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-213Б-23

Студент: Петров М.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 05.12.24

Москва, 2024

Постановка задачи

Вариант 20.

Дан массив координат (x, y, z). Необходимо найти три точки, которые образуют треугольник максимальной площади

Общий метод и алгоритм решения

Использованные системные вызовы:

1. open — открывает файл для чтения.
2. read — читает данные из открытого файла.
3. close — закрывает открытый файл.
4. write — записывает данные в стандартный поток вывода или стандартный поток ошибок.
5. sscanf — преобразует строку в числовые значения.
6. strtok — разбивает строку на лексемы.
7. pthread_create — создает новый поток выполнения.
8. pthread_join — ожидает завершения потока выполнения.
9. pthread_mutex_init — инициализирует мьютекс.
10. pthread_mutex_lock — блокирует мьютекс.
11. pthread_mutex_unlock — разблокирует мьютекс.
12. pthread_mutex_destroy — уничтожает мьютекс.

Программа находит максимальную площадь треугольника, образованного точками в трехмерном пространстве. Она принимает два аргумента: количество потоков и имя файла с координатами точек. Точки считываются из файла, затем каждый поток вычисляет площади треугольников и сравнивает их с текущим максимумом. В конце программа выводит максимальную найденную площадь треугольника.

Код программы

programm.c

```c
#include <pthread.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>
#include <string.h>
#include <math.h>
#include <stdio.h> // Временно для отладки

#define MAX_THREADS 16
#define MAX_POINTS 1000

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

typedef struct {
    Point *points;
    int num_points;
    double max_area;
    pthread_mutex_t *mutex;
```

```
    int thread_id;
} ThreadData;

double triangle_area(Point a, Point b, Point c) {
    double ab = sqrt(pow(b.x - a.x, 2) + pow(b.y - a.y, 2) + pow(b.z - a.z, 2));
    double ac = sqrt(pow(c.x - a.x, 2) + pow(c.y - a.y, 2) + pow(c.z - a.z, 2));
    double bc = sqrt(pow(c.x - b.x, 2) + pow(c.y - b.y, 2) + pow(c.z - b.z, 2));
    double s = (ab + ac + bc) / 2;
    double area = sqrt(s * (s - ab) * (s - ac) * (s - bc));

    // Отладочный вывод
    // printf("Triangle (A: (%f, %f, %f), B: (%f, %f, %f), C: (%f, %f, %f)) - Area: %f\n",
    //         a.x, a.y, a.z, b.x, b.y, b.z, c.x, c.y, c.z, area);

    return area;
}

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

    for (int i = data->thread_id; i < data->num_points; i += MAX_THREADS) {
        for (int j = i + 1; j < data->num_points; j++) {
            for (int k = j + 1; k < data->num_points; k++) {
                double area = triangle_area(data->points[i], data->points[j], data->points[k]);
                if (area > max_area) {
                    max_area = area;
                }
            }
        }
    }

    pthread_mutex_lock(data->mutex);
    if (max_area > data->max_area) {
        data->max_area = max_area;
    }
    pthread_mutex_unlock(data->mutex);

    return NULL;
}

int main(int argc, char *argv[]) {
    if (argc < 3) {
        const char *error_msg = "Usage: <number_of_threads> <input_file>\n";
        write(STDERR_FILENO, error_msg, strlen(error_msg));
        return 1;
    }

    int num_threads = atoi(argv[1]);
    if (num_threads > MAX_THREADS || num_threads < 1) {
        const char *error_msg = "Number of threads should be between 1 and 16\n";
        write(STDERR_FILENO, error_msg, strlen(error_msg));
        return 1;
    }

    Point points[MAX_POINTS]; // Массив фиксированного размера
    int num_points = 0;
```

```
    // Открываем файл с входными данными
    int fd = open(argv[2], O_RDONLY);
    if (fd < 0) {
        const char *error_msg = "Error opening input file\n";
        write(STDERR_FILENO, error_msg, strlen(error_msg));
        return 1;
    }

    // Чтение точек из файла
    char buffer[256];
    while (read(fd, buffer, sizeof(buffer)) > 0) {
        char *line = strtok(buffer, "\n");
        while (line) {
            if (num_points >= MAX_POINTS) {
                break;
            }

            sscanf(line, "%lf %lf %lf", &points[num_points].x, &points[num_points].y,
&points[num_points].z);
            num_points++;
            line = strtok(NULL, "\n");
        }
    }
    close(fd);

    double max_area = 0.0;
    pthread_mutex_t mutex;
    pthread_mutex_init(&mutex, NULL);
    pthread_t threads[MAX_THREADS];
    ThreadData thread_data[MAX_THREADS];

    for (int i = 0; i < num_threads; i++) {
        thread_data[i] = (ThreadData){.points = points, .num_points = num_points, .max_area =
0.0, .mutex = &mutex, .thread_id = i};
        pthread_create(&threads[i], NULL, find_max_triangle_area, &thread_data[i]);
    }

    for (int i = 0; i < num_threads; i++) {
        pthread_join(threads[i], NULL);
    }

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

    // Формируем вывод
    char output[100];
    int len = sprintf(output, "Maximum triangle area: %f\n", max_area);
    write(STDOUT_FILENO, output, len);

    pthread_mutex_destroy(&mutex);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
markvolkov@MacBook-Air-Mark-2 LAB2 % ./prog 4 input.txt
Maximum triangle area: 3.535534
markvolkov@MacBook-Air-Mark-2 LAB2 % ./prog 8 input.txt
Maximum triangle area: 3.535534
markvolkov@MacBook-Air-Mark-2 LAB2 % ./prog 0 input.txt
Number of threads should be between 1 and 16
markvolkov@MacBook-Air-Mark-2 LAB2 % ./prog 123 input.txt
Number of threads should be between 1 and 16
```

Strace:

strace -f ./p 2 input.txt

execve("./p", ["./p", "2", "input.txt"], 0x7ffe73576528 /* 46 vars */) = 0

brk(NULL) = 0x5952142c2000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe33fd2fb0) = -1 EINVAL (Недопустимый аргумент)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x79bcfbc60000

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=58047, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 58047, PROT_READ, MAP_PRIVATE, 3, 0) = 0x79bcfbc51000

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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x79bcfbb6a000

mmap(0x79bcfbb78000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x79bcfbb78000

mmap(0x79bcfbbf4000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x79bcfbbf4000

mmap(0x79bcfbc4f000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x79bcfbc4f000

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) = 0x79bcfb800000

mprotect(0x79bcfb828000, 2023424, PROT_NONE) = 0

mmap(0x79bcfb828000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x79bcfb828000

```
mmap(0x79bcfb9bd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x79bcfb9bd000
mmap(0x79bcfba16000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x79bcfba16000
mmap(0x79bcfba1c000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x79bcfba1c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x79bcfbb67000
arch_prctl(ARCH_SET_FS, 0x79bcfbb67740) = 0
set_tid_address(0x79bcfbb67a10)         = 4071
set_robust_list(0x79bcfbb67a20, 24)     = 0
rseq(0x79bcfbb680e0, 0x20, 0, 0x53053053) = 0
mprotect(0x79bcfba16000, 16384, PROT_READ) = 0
mprotect(0x79bcfbc4f000, 4096, PROT_READ) = 0
mprotect(0x595207566000, 4096, PROT_READ) = 0
mprotect(0x79bcfbc9a000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
munmap(0x79bcfbc51000, 58047)           = 0
openat(AT_FDCWD, "input.txt", O_RDONLY) = 3
read(3, "0.0 0.0 0.0\n1.0 0.0 0.0\n0.0 1.0 "..., 256) = 119
read(3, "", 256)                        = 0
close(3)                                = 0
rt_sigaction(SIGRT_1, {sa_handler=0x79bcfb891870, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO,
sa_restorer=0x79bcfb842520}, 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) = 0x79bcfae00000
mprotect(0x79bcfae01000, 8388608, PROT_READ|PROT_WRITE) = 0
getrandom("\x31\x0d\x52\x79\x49\xef\x38\x9f", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5952142c2000
brk(0x5952142e3000)                     = 0x5952142e3000
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREA
D|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x79bcfb600910, parent_tid=0x79bcfb600910, exit_signal=0, stack=0x79bcfae00000,
stack_size=0x7fff00, tls=0x79bcfb600640}strace: Process 4072 attached
 => {parent_tid=[4072]}, 88) = 4072
[pid  4071] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  4072] rseq(0x79bcfb600fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4071] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4072] <... rseq resumed>)         = 0
[pid  4071] mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x79bcfa400000
[pid  4072] set_robust_list(0x79bcfb600920, 24) = 0
[pid  4071] mprotect(0x79bcfa401000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  4072] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  4071] <... mprotect resumed>)     = 0
[pid  4072] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4071] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  4072] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
```

```
[pid  4071] <... rt_sigprocmask resumed>[], 8) = 0
[pid  4071]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
child_tid=0x79bcfac00910, parent_tid=0x79bcfac00910, exit_signal=0, stack=0x79bcfa400000,
stack_size=0x7fff00, tls=0x79bcfac00640} <unfinished ...>
[pid  4072] <... rt_sigprocmask resumed>NULL, 8) = 0
strace: Process 4073 attached
[pid  4071] <... clone3 resumed> => {parent_tid=[4073]}, 88) = 4073
[pid  4072] madvise(0x79bcfae00000, 8368128, MADV_DONTNEED <unfinished ...>
[pid  4071] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  4073] rseq(0x79bcfac00fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4072] <... madvise resumed>)      = 0
[pid  4071] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4073] <... rseq resumed>)         = 0
[pid  4072] exit(0 <unfinished ...>
[pid  4071] futex(0x79bcfb600910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4072, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  4073] set_robust_list(0x79bcfac00920, 24 <unfinished ...>
[pid  4072] <... exit resumed>)         = ?
[pid  4073] <... set_robust_list resumed>) = 0
[pid  4072] +++ exited with 0 +++
[pid  4071] <... futex resumed>)        = 0
[pid  4073] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid  4071] futex(0x79bcfac00910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
4073, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  4073] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4073] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
[pid  4073] madvise(0x79bcfa400000, 8368128, MADV_DONTNEED) = 0
[pid  4073] exit(0)                     = ?
[pid  4073] +++ exited with 0 +++
<... futex resumed>)                    = 0
write(1, "Maximum triangle area: 3.535534\n", 32Maximum triangle area: 3.535534
) = 32
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Эта программа эффективно вычисляет максимальную площадь треугольника среди множества точек в трёхмерном пространстве, используя многопоточность для ускорения расчётов. Она демонстрирует применение параллельного программирования для решения задач, связанных с геометрией, и может быть полезна в различных инженерных и научных приложениях.