Programmazione e Architetture degli Elaboratori - Foglio 5

Luca Manzoni, Michele Rispoli, Pietro Morichetti

Esercizio 01

Si scriva un programma C in cui, dato un numero intero `N`, vengono creati `N` thread, ciascuno dei quali stampa a schermo il messaggio `"Hello from thread %d!"`, dove `%d` è il thread-id di quel thread (ottenibile tramite la funzione `pthread_t pthread_self(void)`. Il main thread inoltre stampa a schermo `"Hello from main thread!"`.

Esercizio 02

Si scriva un programma C in cui, dato un numero intero `N`, viene creato un array `A` di interi, da 1 a N, dopodiché viene creato un thread, che dovrà calcolare la media dei valori contenuti in `A` e salvare il valore in un float. Il thread che calcola la media deve ricevere in input sia `A`,`N` e l'indirizzo dove scrivere il risultato. Una volta terminato il calcolo il main thread deve stampare a schermo il valore della media. (Suggerimento: per passare più argomenti a un thread possiamo utilizzare le `struct`)

Esercizio 03

Implementare l'algoritmo mergeSort (link alla pagina wikipedia) in maniera che le chiamate ricorsive vengano eseguite da dei nuovi thread a ogni passo. Una possibile implementazione dell'algoritmo può essere costituita da metodi con le seguenti signature:

```
void merge(int* A, int N);
void* mergeSort(void* args);
```

dove l'unico argomento della routine `mergeSort` potrebbe essere lo `struct`

```
// arg pack
struct Args{
    int* A;
    int N;
};
```

Per testare l'algoritmo potete inizializzare un array ordinato come al solito e darlo poi in pasto alla seguente routine che lo rimescola:

```
// array shuffling utility
void naiveShuffle(int*A,int N){
    srand(time(NULL));
    for (int iii=0;iii<N;++iii){
        int jjj;
        do{
            jjj=rand()%N;
        } while(iii==jjj);

        int buff = A[iii];
        A[iii]=A[jjj];
        A[jjj]=buff;
    }
}
```