Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**УПРАВЛЕНИЕ ПОТОКАМИ**

Студент: Климов Иван Павлович

Группа: М8О–212Б–22

Вариант: 13

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

Приобретение практических навыков в:

* Управление процессами в ОС
* Обеспечение обмена данных между процессами посредством каналов

## Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, 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 входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

13 вариант) Наложить K раз фильтр, использующий матрицу свертки, на матрицу, состоящую из

вещественных чисел. Размер окна задается пользователем

**Общие сведения о программе**

Программа компилируется из файла main.c.

**int** main(**int** argc,**char**\* argv[])принимает количество используемых потоков. А также есть процедура **void** processing(), которая накладывает матрицу свертки на данную матрицу. В цикле for() создаем n потоков, каждый из которых обрабатывает свой сектор матрицы.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы мютексов и потоков в си.
2. Реализовать функцию, проводящую наложение: В функцию поступает данная матрица, матрица свертки, результурующая матрица, количество потоков, актуальные граница результирующей матрицы, мьютекс. Каждый поток обрабатывает свой сектор матрица, а затем заносит реультат в результирующую матрицу. Затем налодение повторяет к раз.
3. Реализовать наложение матрицы свертки на данную матрицу.
4. Проверить, какой вариант решения задачи в данном случае лучше: однопоточное решение или многопоточное.

**Основные файлы программы**

**main.c:**

#include <iostream>

#include <fstream>

#include <vector>

#include <thread>

#include <chrono>

#include <mutex>

#include <cmath>

**using** **namespace** std;

**void** processing(vector<vector<**double** > >& matrix, vector<vector<**double** > >& convolution\_Matrix, vector<vector<**double**> >& rez\_Matrix, **int** V, mutex& mtx, **int** step, **int** A, **int** B);

// Функция печати итоговых матриц

**void** print(**const** vector<vector<**double** > >& matrix) {

**int** rows = matrix.size();

**int** cols = matrix[0].size();

**for** (**int** i = 0; i < rows; i++) {

**for** (**int** j = 0; j < cols; j++) {

cout << matrix[i][j] << " ";

}

cout << "\n";

}

cout << "\n";

}

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

**int** A = 0, B = 0;

cout << "start" << endl;

cout << "Максимально возможное количество потоков: " << thread::hardware\_concurrency() << "\n";

**if** ( atoi(argv[1]) > 12 || atoi(argv[1]) <= 0){

cout << "Заданное число потоков не может быть использовано." << "\n";

**return** 1;

}

**const** **int** Max = atoi(argv[1]);

**int** rows, cols, p, q, K;

vector<thread> th(Max);

cout << "Введите количество раз для наложения фильтров: ";

cin >> K;

cout << "\n";

cout << "Введите i & j: ";

cin >> rows >> cols;

cout << "Введите p & q: ";

cin >> p >> q;

//A = rows; B = cols;

vector<vector<**double** > > matrix(rows, vector<**double**>(cols));

**for** (**int** i = 0; i < rows; i++) {

**for** (**int** j = 0; j < cols; j++) {

matrix[i][j] = rand() % 10;

}

}

print(matrix);

vector<vector<**double** > > convolution\_Matrix(p, vector<**double**>(q));

**for** (**int** i = 0; i < p; i++) {

**for** (**int** j = 0; j < q; j++) {

convolution\_Matrix[i][j] = rand() % 3;

}

}

print(convolution\_Matrix);

vector<vector<**double**> > rez\_matrix(rows-p+1, vector<**double**>(cols-q+1));

//print(rez\_matrix);

// Начало отсчета времени

chrono::steady\_clock::time\_point start\_time = chrono::steady\_clock::now();

cout << "ID потока main: " << this\_thread::get\_id() << "\n";

std::mutex mtx; // Мьютекс для синхронизации доступа к матрице

// Работа единственного потока

**if** (Max == 1){

**for**(**int** i = 0; i < K; i++){

processing(matrix, convolution\_Matrix, rez\_matrix, 0, ref(mtx), 1, 0, 0);

A = A + p - 1;

B = B + q - 1;

}

}

**if** (Max > 1){

**for**(**int** i = 0; i < K; i++){

**for**(**int** t = 0; t < Max; t++){

th[t] = thread(processing, ref(matrix), ref(convolution\_Matrix), ref(rez\_matrix), t, ref(mtx), Max, A, B);

}

**for**(**int** t = 0; t < Max; t++){

th[t].join();

}

A = A + p - 1;

B = B + q - 1;

}

}

cout << "Матрица после наложения " << K << " раз матрицы свертки: " << "\n";

**for** (**int** i = 0; i < rows - A; i++) {

**for** (**int** j = 0; j < cols - B; j++) {

cout << matrix[i][j] << " ";

}

cout << "\n";

}

cout << "\n";

// Конец отсчета времени

chrono::steady\_clock::time\_point end\_time = chrono::steady\_clock::now();

chrono::microseconds duration = chrono::duration\_cast<chrono::microseconds>(end\_time - start\_time);

cout << "Время работы программы: " << duration.count()/Max << " микросекунд. " << "\n";

**return** 0;

}

**void** processing(vector<vector<**double**> >& matrix, vector<vector<**double**> >& convolution\_Matrix, vector<vector<**double**> >& rez\_matrix, **int** V, mutex& mtx, **int** step, **int** A, **int** B){

**int** rows = matrix.size() - A;

**int** cols = matrix[0].size() - B;

**int** q = convolution\_Matrix.size();

**int** p = convolution\_Matrix[0].size();

**for**(**int** i = 0; i < (rows-p+1); i++){

**for**(**int** j = V; j <= (cols-q); j+=step){

**int** summ = 0;

**for**(**int** a = 0; a < q; a++){

**for**(**int** b = 0; b < p; b++){

summ += fmod(convolution\_Matrix[a][b] \* matrix[a+i][b+j], 1000);

}

}

mtx.lock();

rez\_matrix[i][j] = summ;

mtx.unlock();

}

}

matrix = rez\_matrix;

}

**Пример работы**

ivanklimov@MacBook-Air-Ivan-2 src % ./main 1

start

Максимально возможное количество потоков: 8

Введите количество раз для наложения фильтров: 3

Введите i & j: 10 10

Введите p & q: 2 2

7 9 3 8 0 2 4 8 3 9

0 5 2 2 7 3 7 9 0 2

3 9 9 7 0 3 9 8 6 5

7 6 2 7 0 3 9 9 9 1

7 2 3 6 5 5 8 1 4 7

1 3 8 4 8 0 4 6 0 3

2 6 9 4 1 3 7 8 8 3

8 1 5 3 5 4 3 6 5 9

5 4 9 1 7 5 5 4 1 8

8 3 5 2 2 6 6 7 8 4

0 0

2 2

ID потока main: 0x1042b8580

Матрица после наложения 3 раз матрицы свертки:

304 264 208 200 360 528 512

224 272 328 352 360 288 240

304 376 352 256 208 240 200

408 368 216 184 312 448 464

232 240 264 264 256 288 360

360 328 304 336 328 264 224

272 208 184 256 360 424 440

Время работы программы: 125 микросекунд.

ivanklimov@MacBook-Air-Ivan-2 src % ./main 5

start

Максимально возможное количество потоков: 8

Введите количество раз для наложения фильтров: 3

Введите i & j: 15 15

Введите p & q: 3 3

7 9 3 8 0 2 4 8 3 9 0 5 2 2 7

3 7 9 0 2 3 9 9 7 0 3 9 8 6 5

7 6 2 7 0 3 9 9 9 1 7 2 3 6 5

5 8 1 4 7 1 3 8 4 8 0 4 6 0 3

2 6 9 4 1 3 7 8 8 3 8 1 5 3 5

4 3 6 5 9 5 4 9 1 7 5 5 4 1 8

8 3 5 2 2 6 6 7 8 4 1 7 1 8 7

8 7 7 4 4 9 1 5 5 5 8 2 9 8 2

0 7 4 8 5 8 3 0 6 2 2 5 2 2 7

1 5 2 1 1 0 1 8 7 6 0 0 9 5 6

2 8 5 9 4 3 9 0 5 6 3 6 3 0 8

4 0 4 6 4 6 4 3 8 4 0 0 9 3 7

5 7 2 9 7 5 6 5 3 0 4 8 5 5 5

4 2 1 9 2 4 5 8 6 2 5 1 3 9 7

3 7 5 0 2 1 8 7 1 5 5 8 5 6 7

1 1 0

0 1 1

2 2 1

ID потока main: 0x104644580

ID потока: 0x16bae7000

ID потока: 0x16bb73000

ID потока: 0x16bbff000

ID потока: 0x16bc8b000

ID потока: 0x16bd17000

ID потока: 0x16bae7000

ID потока: 0x16bb73000

ID потока: 0x16bbff000

ID потока: 0x16bc8b000

ID потока: 0x16bd17000

ID потока: 0x16bae7000

ID потока: 0x16bb73000

ID потока: 0x16bbff000

ID потока: 0x16bc8b000

ID потока: 0x16bd17000

Матрица после наложения 3 раз матрицы свертки:

2746 3348 3546 3137 3119 2349 3769 574 875

2512 2709 2496 2683 3143 2221 3613 618 929

2189 2115 1772 3067 4227 2092 3405 630 915

2120 2812 2147 3206 4428 1835 3970 572 869

2130 2842 2288 1905 3795 1836 3849 538 846

2268 2180 3340 2479 2375 1825 2980 468 699

2477 2432 3092 2257 1623 2080 3254 496 788

102 172 684 1080 920 89 140 540 750

119 189 728 1074 812 136 150 580 888

Время работы программы: 153 микросекунд.

**Вывод**

Анализируя результаты выполненной работы, можно сказать, что увеличение количества потоков влияет на производительность только тогда, когда мы обрабатываем большие матрицы(100 на 100)

Пример обработки матрицы 100 на 100; количество наложений 10; матрица свертки 7 на 7:

|  |  |
| --- | --- |
| Потоки | Миллисекунды |
| 1 | 115357 |
| 2 | 23086 |
| 4 | 7681 |
| 7 | 4540 |

Когда матрица маленькая, то разницы никакой во времени обработки нет.