TANKERINO.com

Lezione 179

https://www.tankerino.com/it/corsi/10/tepi-4/lezioni/179/thread-a-livello-utente

Thread a Livello Utente

I thread a livello utente sono gestiti direttamente dal programma, senza il coinvolgimento del kernel del sistema operativo. Questo significa che la gestione della loro creazione, sincronizzazione e terminazione è interamente a carico dell'applicazione. Il sistema operativo vede solo il processo principale, senza essere consapevole dei vari thread che lavorano al suo interno.

Immaginate un programma come un grande ufficio, e i thread come impiegati. Con i thread a livello utente, è come se l'ufficio gestisse i suoi impiegati senza coinvolgere un supervisore esterno (il sistema operativo). Ogni impiegato (thread) sa cosa deve fare e si coordina con gli altri all'interno dell'ufficio stesso.

Questi thread sono più leggeri da gestire rispetto ai thread a livello kernel perché non devono interagire direttamente con il sistema operativo ogni volta che viene creato o terminato un thread. Questo li rende ideali per situazioni in cui si vogliono eseguire molte operazioni che non necessitano di interazione diretta con le risorse del sistema.

I thread a livello utente sono gestiti interamente dal programma e non sono visibili al sistema operativo.

Un aspetto importante dei thread a livello utente è che, essendo gestiti dal programma, possono essere molto flessibili. Tuttavia, questa flessibilità porta con sé anche delle sfide, come la gestione dei conflitti tra i thread e la necessità di sincronizzazione tra di essi.

Vantaggi dei thread a livello utente

Un vantaggio significativo dei thread a livello utente è la loro efficienza. Creare e terminare un thread a livello utente è molto più rapido rispetto a un thread a livello kernel. Questo perché non è necessario richiedere l'intervento del sistema operativo per ogni operazione di gestione del thread.

Immaginate un team di persone che devono completare dei compiti. Se ciascuno può decidere

autonomamente quando iniziare e terminare un compito, il lavoro procede più velocemente rispetto a dover chiedere a un supervisore per ogni decisione. Allo stesso modo, i thread a livello utente sono più veloci perché non devono attendere l'approvazione del sistema operativo.

Un altro vantaggio è la portabilità. Dal momento che i thread sono gestiti dalla libreria del programma, è più facile mantenere la stessa logica su diverse piattaforme. Questo rende i thread a livello utente utili in ambienti dove si ha bisogno di un'applicazione che possa funzionare su diversi sistemi operativi senza dover riscrivere il codice per ogni piattaforma.

I thread a livello utente sono ideali per applicazioni che necessitano di molte operazioni parallele, ma che non devono necessariamente sfruttare al massimo le capacità del processore. Ad esempio, possono essere usati per simulazioni, animazioni, e gestione di task di background.

I thread a livello utente sono più veloci e portabili perché non richiedono l'intervento del sistema operativo per la loro gestione.

Uno dei principali limiti dei thread a livello utente è che, essendo invisibili al sistema operativo, non possono sfruttare appieno il multiprocessing o il multicore del processore. Questo significa che, anche se un programma ha molti thread, essi possono essere eseguiti su un solo core alla volta, rallentando l'esecuzione se il programma richiede molta potenza di calcolo.

Immaginate una squadra di operai che lavorano su una strada. Se c'è solo una carreggiata, anche se ci sono molti operai, possono lavorare solo uno alla volta. Questo è simile ai thread a livello utente: ci sono molti thread, ma possono essere eseguiti solo uno per volta su un singolo core.

Un altro problema è la gestione delle chiamate bloccanti. Se un thread a livello utente esegue una chiamata che richiede l'attesa di una risorsa, come la lettura di un file o l'accesso a una rete, l'intero processo può rimanere bloccato, anche se ci sono altri thread che potrebbero continuare a lavorare.

Inoltre, la sincronizzazione tra thread a livello utente è interamente a carico del programmatore. Questo può rendere il codice più complicato e difficile da mantenere, poiché bisogna gestire manualmente i conflitti e assicurarsi che i thread non interferiscano tra loro.

I thread a livello utente possono essere eseguiti solo su un core alla volta e possono bloccare l'intero processo se uno di essi si blocca.

Implementazione dei thread a livello utente in C#

In C#, i thread a livello utente possono essere gestiti tramite la classe Task e l'uso di async e await. Questa libreria permette di creare thread leggeri che eseguono operazioni in parallelo senza coinvolgere direttamente il sistema operativo.

Ecco un esempio di utilizzo di thread a livello utente in C#:

```
using System;

using System.Threading.Tasks;

class Program{

    static async Task Main(string[] args) {

        Task task1 = DoWorkAsync("Task 1", 2000);

        Task task2 = DoWorkAsync("Task 2", 1000);

        await Task.WhenAll(task1, task2);

        Console.WriteLine("Tutti i task sono stati completati.");

   }

   static async Task DoWorkAsync(string name, int delay) {

        Console.WriteLine($"{name} iniziato.");

        await Task.Delay(delay);

        Console.WriteLine($"{name} completato.");

  }

}
```

In questo esempio, abbiamo due task che vengono eseguiti contemporaneamente. Ogni task è come un thread a livello utente, gestito dalla libreria di runtime di C#.

Il vantaggio di questo approccio è che possiamo creare molte operazioni parallele senza sovraccaricare il sistema operativo. Tuttavia, se uno di questi task si blocca, può influire sul comportamento complessivo del programma.

In C#, i thread a livello utente possono essere implementati tramite i Task e la gestione asincrona usando async e await.

Quando Usare i Thread a Livello Utente?

I thread a livello utente sono particolarmente utili in alcune situazioni specifiche. Ad esempio, quando stiamo lavorando su applicazioni leggere che devono essere altamente reattive e veloci, l'utilizzo di thread a livello utente può migliorare significativamente le prestazioni. Questo perché il costo associato alla gestione dei thread è molto inferiore rispetto a quello dei thread a livello kernel.

Immagina di creare un'applicazione di calcolo per l'analisi di dati finanziari. Ogni utente esegue numerose simulazioni simultaneamente. Utilizzare thread a livello utente ti permette di far girare più simulazioni contemporaneamente senza la necessità di far intervenire il kernel ogni volta, migliorando così le prestazioni.

Inoltre, i thread a livello utente possono essere utili quando si lavora su sistemi con risorse limitate. Poiché il costo di gestione è molto basso, possiamo eseguire molti più thread senza consumare una quantità significativa di memoria o di risorse di CPU. Questo è particolarmente utile nei dispositivi mobili o nei sistemi embedded, dove la potenza computazionale e la memoria sono limitate.

I thread a livello utente sono adatti per applicazioni che non richiedono frequenti operazioni di I/O e che devono essere gestite in ambienti a bassa latenza.

Infine, i thread a livello utente possono essere utilizzati in situazioni in cui desideriamo un controllo preciso su come i thread sono gestiti. Ad esempio, per applicazioni scientifiche in cui l'ottimizzazione del carico di lavoro è fondamentale, utilizzare thread a livello utente può consentire di avere un maggiore controllo su come distribuire e gestire i compiti.

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