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:

- fork() – создает дочерний процесс.
- pipe() – для организации межпроцессного взаимодействия.
- write() - для вывода данных в консоль.
- exit() - для завершения дочернего процесса.
- wait() - для ожидания завершения дочерних процессов.
- memcpy() - для копирования данных из одного места памяти в другое.

В рамках лабораторной работы была реализована программа для вычисления максимальной площади треугольника, образованного тремя точками из набора точек в 3D пространстве. Для улучшения производительности задачи использовалась параллельная обработка. Программа разбивает набор точек на несколько частей и создает несколько процессов с помощью системного вызова `fork()`. Каждый процесс обрабатывает свою часть точек и вычисляет площади всех возможных треугольников, образованных тремя точками. Для обмена данными между процессами используется канала связи `pipe()`. Каждый процесс передает результат в родительский процесс, который после завершения всех дочерних процессов собирает данные, находит максимальную площадь и выводит координаты точек, образующих этот треугольник. Для синхронизации работы процессов используется системный вызов `wait()`, который обеспечивает выполнение программы в правильном порядке.

Код программы

lab02.c

```
#include <stdlib.h>
#include <math.h
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <string.h>

#define MAX_POINTS 100
#define FLOAT_PRECISION 6

typedef enum StatusCode {
SUCCESS = 0,
ERROR_ARGS_COUNT
} StatusCode;

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

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));
}

StatusCode find_max_triangle(Point *points, int n_points, int start, int end, int pipe_fd) {
double max_area = 0.0;
Point max_triangle[3];

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

```
}

write(pipe_fd, &max_area, sizeof(double));
write(pipe_fd, &max_triangle, sizeof(max_triangle));
return SUCCESS;
}

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_ARGS_COUNT;
}

int max_processes = atoi(argv[1]);
if (max_processes <= 0) {
return 1;
}

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_process = n_points / max_processes;
pid_t pid;
int pipe_fds[2];
pipe(pipe_fds);

for (int i = 0; i < max_processes; i++) {
int start = i * points_per_process;
int end = (i == max_processes - 1) ? n_points : (i + 1) * points_per_process;

pid = fork();
if (pid == 0) {
close(pipe_fds[0]);
find_max_triangle(points, n_points, start, end, pipe_fds[1]);
close(pipe_fds[1]);
exit(0);
}
}

for (int i = 0; i < max_processes; i++) {
```

```
wait(NULL);
}

close(pipe_fds[1]);

double max_area = 0.0;
Point max_triangle[3];
for (int i = 0; i < max_processes; i++) {
double area;
Point triangle[3];
read(pipe_fds[0], &area, sizeof(double));
read(pipe_fds[0], &triangle, sizeof(triangle));

if (area > max_area) {
max_area = area;
memcpy(max_triangle, triangle, sizeof(triangle));
}
}

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

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

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

print_str("Координата точки 3: ");
print_point(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 ./lab02 4
execve("./lab02", ["./lab02", "4"], 0x7ffce4fb0118 /* 77 vars */) = 0
brk(NULL)                               = 0x6071455e2000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fa3f5842000
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) = 0x7fa3f582d000
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) = 0x7fa3f5744000
mmap(0x7fa3f5754000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7fa3f5754000
mmap(0x7fa3f57d3000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x7fa3f57d3000
mmap(0x7fa3f582b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x7fa3f582b000
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) = 0x7fa3f5400000
```

```
mmap(0x7fa3f5428000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fa3f5428000
mmap(0x7fa3f55b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7fa3f55b0000
mmap(0x7fa3f55ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7fa3f55ff000
mmap(0x7fa3f5605000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fa3f5605000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa3f5741000
arch_prctl(ARCH_SET_FS, 0x7fa3f5741740) = 0
set_tid_address(0x7fa3f5741a10)         = 9595
set_robust_list(0x7fa3f5741a20, 24)     = 0
rseq(0x7fa3f5742060, 0x20, 0, 0x53053053) = 0
mprotect(0x7fa3f55ff000, 16384, PROT_READ) = 0
mprotect(0x7fa3f582b000, 4096, PROT_READ) = 0
mprotect(0x6071443f9000, 4096, PROT_READ) = 0
mprotect(0x7fa3f587a000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fa3f582d000, 83867)           = 0
pipe2([3, 4], 0)                        = 0
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fa3f5741a10) = 9596
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fa3f5741a10) = 9597
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fa3f5741a10) = 9598
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0x7fa3f5741a10) = 9599
wait4(-1, NULL, 0, NULL)                = 9599
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=9599, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
wait4(-1, NULL, 0, NULL)                = 9598
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=9598, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
wait4(-1, NULL, 0, NULL)                = 9597
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=9597, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
wait4(-1, NULL, 0, NULL)                = 9596
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=9596, si_uid=1000, si_status=0, si_utime=0, si_stime=0} ---
close(4)                                = 0
read(3, "\0\0\0\0\0\0\0\0", 8)          = 8
read(3, "q\23491\375\177\0\0\1\0\0\0\0\0\0\5\7\0\0\0\4\0\0\0\0\225 \333&\345\375["..., 72) = 72
read(3, "\0\0\0\0\0\0\0\0", 8)          = 8
read(3, "q\23491\375\177\0\0\1\0\0\0\0\0\0\5\7\0\0\0\4\0\0\0\0\225 \333&\345\375["..., 72) = 72
read(3, "\0\0\0\0\0\0\0\0", 8)          = 8
read(3, "q\23491\375\177\0\0\1\0\0\0\0\0\0\5\7\0\0\0\4\0\0\0\0\225 \333&\345\375["..., 72) = 72
read(3, "c\326\10\v\f\203/@", 8)        = 8
read(3, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\360?\0\0\0\0\0\0\20@"..., 72) = 72
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, ", ", 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

В ходе выполнения лабораторной работы была разработана программа, которая использует многопроцессность для поиска треугольника с максимальной площадью из множества точек в 3D-пространстве. В процессе работы были использованы системные вызовы `fork`, `pipe`, `write`, `read`, а также математические функции для вычисления расстояний и площади треугольников. Основной проблемой стало корректное распределение точек между процессами и синхронизация работы с пайпами для передачи результатов. В будущем, хотелось бы улучшить структуру программы для более эффективной обработки ошибок и оптимизировать вычисления для больших наборов данных.