Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу

«Операционные системы»

Тема работы

“Потоки”

Студент: Зубко Дмитрий Валерьевич
Группа: М8О-208Б-20
Вариант: 17
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/usernameMAI/OS/tree/main/os_lab3

Постановка задачи

Цель работы

Целью является приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме.
Вариант 17. Найти в большом целочисленном массиве минимальный элемент

Общие сведения о программе

Программа написана на языке C++. Для компиляции требуется указать ключ –pthread.

Общий метод и алгоритм решения

При запуске программы требуется указать количество потоков *th_n* и количество элементов *el_n* в массиве, где будет происходить поиск минимального элемента. Этот массив разбивается на $\frac{el\_n}{th\_n}$ частей. В каждой части разные потоки находят минимальный элемент и заносят его в отдельный массив. В конце программы просматривается этот отдельный массив и выводится минимальный элемент. Также выводятся минимальные элементы в каждой из частей исходного массива чисел.
Создаём поток с помощью thread(функция, параметры функции).

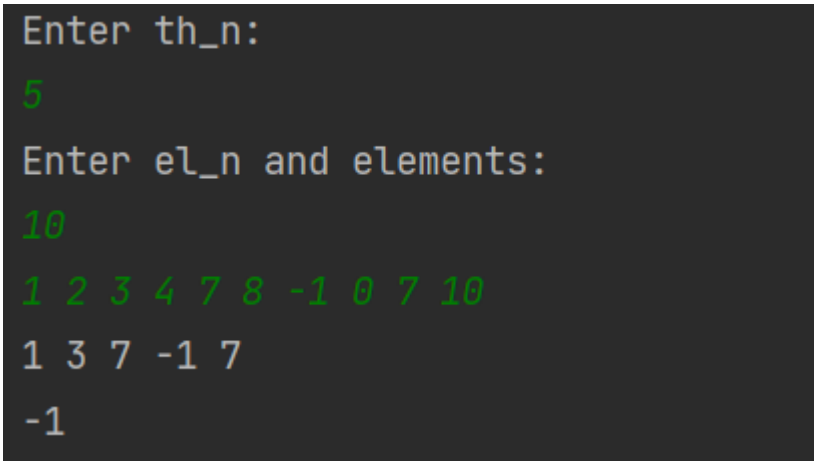
Исходный код

```
#include <iostream>
#include <thread>
#include <vector>

using namespace std;

void min_elem(vector<int> const& elements, vector<int>& min, const
int l, const int r, const int i) {
```

```
    int min_el = elements[l];

    for (int j = l; j <= r; ++j)
    {
        if (elements[j] < min_el)
            min_el = elements[j];
    }
//    cout << this_thread::get_id() << endl;
    min[i] = min_el;
}

int main() {
    int th_n, el_n;
    cout << "Enter th_n: \n";
    cin >> th_n;
    cout << "Enter el_n and elements: \n";
    cin >> el_n;

    vector<int> elements(el_n);
    vector<int> min(th_n);
    vector<thread> threads(th_n);

    for (int i = 0; i < el_n; ++i)
        cin >> elements[i];

    int l = 0;
    int delta = el_n / th_n;
    int last_id = th_n - 1;

    for (int i = 0; i < th_n; ++i)
    {
        if (i != last_id)
        {
            threads[i] = thread(min_elem, ref(elements), ref(min),
l, l + delta - 1, i);
            l += delta;
        } else
        {
            threads[i] = thread(min_elem, ref(elements), ref(min),
l, el_n - 1, i);
        }
    }

    for (int i = 0; i < th_n; ++i)
    {
        threads[i].join();
    }

    for (int i = 0; i < th_n; ++i)
        cout << min[i] << ' ';

    cout << endl;

    int ans = min[0];
    for (int i = 1; i < th_n; ++i)
```

```
        if (min[i] < ans)
            ans = min[i];

    cout << ans << endl;
}
```

Демонстрация работы программы



Выводы

Лабораторная работа ознакомила и научила меня работать с потоками. Я изучил некоторые функции из библиотеки thread. Понял, что при выполнении сложных задач можно распараллеливать их для увеличения производительности.