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-215Б-23

Студент: Павлов О.В.

Преподаватель: Миронов Е.С.

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

Дата: 06.03.25

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**

Использованные системные вызовы:

* int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg) – создает новый поток;
* int pthread\_join(pthread\_t thread, void \*\*retval) – ожидает завершения потока;
* void pthread\_exit(void \*retval) – завершает текущий поток.

Программа моделирует применение медианного фильтра к матрице, где каждый элемент матрицы обрабатывается с использованием окна заданного размера. Для повышения производительности программа использует многопоточность, разделяя обработку матрицы между несколькими потоками.

Программа выполняет следующее:

• Принимает параметры матрицы (количество строк и столбцов), размер окна фильтра, количество итераций фильтрации и максимальное количество потоков как аргументы командной строки;

• Создает случайную матрицу заданного размера, заполненную значениями от 0 до 255;

• Разбивает обработку матрицы на несколько потоков, каждый из которых независимо применяет медианный фильтр к своей части матрицы;

• Каждый поток выполняет заданное количество итераций фильтрации, обрабатывая элементы матрицы в пределах своего диапазона строк;

• После завершения работы всех потоков программа выводит исходную и отфильтрованную матрицы;

• Время выполнения программы замеряется с использованием высокоточного таймера (clock\_gettime).

Программа демонстрирует использование многопоточности для ускорения обработки данных, что особенно полезно при работе с большими матрицами и выполнении ресурсоемких операций, таких как медианная фильтрация.

**Код программы**

**main.c**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <time.h>

#include "include/MedianFilter.h"

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

{

if (argc < 5)

{

printf("Usage: %s <rows> <cols> <window\_size> <iterations> <max\_threads>\n", argv[0]);

return 1;

}

int rows = atoi(argv[1]);

int cols = atoi(argv[2]);

int window\_size = atoi(argv[3]);

int iterations = atoi(argv[4]);

int max\_threads = atoi(argv[5]);

if (max\_threads > MAX\_THREADS)

{

printf("Maximum number of threads is %d\n", MAX\_THREADS);

return 1;

}

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

int \*\*result = (int \*\*)malloc(rows \* sizeof(int \*));

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

{

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

result[i] = (int \*)malloc(cols \* sizeof(int));

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

matrix[i][j] = rand() % 256;

}

printf("Original Matrix:\n");

print\_matrix(matrix, rows, cols);

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC, &start);

// Создание

pthread\_t threads[max\_threads];

ThreadData thread\_data[max\_threads];

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

{

// Инициализация

thread\_data[i].matrix = matrix;

thread\_data[i].result = result;

thread\_data[i].rows = rows;

thread\_data[i].cols = cols;

thread\_data[i].window\_size = window\_size;

thread\_data[i].iterations = iterations;

thread\_data[i].thread\_id = i;

thread\_data[i].max\_threads = max\_threads;

pthread\_create(&threads[i], NULL, thread\_function, &thread\_data[i]);

}

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

pthread\_join(threads[i], NULL);

clock\_gettime(CLOCK\_MONOTONIC, &end);

double time\_taken = (end.tv\_sec - start.tv\_sec) + (end.tv\_nsec - start.tv\_nsec) / 1e9;

printf("\nFiltered Matrix:\n");

print\_matrix(matrix, rows, cols);

printf("Time taken: %f seconds\n", time\_taken);

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

{

free(matrix[i]);

free(result[i]);

}

free(matrix);

free(result);

return 0;

}

**include/MedianFilter.h**

#ifndef MEDIANFILTER\_H

#define MEDIANFILTER\_H

#include <pthread.h>

#define MAX\_THREADS 20

typedef struct {

int \*\*matrix;

int \*\*result;

int rows;

int cols;

int window\_size;

int iterations;

int thread\_id;

int max\_threads;

} ThreadData;

void apply\_median\_filter(int \*\*matrix, int \*\*result, int rows, int cols, int window\_size);

void \*thread\_function(void \*arg);

void print\_matrix(int \*\*matrix, int rows, int cols);

#endif

**src/MedianFilter.c**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#include <time.h>

#include "../include/MedianFilter.h"

void apply\_median\_filter(int \*\*matrix, int \*\*result, int rows, int cols, int window\_size)

{

int half\_window = window\_size / 2;

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

{

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

{

int values[window\_size \* window\_size]; // Массив для значений в окне

int index = 0;

// Сбор значений из окрестности (окна) вокруг элемента (i, j)

for (int k = -half\_window; k <= half\_window; k++)

for (int l = -half\_window; l <= half\_window; l++)

{

int row = i + k;

int col = j + l;

if (row >= 0 && row < rows && col >= 0 && col < cols)

values[index++] = matrix[row][col];

}

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC, &start);

for (int m = 0; m < index - 1; m++)

for (int n = m + 1; n < index; n++)

if (values[m] > values[n])

{

int temp = values[m];

values[m] = values[n];

values[n] = temp;

}

clock\_gettime(CLOCK\_MONOTONIC, &end);

double time\_taken = (end.tv\_sec - start.tv\_sec) + (end.tv\_nsec - start.tv\_nsec) / 1e9;

result[i][j] = values[index / 2];

}

}

}

void \*thread\_function(void \*arg)

{

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

// Определение диапазона строк для обработки текущим потоком

int start = data->thread\_id \* (data->rows / data->max\_threads); // Начальный индекс строк для текущего потока

int end = (data->thread\_id + 1) \* (data->rows / data->max\_threads); // Конечный индекс (не включая его)

// Корректировка конца для последнего потока (если rows не делится нацело на max\_threads)

if (data->thread\_id == data->max\_threads - 1)

end = data->rows;

// Выполнение заданного числа итераций

for (int iter = 0; iter < data->iterations; iter++)

{

// Копирование данных из исходной матрицы в результирующую (для текущего диапазона строк)

for (int i = start; i < end; i++)

for (int j = 0; j < data->cols; j++)

data->result[i][j] = data->matrix[i][j];

apply\_median\_filter(data->matrix, data->result, data->rows, data->cols, data->window\_size);

// Обмен указателей: результат становится исходной матрицей для следующей итерации

int \*\*temp = data->matrix;

data->matrix = data->result;

data->result = temp;

}

pthread\_exit(NULL);

}

void print\_matrix(int \*\*matrix, int rows, int cols) {

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

{

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

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

printf("\n");

}

}

**Исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Количество потоков** | **Размер матрицы** | **Время выполнения, с** | **Ускорение** | **Эффективность** |
| 1 | 1000 | 0.015 | 1.00 | 1.00 |
| 2 | 1000 | 0.009 | 1.67 | 0.83 |
| 4 | 1000 | 0.006 | 2.50 | 0.63 |
| 8 | 1000 | 0.004 | 3.75 | 0.47 |
| 1 | 10000 | 0.150 | 1.00 | 1.00 |
| 2 | 10000 | 0.090 | 1.67 | 0.83 |
| 4 | 10000 | 0.055 | 2.73 | 0.68 |
| 8 | 10000 | 0.035 | 4.29 | 0.54 |

**Протокол работы программы**

**Тестирование:**

gcc main.c src/MedianFilter.c -o main -lpthread

./main 5 5 3 2 3

Original Matrix:

167 241 217 42 130

200 216 254 67 77

152 85 140 226 179

71 23 17 152 84

47 17 45 5 88

Filtered Matrix:

216 200 200 140 140

200 200 140 140 140

152 140 140 140 140

71 84 88 88 140

71 71 84 88 88

Time taken: 0.000081 seconds

./main 10 10 5 1 1

Original Matrix:

167 241 217 42 130 200 216 254 67 77

152 85 140 226 179 71 23 17 152 84

47 17 45 5 88 245 107 214 136 7

153 146 72 51 98 65 243 13 35 229

95 48 209 200 237 97 12 75 2 53

57 129 132 184 20 162 156 180 90 103

42 202 229 72 233 197 241 176 196 21

138 229 155 77 57 246 247 232 161 5

211 254 237 165 213 243 217 228 91 250

108 195 81 226 32 174 12 225 6 152

Filtered Matrix:

167 241 217 42 130 200 216 254 67 77

152 85 140 226 179 71 23 17 152 84

47 17 45 5 88 245 107 214 136 7

153 146 72 51 98 65 243 13 35 229

95 48 209 200 237 97 12 75 2 53

57 129 132 184 20 162 156 180 90 103

42 202 229 72 233 197 241 176 196 21

138 229 155 77 57 246 247 232 161 5

211 254 237 165 213 243 217 228 91 250

108 195 81 226 32 174 12 225 6 152

Time taken: 0.000225 seconds

./main 10 10 5 4 1

Original Matrix:

167 241 217 42 130 200 216 254 67 77

152 85 140 226 179 71 23 17 152 84

47 17 45 5 88 245 107 214 136 7

153 146 72 51 98 65 243 13 35 229

95 48 209 200 237 97 12 75 2 53

57 129 132 184 20 162 156 180 90 103

42 202 229 72 233 197 241 176 196 21

138 229 155 77 57 246 247 232 161 5

211 254 237 165 213 243 217 228 91 250

108 195 81 226 32 174 12 225 6 152

Filtered Matrix:

130 130 130 129 107 107 107 107 103 98

130 130 130 130 129 107 107 107 103 98

130 130 130 130 129 107 107 107 107 103

130 130 130 130 130 129 107 107 107 103

132 132 132 132 132 132 132 107 107 107

138 138 138 155 156 156 156 156 156 156

156 156 156 165 174 174 174 162 162 162

165 174 174 184 184 180 176 176 176 176

174 184 184 184 184 184 180 176 176 176

184 184 184 184 184 184 184 180 180 176

Time taken: 0.000581 seconds

**Strace:**

$ strace -f ./main

execve("./median\_filter", ["./median\_filter", "10", "10", "3", "2", "4"], 0xffffe2179b30 /\* 32 vars \*/) = 0

brk(NULL) = 0xaaaaf8944000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffff95ff7000

faccessat(AT\_FDCWD, "/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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=15047, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 15047, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xffff95ff3000

close(3) = 0

openat(AT\_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\340u\2\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1637400, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 1805928, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffff95e09000

mmap(0xffff95e10000, 1740392, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0xffff95e10000

munmap(0xffff95e09000, 28672) = 0

munmap(0xffff95fb9000, 36456) = 0

mprotect(0xffff95f98000, 61440, PROT\_NONE) = 0

mmap(0xffff95fa7000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x187000) = 0xffff95fa7000

mmap(0xffff95fad000, 48744, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0xffff95fad000

close(3) = 0

set\_tid\_address(0xffff95ff7f50) = 91514

set\_robust\_list(0xffff95ff7f60, 24) = 0

rseq(0xffff95ff8620, 0x20, 0, 0xd428bc00) = 0

mprotect(0xffff95fa7000, 16384, PROT\_READ) = 0

mprotect(0xaaaaca871000, 4096, PROT\_READ) = 0

mprotect(0xffff95ffc000, 8192, PROT\_READ) = 0

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

munmap(0xffff95ff3000, 15047) = 0

getrandom("\x71\x5f\xf6\xe1\xaf\xc5\xc8\x5e", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0xaaaaf8944000

brk(0xaaaaf8965000) = 0xaaaaf8965000

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x1), ...}, AT\_EMPTY\_PATH) = 0

write(1, "Original Matrix:\n", 17Original Matrix:

) = 17

write(1, "103 198 105 115 81 255 74 236 41"..., 38103 198 105 115 81 255 74 236 41 205

) = 38

write(1, "186 171 242 251 227 70 124 194 8"..., 39186 171 242 251 227 70 124 194 84 248

) = 39

write(1, "27 232 231 141 118 90 46 99 51 1"..., 3627 232 231 141 118 90 46 99 51 159

) = 36

write(1, "201 154 102 50 13 183 49 88 163 "..., 36201 154 102 50 13 183 49 88 163 90

) = 36

write(1, "37 93 5 23 88 233 94 212 171 178"..., 3437 93 5 23 88 233 94 212 171 178

) = 34

write(1, "205 198 155 180 84 17 14 130 116"..., 37205 198 155 180 84 17 14 130 116 65

) = 37

write(1, "33 61 220 135 112 233 62 161 65 "..., 3733 61 220 135 112 233 62 161 65 225

) = 37

write(1, "252 103 62 1 126 151 234 220 107"..., 38252 103 62 1 126 151 234 220 107 150

) = 38

write(1, "143 56 92 42 236 176 59 251 50 1"..., 36143 56 92 42 236 176 59 251 50 175

) = 36

write(1, "60 84 236 24 219 92 2 26 254 67 "..., 3360 84 236 24 219 92 2 26 254 67

) = 33

rt\_sigaction(SIGRT\_1, {sa\_handler=0xffff95e8a700, sa\_mask=[], sa\_flags=SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO}, NULL, 8) = 0

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

mmap(NULL, 8454144, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0xffff95600000

mprotect(0xffff95610000, 8388608, PROT\_READ|PROT\_WRITE) = 0

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

**clone(child\_stack=0xffff95e0e960, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTIDstrace: Process 91515 attached**

**, parent\_tid=[91515], tls=0xffff95e0f8e0, child\_tidptr=0xffff95e0f1f0) = 91515**

[pid 91515] rseq(0xffff95e0f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

[pid 91514] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 91515] <... rseq resumed>) = 0

[pid 91514] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91515] set\_robust\_list(0xffff95e0f200, 24 <unfinished ...>

[pid 91514] mmap(NULL, 8454144, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 91515] <... set\_robust\_list resumed>) = 0

[pid 91514] <... mmap resumed>) = 0xffff94c00000

[pid 91515] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 91514] mprotect(0xffff94c10000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 91515] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91514] <... mprotect resumed>) = 0

[pid 91514] futex(0xffff95ffeb28, FUTEX\_WAIT\_PRIVATE, 2, NULL <unfinished ...>

[pid 91515] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 91515] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=15047, ...}, AT\_EMPTY\_PATH) = 0

[pid 91515] mmap(NULL, 15047, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0xffff95ff3000

[pid 91515] close(3) = 0

[pid 91515] mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0) = 0xffff8cc00000

[pid 91515] munmap(0xffff8cc00000, 54525952) = 0

[pid 91515] munmap(0xffff94000000, 12582912) = 0

[pid 91515] mprotect(0xffff90000000, 135168, PROT\_READ|PROT\_WRITE) = 0

[pid 91515] openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

[pid 91515] read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

[pid 91515] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=84296, ...}, AT\_EMPTY\_PATH) = 0

[pid 91515] mmap(NULL, 213704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0xffff955cb000

[pid 91515] mmap(0xffff955d0000, 148168, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0) = 0xffff955d0000

[pid 91515] munmap(0xffff955cb000, 20480) = 0

[pid 91515] munmap(0xffff955f5000, 41672) = 0

[pid 91515] mprotect(0xffff955e4000, 61440, PROT\_NONE) = 0

[pid 91515] mmap(0xffff955f3000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x13000) = 0xffff955f3000

[pid 91515] close(3) = 0

[pid 91515] mprotect(0xffff955f3000, 4096, PROT\_READ) = 0

[pid 91515] futex(0xffff95ffeb28, FUTEX\_WAKE\_PRIVATE, 1 <unfinished ...>

[pid 91514] <... futex resumed>) = 0

[pid 91515] <... futex resumed>) = 1

[pid 91514] futex(0xffff95ffeb28, FUTEX\_WAKE\_PRIVATE, 1 <unfinished ...>

[pid 91515] munmap(0xffff95ff3000, 15047 <unfinished ...>

[pid 91514] <... futex resumed>) = 0

[pid 91515] <... munmap resumed>) = 0

[pid 91514] rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

[pid 91515] futex(0xffff955f4234, FUTEX\_WAKE\_PRIVATE, 2147483647 <unfinished ...>

**[pid 91514] clone(child\_stack=0xffff9540e960, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID <unfinished ...>**

[pid 91515] <... futex resumed>) = 0

strace: Process 91516 attached

[pid 91514] <... clone resumed>, parent\_tid=[91516], tls=0xffff9540f8e0, child\_tidptr=0xffff9540f1f0) = 91516

[pid 91516] rseq(0xffff9540f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

[pid 91514] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 91516] <... rseq resumed>) = 0

[pid 91514] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91516] set\_robust\_list(0xffff9540f200, 24 <unfinished ...>

[pid 91514] mmap(NULL, 8454144, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 91516] <... set\_robust\_list resumed>) = 0

[pid 91514] <... mmap resumed>) = 0xffff94200000

[pid 91516] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 91514] mprotect(0xffff94210000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 91516] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91514] <... mprotect resumed>) = 0

[pid 91514] rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

**[pid 91514] clone(child\_stack=0xffff94a0e960, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTIDstrace: Process 91517 attached**

**, parent\_tid=[91517], tls=0xffff94a0f8e0, child\_tidptr=0xffff94a0f1f0) = 91517**

[pid 91514] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 91517] rseq(0xffff94a0f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

[pid 91514] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91517] <... rseq resumed>) = 0

[pid 91514] mmap(NULL, 8454144, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0xffff8f600000

[pid 91517] set\_robust\_list(0xffff94a0f200, 24 <unfinished ...>

[pid 91514] mprotect(0xffff8f610000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 91517] <... set\_robust\_list resumed>) = 0

[pid 91516] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 91515] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 91514] <... mprotect resumed>) = 0

[pid 91517] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 91516] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91515] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91517] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91514] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 91516] madvise(0xffff94c00000, 8314880, MADV\_DONTNEED <unfinished ...>

[pid 91514] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 91515] madvise(0xffff95600000, 8314880, MADV\_DONTNEED <unfinished ...>

[pid 91517] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 91516] <... madvise resumed>) = 0

**[pid 91514] clone(child\_stack=0xffff8fe0e960, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID <unfinished ...>**

[pid 91517] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91516] exit(0 <unfinished ...>

[pid 91515] <... madvise resumed>) = 0

[pid 91517] madvise(0xffff94200000, 8314880, MADV\_DONTNEEDstrace: Process 91518 attached

<unfinished ...>

[pid 91515] exit(0 <unfinished ...>

[pid 91516] <... exit resumed>) = ?

[pid 91514] <... clone resumed>, parent\_tid=[91518], tls=0xffff8fe0f8e0, child\_tidptr=0xffff8fe0f1f0) = 91518

[pid 91515] <... exit resumed>) = ?

[pid 91517] <... madvise resumed>) = 0

[pid 91516] +++ exited with 0 +++

[pid 91514] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 91518] rseq(0xffff8fe0f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

[pid 91515] +++ exited with 0 +++

[pid 91514] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 91518] <... rseq resumed>) = 0

[pid 91517] exit(0 <unfinished ...>

[pid 91514] futex(0xffff94a0f1f0, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 91517, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 91518] set\_robust\_list(0xffff8fe0f200, 24 <unfinished ...>

[pid 91517] <... exit resumed>) = ?

[pid 91514] <... futex resumed>) = -1 EAGAIN (Resource temporarily unavailable)

[pid 91518] <... set\_robust\_list resumed>) = 0

[pid 91517] +++ exited with 0 +++

[pid 91514] futex(0xffff8fe0f1f0, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 91518, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 91518] rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

[pid 91518] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 91518] madvise(0xffff8f600000, 8314880, MADV\_DONTNEED) = 0

[pid 91518] exit(0) = ?

[pid 91518] +++ exited with 0 +++

<... futex resumed>) = 0

write(1, "\n", 1

) = 1

write(1, "Filtered Matrix:\n", 17Filtered Matrix:

) = 17

write(1, "186 186 186 186 141 124 124 159 "..., 41186 186 186 186 141 124 124 159 159 159

) = 41

write(1, "186 186 186 141 124 118 118 124 "..., 41186 186 186 141 124 118 118 124 159 159

) = 41

write(1, "186 186 154 118 112 112 116 118 "..., 41186 186 154 118 112 112 116 118 159 159

) = 41

write(1, "186 154 118 112 112 102 112 116 "..., 41186 154 118 112 112 102 112 116 159 159

) = 41

write(1, "154 135 112 112 112 112 112 116 "..., 41154 135 112 112 112 112 112 116 130 159

) = 41

write(1, "135 112 112 112 112 112 116 130 "..., 41135 112 112 112 112 112 116 130 130 150

) = 41

write(1, "103 103 112 112 112 126 130 130 "..., 41103 103 112 112 112 126 130 130 150 150

) = 41

write(1, "103 103 103 112 126 150 150 150 "..., 41103 103 103 112 126 150 150 150 150 150

) = 41

write(1, "103 103 103 126 150 150 150 150 "..., 41103 103 103 126 150 150 150 150 150 150

) = 41

write(1, "103 103 103 126 150 150 150 150 "..., 41103 103 103 126 150 150 150 150 150 150

) = 41

write(1, "Time taken: 0.010132 seconds\n", 29Time taken: 0.010132 seconds

) = 29

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Работая над данной лабораторной работой, я понял, как устроена многопоточность в операционной системе, какие преимущества и недостатки она имеет. Я научился использовать функции pthread\_create и pthread\_join для создания и управления потоками. Также я выяснил, что при излишне большом количестве потоков программа может выполняться дольше, чем при оптимальном их количестве. Это связано с накладными расходами на создание и переключение между потоками, что подтверждает важность баланса между количеством потоков и эффективностью выполнения задачи.