Esercitazione n.2
LPR-B-08
Thread Pooling

5/10/2008

Vincenzo Gervasi

ESERCIZIO 1 – CALCOLATORE ASINCRONO

- Si realizzi un **calcolatore asincrono**, in grado cioè di calcolare espressioni senza dover attendere la fine di un calcolo prima di accettare l'espressione successiva.
- Il programma deve accettare in input un'espressione in una delle seguenti forme:
  - PI:*precisione* – calcola il valore di PI greco, come abbiamo visto in precedenza, fino all'approssimazione di $1/10^{precisione}$
  - FIB:*n* – calcola il valore dell'*n*-esimo numero di Fibonacci, col metodo ricorsivo (con due chiamate)
  - FACT:*n* – calcola il fattoriale di *n*, col metodo ricorsivo (con una chiamate)
  - QUIT – termina il programma (una volta finiti i calcoli in sospeso)
- Ad ogni espressione, il programma aggiunge un task a un threadpool, che effettuerà il calcolo e stamperà il risultato a video, e torna immediatamente a chiedere il prossimo input
- I thread di calcolo devono stampare, a fine esecuzione: il nome del loro thread, l'istante di creazione, l'istante di inizio esecuzione, l'istante di completamento, e il risultato del calcolo

ESERCIZIO 2 – RAFFINAMENTI

- Con riferimento all'esercizio precedente:
  - Si faccia in modo che il ciclo principale del calcolatore stampi un **prompt** per indicare la disponibilità a ricevere input
    - Per esempio: "Inserisci espressione: "
  - Quando un calcolo termina, verrà stampato il risultato su video. Fare in modo che la stampa non confonda il prompt
    - In particolare, occorre che il ciclo principale **ristampi il prompt** se è stato stampato un risultato
- Si sperimenti poi con **diverse politiche di pooling**
  - In particolare: qual'è il numero di thread eseguibili contemporaneamente su queste macchine, in maniera tale da minimizzare i tempi d'attesa?
    - Tempo fra l'invio di una richiesta e il suo completamento
    - Tempo fra l'inizio del servizio di una richiesta e il suo completamento

ESERCIZIO 3 – UN EXECUTOR CUSTOM

- Si usi una struttura dati "**coda ordinata**" per implementare un Executor che scheduli i task in base a un campo priorità, indicato dal task stesso
- La priorità (che viene stabilita in fase di creazione e/o sottomissione del task) deve anche essere usata come **priorità del thread** relativo.
- Si tenga presente che i thread possono essere riutilizzati per task a priorità diversa (la priorità va quindi stabilita al momento della "presa in carico" di un task).

- Si crei infine un main di prova che sottometta vari task al vostro Executor, e ne visualizzi l'ordine di sottomissione, inizio e fine esecuzione (come negli esempi visti a lezione)