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

Лабораторная работа №3 по курсу

«Операционные системы»

Студент: Копылов Михаил Юрьевич
Группа: М80-201Б-21
Вариант: 6
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Исходный код
5. Выводы

Репозиторий

https://github.com/Mikhail-cWc/OS_mai/tree/main/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. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 6

Произвести перемножение 2-ух матриц, содержащих комплексные числа

Общие сведения о программе

Программа компилируется из файла main.cpp. Используются системные вызовы Windows.

Исходный код

Main.cpp

```
#include "include\\lab3.h"

int main()
{
    srand(time(0));
    int NumberOfThreads, n, m, k;

    std::cout << "Введите количество потоков, N, M, K:\n";
    std::cin >> NumberOfThreads;
    std::cin >> n;
    std::cin >> m;
    std::cin >> k;
    if (n <= 0 || m <= 0 || k <= 0 || NumberOfThreads <= 0)
    {
        std::cout << "ERROR CREATED COMPLEX MATRIX or NUM OF THREADS\n";
        return -1;
    }

    TComplexMatrix matr_1(n, std::vector<std::complex<double>>(m, 0));
    TComplexMatrix matr_2(m, std::vector<std::complex<double>>(k, 0));
```

```
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            matr_1[i][j] = std::complex<double>(rand() % 15, rand() % 30);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
            std::cout << matr_1[i][j] << " ";
        std::cout << std::endl;
    }
    std::cout << std::endl;

    for (int i = 0; i < m; i++)
        for (int j = 0; j < k; j++)
            matr_2[i][j] = std::complex<double>(rand() % 15, rand() % 30);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
            std::cout << matr_2[i][j] << " ";
        std::cout << std::endl;
    }
    std::cout << std::endl;

    double avg = 0;
    auto start = std::chrono::high_resolution_clock::now();

    auto res = Parallelization(matr_1, matr_2, NumberOfThreads);

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < k; j++)
            std::cout << res[i][j] << " ";
        std::cout << std::endl;
    }

    auto end = std::chrono::high_resolution_clock::now();
    avg += std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count();
    printf("Время выполнения программы(миллисекунды): %f\n", avg);
    system("PAUSE");
    return 0;
}
```

Lab3.cpp

```
#include "lab3.h"

MYDATA *mydata;
unsigned long WINAPI MatrixMultiplication(LPVOID param)
{
    int iNumm = *(reinterpret_cast<int *>(param));

    for (int i = mydata->from[iNumm]; i < mydata->to[iNumm]; i++)
    {
        for (int j = 0; j < mydata->res[0].size(); j++)
        {
            mydata->res[i][j] = 0;
            for (int z = 0; z < mydata->res.size(); z++)
                mydata->res[i][j] += mydata->left[i][z] * mydata->right[z][j];
        }
    }

    return 0;
}
```

```
TComplexMatrix Parallelization(TComplexMatrix left, TComplexMatrix right, int threads)
{
    mydata = (MYDATA *)HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
                                 sizeof(MYDATA));
    std::vector<int> to(threads, 0);
    std::vector<int> from(threads, 0);
    TComplexMatrix res(left.size(), std::vector<std::complex<double>>(right[0].size(),
0));
    mydata->left = left;
    mydata->right = right;
    mydata->res = res;

    // Данные переменные нужны для распределения вычислений по потокам
    int q = left.size() / threads;
    int r = left.size() % threads;
    mydata->to = to;
    mydata->from = from;

    HANDLE hThreads[threads];
    LPDWORD id;
    int noms[threads];

    for (int i = 0; i < threads; i++)
    {
        // У последнего потока будет четное кол-во строк для вычислений
        // При условии что в матрице строк нечетное кол-во
        mydata->to[i] = mydata->from[i] + q + (i < r ? 1 : 0);
        noms[i] = i;
        hThreads[i] = CreateThread(NULL, 0, MatrixMultiplication, (LPVOID)(noms + i),
0, id);
        if (i < threads - 1)
            mydata->from[i + 1] = mydata->to[i];

        if (hThreads[i] == NULL)
            std::cout << "ERROR CREATE THREADS - " << i << std::endl;
    }

    WaitForMultipleObjects(threads, hThreads, TRUE, INFINITE);

    for (int i = 0; i < threads; i++)
        CloseHandle(hThreads[i]);

   TComplexMatrix result = mydata->res;

    HeapFree(GetProcessHeap(), 0, mydata);
    mydata = NULL;

    return result;
}
```

Lab3.h

```
#ifndef OS_LABS_LAB3_H
#define OS_LABS_LAB3_H

#include <vector>
#include <complex>
#include <windows.h>
#include <process.h>
#include <ctime>
#include <iostream>
#include <chrono>
```

```
using TComplexMatrix = std::vector<std::vector<std::complex<double>>>;

struct MYDATA
{

	TComplexMatrix right;
	TComplexMatrix left;
	TComplexMatrix res;
	std::vector<int> from, to;
};

DWORD WINAPI MatrixMultiplication(LPVOID param);

TComplexMatrix Parallelization(TComplexMatrix left, TComplexMatrix right, int threads);

#endif // OS_LABS_LAB3_H
```

Выводы

Составлена и отлажена многопоточная программа на языке Си, выполняющая умножение матрицы на матрицу. Тем самым, приобретены навыки в распараллеливании вычислений, управлении потоками и обеспечении синхронизации между ними.