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-209БВ-24

Студент: Касаева Я.М.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 05.11.25

Москва, 2025

Постановка задачи

Вариант 16.

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

Общий метод и алгоритм решения

Использованные системные вызовы:

- pthread_create() - создание потоков

- pthread_join() - ожидание завершения потоков

- pthread_mutex_lock()/unlock() - синхронизация доступа к общим данным

- clock() - измерение времени выполнения

- getpid() - получение идентификатора процесса

Алгоритм работы программы:

- Инициализация: чтение параметров (радиус, количество точек, потоков)

- Распределение работы: равномерное разделение точек между потоками

- Создание потоков: каждый поток независимо генерирует точки и считает попадания

- Синхронизация: безопасное суммирование результатов через мьютекс

- Вычисление результата: расчет площади круга статистическим методом

- Вывод: отображение результатов и метрик производительности

Код программы

monte_carlo.c

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

typedef struct {
    long long points;
    double radius;
    long long inside;
} thread_data_t;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
long long total_inside = 0;

void* monte_carlo_thread(void* arg) {
```

```c
    thread_data_t* data = (thread_data_t*)arg;
    long long local_inside = 0;

    unsigned int seed = (unsigned int)time(NULL) + (unsigned int)(uintptr_t)arg;

    for (long long i = 0; i < data->points; i++) {
        double x = ((double)rand_r(&seed) / RAND_MAX) * 2 * data->radius - data-
>radius;
        double y = ((double)rand_r(&seed) / RAND_MAX) * 2 * data->radius - data-
>radius;

        if (x*x + y*y <= data->radius * data->radius) {
            local_inside++;
        }
    }

    pthread_mutex_lock(&mutex);
    total_inside += local_inside;
    pthread_mutex_unlock(&mutex);

    return NULL;
}

int main(int argc, char* argv[]) {
    if (argc != 4) {
        printf("Ввод: %s <радиус> <количество_точек> <макс_потоков>\n", argv[0]);
        return 1;
    }

    double radius = atof(argv[1]);
    long long total_points = atoll(argv[2]);
    int max_threads = atoi(argv[3]);

    if (radius <= 0 || total_points <= 0 || max_threads <= 0) {
        printf("Ошибка: все параметры должны быть положительными\n");
        return 1;
    }

    printf("\nПараметры:\n");
    printf("Радиус: %.2f\n", radius);
    printf("Всего точек: %lld\n", total_points);
    printf("Максимум потоков: %d\n", max_threads);

    clock_t start = clock();

    pthread_t threads[max_threads];
    thread_data_t thread_data[max_threads];

    long long points_per_thread = total_points / max_threads;

    for (int i = 0; i < max_threads; i++) {
        thread_data[i].points = points_per_thread;
        thread_data[i].radius = radius;
        thread_data[i].inside = 0;
        pthread_create(&threads[i], NULL, monte_carlo_thread, &thread_data[i]);
    }

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

    double side = 2 * radius;
    double square_area = side * side;
    double circle_area = ((double)total_inside / total_points) * square_area;
    double theoretical_area = M_PI * radius * radius;

    clock_t end = clock();
```

```
    double time_spent = ((double)(end - start)) / CLOCKS_PER_SEC;

    printf("\nРезультаты:\n");
    printf("Точек внутри круга: %lld\n", total_inside);
    printf("Вычисленная площадь: %.6f\n", circle_area);
    printf("Теоретическая площадь: %.6f\n", theoretical_area);
    printf("Погрешность: %.4f%%\n", fabs(circle_area - theoretical_area) /
theoretical_area * 100);
    printf("Время выполнения: %.3f секунд\n", time_spent);

    return 0;
}
```

Протокол работы программы

Тестирование 1:

(base) yanakasaeva@MacBook-Air--YanaK gcc -o monte_carlo monte_carlo.c -lpthread -lm

./monte_carlo 5.0 1000000 4

Параметры:

Радиус: 5.00

Всего точек: 1000000

Максимум потоков: 4

Результаты:

Точек внутри круга: 785360

Вычисленная площадь: 78.536000

Теоретическая площадь: 78.539816

Погрешность: 0.0049%

Время выполнения: 0.032 секунд

Тестирование 2:

(base) yanakasaeva@MacBook-Air--YanaK src % echo "Исследование зависимости от количества потоков:"

for threads in 1 2 4 8; do

echo "Потоков: $threads"

./monte_carlo 5.0 1000000 $threads

echo

done

Исследование зависимости от количества потоков:

Потоков: 1

Параметры:

Радиус: 5.00

Всего точек: 1000000

Максимум потоков: 1

Результаты:

Точек внутри круга: 786379

Вычисленная площадь: 78.637900

Теоретическая площадь: 78.539816

Погрешность: 0.1249%

Время выполнения: 0.031 секунд

Потоков: 2

Параметры:

Радиус: 5.00

Всего точек: 1000000

Максимум потоков: 2

Результаты:

Точек внутри круга: 786009

Вычисленная площадь: 78.600900

Теоретическая площадь: 78.539816

Погрешность: 0.0778%

Время выполнения: 0.022 секунд

Потоков: 4

Параметры:

Радиус: 5.00

Всего точек: 1000000

Максимум потоков: 4

Результаты:

Точек внутри круга: 785761

Вычисленная площадь: 78.576100

Теоретическая площадь: 78.539816

Погрешность: 0.0462%

Время выполнения: 0.020 секунд

Потоков: 8

Параметры:

Радиус: 5.00

Всего точек: 1000000

Максимум потоков: 8

Результаты:

Точек внутри круга: 785690

Вычисленная площадь: 78.569000

Теоретическая площадь: 78.539816

Погрешность: 0.0372%

Время выполнения: 0.021 секунд

Число потоков	Время выполнения (с)	Ускорение	Эффективность
1	0.031	1.00	100%
2	0.022	1.41	70.5%
4	0.020	1.55	38.8%
8	0.021	1.48	18.5%

Тестирование 3:

(base) yanakasaeva@MacBook-Air--YanaK src % echo "Исследование зависимости от объема данных:"

for points in 1000000 10000000 100000000; do

echo "Точек: $points"

./monte_carlo 5.0 $points 4

echo

done

Исследование зависимости от объема данных:

Точек: 1000000

Параметры:

Радиус: 5.00

Всего точек: 1000000

Максимум потоков: 4

Результаты:

Точек внутри круга: 785205

Вычисленная площадь: 78.520500

Теоретическая площадь: 78.539816

Погрешность: 0.0246%

Время выполнения: 0.043 секунд

Точек: 10000000

Параметры:

Радиус: 5.00

Всего точек: 10000000

Максимум потоков: 4

Результаты:

Точек внутри круга: 7853733

Вычисленная площадь: 78.537330

Теоретическая площадь: 78.539816

Погрешность: 0.0032%

Время выполнения: 0.225 секунд

Точек: 100000000

Параметры:

Радиус: 5.00

Всего точек: 100000000

Максимум потоков: 4

Результаты:

Точек внутри круга: 78540200

Вычисленная площадь: 78.540200

Теоретическая площадь: 78.539816

Погрешность: 0.0005%

Время выполнения: 1.833 секунд

Число точек	Время выполнения (с) 4 потока	Погрешность	Относительная скорость
1000000	0.043	0.0246%	1.00x
10000000	0.225	0.0032%	0.19x
100000000	1.833	0.0005%	0.023x

Theards.log:

```
UID        PID  PPID   LWP  C NLWP STIME TTY          TIME CMD
root      1087     1  1087  0    5 20:52 pts/0    00:00:00 ./monte_carlo 5.0 1000000000 4
root      1087     1  1089 99    5 20:52 pts/0    00:00:01 ./monte_carlo 5.0 1000000000 4
root      1087     1  1090 99    5 20:52 pts/0    00:00:01 ./monte_carlo 5.0 1000000000 4
root      1087     1  1091 99    5 20:52 pts/0    00:00:01 ./monte_carlo 5.0 1000000000 4
root      1087     1  1092 99    5 20:52 pts/0    00:00:01 ./monte_carlo 5.0 1000000000 4
```

Strace:

```
1097  execve("./monte_carlo", ["./monte_carlo", "5.0", "10000000", "4"], 0xffffc7a1cb10 /* 8 vars */) = 0
1097  brk(NULL)                         = 0xaaaae9711000
1097  mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xffffaee58000
1097  faccessat(AT_FDCWD, "/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
1097  openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
1097  fstat(3, {st_mode=S_IFREG|0644, st_size=8467, ...}) = 0
1097  mmap(NULL, 8467, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffffaee55000
1097  close(3)                          = 0
1097  openat(AT_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
1097  read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\360\206\2\0\0\0\0\0"..., 832) = 832
1097  fstat(3, {st_mode=S_IFREG|0755, st_size=1722920, ...}) = 0
1097  mmap(NULL, 1892240, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_DENYWRITE, -1, 0) = 0xffffaec51000
1097  mmap(0xffffaec60000, 1826704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffffaec60000
1097  munmap(0xffffaec51000, 61440)     = 0
```

1097 munmap(0xffffaee1e000, 3984) = 0

1097 mprotect(0xffffaedf9000, 81920, PROT_NONE) = 0

1097 mmap(0xffffaee0d000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x19d000) = 0xffffaee0d000

1097 mmap(0xffffaee12000, 49040, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffffaee12000

1097 close(3) = 0

1097 set_tid_address(0xffffaee58fb0) = 1097

1097 set_robust_list(0xffffaee58fc0, 24) = 0

1097 rseq(0xffffaee59600, 0x20, 0, 0xd428bc00) = 0

1097 mprotect(0xffffaee0d000, 12288, PROT_READ) = 0

1097 mprotect(0xaaaad4acf000, 4096, PROT_READ) = 0

1097 mprotect(0xffffaee5d000, 8192, PROT_READ) = 0

1097 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

1097 munmap(0xffffaee55000, 8467) = 0

1097 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

1097 getrandom("\x75\x07\xbf\x03\xc7\x73\x9c\x99", 8, GRND_NONBLOCK) = 8

1097 brk(NULL) = 0xaaaae9711000

1097 brk(0xaaaae9732000) = 0xaaaae9732000

1097 write(1, "\320\237\320\260\321\200\320\260\320\274\320\265\321\202\321\200\321\213:\n", 20) = 20

1097 write(1, "\320\240\320\260\320\264\320\270\321\203\321\201: 5.00\n", 19) = 19

1097 write(1, "\320\222\321\201\320\265\320\263\320\276\321\202\320\276\321\207\320\265\320\272: 10000000\n", 32) = 32

1097 write(1, "\320\234\320\260\320\272\321\201\320\270\320\274\321\203\320\274\320\277\320\276\321\202\320\276\320\272\320\276\320\262:"..., 35) = 35

1097 clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=523832}) = 0

1097 rt_sigaction(SIGRT_1, {sa_handler=0xffffaece2840, sa_mask=[], sa_flags=SA_ONSTACK|SA_RESTART|SA_SIGINFO}, NULL, 8) = 0

1097 rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

1097 mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0xffffae400000

1097 mprotect(0xffffae410000, 8388608, PROT_READ|PROT_WRITE) = 0

1097 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

1097 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0xffffaec0f270, parent_tid=0xffffaec0f270, exit_signal=0, stack=0xffffae400000, stack_size=0x80ea60, tls=0xffffaec0f8e0} => {parent_tid=[1098]}, 88) = 1098

1097 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

1098 rseq(0xffffaec0f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

1097 <... rt_sigprocmask resumed>NULL, 8) = 0

1098 <... rseq resumed>) = 0

1097 mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

1098 set_robust_list(0xffffaec0f280, 24 <unfinished ...>

1097 <... mmap resumed>) = 0xffffada00000

1098 <... set_robust_list resumed>) = 0

1097 mprotect(0xffffada10000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

1098 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

1097 <... mprotect resumed>) = 0

1098 <... rt_sigprocmask resumed>NULL, 8) = 0

1097 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

1097 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0xffffae20f270, parent_tid=0xffffae20f270, exit_signal=0, stack=0xffffada00000, stack_size=0x80ea60, tls=0xffffae20f8e0} => {parent_tid=[1099]}, 88) = 1099

1097 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

1099 rseq(0xffffae20f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

1097 mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

1099 <... rseq resumed>) = 0

1097 <... mmap resumed>) = 0xffffad000000

1099 set_robust_list(0xffffae20f280, 24 <unfinished ...>

1097 mprotect(0xffffad010000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

1099 <... set_robust_list resumed>) = 0

1097 <... mprotect resumed>) = 0

1099 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

1097 rt_sigprocmask(SIG_BLOCK, ~[], <unfinished ...>

1099 <... rt_sigprocmask resumed>NULL, 8) = 0

1097 <... rt_sigprocmask resumed>[], 8) = 0

1097 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0xffffad80f270, parent_tid=0xffffad80f270, exit_signal=0, stack=0xffffad000000, stack_size=0x80ea60, tls=0xffffad80f8e0} => {parent_tid=[1100]}, 88) = 1100

1097 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

1100 rseq(0xffffad80f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

1097 <... rt_sigprocmask resumed>NULL, 8) = 0

1100 <... rseq resumed>) = 0

1097 mmap(NULL, 8454144, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

1100 set_robust_list(0xffffad80f280, 24 <unfinished ...>

1097 <... mmap resumed>) = 0xffffac600000

1100 <... set_robust_list resumed>) = 0

1097 mprotect(0xffffac610000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

1100 rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>

1097 <... mprotect resumed>) = 0

1100 <... rt_sigprocmask resumed>NULL, 8) = 0

1097 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

1097 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0xffffface0f270, parent_tid=0xffffface0f270, exit_signal=0, stack=0xffffac600000, stack_size=0x80ea60, tls=0xffffface0f8e0} => {parent_tid=[1101]}, 88) = 1101

1097 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

1101 rseq(0xffffface0f8c0, 0x20, 0, 0xd428bc00 <unfinished ...>

1097 futex(0xffffaec0f270, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1098, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1101 <... rseq resumed>) = 0

1101 set_robust_list(0xffffface0f280, 24) = 0

1101 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

1098 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

1098 madvise(0xffffae400000, 8314880, MADV_DONTNEED <unfinished ...>

1099 rt_sigprocmask(SIG_BLOCK, ~[RT_1], <unfinished ...>

1098 <... madvise resumed>) = 0

1099 <... rt_sigprocmask resumed>NULL, 8) = 0

1098 exit(0 <unfinished ...>

1099 madvise(0xffffada00000, 8314880, MADV_DONTNEED <unfinished ...>

1098 <... exit resumed>) = ?

1099 <... madvise resumed>) = 0

1097 <... futex resumed>) = 0

1098 +++ exited with 0 +++

1097 futex(0xffffae20f270, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1099, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1099 exit(0) = ?

1100 rt_sigprocmask(SIG_BLOCK, ~[RT_1], <unfinished ...>

1097 <... futex resumed>) = 0

1099 +++ exited with 0 +++

1097 futex(0xffffad80f270, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1100, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1100 <... rt_sigprocmask resumed>NULL, 8) = 0

1100 madvise(0xffffad000000, 8314880, MADV_DONTNEED) = 0

1100 exit(0) = ?

1097 <... futex resumed>) = 0

1100 +++ exited with 0 +++

1097 futex(0xffffface0f270, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 1101, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

1101 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

1101 madvise(0xffffac600000, 8314880, MADV_DONTNEED) = 0

1101 exit(0) = ?

1097 <... futex resumed>) = 0

1101 +++ exited with 0 +++

1097 clock_gettime(CLOCK_PROCESS_CPUTIME_ID, {tv_sec=0, tv_nsec=124337250}) = 0

1097 write(1, "\n", 1) = 1

1097 write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202\321\213:\n", 22) = 22

```
**1097  write(1, "\320\242\320\276\321\207\320\265\320\272
\320\262\320\275\321\203\321\202\321\200\320\270 \320\272\321\200\321\203\320\263"..., 44) = 44**

**1097  write(1,
"\320\222\321\213\321\207\320\270\321\201\320\273\320\265\320\275\320\275\320\260\321\217
\320\277\320\273\320\276\321\211\320"..., 49) = 49**

**1097  write(1,
"\320\242\320\265\320\276\321\200\320\265\321\202\320\270\321\207\320\265\321\201\320\272\320\2
60\321\217 \320\277\320\273\320"..., 53) = 53**

**1097  write(1,
"\320\237\320\276\320\263\321\200\320\265\321\210\320\275\320\276\321\201\321\202\321\214:
0.0169%\n", 32) = 32**

**1097  write(1, "\320\222\321\200\320\265\320\274\321\217
\320\262\321\213\320\277\320\276\320\273\320\275\320\265\320\275\320\270\321\217:"..., 52) = 52**

**1097  write(1, "\n", 1)               = 1**

**1097  write(1, "\320\220\320\275\320\260\320\273\320\270\320\267
\320\277\321\200\320\276\320\270\320\267\320\262\320\276\320\264\320\270\321"..., 51) = 51**

**1097  write(1,
"\320\230\321\201\320\277\320\276\320\273\321\214\320\267\320\276\320\262\320\260\320\275\320\2
76 \320\277\320\276\321\202\320"..., 43) = 43**

**1097  getpid()                      = 1097**

**1097  write(1, "ID \320\277\321\200\320\276\321\206\320\265\321\201\321\201\320\260:
1097\n", 26) = 26**

**1097  getpid()                      = 1097**

**1097  write(1, "\n\320\224\320\273\321\217
\320\277\321\200\320\276\321\201\320\274\320\276\321\202\321\200\320\260
\320\277\320\276\321"..., 73) = 73**

**1097  exit_group(0)                  = ?**

1097  +++ exited with 0 +++
```

Вывод

В ходе лабораторной работы была реализована многопоточная программа для вычисления площади круга методом Монте-Карло. Программа использует мьютексы для синхронизации потоков и корректно работает с разным количеством потоков.

Результаты исследований показали: многопоточность ускоряет вычисления, но не так эффективно, как хотелось бы. При 4 потоках программа работает в 1.55 раза быстрее, чем с одним потоком. Однако эффективность использования процессорного времени при этом составляет всего 38.8%. Главная причина низкой эффективности - накладные расходы. Создание потоков и постоянная блокировка мьютекса для защиты общего счетчика занимают значительное время. Когда потоков становится больше (8), ситуация ухудшается - эффективность падает до 18.5%, так как процессор тратит больше времени на переключение между потоками, чем на полезные вычисления.

С увеличением объема данных точность расчетов закономерно растет: при 100 миллионов точек погрешность составляет всего 0.0005%. Относительная скорость выполнения закономерно снижается с ростом объема данных, демонстрируя пропорциональность времени вычислений количеству обрабатываемых точек.