Московский Авиационный Институт (Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

Группа: М8О-215Б-23

Студент: Венгер Ирина Витальевна Преподаватель: Миронов Е.С. Оценка:

Дата:

Москва, 2024.

**Содержание**

1. Постановка задачи.
2. Общие сведения о программе.
3. Общий метод и алгоритм решения.
4. Код программы.
5. Демонстрация работы программы.
6. Вывод.

# Постановка задачи

Требуется создать динамические библиотеки, которые реализуют заданный вариантом функционал. Далее использовать данные библиотеки 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В конечном итоге, в лабораторной работе необходимо получить следующие части:

1. Динамические библиотеки, реализующие контракты, которые заданы вариантом;
2. Тестовая программа (*программа №1*), которая используют одну из библиотек, используя информацию полученные на этапе компиляции;
3. Тестовая программа (*программа №2*), которая загружает библиотеки, используя только их относительные пути и контракты.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для *программы №2*).
2. «1 arg1 arg2 … argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;
3. «2 arg1 arg2 … argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

Задание варианта:

1. Описание: Рассчёт производной функции cos(x) в точке A с приращением deltaX;

Сигнатура: Float Derivative(float A, float deltaX); Реализация 1: f`(x) = (f(A + deltaX) – f(A))/deltaX;

Реализация 2: f`(x) = (f(A + deltaX) – f(A-deltaX))/(2\*deltaX).

1. Описание: Рассчет значения числа е(основание натурального логарифма); Сигнатура: Float E(int x);

Реализация 1: (1 + 1/x) ^ x;

Реализация 2: Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!)).

# Общие сведения о программе

Программа состоит из файлов: func.h, func1.c, func2.c, main1.c, main2.c и Makefile. Файл func.h содержит объявления функций для вычисления производной функции cos(x) и числа e. Файл func1.c реализует первую версию этих функций, используя методы для вычисления производной и числа e. В файле func2.c представлены альтернативные реализации этих функций с использованием других методов.

func1.c:

* 1. Реализация производной функции cos(x) с использованием формулы: f'(x) = (f(A + deltaX) - f(A)) / deltaX.
  2. Реализация вычисления числа e с использованием формулы: (1 + 1/x) ^ x. func2.c:
     1. Реализация производной функции cos(x) с использованием формулы: f'(x) = (f(A + deltaX) - f(A - deltaX)) / (2 \* deltaX).
     2. Реализация вычисления числа e с использованием суммы ряда: e = sum(1/n!, от n=0 до x).

main1.c: Программа для использования библиотеки во время компиляции. В этой программе функции из библиотеки libfunc1.so загружаются на этапе компиляции и используются для вычислений производной и числа e.

main2.c: Программа для динамической загрузки библиотеки во время исполнения. В этой программе используется интерфейс ОС для загрузки библиотек libfunc1.dylib и libfunc2.dylib, переключение между ними осуществляется с помощью команды 0, а затем вызываются функции для вычислений.

Makefile компилирует две библиотеки libfunc1.dylib и libfunc2.dylib, а также две тестовые программы: main1 и main2. Для работы с математическими функциями в программе используется библиотека math.h, а для динамической загрузки библиотек

* dlfcn.h.
  1. **#include <stdio.h>** — для ввода-вывода. Из этого заголовка используются функции printf, fgets, sscanf, fprintf.
  2. **#include <dlfcn.h>** — для динамической загрузки библиотек. Из этого заголовка используются функции dlopen, dlsym, dlclose, dlerror.
  3. **#include <stdlib.h>** — для работы с памятью и управления процессами. Из этого заголовка используется функция exit.
  4. **#include <math.h>** — для математических вычислений. Из этого заголовка используются функции cos (для вычисления косинуса), pow (для вычисления степени).

# Общий метод и алгоритм решения

**Метод решения задачи:**

1. **Создание динамических библиотек:**

Для решения задачи создаются две динамические библиотеки (libfunc1.dylib и libfunc2.dylib), каждая из которых реализует два математических вычисления: производную функции cos(x) и вычисление числа e. Каждая библиотека предоставляет альтернативные реализации этих вычислений.

1. **Реализация математических функций:**

В библиотеке libfunc1.dylib реализуется первая версия функций:

* + **Вычисление производной функции cos(x)** с использованием формулы: f'(x) = (cos(A + Δx) - cos(A)) / Δx
  + **Вычисление числа e** с использованием формулы: e = (1 + 1/x)^x В библиотеке libfunc2.dylib реализуется вторая версия функций:
  + **Вычисление производной функции cos(x)** с использованием формулы: f'(x) = (cos(A + Δx) - cos(A - Δx)) / (2 \* Δx)
  + **Вычисление числа e** с использованием суммы ряда: e = sum(1 / n!) для n от 0 до x

1. **Тестовые программы:**

В первой тестовой программе (main1.c) используется статическая привязка к библиотеке libfunc1.dylib, и функции для вычисления производной и числа e вызываются напрямую.

Во второй тестовой программе (main2.c) динамическая загрузка библиотек с помощью интерфейса ОС (функции dlopen, dlsym, dlclose) позволяет загружать и использовать библиотеки во время выполнения программы. В этой программе также реализована возможность переключения между библиотеками с помощью команды "0", чтобы динамически выбирать одну из двух реализаций.

1. **Взаимодействие с пользователем:**

Пользователь взаимодействует с программой через командный ввод. Для выполнения функций необходимо ввести команду:

* + "1 arg1 arg2" — для вычисления производной с переданными аргументами (A, deltaX).
  + "2 arg1" — для вычисления числа e с переданным значением x.
  + "0" — для переключения между библиотеками libfunc1.dylib и libfunc2.dylib во второй программе.

**Алгоритм решения:**

1. **Компиляция библиотеки:**

Для каждой библиотеки (файлы func1.c и func2.c) компилируются динамические библиотеки с использованием флага -fPIC для создания объектных файлов с независимым кодом и флага -shared для создания динамических библиотек. С помощью Makefile библиотеки компилируются и связываются с соответствующими программами.

1. **Основная программа (main1.c):**

В этой программе используется статическое связывание с библиотекой libfunc1.dylib. При запуске программы пользователю предлагается вводить команды:

* + При вводе команды "1" программа запрашивает значения для вычисления производной функции cos(x).
  + При вводе команды "2" программа запрашивает значение для вычисления числа e.
  + Все вычисления выполняются с использованием функций, доступных из библиотеки libfunc1.dylib.

1. **Основная программа с динамической загрузкой (main2.c):**

Программа динамически загружает библиотеку (libfunc1.dylib) с помощью функции dlopen. После загрузки библиотеки функции для вычисления производной и числа e извлекаются через dlsym. Программа выполняет вычисления по выбранной пользователем команде.

Вторая программа позволяет переключаться между библиотеками. При вводе команды "0" текущая библиотека закрывается через dlclose, и затем загружается другая библиотека (например, libfunc2.dylib).

1. **Обработка команд:**

В программе с динамической загрузкой пользователь может ввести команду "0" для переключения между библиотеками и команды "1" и "2" для выполнения соответствующих вычислений (производной и числа e).

Результаты вычислений выводятся на экран после выполнения команд.

1. **Завершение работы программы:**

Для завершения работы программы пользователю предлагается ввести команду "exit". В это время открытые библиотеки закрываются через dlclose, и программа завершает выполнение.

# Код программы

Код программы приведён в приложении 1.

# Использование утилиты strace

Листинг утилиты strace приведен в приложении 2. В нем можно увидеть системные вызовы:

**main1.c**

1. execve("./main1", ["./main1"], ...)

Этот системный вызов запускает программу main1 с передачей в нее аргументов. Вернувшийся результат = 0 означает, что процесс был успешно запущен.

1. mmap(NULL, 8192, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0)

Выделение памяти для процесса (8192 байта). Это используется для размещения данных в процессе.

1. openat(AT\_FDCWD, "./lib/libfunc1.so", O\_RDONLY|O\_CLOEXEC)

Попытка открыть динамическую библиотеку libfunc1.so для загрузки, но она не найдена, о чем сообщает ошибка ENOENT. Строка повторяется для других путей, пока не будет найдено успешное открытие библиотеки.

1. mmap(0x7ff94b29e000, 4096, PROT\_READ|PROT\_EXEC,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000)

Загрузка библиотеки в память, с установкой прав доступа (чтение и выполнение) для адреса в памяти.

1. close(3)

Закрытие файлового дескриптора после завершения работы с библиотекой.

1. openat(AT\_FDCWD, "/lib/libm.so.6", O\_RDONLY|O\_CLOEXEC)

Открытие системной библиотеки libm.so.6, что является обязательным для большинства программ, использующих стандартные функции.

1. mmap(0x7ff94b06e000, 2264656, PROT\_READ,

MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0)

Загрузка библиотеки libm.so.6 в память. Процесс также меняет права доступа и фиксирует области памяти.

8. +++ exited with 0 +++

Процесс завершился с кодом возврата 0, что означает успешное выполнение программы.

**main2.c**

1. execve("./main2", ["./main2"], ...)

Запуск программы main2, аналогично первому случаю. Программа запускается без ошибок.

1. mmap(NULL, 8192, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0)

Как и в main1, происходит выделение памяти для программы.

1. openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC)

Открытие кэша динамических библиотек для загрузки необходимых зависимостей.

1. openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC)

Попытка загрузки системной библиотеки libc.so.6. Эта библиотека критична для работы программы, она загружается в память.

1. mmap(0x7fe7cc965000, 2264656, PROT\_READ,

MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0)

Загружается libc.so.6 с изменением прав доступа и установкой соответствующих областей памяти.

1. mprotect(0xffff92bbd000, 77824, PROT\_NONE)

Операция, блокирующая доступ к части памяти для защиты от изменений (для безопасности).

1. openat(AT\_FDCWD, "./lib/libfunc1.so", O\_RDONLY|O\_CLOEXEC)

Открытие библиотеки libfunc1.so для загрузки. Система пытается найти ее в различных путях, включая локальные каталоги.

1. mmap(0x7fe7ccb90000, 4096, PROT\_READ|PROT\_EXEC,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1000)

Загрузка libfunc1.so в память с правами доступа на чтение и выполнение.

1. openat(AT\_FDCWD, "/lib/aarch64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC)

Загрузка математической библиотеки libm.so.6, которая обычно требуется для выполнения математических операций.

1. openat(AT\_FDCWD, "./lib/libfunc2.so", O\_RDONLY|O\_CLOEXEC)

Попытка загрузки другой динамической библиотеки libfunc2.so, возможно, для использования других функций, специфичных для программы.

1. +++ exited with 0 +++

Программа завершена успешно с кодом 0.

# Демонстрация работы программы

(base) boopie@MacBook-Air-Irina lab4 % ./main1 Введите 'выход' для выхода.

Введите команду: 1

Введите параметры (A deltaX): 0.5 0.9

Результат производной: -0.786239

Введите команду: 2 Введите параметр (x): -1 Результат E: inf

Введите команду: выход

(base) boopie@MacBook-Air-Irina lab4 % ./main2 Введите 'выход' для выхода.

Введите '0', чтобы переключиться между библиотеками. Введите команду: 1

Введите параметры (A deltaX): 0.3 1

Результат производной: -0.687838

Введите команду: 2 Введите параметр (x): 10 Результат E: 2.593742

Введите команду: 0

переключиться в библиотеку: ./lib/libfunc2.so Введите команду: 1

Введите параметры (A deltaX): 0.3 1

Результат производной: -0.248672

Введите команду: 2 Введите параметр (x): 9 Результат E: 2.718282 Введите команду: выход

# Вывод

В ходе выполнения лабораторной работы была реализована программа, демонстрирующая два подхода к вычислению производной функции cos(x) и числа “e” с использованием динамических библиотек. Первый подход, реализованный в файле main1.c, использует подключение библиотеки на этапе компиляции. Этот метод является простым и надежным, так как все функции подключаются заранее, что гарантирует корректность работы программы.

Второй подход, реализованный в файле main2.c, использует динамическую загрузку библиотеки с помощью функций из библиотеки dlfcn.h. Этот метод предоставляет гибкость, позволяя загружать библиотеку в процессе выполнения программы, что делает возможным обновление или замену библиотеки без необходимости перекомпиляции всей программы.

Оба подхода продемонстрировали свою эффективность и корректность в вычислениях, а также показали различия в подходах к взаимодействию с библиотеками. Работа над лабораторной была полезна для изучения принципов работы с динамическими библиотеками и их применения в реальных задачах.

## Приложение 1

**src/func.h**

#ifndef FUNC\_H #define FUNC\_H

float Derivative(float A, float deltaX); float E(int x);

#endif // FUNC\_H

**src/func1.c**

**#include <math.h> #include "func.h"**

**// Первая реализация производной**

**float Derivative(float A, float deltaX) {**

**return (cos(A + deltaX) - cos(A)) / deltaX;**

**}**

**// Первая реализация e float E(int x) {**

**return pow(1 + 1.0 / x, x);**

**}**

**src/func2.c**

#include <math.h> #include "func.h"

// Вторая реализация производной

float Derivative(float A, float deltaX) {

return (cos(A + deltaX) - cos(A - deltaX)) / (2 \* deltaX);

}

// Вторая реализация e float E(int x) {

float result = 1.0, term = 1.0; for (int n = 1; n <= x; n++) {

term /= n;

result += term;

}

return result;

}

**src/main1.c**

#include <stdio.h> #include <string.h> #include "func.h"

int main() {

char command[256];

printf("Введите 'выход' для выхода.\n");

while (1) {

printf("Введите команду: "); scanf("%s", command);

if (strcmp(command, "выход") == 0) { break;

}

if (strcmp(command, "1") == 0) { float A, deltaX;

printf("Введите параметры (A deltaX): "); scanf("%f %f", &A, &deltaX);

printf("Результат производной: %f\n", Derivative(A, deltaX));

} else if (strcmp(command, "2") == 0) { int x;

printf("Введите параметр (x): "); scanf("%d", &x);

printf("Результат E: %f\n", E(x));

} else {

printf("Неизвестная команда.\n");

}

}

return 0;

}

**src/main2.c**

#include <stdio.h> #include <stdlib.h> #include <dlfcn.h> #include <string.h> #include "func.h"

int main() {

void \*libHandle;

float (\*Derivative)(float, float); float (\*E)(int);

char command[256];

//char currentLib[256] = "./lib/libfunc1.so"; char currentLib[256] = "./lib/libfunc1.dylib";

printf("Введите 'выход' для выхода.\n");

printf("Введите '0', чтобы переключиться между библиотеками.\n");

// Загрузка первой библиотеки

libHandle = dlopen(currentLib, RTLD\_LAZY); if (!libHandle) {

fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror()); return 1;

}

while (1) {

printf("Введите команду: "); scanf("%s", command);

if (strcmp(command, "выход") == 0) { break;

}

if (strcmp(command, "0") == 0) { dlclose(libHandle);

//if (strcmp(currentLib, "./lib/libfunc1.so") == 0) {

// strcpy(currentLib, "./lib/libfunc2.so");

//} else {

// strcpy(currentLib, "./lib/libfunc1.so");

//}

if (strcmp(currentLib, "./lib/libfunc1.dylib") == 0) { strcpy(currentLib, "./lib/libfunc2.dylib");

} else {

strcpy(currentLib, "./lib/libfunc1.dylib");

}

libHandle = dlopen(currentLib, RTLD\_LAZY); if (!libHandle) {

fprintf(stderr, "Ошибка загрузки библиотеки: %s\n", dlerror()); return 1;

}

printf("переключиться в библиотеку: %s\n", currentLib); continue;

}

// Загрузка функций из библиотеки

Derivative = dlsym(libHandle, "Derivative"); E = dlsym(libHandle, "E");

if (!Derivative || !E) {

fprintf(stderr, "Функции не найдены: %s\n", dlerror()); dlclose(libHandle);

return 1;

}

if (strcmp(command, "1") == 0) { float A, deltaX;

printf("Введите параметры (A deltaX): "); scanf("%f %f", &A, &deltaX);

printf("Результат производной: %f\n", Derivative(A, deltaX));

} else if (strcmp(command, "2") == 0) { int x;

printf("Введите параметр (x): "); scanf("%d", &x);

printf("Результат E: %f\n", E(x));

} else {

printf("Неизвестная команда.\n");

}

}

// Закрытие библиотеки перед выходом

dlclose(libHandle); return 0;

}

**Makefile**

# Компилятор

CC = gcc

# Флаги для создания динамических библиотек

CFLAGS = -fPIC -shared

LDFLAGS = -lm # Подключение math.h

# Правило по умолчанию

all: lib/libfunc1.dylib lib/libfunc2.dylib main1 main2 #all: lib/libfunc1.so lib/libfunc2.so main1 main2

# Создание динамической библиотеки libfunc1.dylib lib/libfunc1.dylib: src/func1.c src/func.h

$(CC) $(CFLAGS) -o $@ $< $(LDFLAGS)

# Создание динамической библиотеки libfunc2.dylib lib/libfunc2.dylib: src/func2.c src/func.h

$(CC) $(CFLAGS) -o $@ $< $(LDFLAGS)

# Создание динамической библиотеки libfunc1.so #lib/libfunc1.so: src/func1.c src/func.h

# $(CC) $(CFLAGS) -o $@ $< $(LDFLAGS) # <- Добавлен -lm

# Создание динамической библиотеки libfunc2.so #lib/libfunc2.so: src/func2.c src/func.h

# $(CC) $(CFLAGS) -o $@ $< $(LDFLAGS) # <- Добавлен -lm

# Компиляция main1 с использованием libfunc1.dylib main1: src/main1.c

$(CC) -o $@ $< -L./lib -lfunc1 $(LDFLAGS) -Wl,-rpath,./lib

# Компиляция main2 (если требуется динамическая загрузка) main2: src/main2.c

$(CC) -o $@ $< $(LDFLAGS)

# Компиляция main1 с использованием libfunc1.so #main1: src/main1.c

# $(CC) -o $@ $< -L./lib -lfunc1 $(LDFLAGS) -Wl,-rpath=./lib # <- Добавлен -lm

# Компиляция main2 (если требуется динамическая загрузка) #main2: src/main2.c

# $(CC) -o $@ $< $(LDFLAGS)

# Очистка

clean:

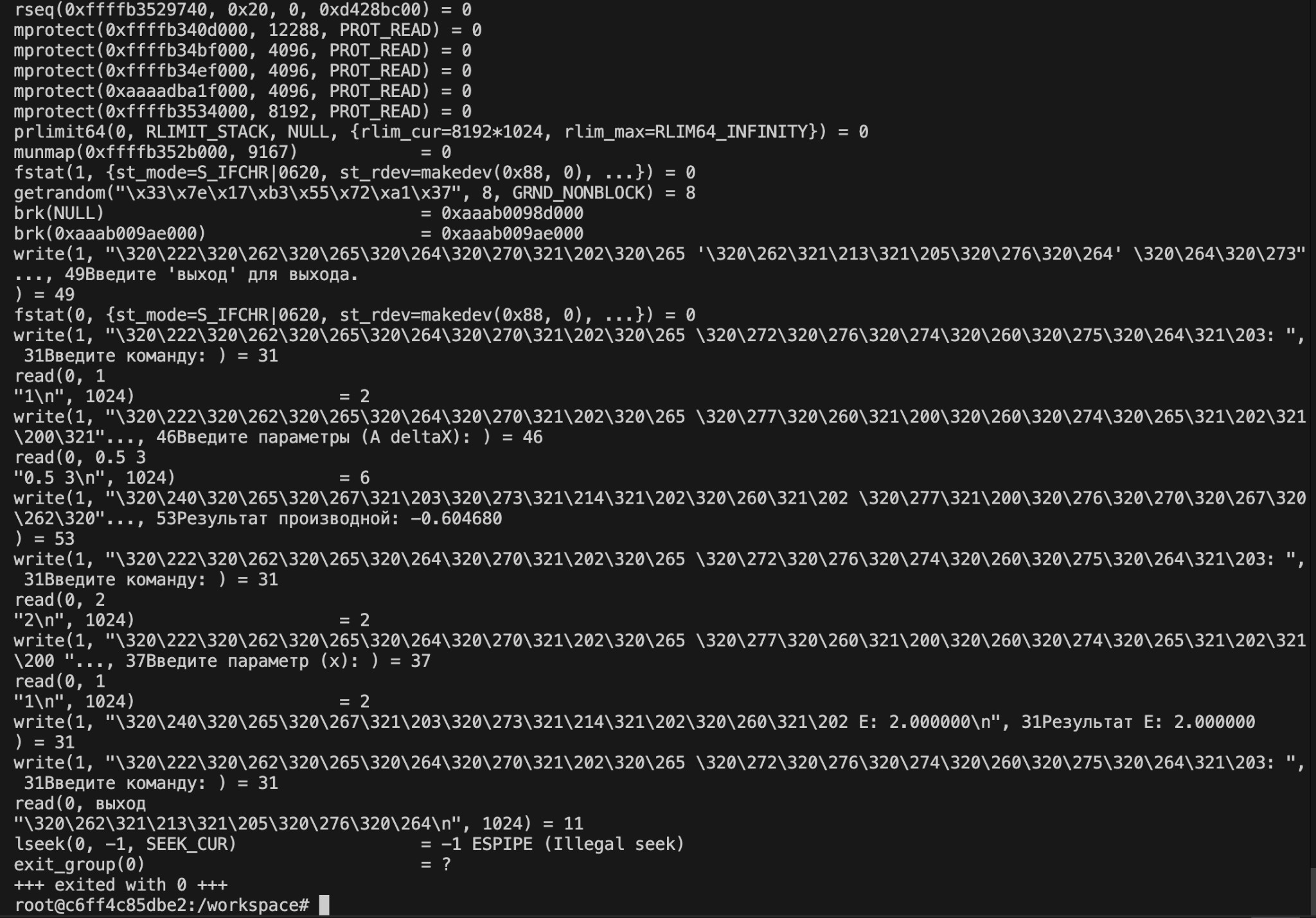
rm -f main1 main2 lib/\*.dylib # Очистка

#clean:

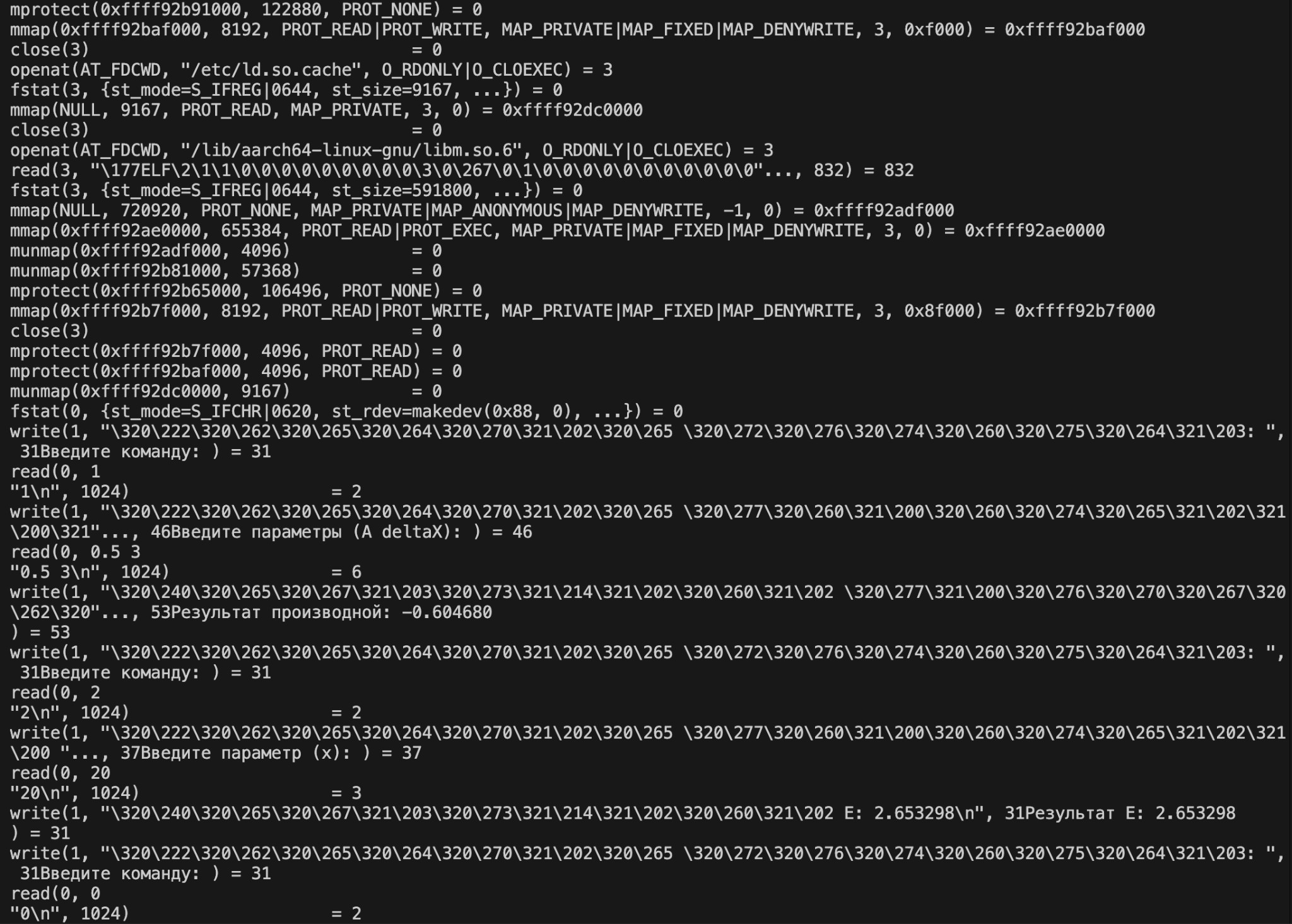
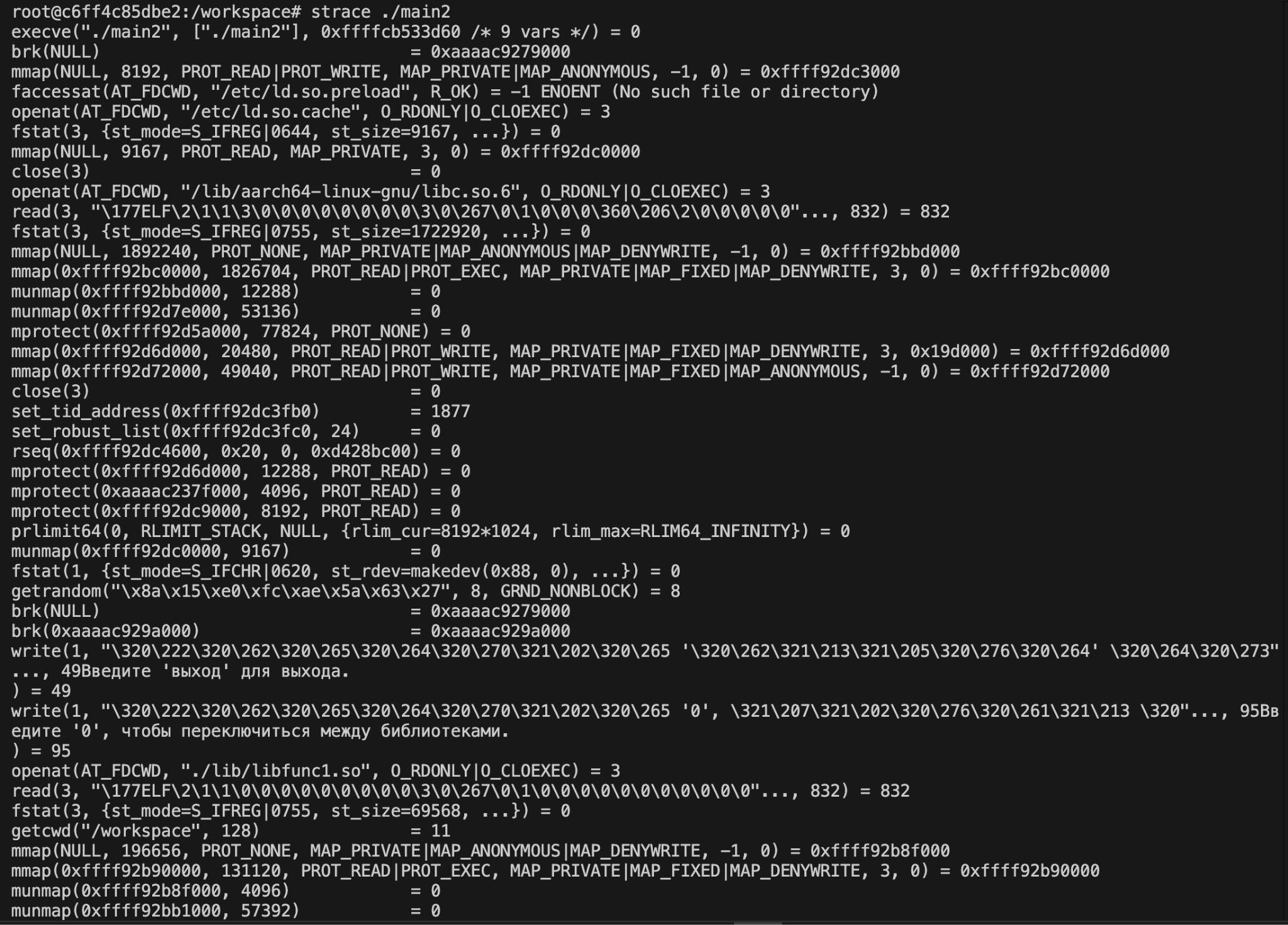
# rm -f main1 main2 lib/\*.so

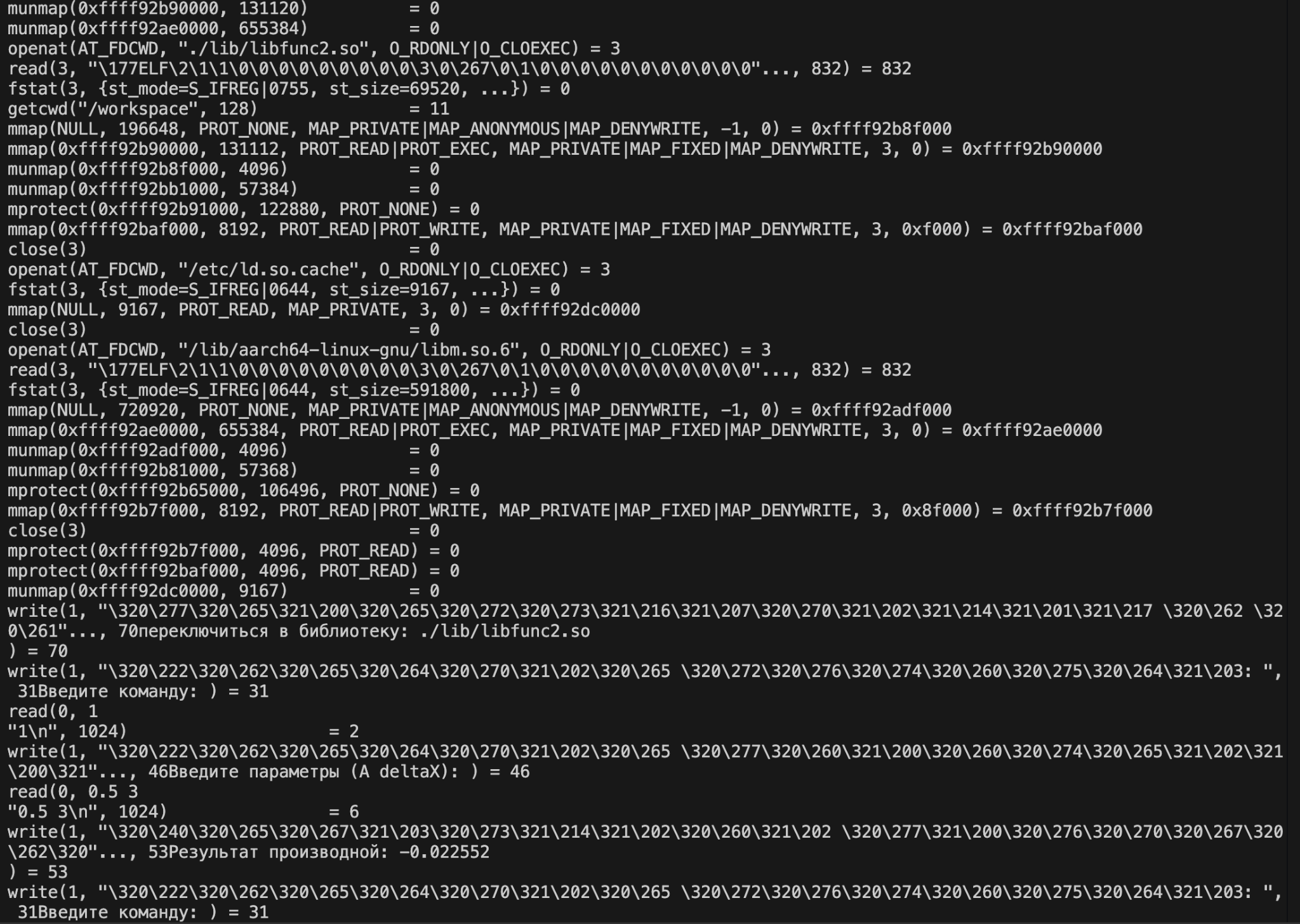
## Приложение 2

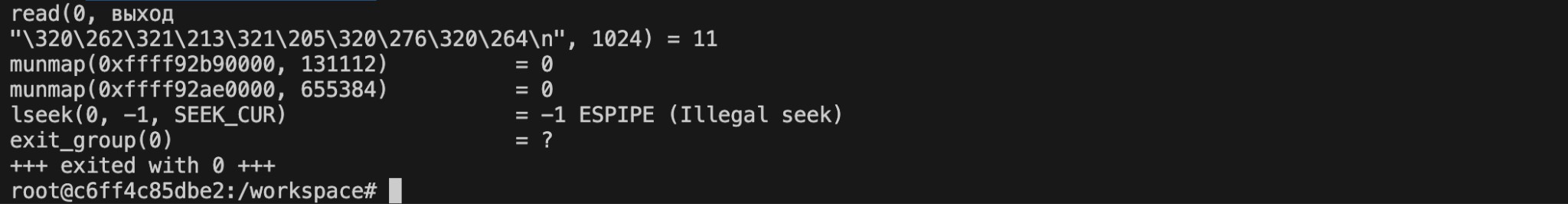
**main1.c**

****

**main2.c**

****



****