Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа: М8О-214Б-23

Студент: Маркелов Я.И.

Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 23.11.24

Москва, 2024

**Постановка задачи**

**Вариант 11.**

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

**Общий метод и алгоритм решения**

write: Используется для вывода строк в стандартный вывод (STDOUT\_FILENO) и стандартный вывод ошибок (STDERR\_FILENO). Например, для вывода сообщений об ошибках и использования программы.

pthread\_create: Используется для создания новых потоков. Каждый поток выполняет функцию threadFunction, которая обрабатывает часть матрицы.

pthread\_join: Используется для ожидания завершения всех созданных потоков. Это гарантирует, что программа не завершится до того, как все потоки завершат свою работу.

pthread\_mutex\_init: Используется для инициализации мьютекса, который используется для синхронизации доступа к общим данным между потоками.

pthread\_mutex\_lock и pthread\_mutex\_unlock: Используются для блокировки и разблокировки мьютекса, чтобы обеспечить безопасный доступ к общим данным между потоками.

pthread\_mutex\_destroy: Используется для уничтожения мьютекса после завершения работы с ним.

malloc и calloc: Используются для выделения памяти под матрицы и другие структуры данных.

free: Используется для освобождения памяти, выделенной ранее с помощью malloc и calloc

qsort: Используется для сортировки массива элементов, что необходимо для применения медианного фильтра.

srand и rand: Используются для инициализации генератора случайных чисел и генерации случайных значений для заполнения матрицы

timespec\_get: Используется для получения текущего времени с высокой точностью, что может быть полезно для измерения производительности.

strlen: Используется для вычисления длины строки перед её выводом с помощью write.

atoi: Используется для преобразования строковых аргументов командной строки в целые числа.

isdigit: Используется для проверки, является ли символ цифрой, что полезно для проверки входных данных.

Эта программа реализует параллельное применение медианного фильтра к матрице с использованием нескольких потоков. Вот подробное описание работы программы:

Структуры данных

ThreadData: Структура, содержащая данные, которые передаются каждому потоку. Включает в себя:

Указатель на матрицу (matrix).

Размер окна фильтра (window\_size).

Размер матрицы (matrix\_size).

Начальные и конечные индексы строк и столбцов для обработки (line\_start, line\_end, column\_start, column\_end).

Количество итераций (K).

Идентификатор потока (id).

pthread\_mutex\_t output\_mutex: Мьютекс для синхронизации доступа к общим данным между потоками.

Функции

compare: Функция сравнения для сортировки элементов массива.

write\_string: Функция для вывода строки в указанный файловый дескриптор.

is\_number: Функция для проверки, является ли строка числом.

apply\_median\_filter: Функция, которая применяет медианный фильтр к части матрицы, определенной в ThreadData.

threadFunction: Функция, которую выполняет каждый поток. Она создает локальную матрицу, применяет медианный фильтр к части матрицы, затем обновляет общую матрицу с помощью мьютекса.

printMatrix: Функция для вывода матрицы на экран.

millis: Функция для получения текущего времени в миллисекундах.

Основная логика программы

Проверка аргументов командной строки: Программа ожидает 4 аргумента: размер окна фильтра, размер матрицы, количество итераций и количество потоков. Если аргументы некорректны, программа выводит сообщение об ошибке и завершает работу.

Инициализация матрицы: Матрица заполняется случайными значениями или значениями, заданными в коде.

Обработка случая, когда размер окна равен размеру матрицы: Если размер окна фильтра равен размеру матрицы, программа сортирует все элементы матрицы и выводит медианное значение в центре матрицы, а остальные элементы заменяет нулями.

Инициализация мьютекса: Мьютекс инициализируется для синхронизации доступа к общим данным.

Создание потоков: Для каждого потока создается структура ThreadData, которая содержит данные о части матрицы, которую будет обрабатывать поток.

Запуск потоков: Программа создает потоки, каждый из которых выполняет функцию threadFunction.

Ожидание завершения потоков: Программа ожидает завершения всех потоков с помощью pthread\_join.

Обновление матрицы: После завершения работы всех потоков, программа обновляет общую матрицу, объединяя результаты работы всех потоков.

Вывод результата: Программа выводит обработанную матрицу на экран.

Освобождение памяти: Программа освобождает выделенную память и уничтожает мьютекс.

**Код программы**

**main.c**

***#include* <string.h>**

***#include* <unistd.h>**

***#include* <ctype.h>**

***#include* <stdlib.h>**

***#include* <time.h>**

***#include* <pthread.h>**

***#include* <stdio.h>**

***#include* <stdbool.h>**

***typedef struct* {**

***int* \*\*matrix;**

***int* window\_size;**

***int* matrix\_size;**

***int* line\_start;**

***int* line\_end;**

***int* column\_start;**

***int* column\_end;**

***int* K;**

***int* id;**

**} ThreadData;**

**pthread\_mutex\_t output\_mutex;**

***int* compare(*const void* \*a, *const void* \*b) {**

***return* (\*(*int* \*)a - \*(*int* \*)b);**

**}**

***void* write\_string(*int* fd, *const char* \*str) {**

**write(fd, str, strlen(str));**

**}**

**bool is\_number(*const char* \*str) {**

***if* (\*str == '\0') *return* false;**

***if* (\*str == '-') *return* false;**

***while* (\*str) {**

***if* (\*str == '.' || !isdigit(\*str))**

***return* false;**

**str++;**

**}**

***return* true;**

**}**

***void* apply\_median\_filter(*int* \*\*input, *int* \*\*output, ThreadData\* data){**

***int* line = data->line\_start;**

***int* col = data->column\_start;**

***for* (*int* l = line; l < data->line\_end; ++l){**

***for* (*int* i = col; i < data->column\_end; ++i){**

***//++q;***

***int* tmp[data->window\_size \* data->window\_size];**

***int* index = 0;**

***for* (*int* fx = 0; fx < 2; ++fx){**

***for* (*int* fy = 0; fy < 2; ++fy){**

***if* (fx == 0 && fy == 0){**

**tmp[index++] = input[l + fx][i + fy];**

**} *else if* (fx == 1 && fy == 1) {**

**tmp[index++] = input[l + fx][i + fy];**

**tmp[index++] = input[l - fx][i - fy];**

**tmp[index++] = input[l - fx][i + fy];**

**tmp[index++] = input[l + fx][i - fy];**

**} *else* {**

**tmp[index++] = input[l + fx][i + fy];**

**tmp[index++] = input[l - fx][i - fy];**

**}**

**}**

**}**

**qsort(tmp, index, *sizeof*(*int*), compare);**

**output[l][i] = tmp[index / 2];**

**}**

**}**

**}**

***void* \*threadFunction(*void* \*arg){**

**ThreadData \*data = (ThreadData\*)arg;**

***int* size = data->matrix\_size;**

***int* \*\*localOutput = (*int* \*\*)calloc(size, *sizeof*(*int* \*));**

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

**localOutput[i] = (*int* \*)calloc(size, *sizeof*(*int*));**

**}**

**apply\_median\_filter(data->matrix, localOutput, data);**

**pthread\_mutex\_lock(&output\_mutex);**

**data->matrix = localOutput;**

**pthread\_mutex\_unlock(&output\_mutex);**

***return* NULL;**

**}**

***void* printMatrix(*int* \*\*matrix, *int* width, *int* height) {**

**printf("Матрица:\n");**

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

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

**printf("%3d ", matrix[i][j]);**

**}**

**printf("\n");**

**}**

**printf("\n");**

**}**

**int64\_t millis(){**

***struct* timespec now;**

**timespec\_get(&now, TIME\_UTC);**

***return* ((int64\_t) now.tv\_sec) \* 1000 + ((int64\_t) now.tv\_nsec) / 1000000;**

**}**

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

***if* (argc != 5){**

**write\_string(STDOUT\_FILENO, "Usage: ./program <window\_size> <matrix\_size> <K> <num\_threads>");**

***return* 1;**

**}**

***if* (!is\_number(argv[1]) || !is\_number(argv[2]) || !is\_number(argv[3])){**

**write\_string(STDOUT\_FILENO, "Wrong input");**

***return* 2;**

**}**

***int* window\_size = atoi(argv[1]);**

***int* matrix\_size = atoi(argv[2]);**

***int* K = atoi(argv[3]);**

***int* num\_threads = atoi(argv[4]);**

***if* (num\_threads != window\_size){**

**num\_threads = 1;**

**}**

***if* (window\_size % 2 == 0 || matrix\_size % 2 == 0){**

**write\_string(STDOUT\_FILENO, "Must be odd");**

***return* 1;**

**}**

***if* (matrix\_size <= 1){**

**write\_string(STDOUT\_FILENO, "Must be >= 2");**

***return* 2;**

**}**

***if* (window\_size <= 1 || window\_size > matrix\_size){**

**write\_string(STDOUT\_FILENO, "Must be 3 <= window size <= matrix size");**

***return* 3;**

**}**

**srand(time(NULL));**

***int* \*\*matrix = (*int* \*\*)malloc(matrix\_size \* *sizeof*(*int* \*));**

***for* (*int* i = 0; i < matrix\_size; i++) {**

**matrix[i] = (*int* \*)malloc(matrix\_size \* *sizeof*(*int*));**

***for* (*int* j = 0; j < matrix\_size; j++) {**

**matrix[i][j] = i \* 5 + (j + 1);**

**}**

**}**

**printMatrix(matrix, matrix\_size, matrix\_size);**

***if* (window\_size == matrix\_size){**

***int* idx = 0;**

***int* data[matrix\_size \* matrix\_size];**

***for* (*int* i = 0; i < matrix\_size; ++i){**

***for* (*int* j = 0; j < matrix\_size; ++j){**

**data[idx++] = matrix[i][j];**

**}**

**}**

**qsort(data, idx, *sizeof*(*int*), compare);**

***int* output = data[(matrix\_size \* matrix\_size) / 2];**

***for* (*int* i = 0; i < matrix\_size; ++i){**

***for* (*int* j = 0; j < matrix\_size; ++j){**

***if* (i == matrix\_size / 2 && j == matrix\_size / 2){**

**printf("%d ", output);**

**} *else*{**

**printf("0 ");**

**}**

**}**

**printf("\n");**

**}**

***return* 0;**

**}**

**pthread\_mutex\_init(&output\_mutex, NULL);**

**pthread\_t threads[num\_threads];**

**ThreadData threadData[num\_threads];**

***for* (*int* i = 0; i < num\_threads; ++i){**

**threadData[i].matrix = matrix;**

**threadData[i].window\_size = window\_size;**

**threadData[i].matrix\_size = matrix\_size;**

**threadData[i].K = K;**

**threadData[i].column\_start = (matrix\_size - window\_size) / 2; *// 1***

**threadData[i].column\_end = matrix\_size - ((matrix\_size - window\_size) / 2); *// 8***

**threadData[i].line\_start = (window\_size / num\_threads == 1) ? ((matrix\_size - window\_size) / 2) + i : (matrix\_size - window\_size) / 2; *// 3***

**threadData[i].line\_end = (window\_size / num\_threads == 1) ? ((matrix\_size - window\_size) / 2) + i + 1 : matrix\_size - ((matrix\_size - window\_size) / 2);**

**threadData[i].id = i + 1;**

**}**

***// long long res = 0;***

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

***//long long q = millis();***

***for* (*int* i = 0; i < num\_threads; ++i){**

***if* (pthread\_create(&threads[i], NULL, threadFunction, &threadData[i])){**

**write\_string(STDERR\_FILENO, "Error thread creation");**

***return* 4;**

**}**

**}**

***// Ожидание завершения всех потоков***

***for* (*int* i = 0; i < num\_threads; i++) {**

***if* (pthread\_join(threads[i], NULL) != 0) {**

**fprintf(stderr, "Error waiting thread\n");**

***return* 1;**

**}**

**}**

***// long long q1 = millis();***

***// res += (q1 - q);***

***int* \*\*matrix1 = (*int* \*\*)calloc(matrix\_size, *sizeof*(*int* \*));**

***for* (*int* i = 0; i < matrix\_size; i++) {**

**matrix1[i] = (*int* \*)calloc(matrix\_size, *sizeof*(*int*));**

**}**

***for* (*int* i = 0; i < num\_threads; ++i){**

***for* (*int* l = threadData[i].line\_start; l <= threadData[i].column\_end; ++l){**

***for* (*int* j = threadData[i].column\_start; j < threadData[i].column\_end; ++j){**

**matrix1[l][j] = threadData[i].matrix[l][j];**

**}**

**}**

**}**

***for* (*int* i = 0; i < num\_threads; ++i){**

***for* (*int* j = 0; j < matrix\_size; ++j){**

**free(threadData[i].matrix[j]);**

**}**

**free(threadData[i].matrix);**

**threadData[i].matrix = matrix1;**

**}**

**}**

***//***

***// printf("TiMe: %lld\n", res);***

**printMatrix(threadData[num\_threads - 1].matrix, matrix\_size, matrix\_size);**

***for* (*int* i = 0; i < matrix\_size; i++) {**

**free(matrix[i]);**

**}**

**free(matrix);**

**pthread\_mutex\_destroy(&output\_mutex);**

***return* 0;**

**}**

**Протокол работы программы**

**Тесты**

**$ ./2 3 5 1 3**

**Матрица:**

**1 2 3 4 5**

**6 7 8 9 10**

**11 12 13 14 15**

**16 17 18 19 20**

**21 22 23 24 25**

**Матрица:**

**0 0 0 0 0**

**0 7 8 9 0**

**0 12 13 14 0**

**0 17 18 19 0**

**0 0 0 0 0**

**Strace:**

$ strace -f ./2 3 5 1 3

execve("./2", ["./2", "3", "5", "1", "3"], 0x7ffc8b472348 /\* 64 vars \*/) = 0

brk(NULL) = 0x5642e9c9c000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffdffaa11b0) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2279df9000

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) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=82523, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 82523, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f2279de4000

close(3) = 0

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\0P\237\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\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(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 68, 896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, AT\_EMPTY\_PATH) = 0

pread64(3, "\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

mmap(NULL, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f2279bbb000

mprotect(0x7f2279be3000, 2023424, PROT\_NONE) = 0

mmap(0x7f2279be3000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f2279be3000

mmap(0x7f2279d78000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f2279d78000

mmap(0x7f2279dd1000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7f2279dd1000

mmap(0x7f2279dd7000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f2279dd7000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f2279bb8000

arch\_prctl(ARCH\_SET\_FS, 0x7f2279bb8740) = 0

set\_tid\_address(0x7f2279bb8a10) = 6414

set\_robust\_list(0x7f2279bb8a20, 24) = 0

rseq(0x7f2279bb90e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f2279dd1000, 16384, PROT\_READ) = 0

mprotect(0x5642e841a000, 4096, PROT\_READ) = 0

mprotect(0x7f2279e33000, 8192, PROT\_READ) = 0

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

munmap(0x7f2279de4000, 82523) = 0

getrandom("\xa3\x31\x3c\xb3\x6b\xaf\x36\x29", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x5642e9c9c000

brk(0x5642e9cbd000) = 0x5642e9cbd000

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

write(1, "\320\234\320\260\321\202\321\200\320\270\321\206\320\260:\n", 16Матрица:

) = 16

write(1, " 1 2 3 4 5 \n", 21 1 2 3 4 5

) = 21

write(1, " 6 7 8 9 10 \n", 21 6 7 8 9 10

) = 21

write(1, " 11 12 13 14 15 \n", 21 11 12 13 14 15

) = 21

write(1, " 16 17 18 19 20 \n", 21 16 17 18 19 20

) = 21

write(1, " 21 22 23 24 25 \n", 21 21 22 23 24 25

) = 21

write(1, "\n", 1

) = 1

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f2279c4c870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f2279bfd520}, NULL, 8) = 0

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f22793b7000

mprotect(0x7f22793b8000, 8388608, PROT\_READ|PROT\_WRITE) = 0

rt\_sigprocmask(SIG\_BLOCK, ~[], [QUIT], 8) = 0

**clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f2279bb7910, parent\_tid=0x7f2279bb7910, exit\_signal=0, stack=0x7f22793b7000, stack\_size=0x7fff00, tls=0x7f2279bb7640}strace: Process 6415 attached**

=> {parent\_tid=[6415]}, 88) = 6415

[pid 6415] rseq(0x7f2279bb7fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 6414] rt\_sigprocmask(SIG\_SETMASK, [QUIT], <unfinished ...>

[pid 6415] <... rseq resumed>) = 0

[pid 6414] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 6415] set\_robust\_list(0x7f2279bb7920, 24 <unfinished ...>

[pid 6414] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 6415] <... set\_robust\_list resumed>) = 0

[pid 6414] <... mmap resumed>) = 0x7f2278bb6000

[pid 6415] rt\_sigprocmask(SIG\_SETMASK, [QUIT], <unfinished ...>

[pid 6414] mprotect(0x7f2278bb7000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 6415] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 6414] <... mprotect resumed>) = 0

[pid 6415] mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0 <unfinished ...>

[pid 6414] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 6415] <... mmap resumed>) = 0x7f2270bb6000

[pid 6414] <... rt\_sigprocmask resumed>[QUIT], 8) = 0

[pid 6415] munmap(0x7f2270bb6000, 54829056 <unfinished ...>

[pid 6414] **clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f22793b6910, parent\_tid=0x7f22793b6910, exit\_signal=0, stack=0x7f2278bb6000, stack\_size=0x7fff00, tls=0x7f22793b6640} <unfinished ...>**

[pid 6415] <... munmap resumed>) = 0

[pid 6415] munmap(0x7f2278000000, 12279808strace: Process 6416 attached

) = 0

[pid 6414] <... clone3 resumed> => {parent\_tid=[6416]}, 88) = 6416

[pid 6416] rseq(0x7f22793b6fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 6415] mprotect(0x7f2274000000, 135168, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 6414] rt\_sigprocmask(SIG\_SETMASK, [QUIT], <unfinished ...>

[pid 6416] <... rseq resumed>) = 0

[pid 6414] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 6415] <... mprotect resumed>) = 0

[pid 6414] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 6416] set\_robust\_list(0x7f22793b6920, 24 <unfinished ...>

[pid 6414] <... mmap resumed>) = 0x7f22783b5000

[pid 6416] <... set\_robust\_list resumed>) = 0

[pid 6414] mprotect(0x7f22783b6000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 6415] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 6414] <... mprotect resumed>) = 0

[pid 6416] rt\_sigprocmask(SIG\_SETMASK, [QUIT], <unfinished ...>

[pid 6414] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 6415] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 6414] <... rt\_sigprocmask resumed>[QUIT], 8) = 0

[pid 6416] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 6414] **clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f2278bb5910, parent\_tid=0x7f2278bb5910, exit\_signal=0, stack=0x7f22783b5000, stack\_size=0x7fff00, tls=0x7f2278bb5640} <unfinished ...>**

[pid 6415] madvise(0x7f22793b7000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 6416] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 6415] <... madvise resumed>) = 0

[pid 6416] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 6415] exit(0 <unfinished ...>

[pid 6416] madvise(0x7f2278bb6000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 6414] <... clone3 resumed> => {parent\_tid=[6417]}, 88) = 6417

[pid 6415] <... exit resumed>) = ?

[pid 6414] rt\_sigprocmask(SIG\_SETMASK, [QUIT], <unfinished ...>

[pid 6416] <... madvise resumed>) = 0

[pid 6414] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 6415] +++ exited with 0 +++

[pid 6416] exit(0 <unfinished ...>

[pid 6414] futex(0x7f22793b6910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 6416, NULL, FUTEX\_BITSET\_MATCH\_ANYstrace: Process 6417 attached

<unfinished ...>

[pid 6416] <... exit resumed>) = ?

[pid 6414] <... futex resumed>) = 0

[pid 6417] rseq(0x7f2278bb5fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 6416] +++ exited with 0 +++

[pid 6414] futex(0x7f2278bb5910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 6417, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 6417] <... rseq resumed>) = 0

[pid 6417] set\_robust\_list(0x7f2278bb5920, 24) = 0

[pid 6417] rt\_sigprocmask(SIG\_SETMASK, [QUIT], NULL, 8) = 0

[pid 6417] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 6417] madvise(0x7f22783b5000, 8368128, MADV\_DONTNEED) = 0

[pid 6417] exit(0) = ?

[pid 6414] <... futex resumed>) = 0

[pid 6417] +++ exited with 0 +++

write(1, "\320\234\320\260\321\202\321\200\320\270\321\206\320\260:\n", 16Матрица:

) = 16

write(1, " 0 0 0 0 0 \n", 21 0 0 0 0 0

) = 21

write(1, " 0 7 8 9 0 \n", 21 0 7 8 9 0

) = 21

write(1, " 0 12 13 14 0 \n", 21 0 12 13 14 0

) = 21

write(1, " 0 17 18 19 0 \n", 21 0 17 18 19 0

) = 21

write(1, " 0 0 0 0 0 \n", 21 0 0 0 0 0

) = 21

write(1, "\n", 1

) = 1

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В ходе выполнения данной лабораторной работы я получил ценный опыт работы с многопоточностью и синхронизацией в программах на языке C.