

Threads

Arturo Carpi

Dipartimento di Matematica e Informatica
Università di Perugia

Corso di Sistemi Operativi - a.a. 2021/22

I processi sono caratterizzati da:

Risorse allocate

- spazio di indirizzamento virtuale (immagine in memoria)
Traccia di esecuzione
- RAM
- dispositivi I/O
- file

Scheduling

- Traccia di esecuzione
 - alternata con altri processi
- stato
- priorità

Ma le due caratteristiche potrebbero essere trattate in maniera indipendente:

Processo

(o task)

Possiede le risorse (o processo leggero)

Thread

(o processo leggero)

Unità che viene assegnata al processore

Si dice multithreading la capacità di un SO di permettere più tracce di esecuzione concorrenti in un singolo processo.

Quattro casi:

<u>single-threading</u>	<u>multi-threading</u>
un processo, un thread (MS-DOS)	un processo, più thread (Java run-time)
più processi, un thread per ciascuno (UNIX tradizionale)	più processi, ciascuno con più thread (Windows, Solaris, UNIX)

Definizione

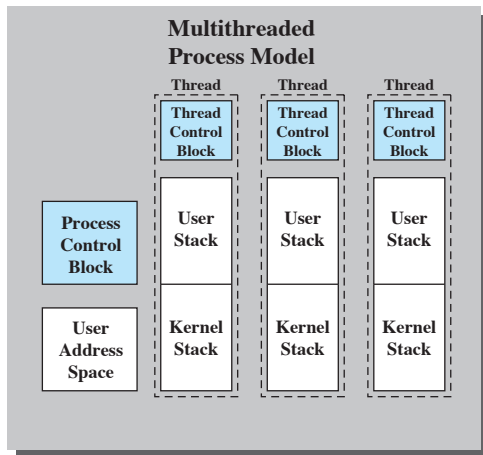
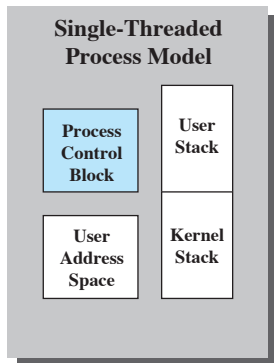
In un ambiente multi-thread, il processo è l'unità di allocazione delle risorse e un'unità di protezione.

Associati al processo

- Spazio di indirizzamento virtuale: contiene l'immagine del processo
- Accesso protetto al processore, ad altri processi (IPC), files, dispositivi I/O
- Uno o più thread

Associati al thread

- stato: pronto, in attesa, in esecuzione
- contesto del thread (immagine del processore)
 - salveto quando non è in esecuzione
- stack di esecuzione
- spazio di memoria statico per le variabili locali,
- accesso a memoria e risorse del processo, condivise con gli altri thread del processo



- la creazione di un thread è più rapida della creazione di un processo
- la terminazione di un thread è più rapida della terminazione di un processo
- lo switch tra due thread del medesimo processo è più rapida dello switch tra due processi
- efficienza della comunicazione fra programmi (eseguiti da thread del medesimo processo)

In un sistema a utente singolo:

- Esecuzione in foreground e background
- Elaborazione asincrona
- Velocità di esecuzione
- Struttura modulare dei programmi

Osservazione

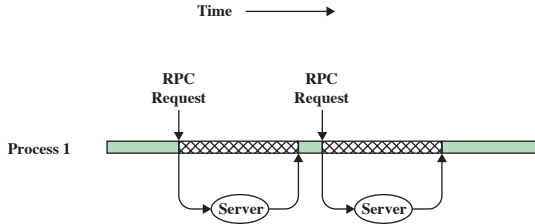
La schedulazione è gestita principalmente a livello di thread.
Ma non la sospensione (memory swap) e la terminazione.

Stati del thread

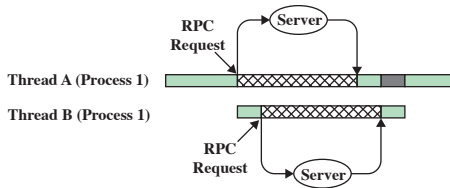
- In esecuzione
- Pronto
- In attesa

Operazioni sui thread




- Creazione
- Sospensione (block)
- Riattivazione (unblock)
- Terminazione



(a) RPC Using Single Thread



(b) RPC Using One Thread per Server (on a uniprocessor)

-  Blocked, waiting for response to RPC
-  Blocked, waiting for processor, which is in use by Thread B
-  Running

Sincronizzazione dei thread

- I thread di un processo condividono spazio di indirizzamento e altre risorse
- Ogni modifica delle risorse influenza gli altri thread
- La modifica simultanea di un dato da parte di due thread può corrompere il dato medesimo

È necessario sincronizzare l'accesso dei thread alle risorse

Thread a livello-utente e a livello kernel

Thread a livello utente

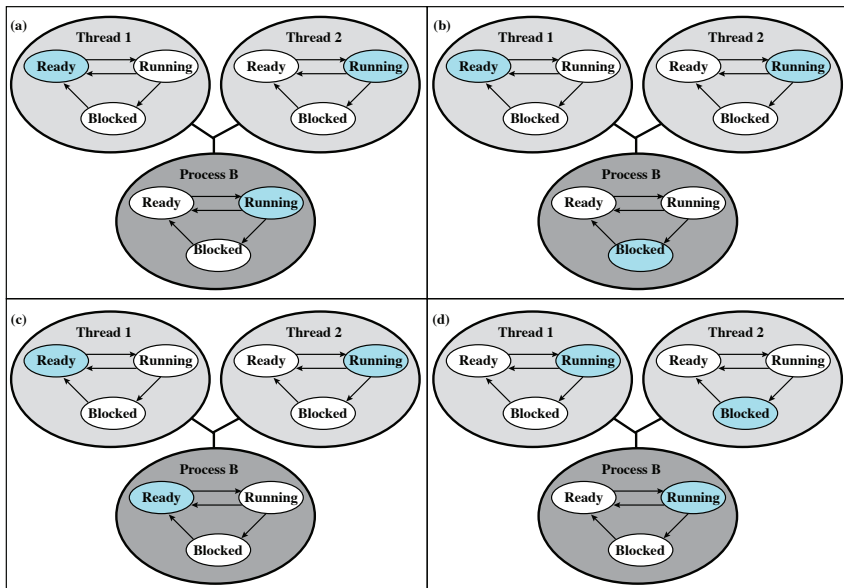
La gestione dei thread è fatta dall'applicazione

Il kernel non sa dell'esistenza dei thread

Thread a livello kernel

La gestione dei thread è fatta dal kernel

L'applicazione utilizza un'API per la gestione dei thread



Thread a livello utente

Vantaggi

- Lo scambio di thread non richiede privilegi della modalità kernel
- La schedulazione può essere specifica per l'applicazione
- Portabilità

Svantaggi

- Le chiamate di sistema sono bloccanti per l'intero processo
- Nessun vantaggio dai multiprocessori

Contromisure

- Jacketing: trasforma una chiamata di sistema bloccante in una non bloccante
- Programmare un'applicazione in processi multipli

Thread a livello kernel

- Il kernel gestisce i PCB dei processi e dei thread
- Schedulazione del processore sulla base dei thread

Vantaggi

- Possibilità di eseguire thread differenti su diversi processori
- Il blocco di un thread non blocca il processo
- Anche le routine del kernel possono essere gestite in multithreading

Svantaggi

- I cambi di modalità (utente/kernel) producono overhead

Operation	User-Level Threads	Kernel-Level Threads	Processes
Null Fork	34	948	11,300
Signal Wait	37	441	1,840

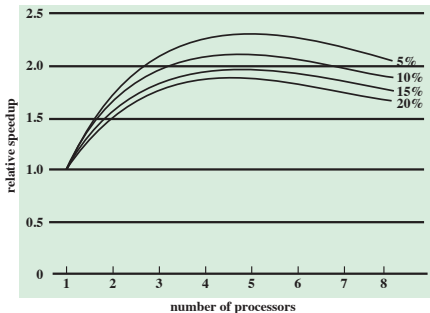
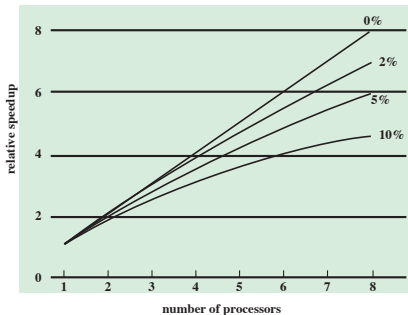
- Creazione dei thread, schedulazione e sincronizzazione nello spazio utente
- Thread modalità utente mappati su thread modalità kernel (in numero minore o uguale)
- Permette esecuzione parallela dei thread e il blocco di un thread non blocca il processo
 - Solaris

Model	Description	Example Systems
1:1	Each thread of execution is a unique process with its own address space and resources.	Traditional UNIX implementations
M:1	A process defines an address space and dynamic resource ownership. Multiple threads may be created and executed within that process.	Windows NT, Solaris, Linux, OS/2, OS/390, MACH
1:M	A thread may migrate from one process environment to another. This allows a thread to be easily moved among distinct systems.	Ra (Clouds), Emerald
M:N	Combines attributes of M:1 and 1:M cases.	TRIX

Multicore e multithreading

Legge di Amdahl

$$\text{speedup} = \frac{\text{tempo di esecuzione su un processore}}{\text{tempo di esecuzione su } N \text{ processori}} = \frac{1}{(1 - f) + \frac{f}{n}}$$



Chi si avvantaggia

- Applicazioni multithread native
 - pochi processi con molti thread
- Applicazioni multiprocesso
 - molti processi single-thread
- Applicazioni Java
 - JVM è un processo multithread e assicura schedulazione e gestione della memoria per le applicazioni Java
- Multiple istanze di un'applicazione

Threads in UNIX

- Il kernel non distingue tra processi e thread
- per ogni **task** (processo o thread) c'è un **task control block**
- **fork()** crea una nuova task, con una copia del **task control block**
- **clone()** crea una nuova task, ma nel suo task control block ci sono puntatori ai campi di quello del genitore (in base ai parametri passati a **clone**)