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-211Б-23

Студент: Лапин Д.А.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 10.01.25

Москва, 2025

Постановка задачи

Вариант 16.

Цель работы

Приобретение практических навыков в:

- Управление процессами в ОС
- Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке 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.

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

Общий метод и алгоритм решения

Использованные системные вызовы:

- ssize_t write(int fd, const void *buf, size_t count); - записывает count байт из буфера в файл.
- int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
- void *(*start)(void *), void *arg)– создание потока
- int pthread_join (pthread_t THREAD_ID, void ** DATA) – ожидание завершения потока
- int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr) – инициализация мьютекса
- int pthread_mutex_lock(pthread_mutex_t *mutex) – блокировка мьютекса ●

int pthread_mutex_unlock(pthread_mutex_t *mutex) – разблокировка мьютекса

- int pthread_mutex_destroy(pthread_mutex_t *mutex) – удаление мьютекса

Программа вычисляет площадь окружности методом Монте-Карло: генерирует случайные точки в квадрате с заданным радиусом и подсчитывает долю точек, попавших внутрь окружности, для оценки площади. Для многопоточности она делит работу на задачи фиксированного размера (`CHUNK_SIZE`) и распределяет их между потоками, число которых задано аргументом командной строки.

Каждый поток генерирует свои точки и подсчитывает, сколько из них находятся внутри окружности, после чего обновляет глобальный счётчик `g_insideCount` с использованием мьютексов для обеспечения потокобезопасности. Генерация и подсчёт продолжаются, пока задачи не завершатся.

После завершения всех потоков программа оценивает площадь окружности как долю точек внутри круга, умноженную на площадь квадрата, и выводит результат с помощью собственных функций преобразования чисел в строки, избегая использования стандартного `printf`.

Количество потоков	Время, с	Ускорение	Эффективность
1	2,912	1	1
2	1,495	1,947826087	0,973913043
3	1,225	2,377142857	0,792380952
4	0,769	3,786736021	0,946684005
5	0,656	4,43902439	0,887804878
6	0,594	4,902356902	0,817059484
7	0,510	5,709803922	0,815686275
8	0,466	6,248927039	0,78111588
9	0,405	7,190123457	0,798902606
10	0,369	7,891598916	0,789159892
11	0,343	8,489795918	0,771799629
12	0,327	8,905198777	0,742099898

Данные подсчитаны при количестве точек = 214483647

Количество логических ядер на моем ПК: 12

Количество точек	Время, с
10	0.001
100	0.001
1000	0.001
10000	0.001
100000	0.001
1000000	0.002
10000000	0.020
100000000	0.155
1000000000	1.526
10000000000	16.615

Код программы

main.c

```
/*****************************************************************************
* Запуск:
*     ./lab2 5.0 4
*  где 5.0  — радиус окружности,
*      4    — максимальное число одновременно работающих потоков.
*****************************************************************************/

#include <stdlib.h>    /* atof, atoi, rand, srand */
#include <unistd.h>    /* write, _exit, getpid */
#include <pthread.h>   /* pthread_create, pthread_join, pthread_mutex_* */
#include <time.h>      /* clock_gettime, time (для srand) */
#include <string.h>    /* для обработки строк в функциях конвертации */

//---------------------------------------------------------------------------
// Настройки «метода Монте-Карло»:

/*
* Общее количество случайных точек, которые будем генерировать.
* Чем больше точек, тем точнее результат, но дольше время вычисления.
*/
#define TOTAL_POINTS 1000000000

/*
* Размер "задачи" — количество точек, обрабатываемых одним потоком за один раз.
* Если TOTAL_POINTS=1_000_000, а CHUNK_SIZE=100_000, то будет 10 "задач".
*/
#define CHUNK_SIZE 10000000

//---------------------------------------------------------------------------
// Глобальные переменные для управления задачами:

static int g_totalTasks = 0;         // Общее число «задач»
static int g_nextTask = 0;          // Индекс следующей «задачи»
```

```
static long g_insideCount = 0;      // Счётчик точек, попавших внутрь окружности
static double g_radius = 1.0;       // Радиус окружности (считывается из argv)


static pthread_mutex_t g_taskMutex   = PTHREAD_MUTEX_INITIALIZER; // для g_nextTask
static pthread_mutex_t g_resultMutex = PTHREAD_MUTEX_INITIALIZER; // для g_insideCount


//--------------------------------------------------------------------------
// Функция my_itoa: преобразует целое число value в десятичную строку в buf.


static int my_itoa(long value, char *buf) {
   if (value < 0) value = -value;


   char temp[32];
   int i = 0;
   do {
       temp[i++] = (char)('0' + (value % 10));
       value /= 10;
   } while (value > 0);


   int len = 0;
   while (i > 0) {
       buf[len++] = temp[--i];
   }
   buf[len] = '\0';
   return len;
}


//--------------------------------------------------------------------------
// Функция my_dtoa: упрощённое преобразование double в строку.


static int my_dtoa(double value, char *buf, int precision) {
   int idx = 0;
   if (value < 0.0) {
       buf[idx++] = '-';
       value = -value;
```

```
}

long whole = (long) value;
double frac = value - (double) whole;

char tmp[64];
int lenWhole = my_itoa(whole, tmp);
for (int i = 0; i < lenWhole; i++) {
    buf[idx++] = tmp[i];
}

// Добавим точку
buf[idx++] = '.';

double mult = 1.0;
for (int i = 0; i < precision; i++) {
    mult *= 10.0;
}
long fracVal = (long) (frac * mult);

char tmpFrac[64];
int lenFrac = my_itoa(fracVal, tmpFrac);

// Дописываем ведущие нули, если надо
if (lenFrac < precision) {
    int zeros = precision - lenFrac;
    while (zeros--) {
        buf[idx++] = '0';
    }
}
for (int i = 0; i < lenFrac; i++) {
    buf[idx++] = tmpFrac[i];
}

buf[idx] = '\0';
return idx;
```

```
}

//---------------------------------------------------------------------------
// Упрощённый вывод строки

static void write_str(const char *s) {
   size_t len = 0;
   while (s[len] != '\0') {
       len++;
   }
   write(STDOUT_FILENO, s, len);
}

//---------------------------------------------------------------------------
// Функция получения индекса следующей задачи:

static int get_next_task(void) {
   pthread_mutex_lock(&g_taskMutex);
   int taskIndex = -1;
   if (g_nextTask < g_totalTasks) {
       taskIndex = g_nextTask;
       g_nextTask++;
   }
   pthread_mutex_unlock(&g_taskMutex);
   return taskIndex;
}

//---------------------------------------------------------------------------
// Функция, которую выполняет каждый поток (worker)

static void *thread_worker(void *arg) {
   (void) arg; // не используем, но аргумент оставить для сигнатуры pthread

   unsigned seed = (unsigned) time(NULL) ^ (unsigned) pthread_self();
   srand(seed);
```

```
    while (1) {
        int taskIndex = get_next_task();
        if (taskIndex < 0) {
            break;
        }

        long pointsPerTask = CHUNK_SIZE;
        long localInside = 0;

        for (long i = 0; i < pointsPerTask; i++) {
            double x = ((double) rand_r(&seed) / (double) RAND_MAX) * 2.0 *
g_radius - g_radius;
            double y = ((double) rand_r(&seed) / (double) RAND_MAX) * 2.0 *
g_radius - g_radius;

            double dist2 = x * x + y * y;
            if (dist2 <= (g_radius * g_radius)) {
                localInside++;
            }
        }

        pthread_mutex_lock(&g_resultMutex);
        g_insideCount += localInside;
        pthread_mutex_unlock(&g_resultMutex);
    }

    return NULL;
}

//---------------------------------------------------------------------------
// Точка входа в программу

int main(int argc, char *argv[]) {
    // Переменные для времени
    struct timespec startTime, endTime;

    // Засекаем время в начале
```

```
clock_gettime(CLOCK_MONOTONIC, &startTime);


if (argc < 3) {
    write_str("Usage: ./lab2 <radius> <max_threads>\n");
    _exit(1);
}


g_radius = atof(argv[1]);
if (g_radius <= 0.0) {
    write_str("Radius must be positive!\n");
    _exit(1);
}


int maxThreads = atoi(argv[2]);
if (maxThreads <= 0) {
    write_str("maxThreads must be positive!\n");
    _exit(1);
}


g_totalTasks = (int)(TOTAL_POINTS / CHUNK_SIZE);


pthread_t *threads = (pthread_t *) malloc(sizeof(pthread_t) * maxThreads);
if (!threads) {
    write_str("Memory allocation error\n");
    _exit(1);
}


g_insideCount = 0;
g_nextTask = 0;


// Запуск потоков
for (int i = 0; i < maxThreads; i++) {
    pthread_create(&threads[i], NULL, thread_worker, NULL);
}


// Ожидание завершения всех потоков
```

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

    free(threads);

    // Оценка площади окружности методом Монте-Карло
    double fraction = (double) g_insideCount / (double) TOTAL_POINTS;
    double area = fraction * 4.0 * (g_radius * g_radius);

    char bufArea[128];
    my_dtoa(area, bufArea, 6);

    write_str("Calculated area = ");
    write_str(bufArea);
    write_str("\n");

    // Засекаем время в конце
    clock_gettime(CLOCK_MONOTONIC, &endTime);

    // Считаем прошедшее время в секундах (double)
    double elapsedSeconds = (endTime.tv_sec - startTime.tv_sec)
                            + (endTime.tv_nsec - startTime.tv_nsec) /
1000000000.0;

    // Выведем время
    char bufTime[128];
    my_dtoa(elapsedSeconds, bufTime, 6);
    write_str("Elapsed time: ");
    write_str(bufTime);
    write_str(" seconds\n");

    _exit(0);
}
```

Протокол работы программы

Тесты:

1) **4 потока**

```
/mnt/c/Users/mrbor/CLionProjects/osi/lab2/cmake-build-debug/lab2 19.0 4
Calculated area = 1134.119045
Elapsed time: 6.210882 seconds

Process finished with exit code 0
```

2) **8 потоков**

```
/mnt/c/Users/mrbor/CLionProjects/osi/lab2/cmake-build-debug/lab2 19.0 8
Calculated area = 1134.112008
Elapsed time: 4.215101 seconds

Process finished with exit code 0
```

Strace:

```
    288280 execve("./main", ["./main", "19", "8"], 0x7fff9b5f8008 /* 26 vars */) = 0

    288280 brk(NULL)                        = 0x55a0055fb000

    288280 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fe5eee18000

    288280 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)

    288280 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

    288280 fstat(3, {st_mode=S_IFREG|0644, st_size=21363, ...}) = 0

    288280 mmap(NULL, 21363, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fe5eee12000

    288280 close(3)                         = 0

    288280 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) =
3

    288280 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"..., 832) = 832

    288280 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

    288280 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
```

```
     288280 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

     288280 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fe5eec00000

     288280 mmap(0x7fe5eec28000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fe5eec28000

     288280 mmap(0x7fe5eedb0000, 323584, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7fe5eedb0000

     288280 mmap(0x7fe5eedff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7fe5eedff000

     288280 mmap(0x7fe5eee05000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fe5eee05000

     288280 close(3)                         = 0

     288280 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0x7fe5eebfd000

     288280 arch_prctl(ARCH_SET_FS, 0x7fe5eebfd740) = 0

     288280 set_tid_address(0x7fe5eebfda10)  = 288280

     288280 set_robust_list(0x7fe5eebfda20, 24) = 0

     288280 rseq(0x7fe5eebfe060, 0x20, 0, 0x53053053) = 0

     288280 mprotect(0x7fe5eedff000, 16384, PROT_READ) = 0

     288280 mprotect(0x559febc01000, 4096, PROT_READ) = 0

     288280 mprotect(0x7fe5eee50000, 8192, PROT_READ) = 0

     288280 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

     288280 munmap(0x7fe5eee12000, 21363)    = 0

     288280 getrandom("\xab\xbb\xfa\x4e\x07\xbb\x2f\x97", 8, GRND_NONBLOCK) = 8

     288280 brk(NULL)                        = 0x55a0055fb000

     288280 brk(0x55a00561c000)              = 0x55a00561c000

     288280 rt_sigaction(SIGRT_1, {sa_handler=0x7fe5eec99520, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7fe5eec45320},
NULL, 8) = 0

     288280 rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0)
= 0x7fe5ee3fc000

     288280 mprotect(0x7fe5ee3fd000, 8388608, PROT_READ|PROT_WRITE) = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5eebfc990,
parent_tid=0x7fe5eebfc990, exit_signal=0, stack=0x7fe5ee3fc000, stack_size=0x7fff80,
tls=0x7fe5eebfc6c0} => {parent_tid=[288281]}, 88) = 288281

     288281 rseq(0x7fe5eebfcfe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
```

```
     288281 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288281 set_robust_list(0x7fe5eebfc9a0, 24 <unfinished ...>

     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     288281 <... set_robust_list resumed>)   = 0

     288280 <... mmap resumed>)              = 0x7fe5edbfb000

     288280 mprotect(0x7fe5edbfc000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     288281 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288280 <... mprotect resumed>)          = 0

     288281 <... rt_sigprocmask resumed>NULL, 8) = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5ee3fb990,
parent_tid=0x7fe5ee3fb990, exit_signal=0, stack=0x7fe5edbfb000, stack_size=0x7fff80,
tls=0x7fe5ee3fb6c0} => {parent_tid=[288282]}, 88) = 288282

     288282 rseq(0x7fe5ee3fbfe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288282 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288282 set_robust_list(0x7fe5ee3fb9a0, 24 <unfinished ...>

     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     288282 <... set_robust_list resumed>)   = 0

     288280 <... mmap resumed>)              = 0x7fe5ed3fa000

     288282 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288280 mprotect(0x7fe5ed3fb000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     288282 <... rt_sigprocmask resumed>NULL, 8) = 0

     288280 <... mprotect resumed>)          = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5edbfa990,
parent_tid=0x7fe5edbfa990, exit_signal=0, stack=0x7fe5ed3fa000, stack_size=0x7fff80,
tls=0x7fe5edbfa6c0} => {parent_tid=[288283]}, 88) = 288283

     288283 rseq(0x7fe5edbfafe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288283 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288283 set_robust_list(0x7fe5edbfa9a0, 24 <unfinished ...>
```

```
     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     288283 <... set_robust_list resumed>)   = 0

     288280 <... mmap resumed>)              = 0x7fe5ecbf9000

     288283 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288280 mprotect(0x7fe5ecbfa000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     288283 <... rt_sigprocmask resumed>NULL, 8) = 0

     288280 <... mprotect resumed>)          = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5ed3f9990,
parent_tid=0x7fe5ed3f9990, exit_signal=0, stack=0x7fe5ecbf9000, stack_size=0x7fff80,
tls=0x7fe5ed3f96c0} => {parent_tid=[288284]}, 88) = 288284

     288284 rseq(0x7fe5ed3f9fe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288284 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288284 set_robust_list(0x7fe5ed3f99a0, 24 <unfinished ...>

     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     288284 <... set_robust_list resumed>)   = 0

     288280 <... mmap resumed>)              = 0x7fe5ec3f8000

     288284 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288280 mprotect(0x7fe5ec3f9000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     288284 <... rt_sigprocmask resumed>NULL, 8) = 0

     288280 <... mprotect resumed>)          = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5ecbf8990,
parent_tid=0x7fe5ecbf8990, exit_signal=0, stack=0x7fe5ec3f8000, stack_size=0x7fff80,
tls=0x7fe5ecbf86c0} => {parent_tid=[288285]}, 88) = 288285

     288285 rseq(0x7fe5ecbf8fe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288285 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288285 set_robust_list(0x7fe5ecbf89a0, 24 <unfinished ...>

     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     288285 <... set_robust_list resumed>)   = 0
```

```
     288280 <... mmap resumed>)              = 0x7fe5ebbf7000

     288285 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288280 mprotect(0x7fe5ebbf8000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     288285 <... rt_sigprocmask resumed>NULL, 8) = 0

     288280 <... mprotect resumed>)          = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5ec3f7990,
parent_tid=0x7fe5ec3f7990, exit_signal=0, stack=0x7fe5ebbf7000, stack_size=0x7fff80,
tls=0x7fe5ec3f76c0} => {parent_tid=[288286]}, 88) = 288286

     288286 rseq(0x7fe5ec3f7fe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288286 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288286 set_robust_list(0x7fe5ec3f79a0, 24 <unfinished ...>

     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     288286 <... set_robust_list resumed>)   = 0

     288280 <... mmap resumed>)              = 0x7fe5eb3f6000

     288286 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288280 mprotect(0x7fe5eb3f7000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

     288286 <... rt_sigprocmask resumed>NULL, 8) = 0

     288280 <... mprotect resumed>)          = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5ebbf6990,
parent_tid=0x7fe5ebbf6990, exit_signal=0, stack=0x7fe5eb3f6000, stack_size=0x7fff80,
tls=0x7fe5ebbf66c0} => {parent_tid=[288287]}, 88) = 288287

     288287 rseq(0x7fe5ebbf6fe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288287 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288287 set_robust_list(0x7fe5ebbf69a0, 24 <unfinished ...>

     288280 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

     288287 <... set_robust_list resumed>)   = 0

     288280 <... mmap resumed>)              = 0x7fe5eabf5000

     288287 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288280 mprotect(0x7fe5eabf6000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
```

```
     288287 <... rt_sigprocmask resumed>NULL, 8) = 0

     288280 <... mprotect resumed>)          = 0

     288280 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

     288280
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fe5eb3f5990,
parent_tid=0x7fe5eb3f5990, exit_signal=0, stack=0x7fe5eabf5000, stack_size=0x7fff80,
tls=0x7fe5eb3f56c0} => {parent_tid=[288288]}, 88) = 288288

     288288 rseq(0x7fe5eb3f5fe0, 0x20, 0, 0x53053053 <unfinished ...>

     288280 rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

     288288 <... rseq resumed>)              = 0

     288280 <... rt_sigprocmask resumed>NULL, 8) = 0

     288288 set_robust_list(0x7fe5eb3f59a0, 24 <unfinished ...>

     288280 futex(0x7fe5eebfc990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 288281,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

     288288 <... set_robust_list resumed>)   = 0

     288288 rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

     288283 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

     288283 madvise(0x7fe5ed3fa000, 8368128, MADV_DONTNEED) = 0

     288283 exit(0)                          = ?

     288283 +++ exited with 0 +++

     288284 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

     288284 madvise(0x7fe5ecbf9000, 8368128, MADV_DONTNEED) = 0

     288284 exit(0)                          = ?

     288284 +++ exited with 0 +++

     288286 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

     288286 madvise(0x7fe5ebbf7000, 8368128, MADV_DONTNEED) = 0

     288286 exit(0)                          = ?

     288286 +++ exited with 0 +++

     288288 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

     288288 madvise(0x7fe5eabf5000, 8368128, MADV_DONTNEED) = 0

     288288 exit(0)                          = ?

     288288 +++ exited with 0 +++

     288281 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

     288281 madvise(0x7fe5ee3fc000, 8368128, MADV_DONTNEED) = 0

     288281 exit(0)                          = ?

     288280 <... futex resumed>)             = 0

     288281 +++ exited with 0 +++
```

```
    288280 futex(0x7fe5ee3fb990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 288282,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

    288285 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

    288285 madvise(0x7fe5ec3f8000, 8368128, MADV_DONTNEED) = 0

    288285 exit(0)                          = ?

    288285 +++ exited with 0 +++

    288287 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

    288287 madvise(0x7fe5eb3f6000, 8368128, MADV_DONTNEED) = 0

    288287 exit(0)                          = ?

    288287 +++ exited with 0 +++

    288282 rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

    288282 madvise(0x7fe5edbfb000, 8368128, MADV_DONTNEED) = 0

    288282 exit(0)                          = ?

    288280 <... futex resumed>)             = 0

    288282 +++ exited with 0 +++

    288280 munmap(0x7fe5ee3fc000, 8392704)  = 0

    288280 munmap(0x7fe5edbfb000, 8392704)  = 0

    288280 munmap(0x7fe5ed3fa000, 8392704)  = 0

    288280 munmap(0x7fe5ecbf9000, 8392704)  = 0

    288280 write(1, "Calculated area = ", 18) = 18

    288280 write(1, "1134.116205", 11)      = 11

    288280 write(1, "\n", 1)                = 1

    288280 write(1, "Elapsed time: ", 14)   = 14

    288280 write(1, "4.023168", 8)          = 8

    288280 write(1, " seconds\n", 9)        = 9

    288280 exit_group(0)                    = ?

    288280 +++ exited with 0 +++
```

Вывод

В ходе лабораторной работы приобретены навыки управления потоками в ОС и их синхронизации с использованием стандартных средств pthread для Unix. Разработана многопоточная программа для расчёта площади круга методом Монте-Карло, с возможностью ограничения количества одновременно работающих потоков. Реализация успешно продемонстрировала корректность вычислений и эффективность синхронизации, а также важность грамотного управления потоками для оптимизации работы программы.