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: Меджидли Исмаил Ибрагим оглы
Группа: М8О-201Б-21
Вариант: 10
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023

Содержание

Репозиторий

https://github.com/imedzhidli/Operational-Systems

Постановка задачи

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.

Наложить K раз медианный фильтр на матрицу, состоящую из целых чисел. Размер окна задается

Общие сведения о программе

CMakeLists.txt - описание процесса сборки проекта

main.cpp - считывание и вывод данных

laba3OS.h - заголовочный файл, описаны функции для работы с матрицей

laba3OS.cpp - реализация функций, которые определены в laba3OS.h

utils.h - полезные функции

lab3_test.cpp - тесты к лабораторной работе

Общий метод и алгоритм решения

Программе в качестве Параметров подаётся общее число потоков. Если оно не задано, то программа запускается в двупоточном режиме. Потом задаётся размер матрицы, размер окна и количество фильтров.

Программа разделяет матрицу на строки и дает обрабатывать одному потоку количество строк так, чтобы каждому потоку досталось одинаковое количество строк (бывают случаи, когда один поток обрабатывает на 1 строку больше чем остальные). Потоки никак не пересекаются, когда обращаются к матрице, ведь программа работает по принципу двух матриц: если нужно наложить 1 раз медианный фильтр, результаты каждой ячейки кладутся во 2-ую матрицу, если же надо 2 раза наложить фильтр, то исходной матрицей для потоков становится матрица 2, а матрица 1 становится той, куда записывается ответ.

Исходный код

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16.3)

project(lab3 LANGUAGES CXX)

find_package(Threads REQUIRED)

add_executable(lab3
       main.cpp
       include/laba3OS.h src/laba3OS.cpp)

target_include_directories(lab3 PRIVATE include)
```

```
target_link_libraries(lab3 PRIVATE Threads::Threads)
```

laba3OS.h

```
#ifndef OS_LABS_LAB3_H
#define OS_LABS_LAB3_H

#include <vector>
#include <iostream>
#include <string>
#include <pthread.h>
#define MAX(a, b) ((a) > (b) ? (a) : (b))
#define MIN(a, b) ((a) < (b) ? (a) : (b))
using namespace std;

using TMatrix = vector<vector<int>>;

struct thread_args{
    int height;
    int width;
    TMatrix *first_matrix;
    TMatrix *second_matrix;
    int area;
    int lower_row;
    int upper_row;
};

void ReadMatrix(TMatrix &matrix);

void WriteMatrix(TMatrix &matrix);

int median(vector <int> &matrix);

void median_filter(int &height, int &width, TMatrix &a, TMatrix &b, int
&area, int &lower_row, int &upper_row);

void First(int &n, int &m, TMatrix &mat1, TMatrix &mat2, int &k, int
&window_size, int &threadCount, TMatrix &res);

#endif //OS_LABS_LAB3_H
```

laba3OS.cpp

```
#include "laba3OS.h"
#include "utils.h"
```

```
void ReadMatrix(TMatrix &matrix) {
    for (int i = 0; i < Isize(matrix); i++) {
        for (int j = 0; j < Isize(matrix[i]); j++) {
            cin >> matrix[i][j];
        }
    }
}

void WriteMatrix(TMatrix &matrix) {
    for (int i = 0; i < Isize(matrix); i++) {
        for (int j = 0; j < Isize(matrix[i]); j++) {
            cout << matrix[i][j] << " ";
        }
        cout << "\n";
    }
}


int median(vector <int> &matrix){
    int n = matrix.size();
    for(int i = 0; i < n; i++){
        for(int j = i; j < n; j++){
            if(matrix[i] > matrix[j]){
                int temp = matrix[i];
                matrix[i] = matrix[j];
                matrix[j] = temp;
            }
        }
    }
    return matrix[n / 2];
}

void median_filter(int &height, int &width, TMatrix &a, TMatrix &b, int
&area, int &lower_row, int &upper_row){
    int radius = (area - 1) / 2;
    for (int y = lower_row; y < upper_row; y++){
        int top = MAX(y - radius, 0);
        int bottom = MIN(y + radius, height - 1);
        for (int x = 0; x < width; x++){
            int left = MAX(x - radius, 0);
            int right = MIN(x + radius, width - 1);
            vector <int> m((bottom - top + 1) * (right - left + 1));
            int k = 0;
            for (int v = top; v <= bottom; v++){
```

```
                for (int u = left; u <= right; u++){
                    m[k] = a[v][u];
                    k++;
                }
            }
            b[y][x] = median(m);
        }
    }
}

void *lineresation(void *args){
    thread_args *arg = (thread_args*) args;
    median_filter(arg->height, arg->width, *arg->first_matrix,
*arg->second_matrix, arg->area, arg->lower_row, arg->upper_row);
    return NULL;
}


void First(int &n, int &m, TMatrix &mat1, TMatrix &mat2, int &k, int
&window_size, int &threadCount, TMatrix &res){
    int imedy = n / threadCount;
    pthread_t threads[threadCount];
    pthread_attr_t attr;
    pthread_attr_init(&attr);
    thread_args p_args[threadCount];
    for (int i  = 0; i < k; i++){
        for(int j = 0; j < threadCount; j++){
            p_args[j].height = n;
            p_args[j].width = m;

            if (i % 2 == 0){
                p_args[j].first_matrix = &mat1;
                p_args[j].second_matrix = &mat2;
            }
            else{
                p_args[j].first_matrix = &mat2;
                p_args[j].second_matrix = &mat1;
            }
            p_args[j].area = window_size;
            p_args[j].lower_row = j * imedy;
            p_args[j].upper_row = (j + 1) * imedy;
            if (j == threadCount - 1)
                p_args[j].upper_row = n;
```

```
            cout << "Поток "<< j+1 << " работает с [" <<
p_args[j].lower_row <<","<< p_args[j].upper_row << ") строкой(ами)" <<
"\n";
        }
        for (int i = 0; i < threadCount; i++){
            int res = pthread_create(&threads[i], &attr, lineresation,
&p_args[i]);
            if (res != 0){
                cout << "Ошибка с созданием потока!";
                exit(-2);
            }
        }
        for (int i = 0; i < threadCount; i++){
            int res = pthread_join(threads[i], NULL);
            if (res != 0){
                cout << "Ошибка с ожиданием выхода!";
                exit(-3);
            }
        }
        printf("-------------------------------------------------\n");
    }

    if (k % 2 == 1){
        res = mat2;}
    else {
        res = mat1;}

}
```

utils.h

```
#ifndef OS_LABS_UTILS_H
#define OS_LABS_UTILS_H

template <typename Container>
inline int Isize(const Container& cont) {
    return static_cast<int>(cont.size());
}

#endif //OS_LABS_UTILS_H
```

main.cpp

```
#include "laba3OS.h"
```

```
int main(int argc, char* argv[]){

    int n, m, threadCount;

    cout << "Введите кол-во потоков: ";
    cin >> threadCount;

    cout << "Введите размеры матрицы: ";
    cin >> n >> m;


    TMatrix mat1(n, vector<int>(m));
    TMatrix mat2(n, vector<int>(m));

    if (threadCount > n) threadCount = n;


    int window_size;

    printf("Введите размер окна (от 3 и более нечетное число): ");
    cin >> window_size;

    printf("Введите кол-во медианных фильтров: ");
    int k;
    cin >> k;

    printf("Введите вашу матрицу:\n");

    ReadMatrix(mat1);
    TMatrix res;

    First(n, m, mat1, mat2, k, window_size, threadCount, res);
    WriteMatrix(res);

    return 0;
}
```

lab3_test.cpp

```
#include <gtest/gtest.h>

#include "laba3OS.h"
#include "utils.h"

#include <chrono>
```

```
namespace {
    TMatrix GenerateMatrix(int n, int m) {
        TMatrix result(n, vector<int>(m));

        srand(time(nullptr));

        for(int i = 0; i < n; ++i) {
            for(int j = 0; j < m; ++j) {
                result[i][j] = rand() % 100;
            }
        }

        return result;
    }
}

bool operator==(const TMatrix& lhs, const TMatrix& rhs) {
    if(lhs.size() != rhs.size()) {
        return false;
    }

    for(int i = 0; i < Isize(lhs); ++i) {
        if(lhs[i].size() != rhs[i].size()) {
            return false;
        }

        for(int j = 0; j < Isize(lhs); ++j) {
            if(lhs[i][j] != rhs[i][j]) {
                return false;
            }
        }
    }

    return true;
}

TEST(Lab3Test, MedianTest) {

    int expectedres = 3;
    int expectedres2 = 2;

    vector <int> beforefunc = {0, 0, 1, 20, 34, 3, 50};
    vector <int> beforefunc2 = {0, 2, 1, 3};
```

```
    int res = median(beforefunc);
    int res2 = median(beforefunc2);
    EXPECT_EQ(expectedres, res);
    EXPECT_EQ(expectedres2, res2);

}

TEST(Lab3Test, FirstTest) {
    srand(time(NULL));
    auto getAvgTime = [](int thread_count) {
    int n = 3; int m = 3;
    int window_size = 3;
    // int thread_count = 2;
    int k = 2;
    TMatrix expectedMatrix{
        {1, 1, 2},
        {1, 1, 2},
        {1, 1, 3}
    };
    TMatrix mat1 = {
        {1, 2, 3},
        {1, 1, 0},
        {1, 4, 5}
    };
    TMatrix mat2(n, vector<int>(m));
    TMatrix res;

    double avg = 0;

    auto begin = chrono::high_resolution_clock::now();
    First(n, m, mat1, mat2, k, window_size, thread_count, res);
    EXPECT_EQ(res, expectedMatrix);
    auto end = chrono::high_resolution_clock::now();
    avg += chrono::duration_cast<chrono::milliseconds>(end -
begin).count();


        return avg / 10;
    };
    auto singleThread = getAvgTime(1);
    auto multiThread = getAvgTime(3);

    cout << "Avg time for 1 thread: " << singleThread << '\n';
    cout << "Avg time for 3 threads: " << multiThread << '\n';
```

```
    EXPECT_LE(singleThread, multiThread);
}
```

Демонстрация работы программы

imedzhidli@imedzhidli:~/Desktop/OS/3Laba/build$
/home/imedzhidli/Desktop/OS/3Laba/build/lab3

Введите кол-во потоков: 3

Введите размеры матрицы: 3 3

Введите размер окна (от 3 и более нечетное число): 3

Введите кол-во медианных фильтров: 2

Введите вашу матрицу:

1 2 3

1 1 0

1 4 5

Поток 1 работает с [0,1) строкой(ами)

Поток 2 работает с [1,2) строкой(ами)

Поток 3 работает с [2,3) строкой(ами)

Поток 1 работает с [0,1) строкой(ами)

Поток 2 работает с [1,2) строкой(ами)

Поток 3 работает с [2,3) строкой(ами)

---

1 1 2

1 1 2

1 1 3

imedzhidli@imedzhidli:~/Desktop/OS/3Laba/build$

Выводы

Я приобрел практические навыки в:

1) Управлении потоками в ОС
2) Обеспечении синхронизации между потоками