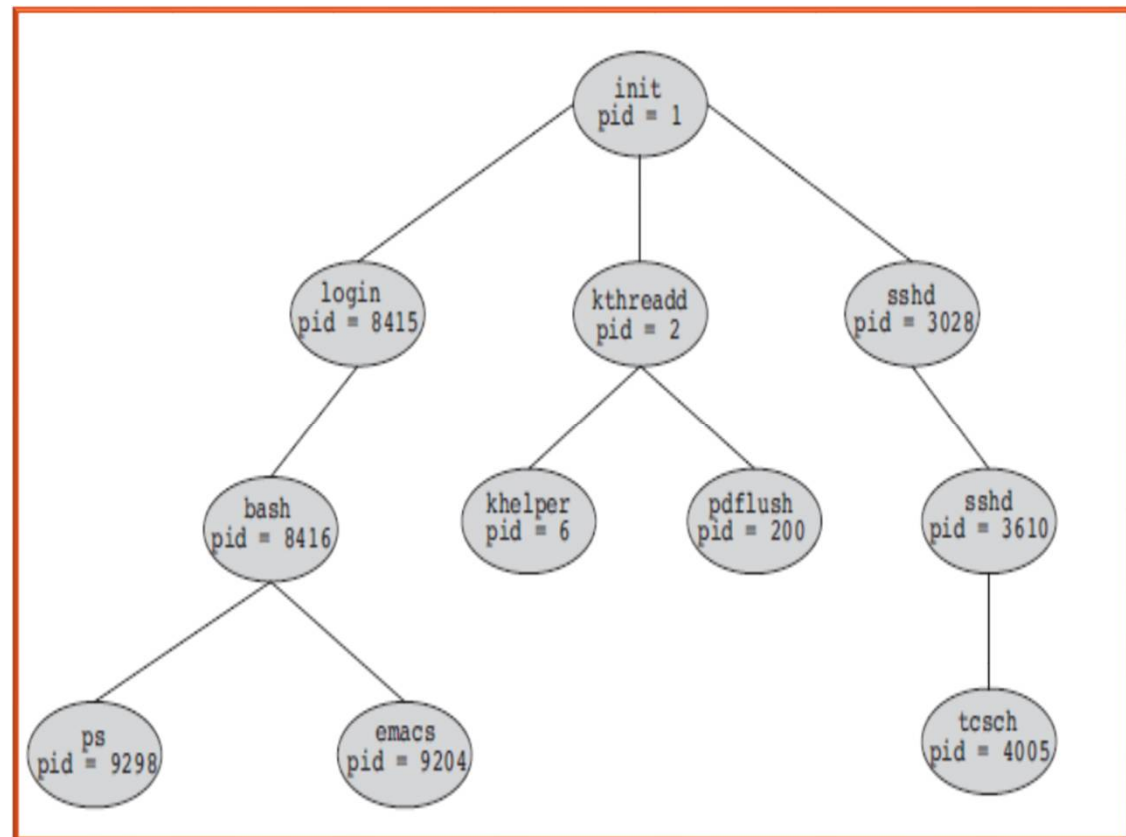
init è il progenitore di tutti i processi utente; dopo l'avvio genera vari processi utente:

-**sshd** = server ssh (gestione dei client che si connettono al sistema attraverso ssh)

-**kthreadd** = crea processi che eseguono attività del kernel

-**login** = gestisce i client che si connettono direttamente al sistema

-nell'esempio: il client utilizza il terminale e quindi **bash**, ed ha avviato il comando **ps** e l'editor **emacs**



Creazione di processi: implementazione

- Condivisione di risorse
 - Padre e figli condividono tutte le risorse
 - figli condividono un sottoinsieme delle risorse del padre
 - Padre e figlio non condividono niente
- Esecuzione
 - Padre e figli vengono eseguiti in concorrenza
 - Padre aspetta fino alla terminazione dei figli

Crazione di processi: implementazione

- Spazio di indirizzamento
 - Lo spazio del figlio è un duplicato di quello del padre
 - Il figlio può caricare in esso un nuovo programma
- Esempi UNIX
 - La chiamata **fork** crea un nuovo processo
 - La chiamata **exec** viene usata dopo una **fork** per sostituire l'immagine in memoria del processo con un nuovo programma
 - La chiamate **wait** permette al padre di aspettare che il figlio termini

Terminazione di un processo

- Un processo esegue l'ultima istruzione e chiede al SO di eliminarlo (**exit**)
 - Trasmette un valore di output al padre (via **wait**)
 - Le risorse del processo sono recuperate dal SO
- Il processo padre può terminare l'esecuzione dei processi figli (e.g., **TerminateProcess()** in Win32)
 - Un figlio ha ecceduto nell'uso delle risorse
 - Il compito assegnato al figlio non è più richiesto
 - Se il padre ha terminato
 - Alcuni SO non permettono ai processi figli di continuare
 - Tutti i figli vengono terminati – *terminazione a cascata*

Multithreading: estensione del concetto di processo

- Alla base del concetto di thread sta la constatazione che la definizione di **processo** è basata su due aspetti:
 - Possesso delle risorse
 - Esecuzione
- I due aspetti sono indipendenti e come tali possono essere gestiti dal SO
 - L'elemento che viene eseguito è il **thread**
 - L'elemento che possiede le risorse è il **processo**
- Il termine multithreading è utilizzato per descrivere la situazione in cui ad un processo sono associati più thread

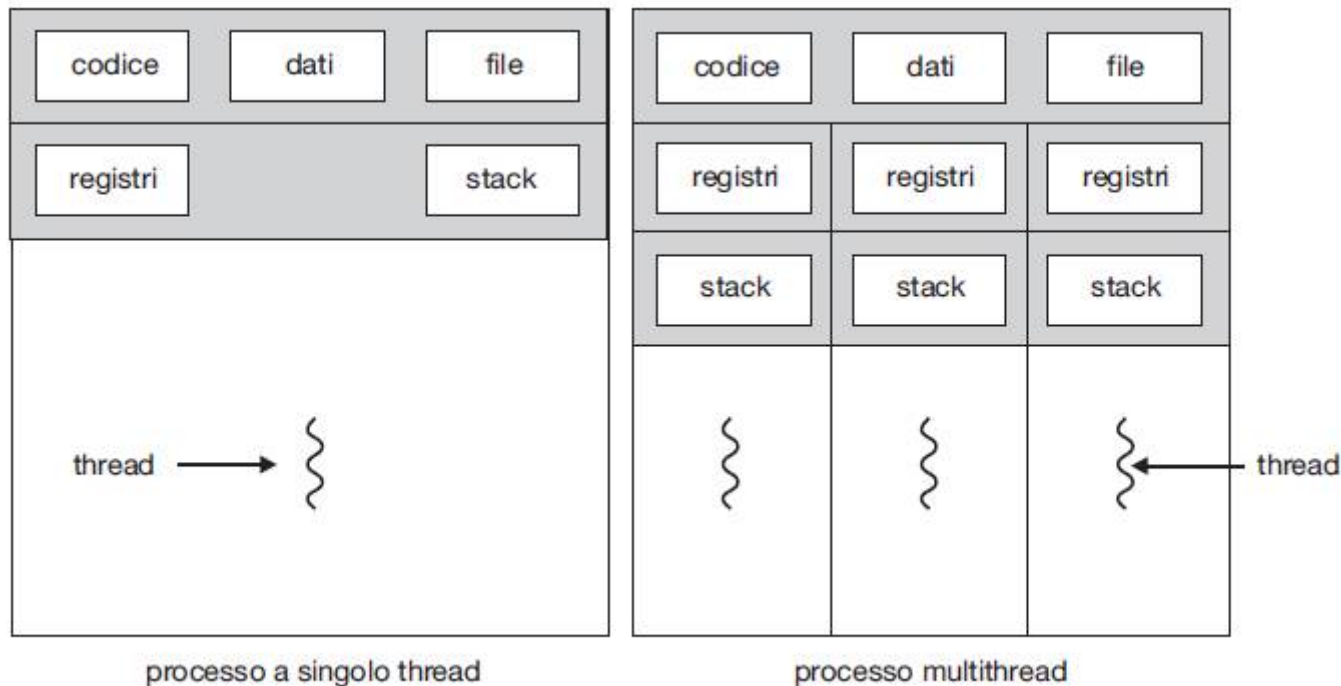
Multithreading: esempi

La maggior parte delle applicazioni attuali sono multithread

- **Web browser** potrebbe avere:
 - un thread per la rappresentazione sullo schermo immagini e testo,
 - un thread per reperire i dati dalla rete.
- **Word processor** potrebbe avere:
 - Possibile avere un thread per ciascun documento aperto (i thread sono tutti uguali ma lavorano su dati diversi)
 - Relativamente ad ognuno è anche possibile avere...
 - Un thread per la rappresentazione dei dati su schermo
 - Un thread per la lettura dei dati immessi da tastiera
 - Un thread per la correzione ortografica e grammaticale
 - Un thread per il salvataggio periodico su disco

Multithreading

- Tutti i thread
 - condividono le risorse del processo,
 - risiedono nello stesso spazio di indirizzamento ed
 - hanno accesso agli stessi dati



Multithreading

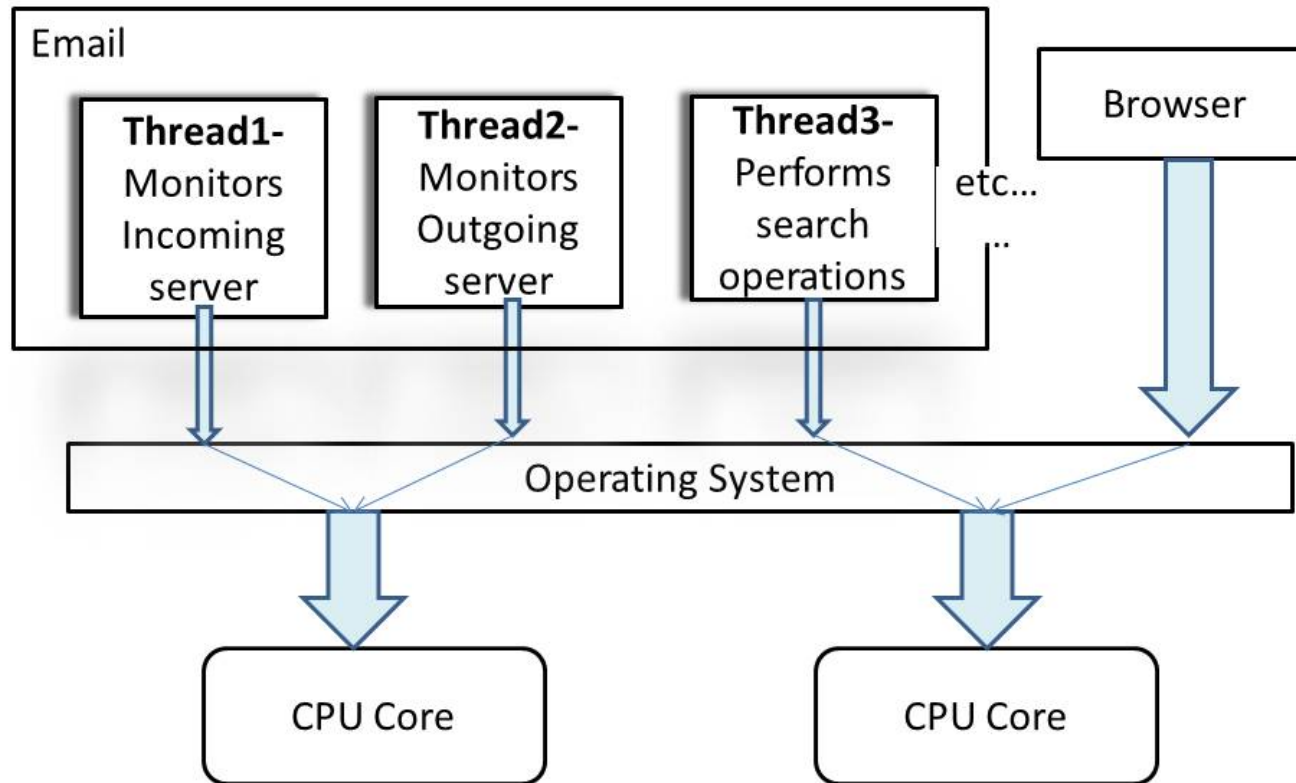
- Ad ogni thread è però associato
 - proprio **descrittore**
 - un program counter,
 - uno stato di esecuzione,
 - uno spazio di memoria per le variabili locali,
 - uno stack,
 - un contesto
- Le informazioni associate al thread sono poche
 - le operazioni di cambio di contesto, di creazione e terminazione sono più semplici

Multithreading

- Esistono applicazioni che, presentando un intrinseco grado di parallelismo, possono essere decomposte in attività che procedono concorrentemente, condividendo un insieme di dati e risorse comuni
- **Esempio:** Applicazioni in tempo reale per il controllo di impianti fisici, in cui si individuano attività quali:
 - Controllo dei singoli dispositivi di I/O dedicati a raccogliere i dati del processo fisico
 - Invio di comandi verso l'impianto
 - Le diverse attività devono frequentemente accedere a strutture dati comuni, che rappresentano lo stato complessivo del sistema da controllare

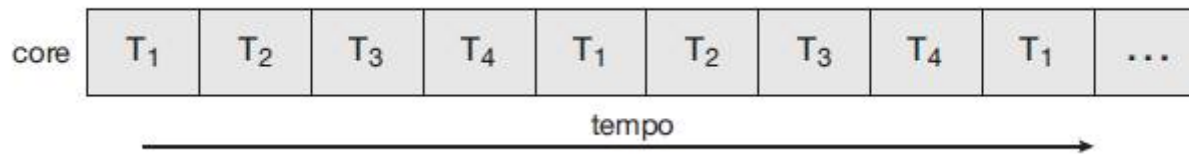
Multithreading

- Il multithreading si adatta perfettamente alla programmazione nei sistemi multicore



concorrenza VS parallelismo

Esecuzione concorrente su un sistema single core



Parallelismo nei sistemi multicore

