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: Миронов Д. А.

Преподаватель: Бахарев В. Д.

Оценка: _____

Дата: 28.10.24

Москва, 2024

Постановка задачи

Вариант 20.

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы. Дан массив координат (x, y, z). Необходимо найти три точки, которые образуют треугольник максимальной площади.

Общий метод и алгоритм решения

Использованные системные вызовы:

- pthread_t threads[num_threads]; - объявляет массив потоков.
- pthread_create(&threads[i], NULL, find_max_area, &thread_data[i]); - создаёт новый поток.
- pthread_join(threads[i], NULL); - ожидание завершения конкретного потока.

Решение:

1. Объявляю функцию, которая будет выполняться внутри потока.
2. Считываю количество точек и количество потоков.
3. Заполняю массив точек случайным образом, используя функцию rand().
4. Инициализирую массив потоков, массив данных потоков и переменную для максимальной площади.
5. В цикле заполняю данные потока и запускаю поток.
6. В цикле жду завершения каждого потока и сравнивыаю его ответ с текущим.
7. Вывожу ответ.

Код программы

server.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <math.h>

int toInt(const char* argv) {
    int res = 0, i = 0;
    while (argv[i] != '\0') {
        res = res * 10 + argv[i] - '0';
        i++;
    }
    return res;
}

typedef struct {
    double x, y, z;
} Point;

typedef struct {
    Point *points;
    int numThreads;
    int end;
    int ind;
    double max_area;
```

```
} ThreadData;

double triangle_area(const Point a, const Point b, const Point c) {
    double ABx = b.x - a.x;
    double ABy = b.y - a.y;
    double ABz = b.z - a.z;

    // Вектор AC
    double ACx = c.x - a.x;
    double ACy = c.y - a.y;
    double ACz = c.z - a.z;

    // Векторное произведение AB и AC
    double cross_x = ABy * ACz - ABz * ACy;
    double cross_y = ABz * ACx - ABx * ACz;
    double cross_z = ABx * ACy - ABy * ACx;

    // Площадь треугольника равна половине длины вектора, полученного от векторного
произведения
    return 0.5 * sqrt(cross_x * cross_x + cross_y * cross_y + cross_z * cross_z);
}

void* find_max_area(void* arg) {
    ThreadData *data = (ThreadData*)arg;
    double max_area = 0.0;

    for (int i = data->ind; i < data->end; i += data->numThreads) {
        for (int j = i + 1; j < data->end; j++) {
            for (int k = j + 1; k < data->end; k++) {
                double area = triangle_area(data->points[i], data->points[j], data-
>points[k]);
                //printf("i: %d | j: %d | k: %d  ||| area: %lf\n", i, j, k, area);
                if (area > max_area) {
                    max_area = area;
                }
            }
        }
    }
    data->max_area = max_area;
    return NULL;
}

int main(int argc, char *argv[]) {
    if (argc < 3) {
        printf("Usage: <number of points> <number of threads>\n");
        return -1;
    }

    const int num_points = toInt(argv[1]);
    const int num_threads = toInt(argv[2]);
    Point *points = malloc(num_points * sizeof(Point));

    for (int i = 0; i < num_points; i++) {
        points[i].x = rand() % 100;
        points[i].y = rand() % 100;
        points[i].z = rand() % 100;
    }

    pthread_t threads[num_threads];
    ThreadData thread_data[num_threads];
    double max_area = 0.0;

    // Creating threads
    for (int i = 0; i < num_threads; i++) {
        thread_data[i].points = points;
        thread_data[i].numThreads = num_threads;
        thread_data[i].end = num_points;
```

```
        thread_data[i].ind = i;
        pthread_create(&threads[i], NULL, find_max_area, &thread_data[i]);
        //printf("Thread %d\n", i + 1);
    }

    for (int i = 0; i < num_threads; i++) {
        pthread_join(threads[i], NULL);
        if (thread_data[i].max_area > max_area) {
            max_area = thread_data[i].max_area;
        }
    }

    printf("Max area: %f\n", max_area);
    free(points);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
$ cc -o test main.c -lm -pthread
$ ./test 256 2
Max area: 6735.965892
Num_Points = 256
```

Число потоков	Время исполнения (мс)	Ускорение	Эффективность
1	23	1	1
2	13	1,76	0,88
3	11	2,09	0,69
4	10	2,3	0,57
5	7	3,28	0,65
6	6	3,83	0,63

```
Num_Points = 1024
```

Число потоков	Время исполнения (мс)	Ускорение	Эффективность
1	1321	1	1
2	715	1,84	0,92
3	447	2,95	0,98
4	361	3,65	0,91
5	339	3,89	0,77
6	283	4,66	0,77

```
Num_Points = 3000
```

Число потоков	Время исполнения (мс)	Ускорение	Эффективность
1	36126	1	1
2	20482	1,76	0,88
3	15362	2,35	0,78
4	9294	3,88	0,97
5	8413	4,29	0,85
6	6905	5,23	0,87

Strace:

```
$ strace ./test 1024 24

execve("./test", ["./test", "1024", "24"], 0x7fff65ffebf0 /* 28 vars */) = 0

brk(NULL)                               = 0x556ab99b8000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0417aa9000

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=20335, ...}) = 0

mmap(NULL, 20335, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0417aa4000

close(3)                                = 0

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\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0

mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f04179bb000

mmap(0x7f04179cb000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x10000) = 0x7f04179cb000

mmap(0x7f0417a4a000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8f000) =
0x7f0417a4a000

mmap(0x7f0417aa2000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0xe7000) = 0x7f0417aa2000

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\0\220\243\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

fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f04177a9000

mmap(0x7f04177d1000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7f04177d1000

mmap(0x7f0417959000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) =
0x7f0417959000

mmap(0x7f04179a8000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1fe000) = 0x7f04179a8000

mmap(0x7f04179ae000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1,
0) = 0x7f04179ae000
```

```
close(3)                                = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f04177a6000

arch_prctl(ARCH_SET_FS, 0x7f04177a6740) = 0

set_tid_address(0x7f04177a6a10)         = 1580

set_robust_list(0x7f04177a6a20, 24)     = 0

rseq(0x7f04177a7060, 0x20, 0, 0x53053053) = 0

mprotect(0x7f04179a8000, 16384, PROT_READ) = 0

mprotect(0x7f0417aa2000, 4096, PROT_READ) = 0

mprotect(0x556aa83e7000, 4096, PROT_READ) = 0

mprotect(0x7f0417ae1000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7f0417aa4000, 20335)           = 0

getrandom("\x9d\x77\x11\xaf\x53\xd8\x6e\x83", 8, GRND_NONBLOCK) = 8

brk(NULL)                               = 0x556ab99b8000

brk(0x556ab99d9000)                     = 0x556ab99d9000

rt_sigaction(SIGRT_1, {sa_handler=0x7f0417842520, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f04177ee320}, 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) = 0x7f0416fa5000

mprotect(0x7f0416fa6000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f04177a5990,
parent_tid=0x7f04177a5990, exit_signal=0, stack=0x7f0416fa5000, stack_size=0x7fff80,
tls=0x7f04177a56c0} => {parent_tid=[1581]}, 88) = 1581

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f04167a4000

mprotect(0x7f04167a5000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0416fa4990,
parent_tid=0x7f0416fa4990, exit_signal=0, stack=0x7f04167a4000, stack_size=0x7fff80,
tls=0x7f0416fa46c0} => {parent_tid=[1582]}, 88) = 1582

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f0415fa3000
```

```
mprotect(0x7f0415fa4000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f04167a3990,
parent_tid=0x7f04167a3990, exit_signal=0, stack=0x7f0415fa3000, stack_size=0x7fff80,
tls=0x7f04167a36c0} => {parent_tid=[1583]}, 88) = 1583

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f04157a2000

mprotect(0x7f04157a3000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0415fa2990,
parent_tid=0x7f0415fa2990, exit_signal=0, stack=0x7f04157a2000, stack_size=0x7fff80,
tls=0x7f0415fa26c0} => {parent_tid=[1584]}, 88) = 1584

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f0414fa1000

mprotect(0x7f0414fa2000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f04157a1990,
parent_tid=0x7f04157a1990, exit_signal=0, stack=0x7f0414fa1000, stack_size=0x7fff80,
tls=0x7f04157a16c0} => {parent_tid=[1585]}, 88) = 1585

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f04147a0000

mprotect(0x7f04147a1000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0414fa0990,
parent_tid=0x7f0414fa0990, exit_signal=0, stack=0x7f04147a0000, stack_size=0x7fff80,
tls=0x7f0414fa06c0} => {parent_tid=[1586]}, 88) = 1586

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f0413f9f000

mprotect(0x7f0413fa0000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f041479f990,
parent_tid=0x7f041479f990, exit_signal=0, stack=0x7f0413f9f000, stack_size=0x7fff80,
tls=0x7f041479f6c0} => {parent_tid=[1587]}, 88) = 1587

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
```

```
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f041379e000

mprotect(0x7f041379f000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0413f9e990,
parent_tid=0x7f0413f9e990, exit_signal=0, stack=0x7f041379e000, stack_size=0x7fff80,
tls=0x7f0413f9e6c0} => {parent_tid=[1588]}, 88) = 1588

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f0412f9d000

mprotect(0x7f0412f9e000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f041379d990,
parent_tid=0x7f041379d990, exit_signal=0, stack=0x7f0412f9d000, stack_size=0x7fff80,
tls=0x7f041379d6c0} => {parent_tid=[1589]}, 88) = 1589

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f041279c000

mprotect(0x7f041279d000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0412f9c990,
parent_tid=0x7f0412f9c990, exit_signal=0, stack=0x7f041279c000, stack_size=0x7fff80,
tls=0x7f0412f9c6c0} => {parent_tid=[1590]}, 88) = 1590

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f0411f9b000

mprotect(0x7f0411f9c000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f041279b990,
parent_tid=0x7f041279b990, exit_signal=0, stack=0x7f0411f9b000, stack_size=0x7fff80,
tls=0x7f041279b6c0} => {parent_tid=[1591]}, 88) = 1591

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f041179a000

mprotect(0x7f041179b000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0411f9a990,
parent_tid=0x7f0411f9a990, exit_signal=0, stack=0x7f041179a000, stack_size=0x7fff80,
tls=0x7f0411f9a6c0} => {parent_tid=[1592]}, 88) = 1592
```

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f0410f99000

mprotect(0x7f0410f9a000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0411799990,
parent_tid=0x7f0411799990, exit_signal=0, stack=0x7f0410f99000, stack_size=0x7fff80,
tls=0x7f04117996c0} => {parent_tid=[1593]}, 88) = 1593

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f0410798000

mprotect(0x7f0410799000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0410f98990,
parent_tid=0x7f0410f98990, exit_signal=0, stack=0x7f0410798000, stack_size=0x7fff80,
tls=0x7f0410f986c0} => {parent_tid=[1594]}, 88) = 1594

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040ff97000

mprotect(0x7f040ff98000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f0410797990,
parent_tid=0x7f0410797990, exit_signal=0, stack=0x7f040ff97000, stack_size=0x7fff80,
tls=0x7f04107976c0} => {parent_tid=[1595]}, 88) = 1595

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040f796000

mprotect(0x7f040f797000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040ff96990,
parent_tid=0x7f040ff96990, exit_signal=0, stack=0x7f040f796000, stack_size=0x7fff80,
tls=0x7f040ff966c0} => {parent_tid=[1596]}, 88) = 1596

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040ef95000

mprotect(0x7f040ef96000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040f795990,
```

```
parent_tid=0x7f040f795990, exit_signal=0, stack=0x7f040ef95000, stack_size=0x7fff80,
tls=0x7f040f7956c0} => {parent_tid=[1597]}, 88) = 1597

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040e794000

mprotect(0x7f040e795000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040ef94990,
parent_tid=0x7f040ef94990, exit_signal=0, stack=0x7f040e794000, stack_size=0x7fff80,
tls=0x7f040ef946c0} => {parent_tid=[1598]}, 88) = 1598

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040df93000

mprotect(0x7f040df94000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040e793990,
parent_tid=0x7f040e793990, exit_signal=0, stack=0x7f040df93000, stack_size=0x7fff80,
tls=0x7f040e7936c0} => {parent_tid=[1599]}, 88) = 1599

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040d792000

mprotect(0x7f040d793000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040df92990,
parent_tid=0x7f040df92990, exit_signal=0, stack=0x7f040d792000, stack_size=0x7fff80,
tls=0x7f040df926c0} => {parent_tid=[1600]}, 88) = 1600

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040cf91000

mprotect(0x7f040cf92000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040d791990,
parent_tid=0x7f040d791990, exit_signal=0, stack=0x7f040cf91000, stack_size=0x7fff80,
tls=0x7f040d7916c0} => {parent_tid=[1601]}, 88) = 1601

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040c790000

mprotect(0x7f040c791000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
```

```
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040cf90990,
parent_tid=0x7f040cf90990, exit_signal=0, stack=0x7f040c790000, stack_size=0x7fff80,
tls=0x7f040cf906c0} => {parent_tid=[1602]}, 88) = 1602

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040bf8f000

mprotect(0x7f040bf90000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040c78f990,
parent_tid=0x7f040c78f990, exit_signal=0, stack=0x7f040bf8f000, stack_size=0x7fff80,
tls=0x7f040c78f6c0} => {parent_tid=[1603]}, 88) = 1603

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7f040b78e000

mprotect(0x7f040b78f000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f040bf8e990,
parent_tid=0x7f040bf8e990, exit_signal=0, stack=0x7f040b78e000, stack_size=0x7fff80,
tls=0x7f040bf8e6c0} => {parent_tid=[1604]}, 88) = 1604

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

futex(0x7f04177a5990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1581, NULL,
FUTEX_BITSET_MATCH_ANY) = 0

futex(0x7f0415fa2990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1584, NULL,
FUTEX_BITSET_MATCH_ANY) = 0

munmap(0x7f0416fa5000, 8392704)         = 0

futex(0x7f0414fa0990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1586, NULL,
FUTEX_BITSET_MATCH_ANY) = 0

munmap(0x7f04167a4000, 8392704)         = 0

futex(0x7f041479f990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1587, NULL,
FUTEX_BITSET_MATCH_ANY) = 0

munmap(0x7f0415fa3000, 8392704)         = 0

munmap(0x7f04157a2000, 8392704)         = 0

munmap(0x7f0414fa1000, 8392704)         = 0

munmap(0x7f04147a0000, 8392704)         = 0

futex(0x7f041279b990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1591, NULL,
FUTEX_BITSET_MATCH_ANY) = 0

munmap(0x7f0413f9f000, 8392704)         = 0

munmap(0x7f041379e000, 8392704)         = 0
```

```
munmap(0x7f0412f9d000, 8392704)         = 0

munmap(0x7f041279c000, 8392704)         = 0

munmap(0x7f0411f9b000, 8392704)         = 0

munmap(0x7f041179a000, 8392704)         = 0

munmap(0x7f0410f99000, 8392704)         = 0

munmap(0x7f0410798000, 8392704)         = 0

munmap(0x7f040ff97000, 8392704)         = 0

munmap(0x7f040f796000, 8392704)         = 0

munmap(0x7f040ef95000, 8392704)         = 0

munmap(0x7f040e794000, 8392704)         = 0

munmap(0x7f040df93000, 8392704)         = 0

futex(0x7f040bf8e990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1604, NULL,
FUTEX_BITSET_MATCH_ANY) = 0

munmap(0x7f040d792000, 8392704)         = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

write(1, "Max area: 7211.060914\n", 22Max area: 7211.060914

) = 22

exit_group(0)                           = ?

+++ exited with 0 +++
```

Вывод

В результате выполнения лабораторной работы удалось познакомиться с многопоточным программированием. Программа успешно реализует задачу обработки данных в многопоточном режиме с использованием стандартных средств операционной системы. Были изучены базовые системные вызовы для создания и управления потоками.