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: Голосов Г.С.

Преподаватель: Миронов Е.С.

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

Дата: 15.11.24

Москва, 2024

**Постановка задачи**

**Вариант 16.**

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.

Задаётся радиус окружности. Необходимо с помощью метода Монте-Карло рассчитать её площадь

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg); – Создает новый поток, возвращает 0 при успехе
* pthread\_join(pthread\_t thread, void \*\*retval); - Ожидает завершения указанного потока. Блокирует вызывающий поток до завершения целевого потока
* pthread\_mutex\_lock(pthread\_mutex\_t \*mutex); - Блокирует мьютекс. Если мьютекс уже заблокирован, поток блокируется до освобождения
* pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex); - Разблокирует мьютекс. Позволяет другим потокам захватить мьютекс
* pthread\_mutex\_destroy(&mutex); - удаление мьютекса
* pthread\_exit(NULL); - завершение работы потока
* pthread\_mutex\_init(&mutex, NULL); - создание мьютекса

Программа выполняет вычисление площади круга методом Монте-Карло. Он заключается в том, что окружность вписывается в квадрат, затем по квадрату случайным образом распределяются точки и подсчитывается количество точек попавших в круг. Для вычисления площади круга, количество точек попавших в круг делится на общее число точек (в программе общее число точек равно 10^9) и умножается на площадь квадрата.

Таким образом, программа принимает на вход радиус окружности и количество потоков. Общее количество точек равномерно распределяется по потокам. Случайным образом генерируются координаты точки (x, y), проверяется находится ли точка внутри окружности. Количество точек, попавшее в окружность, прибавляется к глобальной переменной total\_points\_in\_circle. В целях борьбы с проблемой состязательной ситуации для изменения значения этой переменной будет использоваться мьютекс. После выполнения всех потоков вычисляется площадь окружности.

**Код программы**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <pthread.h>

#include <time.h>

#include <iostream>

#include <random>

struct Thread\_Data{

    double radius;

    int points\_per\_thread;

};

int total\_points\_in\_circle = 0;

pthread\_mutex\_t mutex;

void\* monte\_carlo(void\* arg) {

    struct Thread\_Data\* data = (struct Thread\_Data\*)arg;

    int inside\_circle = 0;

    unsigned int seed = time(NULL) ^ pthread\_self();

    double radius\_square = data->radius \* data->radius;

    for (int i = 0; i < data->points\_per\_thread; ++i) {

        double x = (double)rand\_r(&seed) / RAND\_MAX \* data->radius;

        double y = (double)rand\_r(&seed) / RAND\_MAX \* data->radius;

        if (x \* x + y \* y <= (radius\_square)) {

            inside\_circle++;

        }

    }

    pthread\_mutex\_lock(&mutex);

    total\_points\_in\_circle += inside\_circle;

    pthread\_mutex\_unlock(&mutex);

    pthread\_exit(NULL);

}

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

    struct timespec start, end;

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

    if (argc != 3) {

        std::cerr << "Usage: " << argv[0] << " <radius> <threads\_number>" << std::endl;

        return EXIT\_FAILURE;

    }

    double radius = atof(argv[1]);

    int threads\_number = atoi(argv[2]);

    if (radius <= 0 || threads\_number <= 0) {

        std::cerr << "Radius and threads\_number must be positive numbers." << std::endl;

        return EXIT\_FAILURE;

    }

    srand(time(NULL));

    int total\_points = 1000000000; // 10 ^ 9

    pthread\_t threads[threads\_number];

    int points\_per\_thread = total\_points / threads\_number;

    struct Thread\_Data data;

    data.radius = radius;

    data.points\_per\_thread = points\_per\_thread;

    pthread\_mutex\_init(&mutex, nullptr);

    for (int i = 0; i < threads\_number; ++i) {

        pthread\_create(&threads[i], NULL, monte\_carlo, &data);

    }

    for (int i = 0; i < threads\_number; ++i) {

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

    }

    pthread\_mutex\_destroy(&mutex);

    double area = ((double)total\_points\_in\_circle / total\_points) \* (radius \* radius \* 4);

    std::cout << "Estimated area of the circle with radius " << radius << ": " << area << std::endl;

    // Display the number of threads used

    std::cout << "Number of threads used: " << threads\_number << std::endl;

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

    double elapsed\_time = (end.tv\_sec - start.tv\_sec) +

                          (end.tv\_nsec - start.tv\_nsec) / 1e9;

    std::cout << "Elapsed time: " << elapsed\_time << " seconds" << std::endl;

    return EXIT\_SUCCESS;

}

**Протокол работы программы**

**Тестирование:**

tobiklosj@LAPTOP-C3C2PI9E:~/labs\_OS/lab2/src$ ./a.out 12 1

Estimated area of the circle with radius 12: 452.388

Number of threads used: 1

Elapsed time: 16.8378 seconds

tobiklosj@LAPTOP-C3C2PI9E:~/labs\_OS/lab2/src$ ./a.out 12 2

Estimated area of the circle with radius 12: 452.391

Number of threads used: 2

Elapsed time: 8.33231 seconds

tobiklosj@LAPTOP-C3C2PI9E:~/labs\_OS/lab2/src$ ./a.out 12 3

Estimated area of the circle with radius 12: 452.388

Number of threads used: 3

Elapsed time: 6.16774 seconds

tobiklosj@LAPTOP-C3C2PI9E:~/labs\_OS/lab2/src$ ./a.out 12 4

Estimated area of the circle with radius 12: 452.39

Number of threads used: 4

Elapsed time: 5.40424 seconds

tobiklosj@LAPTOP-C3C2PI9E:~/labs\_OS/lab2/src$ ./a.out 12 5

Estimated area of the circle with radius 12: 452.389

Number of threads used: 5

Elapsed time: 5.50319 seconds

tobiklosj@LAPTOP-C3C2PI9E:~/labs\_OS/lab2/src$ ./a.out 12 10

Estimated area of the circle with radius 12: 452.385

Number of threads used: 10

Elapsed time: 5.47629 seconds

tobiklosj@LAPTOP-C3C2PI9E:~/labs\_OS/lab2/src$ ./a.out 100 3

Estimated area of the circle with radius 100: 31415.9

Number of threads used: 3

Elapsed time: 6.40172 seconds

**Оценка эффективности**

|  |  |  |  |
| --- | --- | --- | --- |
| **Число потоков** | **Время исполнения (мс)** | **Ускорение** | **Эффективность** |
| **1** | 16.8378 | 1 | 1 |
| **2** | 8.33231 | 2.02 | 1.01 |
| **3** | 6.16774 | 2.73 | 0.91 |
| **4** | 5.40424 | 3.12 | 0.78 |
| **5** | 5.50319 | 3.05 | 0.61 |
| **10** | 5.47629 | 3.07 | 0.31 |

Формула расчета ускорения: T1/TN, где Т1 – время выполнения на одном потоке, TN – время выполнения на N потоках.

Эффективность – величина EN = SN / N, где SN – ускорение, N – количество используемых потоков.

**Strace:**

execve("./a.out", ["./a.out", "12", "3"], 0x7fff348c9870 /\* 20 vars \*/) = 0

brk(NULL) = 0x55f97e3d1000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffed158bca0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc3c6dca000

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=36115, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 36115, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fc3c6dc1000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.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"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fc3c6b95000

mprotect(0x7fc3c6c2f000, 1576960, PROT\_NONE) = 0

mmap(0x7fc3c6c2f000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7fc3c6c2f000

mmap(0x7fc3c6d40000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7fc3c6d40000

mmap(0x7fc3c6db0000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7fc3c6db0000

mmap(0x7fc3c6dbe000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fc3c6dbe000

close(3) = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fc3c6b75000

mmap(0x7fc3c6b78000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7fc3c6b78000

mmap(0x7fc3c6b8f000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7fc3c6b8f000

mmap(0x7fc3c6b93000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7fc3c6b93000

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) = 0x7fc3c694c000

mprotect(0x7fc3c6974000, 2023424, PROT\_NONE) = 0

mmap(0x7fc3c6974000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fc3c6974000

mmap(0x7fc3c6b09000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fc3c6b09000

mmap(0x7fc3c6b62000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7fc3c6b62000

mmap(0x7fc3c6b68000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fc3c6b68000

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"..., 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) = 0x7fc3c6865000

mmap(0x7fc3c6873000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7fc3c6873000

mmap(0x7fc3c68ef000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7fc3c68ef000

mmap(0x7fc3c694a000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7fc3c694a000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc3c6863000

arch\_prctl(ARCH\_SET\_FS, 0x7fc3c68643c0) = 0

set\_tid\_address(0x7fc3c6864690) = 1810

set\_robust\_list(0x7fc3c68646a0, 24) = 0

rseq(0x7fc3c6864d60, 0x20, 0, 0x53053053) = 0

mprotect(0x7fc3c6b62000, 16384, PROT\_READ) = 0

mprotect(0x7fc3c694a000, 4096, PROT\_READ) = 0

mprotect(0x7fc3c6b93000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fc3c6861000

mprotect(0x7fc3c6db0000, 45056, PROT\_READ) = 0

mprotect(0x55f97e044000, 4096, PROT\_READ) = 0

mprotect(0x7fc3c6e04000, 8192, PROT\_READ) = 0

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

munmap(0x7fc3c6dc1000, 36115) = 0

getrandom("\x50\xc5\x5d\x7b\x0e\xd9\xba\xe4", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55f97e3d1000

brk(0x55f97e3f2000) = 0x55f97e3f2000

futex(0x7fc3c6dbe77c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=34044, tv\_nsec=489626900}) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7fc3c69dd870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7fc3c698e520}, 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) = 0x7fc3c6060000

mprotect(0x7fc3c6061000, 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=0x7fc3c6860910, parent\_tid=0x7fc3c6860910, exit\_signal=0, stack=0x7fc3c6060000, stack\_size=0x7fff00, tls=0x7fc3c6860640} => {parent\_tid=[1811]}, 88) = 1811

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fc3c585f000

mprotect(0x7fc3c5860000, 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=0x7fc3c605f910, parent\_tid=0x7fc3c605f910, exit\_signal=0, stack=0x7fc3c585f000, stack\_size=0x7fff00, tls=0x7fc3c605f640} => {parent\_tid=[1812]}, 88) = 1812

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

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fc3c505e000

mprotect(0x7fc3c505f000, 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=0x7fc3c585e910, parent\_tid=0x7fc3c585e910, exit\_signal=0, stack=0x7fc3c505e000, stack\_size=0x7fff00, tls=0x7fc3c585e640} => {parent\_tid=[1813]}, 88) = 1813

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

futex(0x7fc3c6860910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 1811, NULL, FUTEX\_BITSET\_MATCH\_ANY) = 0

futex(0x7fc3c605f910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 1812, NULL, FUTEX\_BITSET\_MATCH\_ANY) = 0

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

write(1, "Estimated area of the circle wit"..., 51Estimated area of the circle with radius 12: 452.4

) = 51

write(1, "Number of threads used: 3\n", 26Number of threads used: 3

) = 26

clock\_gettime(CLOCK\_MONOTONIC, {tv\_sec=34050, tv\_nsec=123046900}) = 0

write(1, "Elapsed time: 5.63342 seconds\n", 30Elapsed time: 5.63342 seconds

) = 30

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В ходе лабораторной работы я познакомился с вычислением площади методом Монте-Карло, работой с мьютексами и многопоточностью. По результатам тестирования видно, что использование нескольких потоков может существенно ускорить выполнение программы, однако всё зависит от правильного выбора количества потоков, который нужно делать, исходя из вычислительных возможностей компьютера.