TANKERINO.com

Lezione 178

https://www.tankerino.com/it/corsi/10/tepi-4/lezioni/178/modelli-di-programmazione-multithread

Modelli di Programmazione Multithread

In informatica, la programmazione multithread è una tecnica che permette di eseguire più operazioni contemporaneamente all'interno dello stesso programma. Questo significa che un programma può eseguire diverse parti del codice in parallelo, migliorando la sua efficienza e reattività. Ma cosa sono esattamente i thread? Possiamo immaginarli come dei piccoli flussi di esecuzione indipendenti che lavorano all'interno di un programma più grande.

La programmazione multithread è particolarmente utile quando abbiamo molte operazioni da svolgere e vogliamo risparmiare tempo. Ad esempio, pensiamo a un programma che deve scaricare più file da internet. Se utilizzasse un solo thread, dovrebbe scaricare i file uno alla volta. Con più thread, invece, potrebbe scaricarne diversi contemporaneamente, velocizzando il processo.

Immaginate un ristorante con un solo cuoco. Se il cuoco prepara un piatto alla volta, i clienti devono aspettare a lungo. Ma se il ristorante ha più cuochi (thread), diversi piatti possono essere preparati contemporaneamente, riducendo i tempi di attesa.

In un contesto di programmazione, gestire i thread richiede attenzione, perché bisogna assicurarsi che non ci siano conflitti o interferenze tra di loro. Ad esempio, se due thread cercano di modificare la stessa variabile contemporaneamente, potrebbero creare degli errori nei risultati.

Un thread è un flusso di esecuzione che può essere gestito in parallelo ad altri thread, eseguendo diverse porzioni di codice contemporaneamente all'interno di uno stesso processo.

Perché utilizzare la programmazione multithread?

La programmazione multithread permette di sfruttare al meglio le risorse del computer. Oggi i computer hanno più core, ovvero unità di elaborazione indipendenti che possono eseguire operazioni in parallelo. La programmazione multithread consente di distribuire il carico di lavoro tra questi core, migliorando le prestazioni del programma.

Un esempio pratico è un gioco per computer. I giochi moderni hanno molte attività da gestire contemporaneamente: il movimento dei personaggi, la grafica, la gestione della fisica, e molto altro. Utilizzando la programmazione multithread, ogni attività può essere gestita da un thread diverso, rendendo il gioco più fluido.

Pensiamo a un gioco in cui un thread si occupa di muovere i nemici, un altro thread aggiorna lo schermo e un altro ancora calcola la fisica delle collisioni. Se tutto questo fosse gestito da un solo thread, il gioco sarebbe lento e scattoso.

La multithread può essere utilizzata anche in applicazioni server, dove un server deve rispondere a molte richieste simultaneamente. Ad esempio, un server web che gestisce le richieste di più utenti contemporaneamente usa thread separati per ogni richiesta, rendendo il servizio più veloce e reattivo.

La programmazione multithread consente di eseguire più operazioni simultaneamente, sfruttando al meglio le capacità di calcolo di un computer con più core.

Tipi di modelli di programmazione multithread

Esistono diversi modi per gestire i thread in un programma. I tre principali modelli sono: thread a livello utente, thread a livello kernel e thread in modello ibrido. Ognuno di questi modelli ha vantaggi e svantaggi specifici, e viene utilizzato in situazioni diverse a seconda delle esigenze del programma.

Il modello dei thread a livello utente prevede che i thread siano gestiti direttamente dal programma, senza coinvolgere il sistema operativo. Questo significa che i thread sono più leggeri e facili da creare, ma ci sono dei limiti, come la possibilità di bloccare l'intero programma se un thread si blocca.

Pensiamo a un'applicazione per scaricare file. Utilizzando i thread a livello utente, il programma può creare un thread per ogni download senza dover chiedere al sistema operativo di intervenire.

Al contrario, i thread a livello kernel sono gestiti direttamente dal sistema operativo. Questo offre un miglior controllo sulle risorse e permette di eseguire i thread su più core del processore, ma la gestione richiede più risorse e tempo.

Immaginate un sistema operativo come un direttore d'orchestra che assegna i compiti a diversi musicisti (thread). Con i thread a livello kernel, il direttore ha il controllo completo su quali musicisti suonano e quando, ma il coordinamento richiede tempo.

Infine, il modello ibrido combina i vantaggi dei due modelli precedenti, permettendo di gestire i thread a livello utente e mappandoli sui thread del kernel. Questo modello è spesso utilizzato per ottenere un buon compromesso tra efficienza e flessibilità.

I tre principali modelli di thread sono: thread a livello utente, thread a livello kernel e thread in modello ibrido.

La sicurezza e la sincronizzazione dei thread

Quando si lavora con thread multipli, è essenziale capire come sincronizzare le operazioni per evitare problemi di concorrenza. La concorrenza si verifica quando due o più thread cercano di accedere contemporaneamente alla stessa risorsa, come una variabile o un file.

Per evitare conflitti, si possono utilizzare diversi strumenti come i lock o i mutex, che permettono di bloccare temporaneamente l'accesso a una risorsa mentre un thread la sta utilizzando. Questo assicura che solo un thread alla volta possa modificare la risorsa.

Immaginate due persone che cercano di usare lo stesso sportello automatico per prelevare denaro. Se lo fanno contemporaneamente, potrebbero prelevare più di quanto disponibile. Un "lock" è come un segnale che impedisce a una delle due persone di usare lo sportello finché non ha finito l'altra.

La sincronizzazione dei thread è essenziale per evitare conflitti quando più thread accedono alle stesse risorse contemporaneamente.

Nei prossimi incontri, approfondiremo ciascuno dei tre modelli di thread, analizzando i casi d'uso e come implementarli in C#. Ognuno di questi modelli ha applicazioni pratiche e richiede una comprensione delle necessità dell'applicazione per scegliere quello più adatto.

Questa lezione e' stata realizzata grazie al contributo di:

Baobab.school FrancescoBelloni.it software development

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

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