**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные системы»

**Лабораторная работа № 3**

Тема: Управление потоками в ОС

Студент: Тимофеев А.В.

Группа: 80-207

Преподаватель: Миронов Е. С.

Дата:

Оценка:

**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.

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

Наложить K раз фильтры эрозии и наращивания на матрицу, состоящую из вещественных чисел. На выходе получается 2 результирующие матрицы

**Алгоритм решения**

В файле matrix.h содержится структура матрицы функции инициализации и отчистки.

В файле main.c содержится основной код программы, а именно функции эрозии и расширения матрицы, функции поиска локального минимума и максимума, глобальные переменные матриц, в main.c реализован алгоритм разбиения на потоки эрозии и дилатация матриц. Алгоритм состоит из цикла отвечающего за К раз обработку матриц и внутренних циклов, которые разбивают матрицу построчно и передают в поточные функции эрозии и дилатации.

Эффективность вычисляется по формуле **,** где S – эффективность,  **–** время выполнения программы на одном потоке.  **–** время выполнения на n-ом потоке.   
Тестирование происходит на матрице 10х10 из положительных вещественных чисел, при К равном 5.

|  |  |  |  |
| --- | --- | --- | --- |
| Процессы | Время | Ускорение | Эффективность |
| 1 | 7.330000 | 1,0 | 1 |
| 2 | 6.028000 | 1,22 | 0,88 |
| 3 | 5.884000 | 1,25 | 0,42 |
| 4 | 6.798000 | 1,08 | 0,27 |
| 5 | 6.294000 | 1,16 | 0,232 |
| 6 | 6.014000 | 1,22 | 0,20 |
| 7 | 6.401000 | 1,15 | 0,16 |
| 8 | 7.952000 | 0,92 | 0,115 |
| 9 | 8.508000 | 0,86 | 0,095 |

Можно заметить что после 3-го потока время затраченной на обработку данных начинает расти, это происходит из-за того что я выделил 3 ядра ЦП под виртуальную машину на которой я делал лабораторную работу, поэтому максимальное ускорение происходит при 3-х потоках.

**Листинг программы**

**main.c**

#include <time.h>

#include <stdio.h>

#include "matrix.h"

#include <stdlib.h>

#include <pthread.h>

#include <float.h>

//n - строка

// m - столбец

struct Matrix erM;

struct Matrix dilM;

struct thr\_data{

int numstr; // массив номеров строк которые мы обрабатываем

float out; // массив для ответов из ф-ий matrixMin matrixMax

};

void matrixMin(struct thr\_data \*tmpdata){

tmpdata->out = FLT\_MAX; // MIN

for(int i = 0; i < erM.dimm; i++){

if (tmpdata->out > erM.matr[tmpdata->numstr][i]) {

if (erM.matr[tmpdata->numstr][i] == 0){

erM.matr[tmpdata->numstr][i] += (rand() % 1000);

erM.matr[tmpdata->numstr][i] /= 10000;

}

tmpdata->out = erM.matr[tmpdata->numstr][i];

}

}

}

void matrixMax(struct thr\_data \*tmpdata){

tmpdata->out = -(FLT\_MAX); // MAX

for(int i = 0; i < dilM.dimm; i++){

if (tmpdata->out < dilM.matr[tmpdata->numstr][i]) {

if (dilM.matr[tmpdata->numstr][i] == 0){

dilM.matr[tmpdata->numstr][i] += (rand() % 1000);

dilM.matr[tmpdata->numstr][i] /= 10000;

}

tmpdata->out = dilM.matr[tmpdata->numstr][i];

}

}

}

void erosion(struct thr\_data \*tmpdata){ // struct thr\_data \*tmpdata передает минимум и номер строки которую нужно обработать

for(int i = 0; i < erM.dimm; i++){

erM.matr[tmpdata->numstr][i] -= tmpdata->out;

}

}

void dilatation(struct thr\_data \*tmpdata){

for(int i = 0; i < dilM.dimm; i++){

dilM.matr[tmpdata->numstr][i] += tmpdata->out;

}

}

int k, n, m;

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

unsigned int max\_threads = 1;

if (argc > 1 && atoi(argv[1]) > 0) {

max\_threads = atoi(argv[1]);

}

printf("Threads %d\nEnter size of matrix. Enter N and M\n ", max\_threads);

fflush(stdout);

scanf("%d %d", &n, &m);

fflush(stdout);

if (n == 0 || m == 0) {

printf("I cannot create matrix with this n and m\n");

return 0;

}

scanfMatrix(&erM, n, m);

init\_Matr(&dilM,n,m);

for(int q = 0; q < erM.dimn; q++){ // копируем значения из введенной матрицы во 2-ю

for(int l = 0; l < erM.dimm; l++){

dilM.matr[q][l] = erM.matr[q][l];

}

}

printf("Matrix for erosion\n");

printMatrix(&erM);

printf("Matrix for dilatation\n");

printMatrix(&dilM);

fflush(stdout);

printf("Enter your K\n ");

fflush(stdout);

scanf("%d", &k);

pthread\_t \*threads = (pthread\_t \*) malloc(sizeof(pthread\_t) \* max\_threads);

double start, end;

struct thr\_data arr\_thr\_data[n];

for(int i = 0; i < n; ++i){

arr\_thr\_data->numstr = 0;

arr\_thr\_data->out = 0;

}

int count = 0;

while(count < k) {

count++;

float max = -(FLT\_MAX);

float min = FLT\_MAX;

for (unsigned int i = 0; i < n; i += max\_threads){

for (unsigned int j = 0; j < max\_threads; j++){

if (i + j >= n) {//чтобы не запускался под конец новый процесс

break;

}

arr\_thr\_data[i + j].numstr = i + j;

if (pthread\_create(&threads[j], NULL, matrixMin, /\*(void \*) &args[i + j]\*/ &arr\_thr\_data[i+j]) != 0){

perror("Cannot create thread\n");

return 1;

}

}

//ждем завершения потоков

for (int j = 0; j < max\_threads; ++j){ // пока ждем процессы проверяем массив arr\_thr\_data на минимум

if (i + j >= n) {

break;

}

if (pthread\_join(threads[j], NULL)){

perror("Thread didn't finished\n");

return 1;

}

if(arr\_thr\_data[i + j].out < min){

min = arr\_thr\_data[i + j].out;

}

}

}

for(int i = 0; i < n; i++){ // в arr\_thr\_data заменяем все out на минимальное значение чтобы потом передать в erosion

arr\_thr\_data[i].out = min;

}

for (unsigned int i = 0; i < n; i += max\_threads){

for (unsigned int j = 0; j < max\_threads; j++){

if (i + j >= n) {//чтобы не запускался под конец новый процесс

break;

}

arr\_thr\_data[i + j].numstr = i + j;

if (pthread\_create(&threads[j], NULL, erosion, /\*(void \*) &args[i + j]\*/ &arr\_thr\_data[i+j]) != 0){

perror("Cannot create thread\n");

return 1;

}

}

//ждем завершения потоков

for (int j = 0; j < max\_threads; ++j){ // пока ждем процессы проверяем массив arr\_thr\_data на минимум

if (i + j >= n) {

break;

}

if (pthread\_join(threads[j], NULL)){

perror("Thread didn't finished\n");

return 1;

}

}

}

for (unsigned int i = 0; i < n; i += max\_threads){

for (unsigned int j = 0; j < max\_threads; ++j){

if (i + j >= n){//чтобы не запускался под конец новый процесс

break;

}

arr\_thr\_data[j].numstr = i + j;

if (pthread\_create(&threads[j], NULL, matrixMax, /\*(void \*) &args[i + j]\*/ &arr\_thr\_data[i+j]) != 0){

perror("Cannot create thread\n");

return 1;

}

}

//ждем завершения потоков

for (int j = 0; j < max\_threads; ++j){// пока ждем процессы проверяем массив arr\_thr\_data на максимум

if (i + j >= n){

break;

}

if (pthread\_join(threads[j], NULL)){

perror("Thread didn't finished\n");

return 1;

}

if(arr\_thr\_data[i + j].out > max){

max = arr\_thr\_data[i + j].out;

}

}

}

for(int i = 0; i < n; i++){ // в arr\_thr\_data заменяем все out на минимальное значение чтобы потом передать в erosion

arr\_thr\_data[i].out = max;

}

for (unsigned int i = 0; i < n; i += max\_threads){

for (unsigned int j = 0; j < max\_threads; ++j){

if (i + j >= n) {//чтобы не запускался под конец новый процесс

break;

}

arr\_thr\_data[i + j].numstr = i + j;

if (pthread\_create(&threads[j], NULL, dilatation, /\*(void \*) &args[i + j]\*/ &arr\_thr\_data[i+j]) != 0){

perror("Cannot create thread\n");

return 1;

}

}

//ждем завершения потоков

for (int j = 0; j < max\_threads; j++){ // пока ждем процессы проверяем массив arr\_thr\_data на минимум

if (i + j >= n) {

break;

}

if (pthread\_join(threads[j], NULL)){

perror("Thread didn't finished\n");

return 1;

}

}

}

}

printf("\nITERATION %d\n\n",count);

printf("Matrix for erosion\n");

printMatrix(&erM);

printf("Matrix for dilatation\n");

printMatrix(&dilM);

freematrix(&erM);

freematrix(&dilM);

free(threads);

return 0;

}

**matrix.h**

#ifndef MATFUNC\_H

#define MATFUNC\_H

struct Matrix {

int dimn;

int dimm;

float \*\*matr;

};

void scanfMatrix(struct Matrix \*a, int n, int m) { // Ввод матрицы while (getchar() != '\n');

a->dimn = n;

a->dimm = m;

a->matr = (float\*\*)malloc(sizeof(float\*)\*n);

for (int i=0; i<n; i++) {

a->matr[i] = (float\*) malloc(sizeof(float)\*m);

}

for (int i=0; i<n; i++) {

for (int j=0; j<m; j++) {

scanf("%f", &a->matr[i][j]);

}

}

}

/\*int random(int seed) {

seed \* 123456;

}\*/

void init\_Matr(struct Matrix\* a, int n, int m){

a->dimm = m;

a->dimn = n;

a->matr = (float\*\*)malloc(sizeof(float\*)\*n);

for (int i = 0; i < n; ++i) {

a->matr[i] = (float\*)malloc(sizeof(float)\*m);

}

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

a->matr[i][j] = 0;

}

}

}

void printMatrix(struct Matrix \*a) { // Печать матрицы

printf("\n");

for (int i=0; i<a->dimn; i++) {

for (int j=0; j<a->dimm; j++) {

printf("%f ", a->matr[i][j]);

}

printf("\n");

}

printf("\n");

}

#endif

**Тесты и протокол исполнения**

dude@dude:~/Рабочий стол/oc/OC/laba3$ ./lab3 3| cat >logf

5 5

1.0 2.0 0.5 5.0 8.5

7.5 5.55 0.2 1 8

1 5 6 8 9

10 0.2 10 2 3

6 6 6 6 6

3

dude@dude:~/Рабочий стол/oc/OC/laba3$ cat logf

Threads 3

Enter size of matrix. Enter N and M

Matrix for erosion

1.000000 2.000000 0.500000 5.000000 8.500000

7.500000 5.550000 0.200000 1.000000 8.000000

1.000000 5.000000 6.000000 8.000000 9.000000

10.000000 0.200000 10.000000 2.000000 3.000000

6.000000 6.000000 6.000000 6.000000 6.000000

Matrix for dilatation

1.000000 2.000000 0.500000 5.000000 8.500000

7.500000 5.550000 0.200000 1.000000 8.000000

1.000000 5.000000 6.000000 8.000000 9.000000

10.000000 0.200000 10.000000 2.000000 3.000000

6.000000 6.000000 6.000000 6.000000 6.000000

Enter your K

ITERATION 3

Matrix for erosion

0.711400 1.711400 0.211400 4.711400 8.211401

7.211400 5.261400 0.027400 0.711400 7.711400

0.711400 4.711400 5.711400 7.711400 8.711401

9.711401 0.000000 9.711401 1.711400 2.711400

5.711400 5.711400 5.711400 5.711400 5.711400

Matrix for dilatation

71.000000 72.000000 70.500000 75.000000 78.500000

77.500000 75.550003 70.199997 71.000000 78.000000

71.000000 75.000000 76.000000 78.000000 79.000000

80.000000 70.199997 80.000000 72.000000 73.000000

76.000000 76.000000 76.000000 76.000000 76.000000

dude@dude:~/Рабочий стол/oc/OC/laba3$ ./lab3 4

Threads 4

Enter size of matrix. Enter N and M

2 4

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Matrix for erosion

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Matrix for dilatation

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Enter your K

3

ITERATION 3

Matrix for erosion

0.173100 1.173100 2.173100 3.173100

4.423100 0.000000 3.173100 3.173100

Matrix for dilatation

37.750000 38.750000 39.750000 40.750000

42.000000 37.450001 40.750000 40.750000

dude@dude:~/Рабочий стол/oc/OC/laba3$ ./lab3 2

Threads 2

Enter size of matrix. Enter N and M

2 4

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Matrix for erosion

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Matrix for dilatation

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Enter your K

1

ITERATION 1

Matrix for erosion

0.300000 1.300000 2.300000 3.300000

4.550000 0.000000 3.300000 3.300000

Matrix for dilatation

6.250000 7.250000 8.250000 9.250000

10.500000 5.950000 9.250000 9.250000

dude@dude:~/Рабочий стол/oc/OC/laba3$ ./lab3

Threads 1

Enter size of matrix. Enter N and M

2 4

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Matrix for erosion

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Matrix for dilatation

1.000000 2.000000 3.000000 4.000000

5.250000 0.700000 4.000000 4.000000

Enter your K

8

ITERATION 8

Matrix for erosion

0.000000 0.927900 1.927900 2.927900

4.177900 0.003300 2.927900 2.927900

Matrix for dilatation

1339.750000 1340.750000 1341.750000 1342.750000

1344.000000 1339.449951 1342.750000 1342.750000

**strace:**

dude@dude:~/Рабочий стол/oc/OC/laba3$ cat logf

execve("./lab3", ["./lab3", "3"], 0x7ffe19302c98 /\* 56 vars \*/) = 0

brk(NULL) = 0x555a35385000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fff6814cce0) = -1 EINVAL (Недопустимый аргумент)

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (Нет такого файла или каталога)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 6

fstat(6, {st\_mode=S\_IFREG|0644, st\_size=67057, ...}) = 0

mmap(NULL, 67057, PROT\_READ, MAP\_PRIVATE, 6, 0) = 0x7f3b507e7000

close(6) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) = 6

read(6, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\201\0\0\0\0\0\0"..., 832) = 832

pread64(6, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68

fstat(6, {st\_mode=S\_IFREG|0755, st\_size=157224, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f3b507e5000

pread64(6, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\345Ga\367\265T\320\374\301V)Yf]\223\337"..., 68, 824) = 68

mmap(NULL, 140408, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 6, 0) = 0x7f3b507c2000

mmap(0x7f3b507c9000, 69632, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x7000) = 0x7f3b507c9000

mmap(0x7f3b507da000, 20480, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x18000) = 0x7f3b507da000

mmap(0x7f3b507df000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x1c000) = 0x7f3b507df000

mmap(0x7f3b507e1000, 13432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f3b507e1000

close(6) = 0

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

read(6, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\360q\2\0\0\0\0\0"..., 832) = 832

pread64(6, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(6, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(6, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

fstat(6, {st\_mode=S\_IFREG|0755, st\_size=2029224, ...}) = 0

pread64(6, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

pread64(6, "\4\0\0\0\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32, 848) = 32

pread64(6, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\t\233\222%\274\260\320\31\331\326\10\204\276X>\263"..., 68, 880) = 68

mmap(NULL, 2036952, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 6, 0) = 0x7f3b505d0000

mprotect(0x7f3b505f5000, 1847296, PROT\_NONE) = 0

mmap(0x7f3b505f5000, 1540096, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x25000) = 0x7f3b505f5000

mmap(0x7f3b5076d000, 303104, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x19d000) = 0x7f3b5076d000

mmap(0x7f3b507b8000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 6, 0x1e7000) = 0x7f3b507b8000

mmap(0x7f3b507be000, 13528, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f3b507be000

close(6) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f3b505cd000

arch\_prctl(ARCH\_SET\_FS, 0x7f3b505cd740) = 0

mprotect(0x7f3b507b8000, 12288, PROT\_READ) = 0

mprotect(0x7f3b507df000, 4096, PROT\_READ) = 0

mprotect(0x555a34337000, 4096, PROT\_READ) = 0

mprotect(0x7f3b50825000, 4096, PROT\_READ) = 0

munmap(0x7f3b507e7000, 67057) = 0

set\_tid\_address(0x7f3b505cda10) = 7983

set\_robust\_list(0x7f3b505cda20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f3b507c9bf0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f3b507d73c0}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f3b507c9c90, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f3b507d73c0}, NULL, 8) = 0

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

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

brk(NULL) = 0x555a35385000

brk(0x555a353a6000) = 0x555a353a6000

write(1, "Threads 3\nEnter size of matrix. "..., 46) = 46

write(1, "> ", 2) = 2

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

read(0, "5 5\n", 1024) = 4

read(0, "1.0 2.0 0.5 5.0 8.5\n", 1024) = 20

read(0, "7.5 5.55 0.2 1 8\n", 1024) = 17

read(0, "1 5 6 8 9\n", 1024) = 10

read(0, "10 0.2 10 2 3\n", 1024) = 14

read(0, "6 6 6 6 6\n", 1024) = 10

write(1, "Matrix for erosion\n", 19) = 19

write(1, "\n", 1) = 1

write(1, "1.000000 2.000000 0.500000 5.000"..., 46) = 46

write(1, "7.500000 5.550000 0.200000 1.000"..., 46) = 46

write(1, "1.000000 5.000000 6.000000 8.000"..., 46) = 46

write(1, "10.000000 0.200000 10.000000 2.0"..., 48) = 48

write(1, "6.000000 6.000000 6.000000 6.000"..., 46) = 46

write(1, "\n", 1) = 1

write(1, "Matrix for dilatation\n", 22) = 22

write(1, "\n", 1) = 1

write(1, "1.000000 2.000000 0.500000 5.000"..., 46) = 46

write(1, "7.500000 5.550000 0.200000 1.000"..., 46) = 46

write(1, "1.000000 5.000000 6.000000 8.000"..., 46) = 46

write(1, "10.000000 0.200000 10.000000 2.0"..., 48) = 48

write(1, "6.000000 6.000000 6.000000 6.000"..., 46) = 46

write(1, "\n", 1) = 1

write(1, "Enter your K\n", 13) = 13

write(1, "> ", 2) = 2

read(0, "3\n", 1024) = 2

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f3b4fdcc000

mprotect(0x7f3b4fdcd000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[7990], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 7990

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f3b4f5cb000

mprotect(0x7f3b4f5cc000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[7991], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 7991

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f3b4edca000

mprotect(0x7f3b4edcb000, 8388608, PROT\_READ|PROT\_WRITE) = 0

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[7992], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 7992

futex(0x7f3b4f5ca9d0, FUTEX\_WAIT, 7992, NULL) = -1 EAGAIN (Ресурс временно недоступен)

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[7993], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 7993

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 7994

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[7995], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 7995

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[7996], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 7996

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[7997], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 7997

futex(0x7f3b505cc9d0, FUTEX\_WAIT, 7997, NULL) = -1 EAGAIN (Ресурс временно недоступен)

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 7998

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 7999

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8000

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8001

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8002

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8003

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8004

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8005

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8006

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8007

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8008

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8009

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8010

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8011

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8012

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8013

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8014

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8015

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[8016], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8016

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8017

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8018

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8019

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8020

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[8021], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8021

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8022

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8023

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8024

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8025

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8026

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[8027], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8027

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8028

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8029

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8030

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8031

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8032

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8033

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8034

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8035

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8036

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[8037], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8037

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8038

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8039

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8040

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8041

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8042

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8043

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8044

clone(child\_stack=0x7f3b4f5c9fb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4f5ca700, child\_tidptr=0x7f3b4f5ca9d0) = 8045

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8046

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8047

clone(child\_stack=0x7f3b4fdcafb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b4fdcb700, child\_tidptr=0x7f3b4fdcb9d0) = 8048

clone(child\_stack=0x7f3b505cbfb0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[0], tls=0x7f3b505cc700, child\_tidptr=0x7f3b505cc9d0) = 8049

write(1, "\n", 1) = 1

write(1, "ITERATION 3\n\n", 13) = 13

write(1, "Matrix for erosion\n", 19) = 19

write(1, "\n", 1) = 1

write(1, "0.500000 1.500000 0.700000 4.500"..., 46) = 46

write(1, "7.000000 5.050000 0.400000 0.500"..., 46) = 46

write(1, "0.500000 4.500000 5.500000 7.500"..., 46) = 46

write(1, "9.500000 0.400000 9.500000 1.500"..., 46) = 46

write(1, "5.500000 5.500000 5.500000 5.500"..., 46) = 46

write(1, "\n", 1) = 1

write(1, "Matrix for dilatation\n", 22) = 22

write(1, "\n", 1) = 1

write(1, "71.000000 72.000000 70.500000 75"..., 51) = 51

write(1, "77.500000 75.550003 70.199997 71"..., 51) = 51

write(1, "71.000000 75.000000 76.000000 78"..., 51) = 51

write(1, "80.000000 70.199997 80.000000 72"..., 51) = 51

write(1, "76.000000 76.000000 76.000000 76"..., 51) = 51

write(1, "\n", 1) = 1

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Недопустимая операция смещения)

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**:

Благодаря данной лабораторной работе я научился создавать многопоточные программы с помощью технологии POSIX Threads, а так же научился правильно определять моменты, в которых многопоточность может дать больше плюсов, из минусов могу сказать что 3 ядра и работа на виртуальной машине не очень хорошо сказываются на многопоточности.

**Список литературы**

1. https://habr.com/ru/post/142818/
2. https://habr.com/ru/post/326138/
3. http://rsdn.org/forum/media/669566
4. https://metanit.com/cpp/c/2.3.php