**LAB 05 – THREADS**

**Alan Gleizer – 10416804 & Caio Corsini – 10342005**

**Código:**

#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <time.h>

#define TAM 100

#define UPPER 1001

#define LOWER 0

// Precisa deste struct para passar os argumentos como um ponteiro de void

typedef struct {

    int\* arr;

    int ini;

    int fim;

} ThreadData;

// Funcao principal para achar o maior elemento baseado em um indice inicial e um final

void\* maiorElem(void\* arg){

    ThreadData\* data = (ThreadData\*)arg;

    int\* arr = data->arr;

    int ini = data->ini;

    int fim = data->fim;

    int maior = arr[ini]; // Inicializa com o primeiro elemento do segmento de array

    for(int i = ini; i<fim; i++){

        if(arr[i] > maior) maior = arr[i];

    }

    int\* result = malloc(sizeof(int));

    \*result = maior;

    return (void\*)result;

}

int main(){

    // Criiando o vetor que sera trabalhado

    srand(time(NULL)); // Seed para o rand gerar valores diferentes sempre

    int valores[TAM];

    for(int i=0; i<TAM; i++) valores[i] = rand() % (UPPER - LOWER + 1) + LOWER;

    // Determinando o ponto central do vetor

    int meio = TAM/2;

    // Metade da esquerda que sera enviada para a thread 1

    ThreadData dados1;

    dados1.arr = valores;

    dados1.ini = 0;

    dados1.fim = meio;

    // Metade da direita que sera enviada para a thread 2

    ThreadData dados2;

    dados2.arr = valores;

    dados2.ini = meio;

    dados2.fim = TAM;

    // Ja que serao 2 threads, sao necessarios 2 thread\_id e dois ponteiros de int para armazenar os resultados

    pthread\_t thread\_id1, thread\_id2;

    int \*resultado1, \*resultado2;

    // Criando as threads

    pthread\_create(&thread\_id1, NULL, maiorElem, &dados1);

    pthread\_create(&thread\_id2, NULL, maiorElem, &dados2);

    // Colocando as threads para executar

    pthread\_join(thread\_id1, (void\*\*)&resultado1);

    pthread\_join(thread\_id2, (void\*\*)&resultado2);

    // Determinando qual dos dois resultados encontrados eh o maior

    int oMaior;

    if(\*resultado1 > \*resultado2) oMaior = \*resultado1;

    else oMaior = \*resultado2;

    // Apresentacao dos resultados finais

    printf("O maior elemento da thread 1 eh: %d\n", \*resultado1);

    printf("O maior elemento da thread 2 eh: %d\n", \*resultado2);

    printf("O maior elemento do vetor randomizado eh: %d\n", oMaior);

    // Liberando a memoria

    free(resultado1);

    free(resultado2);

    return 0;

}

**Resultado:**

