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: Арсельгов А. Б.

Преподаватель: Бахарев В. Д.

Оценка: _____

Дата: 10.11.24

Москва, 2024

Постановка задачи

Вариант 20.

Задание

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.

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 входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

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

Общий метод и алгоритм решения

Использованные системные вызовы:

- **write** - для вывода данных в консоль.
- **pthread_create** - создаёт новый поток (thread), выполняющий функцию `find_max_triangle`. В нашем случае он используется для параллельного выполнения расчёта площади треугольника в нескольких потоках, что позволяет распределить вычисления.
- **pthread_join** — Этот вызов ожидает завершения выполнения каждого потока, созданного с помощью `pthread_create`. Основной поток программы ожидает, пока все потоки завершат свои вычисления, прежде чем продолжить выполнение.
- `pthread_mutex_lock` и `pthread_mutex_unlock` — Эти вызовы используются для синхронизации доступа к глобальным переменным между потоками. В данном коде они необходимы для того, чтобы предотвратить одновременное обновление `global_max_area` и `global_max_triangle`, что могло бы привести к некорректным данным в случае параллельного доступа..

Алгоритм:

- Инициализировать массив точек и параметры потоков.
- Создать потоки и разделить данные между ними.
- Каждый поток находит максимальный треугольник в своем диапазоне и при необходимости обновляет глобальный максимум.
- Основной поток ждет завершения всех потоков.

- Выводит глобальные результаты: максимальная площадь и координаты треугольника.

Код программы

lab02.c

```
#unclude <stdlib.h>
#include <math.h>
#include <pthread.h>
#include <string.h>
#include <unistd.h>
#define MAX_POINTS 100
#define FLOAT_PRECISION 6

typedef enum StatusCode {
SUCCESS = 0,
ERROR_COUNT_ARGS,
ERROR_COUNT_THREADS
} StatusCode;

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

typedef struct ThreadData {
Point *points;
int start, end;
double max_area;
Point max_triangle[3];
} ThreadData;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
double global_max_area = 0.0;
Point global_max_triangle[3];

double distance(Point a, Point b) {
return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y) + (a.z - b.z) * (a.z - b.z));
}

double triangle_area(Point a, Point b, Point c) {
double ab = distance(a, b);
double bc = distance(b, c);
double ca = distance(c, a);
double s = (ab + bc + ca) / 2.0;
return sqrt(s * (s - ab) * (s - bc) * (s - ca));
```

```
}

void *find_max_triangle(void *arg) {
ThreadData *data = (ThreadData *)arg;
double max_area = 0.0;
Point max_triangle[3];

for (int i = data->start; i < data->end - 2; i++) {
for (int j = i + 1; j < data->end - 1; j++) {
for (int k = j + 1; k < data->end; k++) {
double area = triangle_area(data->points[i], data->points[j], data->points[k]);
if (area > max_area) {
max_area = area;
max_triangle[0] = data->points[i];
max_triangle[1] = data->points[j];
max_triangle[2] = data->points[k];
}
}
}
}

pthread_mutex_lock(&mutex);
if (max_area > global_max_area) {
global_max_area = max_area;
memcpy(global_max_triangle, max_triangle, sizeof(max_triangle));
}
pthread_mutex_unlock(&mutex);

return NULL;
}

StatusCode print_str(const char *str) {
write(1, str, strlen(str));

return SUCCESS;
}

StatusCode print_int(int num) {
char buf[12];
char *p = buf + sizeof(buf) - 1;
*p = '\0';

int is_negative = num < 0;
if (is_negative) {
num = -num;
}

do {
*(--p) = '0' + (num % 10);
num /= 10;
} while (num > 0);
```

```
if (is_negative) {
*(--p) = '-';
}

print_str(p);

return SUCCESS;
}

StatusCode print_double(double num) {
if (num < 0) {
print_str("-");
num = -num;
}

int int_part = (int)num;
double fraction_part = num - int_part;

print_int(int_part);

print_str(".");
for (int i = 0; i < FLOAT_PRECISION; i++) {
fraction_part *= 10;
int digit = (int)fraction_part;
fraction_part -= digit;
char c = '0' + digit;
write(1, &c, 1);
}

return SUCCESS;
}

StatusCode print_point(Point p) {
print_str("(");
print_double(p.x);
print_str(", ");
print_double(p.y);
print_str(", ");
print_double(p.z);
print_str(")\n");

return SUCCESS;
}

int main(int argc, char *argv[]) {
if (argc != 2) {
write(2, "Неверное количество аргументов.\n", strlen("Неверное количество аргументов.\n"));
return ERROR_COUNT_ARGS;
}

int max_threads = atoi(argv[1]);
if (max_threads <= 0) {
```

```
return ERROR_COUNT_THREADS;
}

Point points[MAX_POINTS] = {
{0.0, 0.0, 0.0}, {1.0, 0.0, 0.0}, {0.0, 1.0, 0.0},
{0.0, 0.0, 1.0}, {1.0, 1.0, 1.0}, {2.0, 2.0, 2.0},
{3.0, 3.0, 3.0}, {1.0, 2.0, 3.0}, {4.0, 4.0, 4.0},
{-1.0, 2.0, 3.0}, {0.0, 5.0, -1.0}, {3.0, 2.0, 1.0}
};

int n_points = sizeof(points) / sizeof(Point);
int points_per_thread = n_points / max_threads;

pthread_t threads[max_threads];
ThreadData thread_data[max_threads];

for (int i = 0; i < max_threads; i++) {
thread_data[i].points = points;
thread_data[i].start = i * points_per_thread;
thread_data[i].end = (i == max_threads - 1) ? n_points : (i + 1) * points_per_thread;

pthread_create(&threads[i], NULL, find_max_triangle, &thread_data[i]);
}

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

print_str("Максимальная площадь: ");
print_double(global_max_area);
print_str("\nТочки треугольника:\n");

print_str("Координата точки 1: ");
print_point(global_max_triangle[0]);

print_str("Координата точки 2: ");
print_point(global_max_triangle[1]);

print_str("Координата точки 3: ");
print_point(global_max_triangle[2]);

return SUCCESS;
}
```

Протокол работы программы

Тестирование:

```
user@adamarselgov:~/MAI_OS/lab02/src$ gcc lab02.c -o lab02 -lm
user@adamarselgov:~/MAI_OS/lab02/src$ ./q 4
bash: ./q: Нет такого файла или каталога
user@adamarselgov:~/MAI_OS/lab02/src$ ./lab02
Неверное количество аргументов.
user@adamarselgov:~/MAI_OS/lab02/src$ ./lab02 4
Максимальная площадь: 15.755951
Точки треугольника:
Координата точки 1: (0.000000, 0.000000, 1.000000)
Координата точки 2: (4.000000, 4.000000, 4.000000)
Координата точки 3: (0.000000, 5.000000, -1.000000)
user@adamarselgov:~/MAI_OS/lab02/src$
```

Strace:

```
user@adamarselgov:~/MAI_OS/lab02/src$ strace ./1 4
execve("./1", ["./1", "4"], 0x7fff87f7def8 /* 77 vars */) = 0
brk(NULL)                               = 0x5e842a5d5000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x71c39caaa000
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
fstat(3, {st_mode=S_IFREG|0644, st_size=83867, ...}) = 0
mmap(NULL, 83867, PROT_READ, MAP_PRIVATE, 3, 0) = 0x71c39ca95000
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\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) = 0x71c39c9ac000
mmap(0x71c39c9bc000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x71c39c9bc000
mmap(0x71c39ca3b000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x71c39ca3b000
mmap(0x71c39ca93000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x71c39ca93000
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) = 0x71c39c600000
     mmap(0x71c39c628000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x71c39c628000
     mmap(0x71c39c7b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x71c39c7b0000
     mmap(0x71c39c7ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x71c39c7ff000
     mmap(0x71c39c805000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x71c39c805000
     close(3)                                = 0
     mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x71c39c9a9000
     arch_prctl(ARCH_SET_FS, 0x71c39c9a9740) = 0
     set_tid_address(0x71c39c9a9a10)         = 12022
     set_robust_list(0x71c39c9a9a20, 24)     = 0
     rseq(0x71c39c9aa060, 0x20, 0, 0x53053053) = 0
     mprotect(0x71c39c7ff000, 16384, PROT_READ) = 0
     mprotect(0x71c39ca93000, 4096, PROT_READ) = 0
     mprotect(0x5e842a567000, 4096, PROT_READ) = 0
     mprotect(0x71c39cae2000, 8192, PROT_READ) = 0
     prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
     munmap(0x71c39ca95000, 83867)           = 0
     rt_sigaction(SIGRT_1, {sa_handler=0x71c39c699520, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x71c39c645320}, 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) = 0x71c39bc00000
     mprotect(0x71c39bc01000, 8388608, PROT_READ|PROT_WRITE) = 0
     getrandom("\x6d\xcd\x1d\x3c\xa2\x58\xb1\x34", 8, GRND_NONBLOCK) = 8
     brk(NULL)                               = 0x5e842a5d5000
     brk(0x5e842a5f6000)                     = 0x5e842a5f6000
     rt_sigprocmask(SIG_BLOCK, ~[], [], 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=0x71c39c400990, parent_tid=0x71c39c400990, exit_signal=0, stack=0x71c39bc00000, stack_size=0x7fff80, tls=0x71c39c4006c0} => {parent_tid=[12023]}, 88) = 12023
     rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
     mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x71c39b200000
     mprotect(0x71c39b201000, 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_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x71c39ba00990, parent_tid=0x71c39ba00990, exit_signal=0, stack=0x71c39b200000, stack_size=0x7fff80, tls=0x71c39ba006c0} => {parent_tid=[12024]}, 88) = 12024
     rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
     mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x71c39a800000
     mprotect(0x71c39a801000, 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_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x71c39b000990, parent_tid=0x71c39b000990, exit_signal=0, stack=0x71c39a800000, stack_size=0x7fff80, tls=0x71c39b0006c0} => {parent_tid=[12025]}, 88) = 12025
     rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
```

```
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x71c399e00000
mprotect(0x71c399e01000, 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_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x71c39a600990, parent_tid=0x71c39a600990, exit_signal=0, stack=0x71c399e00000, stack_size=0x7fff80, tls=0x71c39a6006c0} => {parent_tid=[12026]}, 88) = 12026
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
futex(0x71c39a600990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 12026, NULL, FUTEX_BITSET_MATCH_ANY) = 0
write(1, "\320\234\320\260\320\272\321\201\320\270\320\274\320\260\320\273\321\214\320\275\320\260\321\217 \320\277\320\273\320\276\321"..., 41Максимальная площадь: ) = 41
write(1, "15", 215)                      = 2
write(1, ".", 1.)                        = 1
write(1, "7", 17)                        = 1
write(1, "5", 15)                        = 1
write(1, "5", 15)                        = 1
write(1, "9", 19)                        = 1
write(1, "5", 15)                        = 1
write(1, "1", 11)                        = 1
write(1, "\n\320\242\320\276\321\207\320\272\320\270\321\202\321\200\320\265\321\203\320\263\320\276\320\273\321\214\320\275\320\270"..., 38
Точки треугольника:
) = 38
write(1, "\320\232\320\276\320\276\321\200\320\264\320\270\320\275\320\260\321\202\320\260\321\202\320\276\321\207\320\272\320\270 "..., 35Координата точки 1: ) = 35
write(1, "(", 1()                        = 1
write(1, "0", 10)                        = 1
write(1, ".", 1.)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, ", ", 2, )                      = 2
write(1, "0", 10)                        = 1
write(1, ".", 1.)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, ", ", 2, )                      = 2
write(1, "1", 11)                        = 1
write(1, ".", 1.)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
write(1, "0", 10)                        = 1
```

```
     write(1, "0", 10)                       = 1
     write(1, ")\n", 2)
     )                     = 2
     write(1,
"\320\232\320\276\320\276\321\200\320\264\320\270\320\275\320\260\321\202\320\260
\321\202\320\276\321\207\320\272\320\270 ..., 35Координата точки 2: ) = 35
     write(1, "(", 1()                       = 1
     write(1, "4", 14)                       = 1
     write(1, ".", 1.)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, ", ", 2, )                      = 2
     write(1, "4", 14)                       = 1
     write(1, ".", 1.)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, ", ", 2, )                      = 2
     write(1, "4", 14)                       = 1
     write(1, ".", 1.)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, ")\n", 2)
     )                     = 2
     write(1,
"\320\232\320\276\320\276\321\200\320\264\320\270\320\275\320\260\321\202\320\260
\321\202\320\276\321\207\320\272\320\270 ..., 35Координата точки 3: ) = 35
     write(1, "(", 1()                       = 1
     write(1, "0", 10)                       = 1
     write(1, ".", 1.)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, "0", 10)                       = 1
     write(1, ", ", 2, )                      = 2
     write(1, "5", 15)                       = 1
     write(1, ".", 1.)                       = 1
     write(1, "0", 10)                       = 1
```

```
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, ", ", 2, )                      = 2
write(1, "-", 1-)                       = 1
write(1, "1", 11)                       = 1
write(1, ".", 1.)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, "0", 10)                       = 1
write(1, ")\n", 2)
)                       = 2
exit_group(0)                           = ?
+++ exited with 0 +++
user@adamarselgov:~/MAI_OS/lab02/src$
```

Вывод

Программа находит треугольник с максимальной площадью среди заданных точек в пространстве с использованием многопоточности. Каждый поток обрабатывает свою часть точек, вычисляя площадь треугольников и находя локальный максимум. Затем результаты от каждого потока сравниваются, и выбирается глобальный максимальный треугольник. В конце программа выводит максимальную площадь и координаты вершин треугольника с наибольшей площадью.