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**

Студент: Попов Матвей Романович

Группа: М8О-208Б-20

Вариант: 1

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

[https://github.com/.../os\_lab](https://github.com/.../os_lab2)3

**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.

**Вариант 12**: наложить K раз фильтр, использующий матрицу свёртки, на матрицу, состоящую из вещественных чисел. Размер окна 3x3.

**Общие сведения о программе**

Программа представляет из себя один файл main.cpp.

**Общий метод и алгоритм решения**

Исходная матрица разбивается на множество матриц 3x3, которые преобразуются с помощью матрицы свёртки в вещественные числа, составляющие результирующую матрицу. Каждое такое преобразование является отдельным потоком.

**Исходный код**

#include <iostream>

#include <vector>

#include <thread>

#include <string>

using namespace std;

vector<double> res;

void bebra(double (\*buffer)[3], double(\*conv)[3], int current)

{

    for (int i = 0; i < 3; ++i)

    {

        for (int j = 0; j < 3; ++j)

        {

            res[current] += buffer[i][j] \* conv[i][j];

        }

    }

}

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

{

    int thread\_amount;

    if (argc < 2)

    {

        cout << "Enter thread amount:\n";

        cin >> thread\_amount;

    }

    else

    {

        thread\_amount = stoi(argv[1]);

    }

    cout << "Thread amount is " << thread\_amount << endl;

    res.resize(thread\_amount);

    for (int i = 0; i < thread\_amount; ++i)

    {

        res[i] = 0.0;

    }

    vector<thread> th(thread\_amount);

    int current = 0;

    int k;

    cout << "Enter k:\n";

    cin >> k;

    int lines, columns;

    cout << "Enter amount of lines and columns:\n";

    do

    {

        cin >> lines >> columns;

        if ((lines - 2 \* k <= 0)||(columns - 2 \* k <= 0))

        {

            cout << "Error, try again:\n";

        }

    } while ((lines - 2 \* k <= 0)||(columns - 2 \* k <= 0));

    vector<vector<double>> orig(lines, vector<double> (columns, 0.0));

    cout << "Enter original matrix:\n";

    for (int i = 0; i < lines; ++i)

    {

        for (int j = 0; j < columns; ++j)

        {

            cin >> orig[i][j];

        }

    }

    cout << "Enter conv. 3x3 matrix:\n";

    double conv[3][3];

    for (int i = 0; i < 3; ++i)

    {

        for (int j = 0; j < 3; ++j)

        {

            cin >> conv[i][j];

        }

    }

    vector<vector <double>> result(lines, vector<double> (columns, 0.0));

    for (int t = 1; t <= k; ++t)

    {

        for (int I = 0; I < lines - 2 \* t; ++I)

        {

            for (int J = 0; J < columns - 2 \* t; ++J)

            {

                double buffer[3][3];

                for (int i = 0; i < 3; ++i)

                {

                    for (int j = 0; j < 3; ++j)

                    {

                        buffer[i][j] = orig[i + I][j + J];

                    }

                }

                th[current] = thread(bebra, buffer, conv, current);

                result[I][J] = res[current];

                ++current;

                if (current == thread\_amount)

                {

                    current = 0;

                    for (int i = 0; i < thread\_amount; ++i)

                    {

                        th[i].join();

                        res[i] = 0.0;

                    }

                }

            }

            for (int i = 0; i < current; ++i)

            {

                th[i].join();

                res[i] = 0.0;

            }

            current = 0;

        }

        for (int i = 0; i < lines - 2 \* t; ++i)

        {

            for (int j = 0; j < columns - 2 \* t; ++j)

            {

                orig[i][j] = result[i][j];

            }

        }

    }

    cout << "\nResult:\n";

    for (int i = 0; i < lines - 2 \* k; ++i)

    {

        for (int j = 0; j < columns - 2 \* k; ++j)

        {

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

        }

        cout << endl;

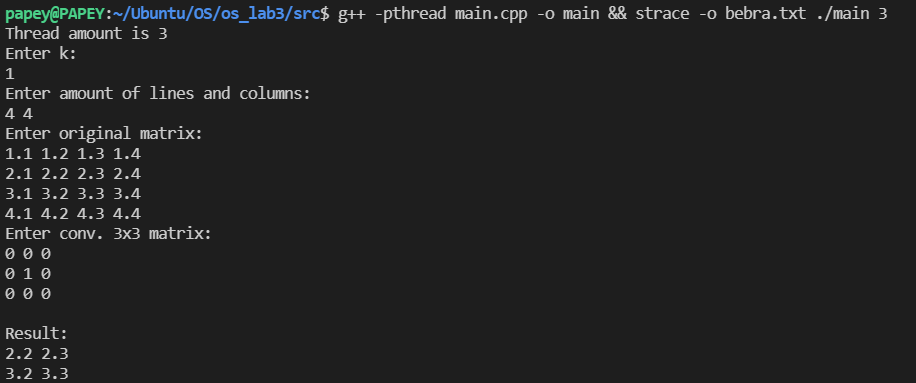
    }

    return 0;

}

**Демонстрация работы программы**

Ввод в консоль:



Содержимое файла bebra.txt:

execve("./main", ["./main", "3"], 0x7fffe8a1f708 /\* 27 vars \*/) = 0

brk(NULL) = 0x7fffdd435000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=34422, ...}) = 0

mmap(NULL, 34422, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f368c8a4000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/usr/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\304\10\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1594864, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f368c8a0000

mmap(NULL, 3702848, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f368c270000

mprotect(0x7f368c3e9000, 2097152, PROT\_NONE) = 0

mmap(0x7f368c5e9000, 49152, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x179000) = 0x7f368c5e9000

mmap(0x7f368c5f5000, 12352, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f368c5f5000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\300\*\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=96616, ...}) = 0

mmap(NULL, 2192432, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f368c050000

mprotect(0x7f368c067000, 2093056, PROT\_NONE) = 0

mmap(0x7f368c266000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x16000) = 0x7f368c266000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000b\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=144976, ...}) = 0

mmap(NULL, 2221184, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f368be30000

mprotect(0x7f368be4a000, 2093056, PROT\_NONE) = 0

mmap(0x7f368c049000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19000) = 0x7f368c049000

mmap(0x7f368c04b000, 13440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f368c04b000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20\35\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030928, ...}) = 0

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f368ba30000

mprotect(0x7f368bc17000, 2097152, PROT\_NONE) = 0

mmap(0x7f368be17000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f368be17000

mmap(0x7f368be1d000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f368be1d000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\200\272\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=1700792, ...}) = 0

mmap(NULL, 3789144, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f368b690000

mprotect(0x7f368b82d000, 2093056, PROT\_NONE) = 0

mmap(0x7f368ba2c000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19c000) = 0x7f368ba2c000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f368c890000

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f368c880000

arch\_prctl(ARCH\_SET\_FS, 0x7f368c880740) = 0

mprotect(0x7f368be17000, 16384, PROT\_READ) = 0

mprotect(0x7f368ba2c000, 4096, PROT\_READ) = 0

mprotect(0x7f368c049000, 4096, PROT\_READ) = 0

mprotect(0x7f368c266000, 4096, PROT\_READ) = 0

mprotect(0x7f368c5e9000, 40960, PROT\_READ) = 0

mprotect(0x7f368cc08000, 4096, PROT\_READ) = 0

mprotect(0x7f368c829000, 4096, PROT\_READ) = 0

munmap(0x7f368c8a4000, 34422) = 0

set\_tid\_address(0x7f368c880a10) = 759

set\_robust\_list(0x7f368c880a20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f368be35cb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f368be42980}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f368be35d50, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f368be42980}, NULL, 8) = 0

rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=8192\*1024}) = 0

brk(NULL) = 0x7fffdd435000

brk(0x7fffdd456000) = 0x7fffdd456000

futex(0x7f368c5f609c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

futex(0x7f368c5f60a8, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

write(1, "Thread amount is 3\n", 19) = 19

write(1, "Enter k:\n", 9) = 9

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

read(0, "1\n", 1024) = 2

write(1, "Enter amount of lines and column"..., 35) = 35

read(0, "4 4\n", 1024) = 4

write(1, "Enter original matrix:\n", 23) = 23

read(0, 0x7fffdd4472c0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, 0x7fffdd4472c0, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)

--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---

read(0, "1.1 1.2 1.3 1.4\n", 1024) = 16

read(0, "2.1 2.2 2.3 2.4\n", 1024) = 16

read(0, "3.1 3.2 3.3 3.4\n", 1024) = 16

read(0, "4.1 4.2 4.3 4.4\n", 1024) = 16

write(1, "Enter conv. 3x3 matrix:\n", 24) = 24

read(0, "0 0 0\n", 1024) = 6

read(0, "0 1 0\n", 1024) = 6

read(0, "0 0 0\n", 1024) = 6

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f368ae80000

mprotect(0x7f368ae81000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f368b67ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f368b6809d0, tls=0x7f368b680700, child\_tidptr=0x7f368b6809d0) = 1102

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f368a670000

mprotect(0x7f368a671000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f368ae6ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f368ae709d0, tls=0x7f368ae70700, child\_tidptr=0x7f368ae709d0) = 1103

clone(child\_stack=0x7f368ae6ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f368ae709d0, tls=0x7f368ae70700, child\_tidptr=0x7f368ae709d0) = 1105

clone(child\_stack=0x7f368b67ffb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tidptr=0x7f368b6809d0, tls=0x7f368b680700, child\_tidptr=0x7f368b6809d0) = 1106

write(1, "\nResult:\n", 9) = 9

write(1, "2.2 2.3 \n", 9) = 9

write(1, "3.2 3.3 \n", 9) = 9

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

exit\_group(0) = ?

+++ exited with 0 +++

**Выводы**

Проделав лабораторную работу, я приобрёл практические навыки в управлении потоками в ОС и обеспечил синхронизацию между ними.