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»

Выполнила: В. А. Гузова
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель работы:

Приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потоками

Задание:

Составить программу на языке C++, 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.

Вариант: 14

Есть набор 512 битных чисел, записанных в шестнадцатеричном представлении, хранящихся в файле. Необходимо посчитать их среднее арифметическое. Округлить результат до целых. Количество используемой оперативной памяти должно задаваться "ключом

Метод решения

Программа реализует параллельное вычисление среднего арифметического для массива 512-битных целых чисел. Массив чисел загружается в память и равномерно распределяется между заданным числом потоков. Каждый поток накапливает локальную частичную сумму (local_sum). Когда все посчиталось локально(в каждом потоке), поток синхронизируется с главным потоком и другими рабочими потоками для обновления глобальной суммы (global_sum) и общего счетчика.Каждый поток сам вносит свой вклад в общую глобальную сумму в защищенной критической секции, используя мьютекс, что обеспечивает корректность данных.

Описание программы

`main.cpp` — точка входа в программу. Парсит аргументы командной строки.Считывает числа из файла, прекращая чтение при превышении заданного лимита памяти.Создает и запускает рабочие потоки, распределяя между ними массив чисел. После завершения всех потоков вычисляет и округляет среднее,выводит результат в шестнадцатеричном и десятичном формате.

`pthread.h` — объявление общего кроссплатформенного интерфейса для работы с потоками.

`bigint.cpp` — арифметика для больших и синхронизация с помощью мьютекса.

Основные функции:

- `Thread::Thread(ThreadFunc func)` — конструктор, создаёт объект `ThreadInfo`
- `void Thread::Run(void* data)` — запускает поток.Обёртка над `pthread_create()`.

- `void Thread::Join()` — ожидает завершения потока.`pthread_join()`.
- `Thread::Thread(Thread&& other) noexcept` и `Thread& operator=(Thread&&)` — передают владение потоком от одного объекта к другому, не запуская поток заново.
- `void Thread::Swap(Thread& other)` — обменивает содержимое двух объектов `Thread`.

В `thread_work` каждый поток:

- последовательно складывает все числа из своего диапазона в локальный `BigInt sum`,
- Поток вызывает `std::lock_guard<std::mutex> lock(BigInt::BigInt::sum_mutex);`
- Внутри критической секции поток выполняет:`global_sum += local_sum;total_count += local_count;`,
- При выходе из `lock_guard` мьютекс автоматически разблокируется, следующий поток вносит свою сумму.

Выводы

В ходе выполнения лабораторной работы были приобретены навыки в организации многопоточной обработки данных в операционных системах, распределения вычислительной нагрузки между потоками. Была составлена и отлажена программа на языке C++, реализующая вычисление среднего арифметического для массива 512-битных целых чисел, представленных в шестнадцатеричном виде.Изучена явная синхронизация с использованием мьютекса (`std::mutex`) и `std::lock_guard` для защиты критической секции. Это гарантировало корректное обновление глобальной суммы (`global_sum`) и счетчика (`total_count`) всеми рабочими потоками. В результате работы программа ограничивает объём загружаемых данных(в соответствии с заданным числом), распределяет массив чисел из файла межу потоками,которые считают локальную сумму и добавляют ее к общей сумме(без ошибок из-за использования мьютекса).Программа считает общую сумму чисел из файла и среднее арифметическое Были обработаны возможные ошибки: некорректные аргументы командной строки, ошибки ввода и вывода, а также сбои при создании потоков. Экспериментально подтверждена эффективность параллельных вычислений: при увеличении числа потоков (до числа ядер) время выполнения сокращается почти линейно, особенно при обработке больших наборов данных.

Количество потоков	Время (мс)	Ускорение	Эффективность
1	23	1.00	1.00
2	11	2.09	1.05
3	8	2.87	0.96
4	7	3.29	0.82
5	5	4.60	0.92
6	6	3.83	0.64

Таблица 1: Зависимость времени выполнения от количества потоков

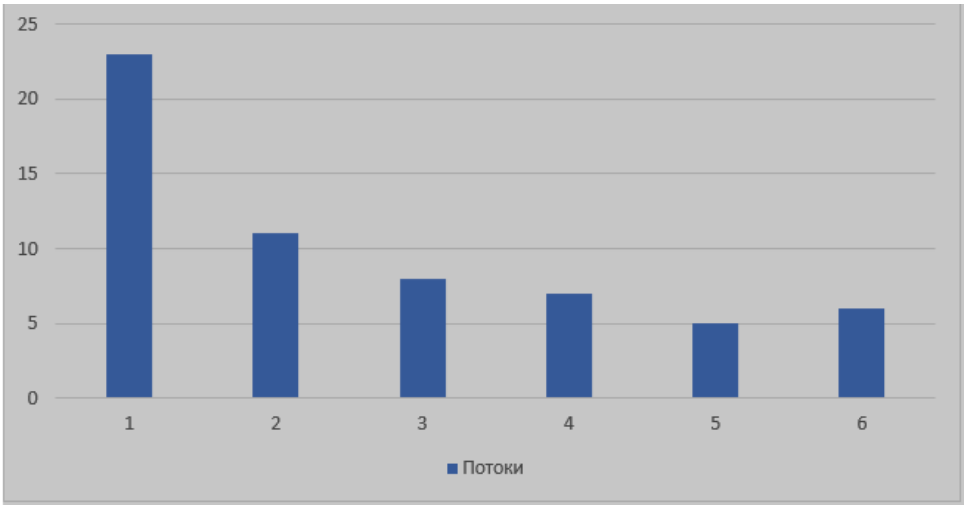

Рис. 1: Диаграмма зависимости времени от количества потоков

Результаты

Для исследования зависимости производительности от количества потоков были проведены замеры времени выполнения программы с различным количеством потоков и фиксированным объемом памяти.Чисел в файле всего 10000,поэтому вычисления происходят очень быстро.

Можно сделать вывод,что оптимальным количеством потоков для данной задачи является количество пяти потоков, так как при большем количестве потоков расходы на управление потоков и конкуренцию за мьютекс начинают превышать выгоду.

Исходная программа

```
#pragma once

#include <iostream>
#include <vector>
#include <string>
#include <cstdint>
#include <mutex>

namespace BigInt {

constexpr int NUM = 8;

class BigInt {
private:
```

```
    uint64_t data_[NUM] = {0};

public:
    BigInt() = default;
    explicit BigInt(const std::string& hex_str);
    BigInt& operator+=(const BigInt& other);
    BigInt operator+(const BigInt& other) const;
    void add_word(uint64_t word);
    void divide_by_block(uint64_t divisor);
    uint64_t divide_by_10();
    std::string to_dec() const;
    const uint64_t* getData() const {
      return data_;
    }
    static std::mutex sum_mutex;
    friend std::ostream& operator<<(std::ostream& os, const BigInt& big_int);
};
}
```

Листинг 1: include/bigint.h

```
#include "bigint.h"
#include <sstream>
#include <iomanip>
#include <algorithm>
#include <cmath>

namespace BigInt {

BigInt::BigInt(const std::string& hex_str) {
    std::string new_str = hex_str;
    if (new_str.length() < NUM * sizeof(uint64_t) * 2) {
        new_str.insert(0, NUM * sizeof(uint64_t) * 2 - new_str.length(), '0');
    }
    for (int i = 0; i < NUM; ++i) {
        int start = new_str.length() - (i + 1) * 16;
        std::string word_hex = new_str.substr(start, 16);
        data_[i] = std::stoull(word_hex, nullptr, 16);
    }
}

BigInt& BigInt::operator+=(const BigInt& other) {
    uint64_t carry = 0;
    for (int i = 0; i < NUM; ++i) {
        uint64_t sum = data_[i] + other.data_[i];
        bool carry1 = (sum < data_[i]);
        data_[i] = sum + carry;
        bool carry2 = (data_[i] < sum);
        carry = carry1 || carry2 ? 1 : 0;
    }
    return *this;
}

BigInt BigInt::operator+(const BigInt& other) const {
    BigInt result = *this;
    result += other;
    return result;
```

```cpp
}

void BigInt::add_word(uint64_t word) {
    if (word == 0) return;
    uint64_t sum = data_[0] + word;
    bool carry1 = (sum < data_[0]);
    data_[0] = sum;
    uint64_t carry = carry1 ? 1 : 0;
    for (int i = 1; i < NUM && carry; ++i) {
        data_[i] += carry;
        if (data_[i] == 0) {
            carry = 1;
        } else {
            carry = 0;
        }
    }
}

void BigInt::divide_by_block(uint64_t divisor) {
    if (divisor == 0) {
        throw std::runtime_error("Division by zero");
    }
    uint64_t remainder = 0;
    for (int i = NUM - 1; i >= 0; --i) {
        __uint128_t current_part = ((__uint128_t)remainder << 64) | data_[i];
        data_[i] = (uint64_t)(current_part / divisor);
        remainder = (uint64_t)(current_part % divisor);
    }
}

std::ostream& operator<<(std::ostream& os, const BigInt& big_int) {
    int first_nonzero = -1;
    for (int i = NUM - 1; i >= 0; --i) {
        if (big_int.data_[i] != 0) {
            first_nonzero = i;
            break;
        }
    }
    if (first_nonzero == -1) {
        return os << "0";
    }
    os << std::hex << big_int.data_[first_nonzero];
    for (int i = first_nonzero - 1; i >= 0; --i) {
        os << std::setw(16) << std::setfill('0') << big_int.data_[i];
    }
    os << std::dec;
    return os;
}

uint64_t BigInt::divide_by_10() {
    const uint64_t divisor = 10;
    uint64_t remainder = 0;
    for (int i = NUM - 1; i >= 0; --i) {
        __uint128_t current_part = ((__uint128_t)remainder << 64) | data_[i];
        data_[i] = (uint64_t)(current_part / divisor);
        remainder = (uint64_t)(current_part % divisor);
    }
    return remainder;
```

```
}

std::string BigInt::to_dec() const {
    BigInt temp = *this;
    bool is_zero = true;
    for (int i = 0; i < NUM; ++i) {
        if (temp.data_[i] != 0) {
            is_zero = false;
            break;
        }
    }
    if (is_zero) {
        return "0";
    }
    std::string result;
    while (!is_zero) {
        uint64_t digit = temp.divide_by_10();
        result.push_back((char)(digit + '0'));
        is_zero = true;
        for (int i = 0; i < NUM; ++i) {
            if (temp.data_[i] != 0) {
                is_zero = false;
                break;
            }
        }
    }
    std::reverse(result.begin(), result.end());
    return result;
}
std::mutex BigInt::BigInt::sum_mutex;
}
```

Листинг 2: src/bigint.cpp

```
namespace thread {
using ThreadFunc = typeof(void*(void*))*;
struct ThreadInfo;
using ThreadHandle = ThreadInfo*;

class Thread {
  private:
    ThreadFunc func_;
    ThreadHandle handle_;
  public:
    Thread(ThreadFunc func);
    Thread(const Thread&) = delete;
    Thread(Thread&& other) noexcept;

    Thread& operator=(const Thread&) = delete;
    Thread& operator=(Thread&& other) noexcept;

    void Swap(Thread& other);
    void Run(void* data);
    void Join();
    ~Thread() noexcept;
};
```

```
}
```

Листинг 3: include/thread.h

```
#include <pthread.h>
#include <utility>
#include <system_error>

#include "thread.h"


namespace thread {
struct ThreadInfo {
    pthread_t thread;
};

Thread::Thread(ThreadFunc func): func_(func) {
    handle_ = new ThreadInfo();
}

Thread::Thread(Thread&& other) noexcept: func_(other.func_), handle_(other.handle_) {
    other.func_ = nullptr;
    other.handle_ = nullptr;
}

Thread::~Thread() noexcept {
    delete handle_;
}

Thread& Thread::operator=(Thread&& other) noexcept {
    if (this == &other) {
        return *this;
    }
    Thread tmp = std::move(other);
    this->Swap(tmp);
    return *this;
}

void Thread::Run(void* data) {
    int res = pthread_create(&(handle_->thread), NULL, func_, data);
    if (res != 0) {
        throw std::system_error(res, std::system_category(), "pthread_create failed");
    }
}

void Thread::Join() {
    int res = pthread_join(handle_->thread, NULL);
    if (res != 0) {
        throw std::system_error(res, std::system_category(), "pthread_join failed");
    }
}

void Thread::Swap(Thread& other) {
    std::swap(func_, other.func_);
    std::swap(handle_, other.handle_);
}

```

```
}
```

Листинг 4: src/thread.cpp

```
#include "bigint.h"
#include "thread.h"
#include <string>
#include <vector>
#include <fstream>
#include <sstream>
#include <iostream>
#include <algorithm>
#include <numeric>
#include <stdexcept>
#include <cmath>
#include <chrono>

constexpr size_t MAX_LEN = 128;
constexpr size_t LINE_SIZE = MAX_LEN + 1;

BigInt::BigInt global_sum;
size_t total_count = 0;

struct ThreadData {
    const std::vector<std::string>* hex_numbers;
    size_t start_index;
    size_t end_index;
};

void* thread_work(void* arg) {
    ThreadData* data = static_cast<ThreadData*>(arg);
    BigInt::BigInt local_sum;
    size_t local_count = 0;
    for (size_t i = data->start_index; i < data->end_index; ++i) {
        try {
            BigInt::BigInt num((*data->hex_numbers)[i]);
            local_sum += num;
            local_count++;
        } catch (const std::exception& e) {
            std::cerr << "Thread error" << e.what() << std::endl;
        }
    }
    {
        std::lock_guard<std::mutex> lock(BigInt::BigInt::sum_mutex);
        global_sum += local_sum;
        total_count += local_count;
    }
    return nullptr;
}


std::vector<std::string> read_data_with_memory_limit(const std::string& filename,
    size_t max_bytes) {
    std::ifstream file(filename);
    if (!file.is_open()) {
        throw std::runtime_error("Could not open file: " + filename);
    }
```

```
    std::vector<std::string> numbers;
    std::string line;
    size_t current_mem_usage = 0;
    constexpr size_t NUM_STR_SIZE = MAX_LEN + 1;

    while (std::getline(file, line)) {
        line.erase(std::remove_if(line.begin(), line.end(), ::isspace), line.end());

        if (line.empty()) continue;
        size_t line_mem = line.capacity() + sizeof(std::string) + sizeof(void*);

        if (current_mem_usage + line_mem > max_bytes) {
            std::cerr << "Memory limit. Stopping file read." << std::endl;
            break;
        }

        numbers.push_back(line);
        current_mem_usage += line_mem;
    }

    if (numbers.empty()) {
        throw std::runtime_error("Error");
    }
    return numbers;
}

int main(int argc, char* argv[]) {
    if (argc != 4) {
        std::cerr << "Error input" << std::endl;
        return 1;
    }
    const std::string filename = argv[1];
    int max_threads = 0;
    size_t max_bytes = 0;
    try {
        max_threads = std::stoi(argv[2]);
        max_bytes = std::stoull(argv[3]);
    } catch (const std::exception& e) {
        std::cerr << "Invalid argument" << e.what() << std::endl;
        return 1;
    }

    if (max_threads <= 0 || max_bytes == 0) {
        std::cerr << "Error argument" << std::endl;
        return 1;
    }

    size_t max_threads = static_cast<size_t>(max_threads);

    std::vector<std::string> hex_numbers;
    try {
        hex_numbers = read_data_with_memory_limit(filename, max_bytes);
    } catch (const std::runtime_error& e) {
        std::cerr << "Error file: " << e.what() << std::endl;
        return 1;
    }
    size_t num_numbers = hex_numbers.size();
    size_t num_my_threads = std::min(num_numbers, max_threads);
```

```
111     if (num_my_threads == 0) {
112         std::cerr << "No numbers to process" << std::endl;
113         return 0;
114     }
115     auto start_time = std::chrono::high_resolution_clock::now();
116     std::vector<ThreadData> all_data(num_my_threads);
117     size_t base_chunk_size = num_numbers / num_my_threads;
118     size_t remainder = num_numbers % num_my_threads;
119     size_t current_start = 0;
120 
121     for (size_t i = 0; i < num_my_threads; ++i) {
122         size_t chunk_size = base_chunk_size + (i < remainder ? 1 : 0);
123         all_data[i] = {&hex_numbers, current_start, current_start + chunk_size};
124         current_start += chunk_size;
125     }
126     std::vector<thread::Thread> threads;
127     threads.reserve(num_my_threads);
128 
129     for (size_t i = 0; i < num_my_threads; ++i) {
130         try {
131             threads.emplace_back(thread::Thread(thread_work));
132             threads.back().Run(&all_data[i]);
133         } catch (const std::system_error& e) {
134             std::cerr << "Error thread " << e.what() << std::endl;
135         }
136     }
137     for (auto& t : threads) {
138         try {
139             t.Join();
140         } catch (const std::system_error& e) {
141             std::cerr << "Error joining thread: " << e.what() << std::endl;
142         }
143     }
144     auto end_time = std::chrono::high_resolution_clock::now();
145     auto answer_time = std::chrono::duration_cast<std::chrono::milliseconds>(end_time
        - start_time).count();
146 
147     std::cout << "time " << answer_time << " ms" << std::endl;
148 
149     if (total_count == 0) {
150         std::cout << "No numbers" << std::endl;
151         return 0;
152     }
153     std::cout << "Sum (HEX): " << global_sum << std::endl;
154     std::cout << "Sum (DEC): " << global_sum.to_dec() << std::endl;
155     uint64_t divisor = static_cast<uint64_t>(total_count);
156     uint64_t half_divisor = divisor / 2;
157     global_sum.add_word(half_divisor);
158     global_sum.divide_by_block(divisor);
159     std::cout << "Rounded: " << global_sum.to_dec() << std::endl;
160     return 0;
161 }
```

Листинг 5: main.cpp

Strace

```
execve("./main", ["./main", "number.hex", "4", "10000000"], 0x7ffc028330e8 /*
↪  26 vars */) = 0
brk(NULL)                               = 0x63c7629ec000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe6f000840) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x76e8d0ec8000
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=29000, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 29000, PROT_READ, MAP_PRIVATE, 3, 0) = 0x76e8d0ec0000
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) = 0x76e8d0c00000
mprotect(0x76e8d0c9a000, 1576960, PROT_NONE) = 0
mmap(0x76e8d0c9a000, 1118208, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x76e8d0c9a000
mmap(0x76e8d0dab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1ab000) = 0x76e8d0dab000
mmap(0x76e8d0e1b000, 57344, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) = 0x76e8d0e1b000
mmap(0x76e8d0e29000, 10432, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x76e8d0e29000
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) = 0x76e8d0ea0000
mmap(0x76e8d0ea3000, 94208, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) = 0x76e8d0ea3000
mmap(0x76e8d0eba000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1a000) = 0x76e8d0eba000
mmap(0x76e8d0ebe000, 8192, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) = 0x76e8d0ebe000
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\00{\f\225\\=\201\327\312\301P\32\230\
↪  266\235"..., 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) = 0x76e8d0800000
mprotect(0x76e8d0828000, 2023424, PROT_NONE) = 0
mmap(0x76e8d0828000, 1658880, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x76e8d0828000
mmap(0x76e8d09bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x1bd000) = 0x76e8d09bd000
mmap(0x76e8d0a16000, 24576, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x76e8d0a16000
mmap(0x76e8d0a1c000, 52816, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x76e8d0a1c000
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"...,
↪  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) = 0x76e8d0b19000
mmap(0x76e8d0b27000, 507904, PROT_READ|PROT_EXEC,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x76e8d0b27000
mmap(0x76e8d0ba3000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
↪  0x8a000) = 0x76e8d0ba3000
mmap(0x76e8d0bfe000, 8192, PROT_READ|PROT_WRITE,
↪  MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x76e8d0bfe000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x76e8d0e9e000
arch_prctl(ARCH_SET_FS, 0x76e8d0e9f3c0) = 0
set_tid_address(0x76e8d0e9f690)         = 2061
set_robust_list(0x76e8d0e9f6a0, 24)     = 0
rseq(0x76e8d0e9fd60, 0x20, 0, 0x53053053) = 0
mprotect(0x76e8d0a16000, 16384, PROT_READ) = 0
mprotect(0x76e8d0bfe000, 4096, PROT_READ) = 0
mprotect(0x76e8d0ebe000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x76e8d0e9c000
mprotect(0x76e8d0e1b000, 45056, PROT_READ) = 0
mprotect(0x63c752747000, 4096, PROT_READ) = 0
mprotect(0x76e8d0f02000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
↪  rlim_max=RLIM64_INFINITY}) = 0
munmap(0x76e8d0ec0000, 29000)           = 0
getrandom("\x1a\x42\xef\x85\xd7\x1c\x15\xf6", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x63c7629ec000
brk(0x63c762a0d000)                     = 0x63c762a0d000
futex(0x76e8d0e2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
openat(AT_FDCWD, "number.hex", O_RDONLY) = 3
read(3, "80b5ce21f1b9f88c4f6ae37f82eb8950"..., 8191) = 8191
read(3, "c14ea4daa92941732d30f84bd2adbeca"..., 8191) = 8191
read(3, "\n307cf9a036a03a82ff9e7990ce453a7"..., 8191) = 8191
read(3, "98a3c8d5a9ecca0ccf72b0c1b707518a"..., 8191) = 8191
```

```
read(3, "d\n34fbdd6a48629090ccb7f01d662ca6"..., 8191) = 8191
brk(0x63c762a30000)                     = 0x63c762a30000
read(3, "baa3a02ff2aa4311d884810958db2ade"..., 8191) = 8191
read(3, "f7\n3342b7a60dc378b507c41d52a9939"..., 8191) = 8191
read(3, "50b0fffc6bb4db15055e979fb6c00af0"..., 8191) = 8191
read(3, "a4a\n73947c26bf8fb6540a5050960871"..., 8191) = 8191
read(3, "517d5eab48bc13ea68b4f85580cd8090"..., 8191) = 8191
read(3, "1046\nbd42eba5757df4a4747d611c723"..., 8191) = 8191
read(3, "afbdd572f2258d356e6397092bf47c48"..., 8191) = 8191
read(3, "be047\n60deed2b584b0ec9ba525b11e6"..., 8191) = 8191
read(3, "6b3befb33bd6ced8cd6e848ca8ba085a"..., 8191) = 8191
read(3, "10125a\ncbacadd894c296978906e73bb"..., 8191) = 8191
read(3, "59dd4b35f1486181e5bd9ee7dc5217fb"..., 8191) = 8191
read(3, "345a742\n87310cf761ada58343b3de28"..., 8191) = 8191
brk(0x63c762a5d000)                     = 0x63c762a5d000
read(3, "af3bac49ef03e44104a31c9ee4ba1182"..., 8191) = 8191
read(3, "61fefeee\ne68e7be499e7a51b3262b81"..., 8191) = 8191
read(3, "78c21cfb3e5b9ee6b7986362803fe7d7"..., 8191) = 8191
read(3, "8a4541838\n5326136d359901ea22c1df"..., 8191) = 8191
read(3, "8868cd9b7db7c03e810ab84381599b58"..., 8191) = 8191
read(3, "37e871de9f\nfdd8586afb050e4c54210"..., 8191) = 8191
read(3, "f382b031820746c63f536f2bc7e4651f"..., 8191) = 8191
read(3, "a144cb263c9\nfdeabaa568afb392eeea"..., 8191) = 8191
read(3, "23c52082f3f3f571f055a8f884b5f708"..., 8191) = 8191
read(3, "3e922064919a\naa2a76a88aa7766f65d"..., 8191) = 8191
read(3, "5fbe51c65cd3680e681a644866a9d9fd"..., 8191) = 8191
read(3, "2cfd89cba201e\n5a90fdf53bec5ba3e1"..., 8191) = 8191
read(3, "ed8a29912569f928c7a661a829be1b7f"..., 8191) = 8191
read(3, "9569ba17c5e467\n2d01f4be7ab307724"..., 8191) = 8191
read(3, "6fcd18979e0fe5e7c402cd2de2ba14c3"..., 8191) = 8191
read(3, "656976588ebe415\n1fb5df2901899f70"..., 8191) = 8191
mmap(NULL, 135168, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x76e8d0e7b000
read(3, "60aff418dffb2869de29eed0ad2870dc"..., 8191) = 8191
read(3, "0b576e4104f22884\n07b08ca5bfafc16"..., 8191) = 8191
read(3, "d3744469d69f44c6a2b3afe766ac1084"..., 8191) = 8191
read(3, "544e9198428bf0dcf\n5a5bdacc2f3d48"..., 8191) = 8191
read(3, "f9025035d0ec40acf8eb45a251fde9d2"..., 8191) = 8191
read(3, "cff38920872196c1bc\n051ac4270eeff"..., 8191) = 8191
read(3, "440de753f92ddb5639246d32f037f485"..., 8191) = 8191
read(3, "ba8f956e33e40ed48ad\na878ee426211"..., 8191) = 8191
read(3, "f6698fcafb341e0dca03e79d48949910"..., 8191) = 8191
brk(0x63c762a7e000)                     = 0x63c762a7e000
read(3, "f53f39935491d9613650\n12457d06da0"..., 8191) = 8191
read(3, "08d8fed63ddfced7671ffe4b394d0f49"..., 8191) = 8191
read(3, "10772a55aed6636e1e686\n0460a73278"..., 8191) = 8191
read(3, "0c2b51f3b1657a6aa3e5e12976372f5a"..., 8191) = 8191
read(3, "bf44e5cf1752c92c10d57d\n1c28b6ddb"..., 8191) = 8191
read(3, "622fe04dd0e6a81faf60d01dcf571b9a"..., 8191) = 8191
read(3, "3b08eb0c142e264ad71c696\n403af045"..., 8191) = 8191
read(3, "a5d83ba9f4be08022546e46395bb827c"..., 8191) = 8191
read(3, "f72c946c87dd2a4875430734\n47ad2d5"..., 8191) = 8191
```

```
read(3, "4f5b0baf597df9eb74cdfdb0f66edffd"..., 8191) = 8191
read(3, "f63bbbe8e6378ac9725bff10a\nbce31e"..., 8191) = 8191
read(3, "b8e1900a31380521c9a7d5f352361431"..., 8191) = 8191
read(3, "a18af28cd3a739760905b6b794\n83b13"..., 8191) = 8191
read(3, "c05fab8c23d56d217ebc7086f001df93"..., 8191) = 8191
read(3, "27e63787ba76b99616fa165ff42\n6a64"..., 8191) = 8191
brk(0x63c762a9f000)                     = 0x63c762a9f000
read(3, "8a5669c2090f0d01d78eeb3d7dc4f420"..., 8191) = 8191
read(3, "7abbf3fa2dfd09de9fbc84f26f92\n4cc"..., 8191) = 8191
read(3, "405394ee094dfe8e193e71544177d29e"..., 8191) = 8191
read(3, "13cc8fd106687f293a62c82b91971\ne4"..., 8191) = 8191
read(3, "89f2f2caed56e057440710752da1d766"..., 8191) = 8191
read(3, "0c8ab8ca3eb5806563acb7c246e455\n7"..., 8191) = 8191
read(3, "295d459308e07980db4e05a5ee9bf22c"..., 8191) = 8191
read(3, "3387573ba4776e11a71d896b6357e6f\n"..., 8191) = 8191
mmap(NULL, 266240, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x76e8d0e3a000
munmap(0x76e8d0e7b000, 135168)          = 0
read(3, "d8e35afe52a475fddf0c8f1b21cb4740"..., 8191) = 8191
read(3, "471a3ad14ac68571cdf3556ab1ff337b"..., 8191) = 8191
read(3, "bfea93191620182a054a2f256d3938df"..., 8191) = 8191
read(3, "df023506e1e696024b86ef1b15e42285"..., 8191) = 8191
read(3, "cc5f6a0ba7fcabd7464a302bbf546bf3"..., 8191) = 8191
read(3, "3e99efb8af925e1f37a1a1e263b17809"..., 8191) = 8191
brk(0x63c762ac0000)                     = 0x63c762ac0000
read(3, "f06f3ddbbdb44e1abf31e1ebd3c0d77d"..., 8191) = 8191
read(3, "5d774be009d492f4536f2f2c71e3fe76"..., 8191) = 8191
read(3, "a31d8fa93903a282fa982c126894fe4d"..., 8191) = 8191
read(3, "d80b36ec17a4fdc62c80c1586361e8e0"..., 8191) = 8191
read(3, "398585e53d0ac6ac493dec635c9b9ebc"..., 8191) = 8191
read(3, "fecae535a48192bb665fd8bb85fcab62"..., 8191) = 8191
read(3, "85b70dd5b1583885e782396dd5caecec"..., 8191) = 8191
read(3, "ba8f85b89c4a9a9cda8295b55ca95d78"..., 8191) = 8191
read(3, "50cf52ab72331dfe41bbd6c3390eb540"..., 8191) = 8191
read(3, "9edcc080ba5516bc2e1b7384e562a7cf"..., 8191) = 8191
read(3, "960481991771ce73cfe8f128358dcdb9"..., 8191) = 8191
read(3, "af264d96ce83d50023445a8b2bb632a5"..., 8191) = 8191
read(3, "5f12cf24ce27d7c968397c9cd0e691c1"..., 8191) = 8191
read(3, "71d388d5ef1dbc025a785ab9ea0b421a"..., 8191) = 8191
read(3, "b6ae4b3abfcc260205b39a8e78ed151c"..., 8191) = 8191
brk(0x63c762ae1000)                     = 0x63c762ae1000
read(3, "92c04daaae539402312810a2bc5f7eb8"..., 8191) = 8191
read(3, "63eacd0d8b56e461f210a610ab25804c"..., 8191) = 8191
read(3, "f7082e5eab7575f002fec0d22d122fb4"..., 8191) = 8191
read(3, "de5678ba092519c11c6eefa21a656f00"..., 8191) = 8191
read(3, "323943f208bc4cd976c272d1c1edc2de"..., 8191) = 8191
read(3, "71cc2444d1cadf663ea4ae475552ac00"..., 8191) = 8191
read(3, "a00270a36d9eb75de049d3a8c13d3d64"..., 8191) = 8191
read(3, "061d235fe69df2364a901ce8748808a1"..., 8191) = 8191
read(3, "ee5819057651d11f0521a1f55768d737"..., 8191) = 8191
read(3, "fc2b7a6adf3718d56bf8bd6c158b607a"..., 8191) = 8191
read(3, "28301e10facd3ca314e454800901e550"..., 8191) = 8191
```

```
read(3, "ba61924da80f2e393a979ae0227f7c2c"..., 8191) = 8191
read(3, "08235a124eda13db12ffaa0769f9631d"..., 8191) = 8191
read(3, "0400ff17fdce26e0332c8f56044c2dbe"..., 8191) = 8191
read(3, "d685264dc1e6618ba4ff2127691478aa"..., 8191) = 8191
brk(0x63c762b02000)                     = 0x63c762b02000
read(3, "235b28e63d163f4376612d2f3b88e6ae"..., 8191) = 8191
read(3, "e051bcc5e030f3192474de20bf7c8267"..., 8191) = 8191
read(3, "f4af7e28f74e6426fd0ffbe4db1ff20f"..., 8191) = 8191
read(3, "56e8f0637e8ab6ddd12e7f5bf7c06bf3"..., 8191) = 8191
read(3, "2b28c4d18dd21a25a1a8a999921528ac"..., 8191) = 8191
read(3, "22b5dd53f1a9fb061b143a13a2333c66"..., 8191) = 8191
read(3, "49d64347ba9876d6c97defbb0df21be4"..., 8191) = 8191
read(3, "f9d1b237000fa52cfec4547ba31ec38b"..., 8191) = 8191
read(3, "04c6c6110649fea0b840742099627f6f"..., 8191) = 8191
read(3, "8d8fbe7f317b4752c5c6354099d7436a"..., 8191) = 8191
read(3, "8916e44a09dd24ceba37f42514b95b42"..., 8191) = 8191
read(3, "83ac94a7669ad86fa78b71b5204f5ebc"..., 8191) = 8191
read(3, "700cfd1fd5bb28c9f4d85f021a303ec1"..., 8191) = 8191
read(3, "af60fb4b6fcaef741c32981d2c7cb07f"..., 8191) = 8191
read(3, "2fcf8103115d051b2c452327f9000ebd"..., 8191) = 8191
brk(0x63c762b23000)                     = 0x63c762b23000
read(3, "91182aa4a90bf4ae0ba7983a0812f0ca"..., 8191) = 8191
read(3, "40d48a47c220f72cc86eb1b245a92094"..., 8191) = 8191
read(3, "8e30a1cc23ec50d72628f49a753521b4"..., 8191) = 8191
read(3, "1961290bfbe42da783fa568fa570be3d"..., 8191) = 8191
read(3, "06d97a647ad36c5371bf672d8db830f9"..., 8191) = 8191
read(3, "e95d3e8a87aa546a38098d827bc15b29"..., 8191) = 8191
read(3, "cee71f446430447c8c135f39b23807ba"..., 8191) = 8191
read(3, "43d9e5daaa3727775bc660d7fd938fba"..., 8191) = 8191
read(3, "ad5972c190a6d8293bd9878a6c044a61"..., 8191) = 8191
read(3, "706a8b48e66579ae2f38a38b5cca8610"..., 8191) = 8191
read(3, "1ca816b57e4b29b7f31858a813d88559"..., 8191) = 8191
read(3, "7bbd641a08b9e405e3fc5e7a56d3f665"..., 8191) = 8191
read(3, "32a3cbb72b8eab2f7c35af91907be652"..., 8191) = 8191
read(3, "3ac9f4f14dbd34f091af9fe217a12f17"..., 8191) = 8191
mmap(NULL, 528384, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
↪  0x76e8d0a98000
munmap(0x76e8d0e3a000, 266240)          = 0
read(3, "593dff4c926efa710a6ec8dbbdeff3df"..., 8191) = 8191
brk(0x63c762b44000)                     = 0x63c762b44000
read(3, "\na020dfad2a48b560954139893440f5d"..., 8191) = 8191
read(3, "c90462d9c78ccff4fc0e4faf03db2d23"..., 8191) = 8191
read(3, "c\n3d372dffe76656befcc16548526173"..., 8191) = 8191
read(3, "8464c51bf7a83490d27ea35fd4a56c01"..., 8191) = 8191
read(3, "8d\ndb691e59d9b90bdc3bcba241ac22e"..., 8191) = 8191
read(3, "6673198ad3ea70281fb256005a361403"..., 8191) = 8191
read(3, "574\n059d299a1393b9953c003efb3b61"..., 8191) = 8191
read(3, "bdd205e9acdb260fef2c83b443adcbae"..., 8191) = 8191
read(3, "0f59\n5741e3b7bf1d9e06004c4be6136"..., 8191) = 8191
read(3, "e0864430b9a3b130724f2b7d2248a57d"..., 8191) = 8191
read(3, "b95e2\nf013eb166690cd3d3b084ee8ad"..., 8191) = 8191
read(3, "fe059f0a567d6c836391c998e7ac7f91"..., 8191) = 8191
```

```
read(3, "62a219\n0a0204d459c6fa22243e49416"..., 8191) = 8191
read(3, "0c923402eeee4334a8aa5da5d2269702"..., 8191) = 8191
brk(0x63c762b65000)                     = 0x63c762b65000
read(3, "bc01da6\n2045fb15e188cf9225da0571"..., 8191) = 8191
read(3, "598818f1f66dc096ddd80cd5b2adbadf"..., 8191) = 8191
read(3, "cdc538ef\na08627fde4afe84d9014bf7"..., 8191) = 8191
read(3, "19f187123eb531b0edd55e98fe6fcfc1"..., 8191) = 8191
read(3, "f7784b2d8\n2c725f629507c83b46d9a1"..., 8191) = 8191
read(3, "2c81ffaf1e60c8c86597a216a34b0416"..., 8191) = 8191
read(3, "1d7a866c7a\nb9e893878931725ec5f39"..., 8191) = 8191
read(3, "1a1eaae5e5d047d32158f98fe9f6d9ac"..., 8191) = 8191
read(3, "d504e56fa8f\nacfaa191643c021de32f"..., 8191) = 8191
read(3, "e64431e2b8c51831dd5adf9bcc5a3ded"..., 8191) = 8191
read(3, "ead1984e8471\n17cc841e51d5c59fda2"..., 8191) = 8191
read(3, "0d7e400e7ca24061cac559704441ce5f"..., 8191) = 8191
read(3, "0d02bbc9ff290\n2cb6d5b8f2f1c3b7f0"..., 8191) = 4013
read(3, "", 8191)                       = 0
close(3)                                = 0
rt_sigaction(SIGRT_1, {sa_handler=0x76e8d0891870, sa_mask=[],
↪  sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO,
↪  sa_restorer=0x76e8d0842520}, 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) =
↪  0x76e8cffff000
mprotect(0x76e8d0000000, 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_S⌋
↪  YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
↪  child_tid=0x76e8d07ff910, parent_tid=0x76e8d07ff910, exit_signal=0,
↪  stack=0x76e8cffff000, stack_size=0x7fff00, tls=0x76e8d07ff640} =>
↪  {parent_tid=[2062]}, 88) = 2062
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
↪  0x76e8cf7fe000
mprotect(0x76e8cf7ff000, 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_S⌋
↪  YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
↪  child_tid=0x76e8cfffe910, parent_tid=0x76e8cfffe910, exit_signal=0,
↪  stack=0x76e8cf7fe000, stack_size=0x7fff00, tls=0x76e8cfffe640} =>
↪  {parent_tid=[2063]}, 88) = 2063
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
↪  0x76e8ceffd000
mprotect(0x76e8ceffe000, 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_S⌋
↪  YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
↪  child_tid=0x76e8cf7fd910, parent_tid=0x76e8cf7fd910, exit_signal=0,
↪  stack=0x76e8ceffd000, stack_size=0x7fff00, tls=0x76e8cf7fd640} =>
↪  {parent_tid=[2064]}, 88) = 2064
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
```

```
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
↪  0x76e8ce7fc000
mprotect(0x76e8ce7fd000, 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_S┘
↪  YSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID,
↪  child_tid=0x76e8ceffc910, parent_tid=0x76e8ceffc910, exit_signal=0,
↪  stack=0x76e8ce7fc000, stack_size=0x7fff00, tls=0x76e8ceffc640} =>
↪  {parent_tid=[2065]}, 88) = 2065
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
futex(0x76e8d07ff910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 2062, NULL,
↪  FUTEX_BITSET_MATCH_ANY) = 0
futex(0x76e8cfffe910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 2063, NULL,
↪  FUTEX_BITSET_MATCH_ANY) = 0
futex(0x76e8cf7fd910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 2064, NULL,
↪  FUTEX_BITSET_MATCH_ANY) = 0
futex(0x76e8ceffc910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 2065, NULL,
↪  FUTEX_BITSET_MATCH_ANY) = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x2), ...},
↪  AT_EMPTY_PATH) = 0
write(1, "time12 ms\n", 10)             = 10
write(1, "Total Sum (HEX): efd42298e4449b9"..., 146) = 146
write(1, "Total Sum (DEC): 125608457591382"..., 173) = 173
write(1, "Rounded: 12560845759138207358773"..., 161) = 161
brk(0x63c762a5d000)                     = 0x63c762a5d000
munmap(0x76e8d0a98000, 528384)          = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```