

**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»**

Выполнил: Ю.В. Павлова
Группа: М80-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель работы: Приобретение практических навыков в:

- Управлении процессами в ОС
- Обеспечение обмена данных между процессами посредством каналов

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.

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.

Вариант 16: Задаётся радиус окружности. Необходимо с помощью метода Монте-Карло рассчитать её площадь.

Метод решения

Алгоритм решения задачи:

1. Пользователь запускает программу с двумя параметрами: радиус окружности и максимальное количество потоков.
2. Программа генерирует 1 000 000 000 (миллиард) случайных точек в квадрате, описанном вокруг окружности заданного радиуса.
3. Точки равномерно распределяются между потоками.
4. Каждый поток независимо генерирует свои точки и подсчитывает, сколько из них попадают внутрь окружности.
5. Для синхронизации доступа к общему счетчику попаданий используется мьютекс.
6. После завершения работы всех потоков вычисляется площадь окружности по формуле:

$$S = 4 \times R^2 \times \frac{\text{количество попавших точек}}{\text{общее количество точек}}$$

Математическая основа метода Монте-Карло:

Метод основан на использовании случайных выборок для решения детерминированных задач. Вероятность того, что случайно выбранная точка внутри квадрата со стороной $2R$ попадет в окружность радиуса R , равна отношению площади окружности к площади квадрата:

$$P = \frac{\pi R^2}{(2R)^2} = \frac{\pi}{4}$$

Следовательно, площадь окружности можно оценить как:

$$S \approx 4 \times R^2 \times \frac{M}{N}$$

где M — количество точек внутри окружности, N — общее количество точек.

Описание программы

Архитектура программы:

```
lab2/  
  build/  
  include/  
    exceptions.h  
    threads.h  
  src/  
    threads.cpp  
main.cpp
```

Основные компоненты:

- `main.cpp` — основная программа, реализующая метод Монте-Карло для вычисления площади окружности.
- `include/exceptions.h` — объявление классов исключений.
- `include/threads.h` — объявление класса `Thread` для работы с потоками.
- `src/threads.cpp` — реализация методов класса `Thread`.

Основные функции и структуры:

- `struct GlobalResult` — хранит общие данные для всех потоков (общее количество попаданий и мьютекс для синхронизации).
- `struct ThreadArgs` — аргументы, передаваемые в каждый поток (радиус, количество точек для генерации, указатель на глобальные данные).
- `void* calculate_area_chunk(void* args)` — функция, выполняемая в каждом потоке. Генерирует случайные точки и подсчитывает попадания внутрь окружности.
- `int main(int argc, char* argv[])` — точка входа в программу, управляет созданием и синхронизацией потоков.

Программа использует мьютекс (`pthread_mutex_t`) для синхронизации доступа к общему счетчику, что предотвращает гонки данных при одновременной записи от нескольких потоков.

Результаты

Для исследования зависимости производительности от количества потоков были проведены замеры времени выполнения программы с различным количеством потоков на фиксированном объеме данных (1 млрд точек) и радиусе окружности $R = 1.0$.

Количество потоков	Время (сек)	Ускорение	Эффективность
1	253.97	1.00	1.00
2	120.96	2.10	1.05
4	60.02	4.23	1.06
8	37.72	6.73	0.84
16	31.80	7.99	0.50
32	28.61	8.88	0.28

Таблица 1: Зависимость времени выполнения от количества потоков

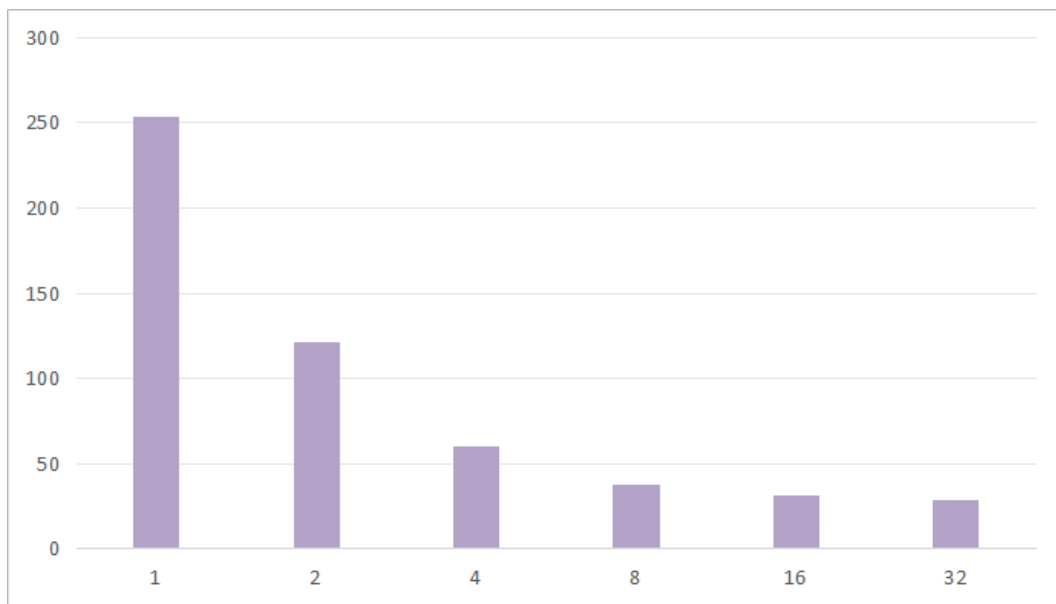


Рис. 1: Диаграмма зависимости времени от количества потоков

Выводы

- Наилучшая эффективность (106%) достигается при 4 потоках
- Максимальное ускорение составляет 8.9 раз при 32 потоках
- Оптимальный диапазон — 4-8 потоков для данного алгоритма
- После 8 потоков наблюдается значительное снижение эффективности
- Алгоритм хорошо масштабируется до количества физических ядер процессора

Исходный код

exceptions.h

```
1 #pragma once
2
3 #include <exception>
4 #include <string>
5
6 namespace exceptions {
7     class ThreadException: public std::exception {
8     public:
9         explicit ThreadException(const std::string& text): error_message_(text)
10         {}
11         const char* what() const noexcept override {
12             return error_message_.c_str();
13         }
14     private:
15         std::string error_message_;
16     };
17
18     class ArgumentException: public std::exception {
19     public:
```

```

19     explicit ArgumentException(const std::string& text): error_message_(
    text) {}
20     const char* what() const noexcept override {
21         return error_message_.c_str();
22     }
23 private:
24     std::string error_message_;
25 };
26 }

```

Листинг 1: Заголовочный файл исключений

threads.h

```

1 #pragma once
2
3 #include "exceptions.h"
4
5 namespace thread {
6     using threadFunc = void* (*)(void*);
7     struct threadInfo;
8
9     class Thread {
10     private:
11         threadFunc func;
12         threadInfo* pimpl;
13         bool is_joined = true;
14
15     public:
16
17         explicit Thread(threadFunc func);
18         Thread(const Thread&) = delete;
19         Thread& operator=(const Thread&) = delete;
20         Thread(Thread&& other) noexcept;
21         Thread& operator=(Thread&& other) noexcept;
22         void Run(void* threadData);
23         void Join();
24         ~Thread() noexcept;
25
26     };
27
28 }

```

Листинг 2: Заголовочный файл потоков

threads.cpp

```

1 #include "threads.h"
2 #include <iostream>
3 #include <utility>
4 #include <pthread.h>
5
6 namespace thread {
7     struct threadInfo {
8         pthread_t thread = 0;
9     };
10
11     Thread::Thread(threadFunc func) : func(func), is_joined(true) {

```

```

12     pimpl = new threadInfo();
13 }
14
15 Thread::Thread(Thread&& other) noexcept
16 : func(other.func), pimpl(other.pimpl), is_joined(other.is_joined) {
17     other.pimpl = nullptr;
18     other.is_joined = true;
19 }
20
21 Thread& Thread::operator=(Thread&& other) noexcept {
22     if (pimpl != nullptr && !is_joined) {
23         pthread_detach(pimpl->thread);
24     }
25     delete pimpl;
26
27     Thread temp = std::move(other);
28     std::swap(func, temp.func);
29     std::swap(pimpl, temp.pimpl);
30     std::swap(is_joined, temp.is_joined);
31
32     return *this;
33 }
34
35 void Thread::Run(void* threadData) {
36     if (pimpl == nullptr) {
37         throw exceptions::ThreadException("Thread is not initialized.")
;
38     }
39
40     if (!is_joined) {
41         throw exceptions::ThreadException("Thread is already running.")
;
42     }
43
44     int result = pthread_create(
45         &(pimpl->thread),
46         nullptr,
47         func,
48         threadData
49     );
50
51     if (result != 0) {
52         pimpl->thread = 0;
53         throw exceptions::ThreadException("Failed to create thread");
54     }
55     is_joined = false;
56 }
57
58 void Thread::Join() {
59     if (pimpl == nullptr || is_joined) {
60         return;
61     }
62
63     int result = pthread_join(pimpl->thread, nullptr);
64
65     if (result != 0) {
66         throw exceptions::ThreadException("Failed to join thread");
67     }
68
69     pimpl->thread = 0;

```

```

70         is_joined = true;
71     }
72
73     Thread::~Thread() noexcept {
74         if (pimpl != nullptr) {
75             if (!is_joined) {
76                 std::cerr << "Warning: Thread resource leaked (not joined).
77 Detaching..." << std::endl;
78                 pthread_detach(pimpl->thread);
79             }
80             delete pimpl;
81         }
82     }

```

Листинг 3: Реализация работы с потоками

main.cpp

```

1  #include "threads.h"
2  #include "exceptions.h"
3
4  #include <iostream>
5  #include <vector>
6  #include <iomanip>
7  #include <cmath>
8  #include <stdexcept>
9  #include <chrono>
10 #include <random>
11 #include <string>
12
13 struct GlobalResult {
14     long long total_hits = 0;
15     pthread_mutex_t mutex;
16 };
17
18 struct ThreadArgs {
19     double radius;
20     long long points_to_generate;
21     GlobalResult* global_res;
22 };
23
24 void* calculate_area_chunk(void* args) {
25     ThreadArgs* thread_args = static_cast<ThreadArgs*>(args);
26
27     double R = thread_args->radius;
28     long long N = thread_args->points_to_generate;
29     double R_squared = R * R;
30     long long local_hits = 0;
31
32     std::random_device rd;
33     std::mt19937 generator(rd());
34     std::uniform_real_distribution<> distrib(-R, R);
35
36     for (long long i = 0; i < N; ++i) {
37         double x = distrib(generator);
38         double y = distrib(generator);
39
40         if ((x * x + y * y) <= R_squared) {

```

```

41         local_hits++;
42     }
43 }
44
45 GlobalResult* global_res = thread_args->global_res;
46
47 pthread_mutex_lock(&global_res->mutex);
48 global_res->total_hits += local_hits;
49 pthread_mutex_unlock(&global_res->mutex);
50
51 return nullptr;
52 }
53
54 int main(int argc, char* argv[]) {
55     double radius;
56     int max_threads;
57     const long long TOTAL_POINTS = 1000000000LL;
58
59     try {
60         if (argc != 3) {
61             throw exceptions::ArgumentException("Incorrect number of arguments.
62 ");
63         }
64
65         radius = std::stod(argv[1]);
66         max_threads = std::stoi(argv[2]);
67
68         if (radius <= 0 || max_threads <= 0) {
69             throw exceptions::ArgumentException("Radius and Max_Threads must be
70 positive numbers.");
71         }
72
73         catch (const std::exception& e) {
74             std::cerr << "Startup Error: " << e.what() << std::endl;
75             std::cerr << "Usage: " << argv[0] << " <Radius> <Max_Threads>" << std::
76 endl;
77             return 1;
78         }
79
80         long long points_per_thread = TOTAL_POINTS / max_threads;
81
82         GlobalResult global_res;
83         if (pthread_mutex_init(&global_res.mutex, nullptr) != 0) {
84             std::cerr << "Fatal error: Failed to initialize mutex." << std::endl;
85             return 1;
86         }
87
88         std::vector<thread::Thread> thread_pool;
89         std::vector<ThreadArgs> args_data(max_threads);
90
91         for (int i = 0; i < max_threads; ++i) {
92             args_data[i].radius = radius;
93             args_data[i].points_to_generate = points_per_thread;
94             args_data[i].global_res = &global_res;
95
96             thread_pool.emplace_back(calculate_area_chunk);
97         }
98
99         auto start_time = std::chrono::high_resolution_clock::now();

```



```

98
99     try {
100         for (int i = 0; i < max_threads; ++i) {
101             thread_pool[i].Run(&args_data[i]);
102         }
103
104         for (int i = 0; i < max_threads; ++i) {
105             thread_pool[i].Join();
106         }
107     }
108     catch (const exceptions::ThreadException& e) {
109         std::cerr << "Critical Error (Create): " << e.what() << std::endl;
110         pthread_mutex_destroy(&global_res.mutex);
111         return 1;
112     }
113
114     auto end_time = std::chrono::high_resolution_clock::now();
115     double execution_time = std::chrono::duration<double>(end_time - start_time)
116     .count();
117
118     long long total_hits = global_res.total_hits;
119     double area_square = 4.0 * radius * radius;
120
121     double estimated_area = area_square * (static_cast<double>(total_hits) /
122     TOTAL_POINTS);
123     double analytical_area = M_PI * radius * radius;
124
125     std::cout << "\n--- Monte Carlo Simulation Results ---" << std::endl;
126     std::cout << "Threads Used:      " << max_threads << std::endl;
127     std::cout << "Total Points:      " << TOTAL_POINTS << std::endl;
128     std::cout << "Total Hits:        " << total_hits << std::endl;
129     std::cout << "Estimated Area:    " << std::fixed << std::setprecision(8) <<
130     estimated_area << std::endl;
131     std::cout << "Analytical Area:   " << std::fixed << std::setprecision(8) <<
132     analytical_area << std::endl;
133     std::cout << "Execution Time:    " << std::fixed << std::setprecision(6) <<
134     execution_time << " seconds" << std::endl;
135
136     pthread_mutex_destroy(&global_res.mutex);
137
138     return 0;
139 }

```

Листинг 4: Основная программа

strace

```

1 execve("./monte_carlo", ["/monte_carlo", "1.0", "4"], 0x7fffa41ae6e8 /* 37
   vars */) = 0
2 brk(NULL)                                = 0x55e38eb1f000
3 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
   x7f69f6291000
4 access("/etc/ld.so.preload", R_OK)        = -1 ENOENT (No such file or directory)
5 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
6 fstat(3, {st_mode=S_IFREG|0644, st_size=21863, ...}) = 0
7 mmap(NULL, 21863, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f69f628b000
8 close(3)                                  = 0
9 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) =
   3

```

```

10 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"...,
    832) = 832
11 fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
12 mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
    x7f69f600d000
13 mmap(0x7f69f60aa000, 1343488, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x9d000) = 0x7f69f60aa000
14 mmap(0x7f69f61f2000, 552960, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
    0x1e5000) = 0x7f69f61f2000
15 mmap(0x7f69f6279000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x26b000) = 0x7f69f6279000
16 mmap(0x7f69f6287000, 12608, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS, -1, 0) = 0x7f69f6287000
17 close(3) = 0
18 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
19 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"...,
    832) = 832
20 fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
21 mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f69f5f24000
22 mmap(0x7f69f5f34000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x10000) = 0x7f69f5f34000
23 mmap(0x7f69f5fb3000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
    0x8f000) = 0x7f69f5fb3000
24 mmap(0x7f69f600b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0xe7000) = 0x7f69f600b000
25 close(3) = 0
26 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
27 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"...,
    832) = 832
28 fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
29 mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f69f5efa000
30 mmap(0x7f69f5efa000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x4000) = 0x7f69f5efa000
31 mmap(0x7f69f5f1e000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
    0x28000) = 0x7f69f5f1e000
32 mmap(0x7f69f5f22000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x2b000) = 0x7f69f5f22000
33 close(3) = 0
34 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
35 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
36 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"...,
    784, 64) = 784
37 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
38 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"...,
    784, 64) = 784
39 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
    x7f69f5ce4000
40 mmap(0x7f69f5d0c000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x28000) = 0x7f69f5d0c000
41 mmap(0x7f69f5e94000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
    0x1b0000) = 0x7f69f5e94000
42 mmap(0x7f69f5ee3000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1fe000) = 0x7f69f5ee3000
43 mmap(0x7f69f5ee9000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS, -1, 0) = 0x7f69f5ee9000
44 close(3) = 0
45 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x7f69f5ce2000

```

```

46 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
   x7f69f5cdf000
47 arch_prctl(ARCH_SET_FS, 0x7f69f5cdf740) = 0
48 set_tid_address(0x7f69f5cdfa10) = 226683
49 set_robust_list(0x7f69f5cdfa20, 24) = 0
50 rseq(0x7f69f5ce0060, 0x20, 0, 0x53053053) = 0
51 mprotect(0x7f69f5ee3000, 16384, PROT_READ) = 0
52 mprotect(0x7f69f5f22000, 4096, PROT_READ) = 0
53 mprotect(0x7f69f600b000, 4096, PROT_READ) = 0
54 mprotect(0x7f69f6279000, 45056, PROT_READ) = 0
55 mprotect(0x55e35cbae000, 4096, PROT_READ) = 0
56 mprotect(0x7f69f62c9000, 8192, PROT_READ) = 0
57 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY
   }) = 0
58 munmap(0x7f69f628b000, 21863) = 0
59 futex(0x7f69f62877bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
60 getrandom("\x1e\xb5\x36\xf6\x43\x81\xf6\x10", 8, GRND_NONBLOCK) = 8
61 brk(NULL) = 0x55e38eb1f000
62 brk(0x55e38eb40000) = 0x55e38eb40000
63 rt_sigaction(SIGRT_1, {sa_handler=0x7f69f5d7d530, sa_mask=[], sa_flags=
   SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f69f5d29330},
   NULL, 8) = 0
64 rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
65 mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0
   x7f69f54de000
66 mprotect(0x7f69f54df000, 8388608, PROT_READ|PROT_WRITE) = 0
67 rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
68 clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
   CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
   child_tid=0x7f69f5cde990, parent_tid=0x7f69f5cde990, exit_signal=0, stack=0
   x7f69f54de000, stack_size=0x7fff80, tls=0x7f69f5cde6c0}strace: Process
   226684 attached
69 => {parent_tid=[226684]}, 88) = 226684
70 [pid 226684] rseq(0x7f69f5cdefe0, 0x20, 0, 0x53053053 <unfinished ...>
71 [pid 226683] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
72 [pid 226684] <... rseq resumed>) = 0
73 [pid 226683] <... rt_sigprocmask resumed>NULL, 8) = 0
74 [pid 226684] set_robust_list(0x7f69f5cde9a0, 24 <unfinished ...>
75 [pid 226683] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK
   , -1, 0 <unfinished ...>
76 [pid 226684] <... set_robust_list resumed>) = 0
77 [pid 226683] <... mmap resumed>) = 0x7f69f4cdd000
78 [pid 226684] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
79 [pid 226683] mprotect(0x7f69f4cde000, 8388608, PROT_READ|PROT_WRITE <unfinished
   ...>
80 [pid 226684] <... rt_sigprocmask resumed>NULL, 8) = 0
81 [pid 226683] <... mprotect resumed>) = 0
82 [pid 226683] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
83 [pid 226683] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
   CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
   CLONE_CHILD_CLEARTID, child_tid=0x7f69f54dd990, parent_tid=0x7f69f54dd990,
   exit_signal=0, stack=0x7f69f4cdd000, stack_size=0x7fff80, tls=0
   x7f69f54dd6c0}strace: Process 226685 attached
84 => {parent_tid=[226685]}, 88) = 226685
85 [pid 226685] rseq(0x7f69f54ddfe0, 0x20, 0, 0x53053053 <unfinished ...>
86 [pid 226683] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
87 [pid 226685] <... rseq resumed>) = 0
88 [pid 226683] <... rt_sigprocmask resumed>NULL, 8) = 0
89 [pid 226685] set_robust_list(0x7f69f54dd9a0, 24 <unfinished ...>

```

```

90 [pid 226683] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK
, -1, 0 <unfinished ...>
91 [pid 226685] <... set_robust_list resumed>) = 0
92 [pid 226683] <... mmap resumed>) = 0x7f69f44dc000
93 [pid 226685] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
94 [pid 226683] mprotect(0x7f69f44dd000, 8388608, PROT_READ|PROT_WRITE <unfinished
...>
95 [pid 226685] <... rt_sigprocmask resumed>NULL, 8) = 0
96 [pid 226683] <... mprotect resumed>) = 0
97 [pid 226683] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
98 [pid 226683] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, child_tid=0x7f69f4cdc990, parent_tid=0x7f69f4cdc990,
exit_signal=0, stack=0x7f69f44dc000, stack_size=0x7fff80, tls=0
x7f69f4cdc6c0}strace: Process 226686 attached
99 => {parent_tid=[226686]}, 88) = 226686
100 [pid 226686] rseq(0x7f69f4cdcfe0, 0x20, 0, 0x53053053 <unfinished ...>
101 [pid 226683] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
102 [pid 226686] <... rseq resumed>) = 0
103 [pid 226683] <... rt_sigprocmask resumed>NULL, 8) = 0
104 [pid 226686] set_robust_list(0x7f69f4cdc9a0, 24 <unfinished ...>
105 [pid 226683] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK
, -1, 0 <unfinished ...>
106 [pid 226686] <... set_robust_list resumed>) = 0
107 [pid 226683] <... mmap resumed>) = 0x7f69f3cdb000
108 [pid 226686] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
109 [pid 226683] mprotect(0x7f69f3cdc000, 8388608, PROT_READ|PROT_WRITE <unfinished
...>
110 [pid 226686] <... rt_sigprocmask resumed>NULL, 8) = 0
111 [pid 226683] <... mprotect resumed>) = 0
112 [pid 226683] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0
113 [pid 226683] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|
CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID, child_tid=0x7f69f44db990, parent_tid=0x7f69f44db990,
exit_signal=0, stack=0x7f69f3cdb000, stack_size=0x7fff80, tls=0
x7f69f44db6c0}strace: Process 226687 attached
114 => {parent_tid=[226687]}, 88) = 226687
115 [pid 226687] rseq(0x7f69f44dbfe0, 0x20, 0, 0x53053053 <unfinished ...>
116 [pid 226683] rt_sigprocmask(SIG_SETMASK, [], <unfinished ...>
117 [pid 226687] <... rseq resumed>) = 0
118 [pid 226683] <... rt_sigprocmask resumed>NULL, 8) = 0
119 [pid 226687] set_robust_list(0x7f69f44db9a0, 24 <unfinished ...>
120 [pid 226683] futex(0x7f69f5cde990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,
226684, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>
121 [pid 226687] <... set_robust_list resumed>) = 0
122 [pid 226687] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
123
124 [pid 226685] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
125 [pid 226685] madvise(0x7f69f4cdd000, 8368128, MADV_DONTNEED) = 0
126 [pid 226685] exit(0) = ?
127 [pid 226685] +++ exited with 0 +++
128 [pid 226687] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
129 [pid 226687] madvise(0x7f69f3cdb000, 8368128, MADV_DONTNEED) = 0
130 [pid 226687] exit(0) = ?
131 [pid 226687] +++ exited with 0 +++
132 [pid 226686] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
133 [pid 226686] madvise(0x7f69f44dc000, 8368128, MADV_DONTNEED) = 0
134 [pid 226686] exit(0) = ?
135 [pid 226686] +++ exited with 0 +++
136 [pid 226684] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

```

```

137 [pid 226684] madvise(0x7f69f54de000, 8368128, MADV_DONTNEED) = 0
138 [pid 226684] exit(0) = ?
139 [pid 226683] <... futex resumed>) = 0
140 [pid 226684] +++ exited with 0 +++
141 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x6), ...}) = 0
142 write(1, "\n", 1
143 ) = 1
144 write(1, "--- Monte Carlo Simulation Resul"... , 39--- Monte Carlo Simulation
    Results ---
145 ) = 39
146 write(1, "Threads Used:      4\n", 21Threads Used:      4
147 ) = 21
148 write(1, "Total Points:      1000000000\n", 30Total Points:      1000000000
149 ) = 30
150 write(1, "Total Hits:      785353352\n", 29Total Hits:      785353352
151 ) = 29
152 write(1, "Estimated Area:    3.14141341\n", 30Estimated Area:    3.14141341
153 ) = 30
154 write(1, "Analytical Area:    3.14159265\n", 30Analytical Area:    3.14159265
155 ) = 30
156 write(1, "Execution Time:    82.268002 sec"... , 37Execution Time:    82.268002
    seconds
157 ) = 37
158 exit_group(0) = ?
159 +++ exited with 0 +++

```