Sistemi operativi – laboratorio

Esercitazione 4: Processi, Memoria condivisa, Pthreads

11 Dicembre 2023 → 18, 21 Dicembre 2023

Si ricorda che per le Pthreads utilizziamo l'implementazione POSIX, mentre per memoria condivisa e semafori utilizziamo l'implementazione System V.

Ese_1:

Scrivere un programma C che calcoli la somma di un array di numeri interi in parallelo utilizzando pthread. Il programma deve prendere in ingresso un array di numeri interi e il numero di thread. Ogni thread deve essere responsabile della somma di una parte dell'array e il thread principale deve attendere che tutti i thread finiscano utilizzando `pthread_join` prima di stampare il risultato finale.

b. Svolgere lo stesso esercizio utilizzando memoria condivisa e processi, e non più le pthreads come nel primo caso.

Ese_2:

Scrivere un programma C che calcoli il fattoriale di un dato numero in parallelo utilizzando pthread. Il programma deve prendere in input un numero intero e ogni thread deve essere responsabile del calcolo di una parte del fattoriale. Il thread principale deve attendere che tutti i thread finiscano utilizzando pthread_join prima di stampare il risultato finale.

b. Svolgere lo stesso esercizio utilizzando memoria condivisa e processi, e non più le pthreads come nel primo caso.

Ese_3:

Scrivere un programma C per calcolare la sequenza di Fibonacci, nelle seguenti varianti:

1. in modo iterativo senza utilizzare processi o thread. Il programma deve prendere in input 'n' e stampare i primi 'n' numeri di Fibonacci;
2. utilizzando processi e memoria condivisa. Il programma deve prendere in input 'n' e stampare i primi 'n' numeri di Fibonacci;
3. utilizzando i thread. Il programma deve prendere in input 'n' e stampare i primi 'n' numeri di Fibonacci. Compilare il programma utilizzando i flag appropriati (ad esempio, `-pthread`) a seconda del vostro compilatore.

Per le tre diverse implementazioni confrontarle in base alle velocità d'esecuzione, concorrenza e parallelismo, utilizzo delle risorse e scalabilità.