TANKERINO.com

Lezione 174

https://www.tankerino.com/it/corsi/10/tepi-4/lezioni/174/definizione-di-thread-e-differenze-con-i-processi

Processi e Thread: Definizione e Differenze

Immagina di aprire un programma sul tuo computer, per esempio, un editor di testo come Notepad. Ogni volta che apri un programma, il sistema operativo crea quello che chiamiamo un processo. Un processo è semplicemente un'istanza di un programma in esecuzione. Puoi pensare al processo come a un contenitore che racchiude tutto il necessario per far funzionare un programma, come il suo codice, i dati e le risorse utilizzate.

Ogni processo è isolato dagli altri, e questo significa che ha il suo spazio di memoria separato. Per esempio, se hai aperto più finestre di Notepad, ciascuna finestra è un processo separato, e ognuna gestisce i propri file e dati in modo indipendente. Se un processo si blocca o fallisce, gli altri non ne risentiranno. Questo isolamento è molto importante per evitare che un errore in un programma possa influenzare l'intero sistema.

Un processo è un'istanza di un programma in esecuzione, con un proprio spazio di memoria e risorse.

Un esempio pratico potrebbe essere quando apriamo due browser diversi sul nostro computer: Chrome e Firefox. Anche se entrambi sono programmi per navigare in Internet, vengono eseguiti come processi separati. Uno può chiudersi o bloccarsi senza influenzare l'altro.

Immagina di avere un programma che esegue un calcolo complesso. Ogni volta che lanci una nuova esecuzione di quel programma, viene creato un nuovo processo, isolato dal precedente.

Cos'è un thread?

Ora, all'interno di ogni processo, possiamo avere uno o più thread.

Un thread è la più piccola unità di esecuzione che può essere gestita dal sistema operativo.

Mentre un processo può essere visto come il "contenitore", un thread è ciò che esegue realmente le istruzioni del programma. In altre parole, se il processo è la macchina, il thread è il motore che fa

muovere la macchina.

Un thread è quindi una sequenza di istruzioni che viene eseguita all'interno del processo. La cosa interessante è che un processo può avere più thread che lavorano contemporaneamente. Questo viene chiamato multithreading. Immagina un programma che deve fare più cose allo stesso tempo: un editor video potrebbe avere un thread che gestisce l'interfaccia grafica e un altro che elabora il video. Grazie ai thread, possiamo eseguire più operazioni contemporaneamente.

Un thread è la più piccola unità di esecuzione all'interno di un processo, che esegue le istruzioni del programma.

Un esempio semplice è quando navighiamo su un sito web. Il browser utilizza più thread per caricare immagini, testo e altri contenuti contemporaneamente, in modo da rendere la navigazione più fluida e veloce.

Immagina un'applicazione che riproduce musica mentre tu navighi nel suo catalogo. La musica viene riprodotta da un thread, mentre un altro thread gestisce la navigazione nell'app.

Le differenze tra processi e thread

Adesso che sappiamo cos'è un processo e cos'è un thread, vediamo le differenze principali. La prima differenza importante è che i processi sono completamente separati l'uno dall'altro. Ogni processo ha il proprio spazio di memoria e le proprie risorse. Questo significa che non possono condividere facilmente dati tra loro. Per far comunicare due processi, dobbiamo usare meccanismi speciali come i socket o la memoria condivisa.

I processi sono separati e non condividono lo stesso spazio di memoria.

I thread, d'altra parte, condividono le risorse del processo che li contiene. Se ci sono più thread all'interno di un processo, possono accedere agli stessi dati e variabili. Questo rende i thread molto più efficienti quando devono lavorare insieme, ma introduce anche dei rischi. Se non si fa attenzione, due thread possono accedere agli stessi dati contemporaneamente, causando errori o risultati imprevedibili.

I thread condividono lo stesso spazio di memoria del processo in cui sono contenuti.

Un altro aspetto è l'overhead. Creare un nuovo processo richiede molte risorse, poiché il sistema

operativo deve allocare memoria e altre risorse per il nuovo processo. Invece, creare un thread è molto più leggero, perché i thread condividono le risorse del processo principale. Questo è il motivo per cui i thread vengono spesso utilizzati per eseguire operazioni parallele in modo efficiente.

Immagina di avere un programma che esegue un lungo calcolo matematico e, allo stesso tempo, aggiorna una barra di avanzamento. Creare un processo separato per ogni operazione sarebbe inefficiente. Utilizzare thread consente di eseguire entrambe le operazioni all'interno dello stesso processo, riducendo il carico di lavoro del sistema.

Infine, i processi possono essere usati per eseguire programmi completamente indipendenti, mentre i thread vengono usati per suddividere un programma in più compiti che possono essere eseguiti contemporaneamente.

Quando usare i processi e quando i thread?

Un'altra domanda importante è: quando dovresti usare un processo e quando dovresti usare un thread? Se hai bisogno di eseguire due programmi completamente separati che non devono condividere dati tra loro, i processi sono la scelta giusta. Questo è il caso tipico dei server web, dove ogni richiesta può essere gestita da un processo separato per garantire che un errore in una richiesta non influenzi le altre.

Invece, se hai bisogno di eseguire più compiti che devono lavorare insieme e condividere dati, i thread sono l'opzione migliore. Molte applicazioni moderne, come giochi o editor di video, usano thread per suddividere il lavoro e migliorare le prestazioni.

Usa i processi quando hai bisogno di eseguire programmi indipendenti. Usa i thread quando vuoi suddividere un programma in più operazioni parallele.

Per esempio, in un editor di video, puoi avere un thread che riproduce il video, uno che gestisce l'interfaccia utente e un altro che applica gli effetti in tempo reale. Tutti questi thread lavorano insieme per migliorare l'esperienza dell'utente.

Immagina un software di fotoritocco che applica filtri alle immagini mentre tu stai ancora modificando altre parti della foto. I thread permettono di gestire tutte queste operazioni in parallelo, senza rallentare l'interfaccia grafica.

(CC BY-NC-SA 3.0) lezione - by tankerino.com
https://www.tankerino.com

Questa lezione e' stata realizzata grazie al contributo di:

Risorse per la scuola
https://www.baobab.school

Siti web a Varese
https://www.francescobelloni.it