Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №4 по курсу

«Операционные системы»

Группа: М8О-216Б-24

Студент: Седов М. А

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 19.11.25

Москва, 2025

Постановка задачи

Вариант 19.

Цель работы:
● Создание динамических библиотек
● Создание программ, которые используют функции динамических библиотек
Задание:
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:
● Динамические библиотеки, реализующие контракты, которые заданы вариантом;
● Тестовая программа (программа №1), которая используют одну из библиотек, используя информацию полученные на этапе компиляции;
● Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:
● Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
● "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 ее выполнения;
● "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 ее выполнения.

Контракты и реализации функций
Подсчёт количества простых чисел на отрезке [a, b] (a, b – натуральные):
Сигнатура функции: int prime_count(int a, int b);
● Реализация №1: Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.
● Реализация №2: Решето Эратосфена

Подсчет площади плоской геометрической фигуры по двум сторонам:
Сигнатура функции: float area(float a, float b);
● Реализация №1: Фигура прямоугольник
● Реализация №2: Фигура прямоугольный треугольник

Общий метод и алгоритм решения

Использованные системные вызовы:

● void *dlopen(const char *filename, int flags) – открывает динамическую библиотеку и возвращает дескриптор. Возвращает NULL при ошибке.

● int dlclose(void *handle) – закрывает динамическую библиотеку. Возвращает 0 при успехе, ненулевое значение при ошибке.

● void *dlsym(void *handle, const char *symbol) – возвращает адрес функции или переменной из библиотеки. Возвращает NULL при ошибке.

● char *dlerror(void) – возвращает строку с описанием последней ошибки dlopen/dlclose/dlsym.

● ssize_t read(int fd, void *buf, size_t count) – читает данные из файлового дескриптора. Возвращает количество прочитанных байт, 0 при достижении конца файла, -1 при ошибке.

● ssize_t write(int fd, const void *buf, size_t count) – записывает данные в файловый дескриптор. Возвращает количество записанных байт, -1 при ошибке.

Алгоритм работы программы:

1. **Программа №1 (использование библиотек на этапе компиляции):**
    - **Компилируется с явным указанием используемой динамической библиотеки**
    - **При запуске отображает доступные команды**
    - **В цикле читает команды пользователя из stdin**
    - **Парсит команды вида "1 a b" или "2 a b"**
    - **Вызывает соответствующие функции из подключенной библиотеки**
    - **Выводит результаты на экран**
    - **Завершается при вводе команды "q"**
2. **Программа №2 (динамическая загрузка библиотек):**
    - **При запуске загружает первую библиотеку с помощью dlopen()**
    - **Получает указатели на функции через dlsym()**
    - **Отображает доступные команды (включая "0" для переключения библиотек)**
    - **В цикле читает команды пользователя из stdin**
    - **При команде "0" закрывает текущую библиотеку и загружает другую**
    - **При командах "1 a b" или "2 a b" вызывает соответствующие функции**
    - **Выводит результаты на экран**
    - **При завершении освобождает ресурсы через dlclose()**
    - **Завершается при вводе команды "q"**

Код программы

functions.h - заголовочный файл с объявлениями функций

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

int prime_count(int a, int b);

float area(float a, float b);

#endif
```

lib1.c - первая реализация библиотеки (наивный алгоритм + прямоугольник)

```c
#include "functions.h"

static int is_prime_naive(int n) {
    if (n < 2) return 0;
    if (n == 2) return 1;
    if (n % 2 == 0) return 0;

    for (int i = 3; i < n; i += 2) {
        if (n % i == 0) {
            return 0;
        }
    }
    return 1;
}

int prime_count(int a, int b) {
    if (a < 1) a = 1;
    if (b < a) return 0;

    int count = 0;
    for (int i = a; i <= b; i++) {
        if (is_prime_naive(i)) {
            count++;
        }
    }
    return count;
}

float area(float a, float b) {
    return a * b;
}
```

lib2.c - вторая реализация библиотеки (решето Эратосфена + прямоугольный треугольник)

```c
#include "functions.h"

#define MAX_SIEVE_SIZE 1000000

int prime_count(int a, int b) {
    if (a < 1) a = 1;
    if (b < 2) return 0;
    if (b < a) return 0;

    // Ограничение на максимальный размер для статического массива
    if (b >= MAX_SIEVE_SIZE) {
        b = MAX_SIEVE_SIZE - 1;
    }

    static char is_prime[MAX_SIEVE_SIZE];

    // Инициализация массива
    for (int i = 0; i <= b; i++) {
        is_prime[i] = 1;
    }
    is_prime[0] = 0;
    is_prime[1] = 0;

    for (int i = 2; i * i <= b; i++) {
        if (is_prime[i]) {
            for (int j = i * i; j <= b; j += i) {
                is_prime[j] = 0;
            }
        }
    }

    int count = 0;
```

```
    for (int i = (a < 2 ? 2 : a); i <= b; i++) {
        if (is_prime[i]) {
            count++;
        }
    }

    return count;
}

float area(float a, float b) {
    return (a * b) / 2.0f;
}
```

program1.c - программа с линковкой библиотек на этапе компиляции

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include "functions.h"

static void write_str(int fd, const char* str) {
    write(fd, str, strlen(str));
}

static void write_int(int fd, int num) {
    char buf[32];
    int len = snprintf(buf, sizeof(buf), "%d", num);
    write(fd, buf, len);
}

static void write_float(int fd, float num) {
    char buf[32];
    int len = snprintf(buf, sizeof(buf), "%.4f", num);
    write(fd, buf, len);
}

static void write_newline(int fd) {
    write(fd, "\n", 1);
}

static ssize_t read_line(char* buffer, size_t size) {
    ssize_t total_read = 0;
    while (total_read < size - 1) {
        char c;
        ssize_t n = read(0, &c, 1);
        if (n <= 0) {
            return -1;
        }
        if (c == '\n') {
            break;
        }
        buffer[total_read++] = c;
    }
    buffer[total_read] = '\0';
    return total_read;
}

static int parse_int(const char* str, int* result) {
    if (!str || !*str) return 0;
    *result = atoi(str);
    return 1;
}

static int parse_float(const char* str, float* result) {
    if (!str || !*str) return 0;
```

```
    *result = atof(str);
    return 1;
}

static int tokenize(char* str, char* tokens[], int max_tokens) {
    int count = 0;
    char* token = strtok(str, " \t");
    while (token && count < max_tokens) {
        tokens[count++] = token;
        token = strtok(NULL, " \t");
    }
    return count;
}

int main(void) {
    write_str(1, "Commands: 1 a b | 2 a b | q\n");

    char input[256];

    while (1) {
        write_str(1, "> ");

        if (read_line(input, sizeof(input)) < 0) {
            break;
        }

        if (input[0] == 'q' || input[0] == 'Q') {
            break;
        }

        char* tokens[4];
        int token_count = tokenize(input, tokens, 4);

        if (token_count == 0) {
            write_str(1, "Error\n");
            continue;
        }

        int command;
        if (!parse_int(tokens[0], &command)) {
            write_str(1, "Error\n");
            continue;
        }

        switch (command) {
            case 1: {
                if (token_count != 3) {
                    write_str(1, "Error\n");
                    break;
                }
                int a_int, b_int;
                if (!parse_int(tokens[1], &a_int) || !parse_int(tokens[2], &b_int)) {
                    write_str(1, "Error\n");
                    break;
                }
                if (a_int < 1 || b_int < 1) {
                    write_str(1, "Error\n");
                    break;
                }
                int count = prime_count(a_int, b_int);
                write_int(1, count);
                write_newline(1);
                break;
            }
```

```
            case 2: {
                if (token_count != 3) {
                    write_str(1, "Error\n");
                    break;
                }
                float a_float, b_float;
                if (!parse_float(tokens[1], &a_float) || !parse_float(tokens[2], &b_float)) {
                    write_str(1, "Error\n");
                    break;
                }
                float result = area(a_float, b_float);
                write_float(1, result);
                write_newline(1);
                break;
            }

            default:
                write_str(1, "Error\n");
                break;
        }
    }

    return 0;
}
```

program2.c - программа с динамической загрузкой библиотек

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>

typedef int (*prime_count_func)(int, int);
typedef float (*area_func)(float, float);

static const char *LIB1_PATH = "./libfunctions1.so";
static const char *LIB2_PATH = "./libfunctions2.so";

static void *current_lib = NULL;
static int current_lib_num = 0;

static prime_count_func fn_prime_count = NULL;
static area_func fn_area = NULL;

static void write_str(int fd, const char* str) {
    write(fd, str, strlen(str));
}

static void write_int(int fd, int num) {
    char buf[32];
    int len = snprintf(buf, sizeof(buf), "%d", num);
    write(fd, buf, len);
}

static void write_float(int fd, float num) {
    char buf[32];
    int len = snprintf(buf, sizeof(buf), "%.4f", num);
    write(fd, buf, len);
}

static void write_newline(int fd) {
    write(fd, "\n", 1);
}

static ssize_t read_line(char* buffer, size_t size) {
```

```
    ssize_t total_read = 0;
    while (total_read < size - 1) {
        char c;
        ssize_t n = read(0, &c, 1);
        if (n <= 0) {
            return -1;
        }
        if (c == '\n') {
            break;
        }
        buffer[total_read++] = c;
    }
    buffer[total_read] = '\0';
    return total_read;
}

static int parse_int(const char* str, int* result) {
    if (!str || !*str) return 0;
    *result = atoi(str);
    return 1;
}

static int parse_float(const char* str, float* result) {
    if (!str || !*str) return 0;
    *result = atof(str);
    return 1;
}

static int tokenize(char* str, char* tokens[], int max_tokens) {
    int count = 0;
    char* token = strtok(str, " \t");
    while (token && count < max_tokens) {
        tokens[count++] = token;
        token = strtok(NULL, " \t");
    }
    return count;
}

static int load_library(const char *path, int lib_num) {
    if (current_lib != NULL) {
        dlclose(current_lib);
        current_lib = NULL;
        fn_prime_count = NULL;
        fn_area = NULL;
    }

    current_lib = dlopen(path, RTLD_NOW);
    if (current_lib == NULL) {
        char error_buf[256];
        const char* dl_error = dlerror();
        if (dl_error) {
            int len = snprintf(error_buf, sizeof(error_buf), "Error loading library %s: %s\n",
path, dl_error);
            write(2, error_buf, len);
        }
        return -1;
    }

    dlerror();

    fn_prime_count = (prime_count_func)dlsym(current_lib, "prime_count");
    const char *error = dlerror();
    if (error != NULL) {
        char error_buf[256];
```

```
        int len = snprintf(error_buf, sizeof(error_buf), "Error loading prime_count: %s\n",
error);
        write(2, error_buf, len);
        dlclose(current_lib);
        current_lib = NULL;
        return -1;
    }

    fn_area = (area_func)dlsym(current_lib, "area");
    error = dlerror();
    if (error != NULL) {
        char error_buf[256];
        int len = snprintf(error_buf, sizeof(error_buf), "Error loading area: %s\n", error);
        write(2, error_buf, len);
        dlclose(current_lib);
        current_lib = NULL;
        fn_prime_count = NULL;
        return -1;
    }

    current_lib_num = lib_num;
    return 0;
}

static void switch_library(void) {
    const char *path;
    int new_lib_num;

    if (current_lib_num != 2) {
        path = LIB2_PATH;
        new_lib_num = 2;
    } else {
        path = LIB1_PATH;
        new_lib_num = 1;
    }

    if (load_library(path, new_lib_num) == 0) {
        write_str(1, "Library ");
        write_int(1, new_lib_num);
        write_newline(1);
    }
}

static void print_current_lib_info(void) {
    if (current_lib_num == 1) {
        write_str(1, "Lib 1");
    } else if (current_lib_num == 2) {
        write_str(1, "Lib 2");
    } else {
        write_str(1, "No lib");
    }
}

int main(void) {
    write_str(1, "Commands: 0 | 1 a b | 2 a b | q\n");

    if (load_library(LIB1_PATH, 1) != 0) {
        write_str(2, "Error\n");
        return 1;
    }

    char input[256];

    while (1) {
        print_current_lib_info();
```

```
        write_str(1, "> ");

        if (read_line(input, sizeof(input)) < 0) {
            break;
        }

        if (input[0] == 'q' || input[0] == 'Q') {
            break;
        }

        char* tokens[4];
        int token_count = tokenize(input, tokens, 4);

        if (token_count == 0) {
            write_str(1, "Error\n");
            continue;
        }

        int command;
        if (!parse_int(tokens[0], &command)) {
            write_str(1, "Error\n");
            continue;
        }

        switch (command) {
            case 0:
                switch_library();
                break;

            case 1:
                if (fn_prime_count == NULL) {
                    write_str(1, "Error\n");
                    break;
                }
                if (token_count != 3) {
                    write_str(1, "Error\n");
                    break;
                }
                int a_int, b_int;
                if (!parse_int(tokens[1], &a_int) || !parse_int(tokens[2], &b_int)) {
                    write_str(1, "Error\n");
                    break;
                }
                if (a_int < 1 || b_int < 1) {
                    write_str(1, "Error\n");
                    break;
                }
                int count = fn_prime_count(a_int, b_int);
                write_int(1, count);
                write_newline(1);
                break;

            case 2:
                if (fn_area == NULL) {
                    write_str(1, "Error\n");
                    break;
                }
                if (token_count != 3) {
                    write_str(1, "Error\n");
                    break;
                }
                float a_float, b_float;
                if (!parse_float(tokens[1], &a_float) || !parse_float(tokens[2], &b_float)) {
                    write_str(1, "Error\n");
                    break;
```

```
                }
                float result = fn_area(a_float, b_float);
                write_float(1, result);
                write_newline(1);
                break;

            default:
                write_str(1, "Error\n");
                break;
        }
    }

    if (current_lib != NULL) {
        dlclose(current_lib);
    }

    return 0;
}
```

Протокол работы программы

Тестирование:

Программа была протестирована на различных входных данных. Примеры тестов:

Тест 1: Program1 (с линковкой libfunctions1.so)

1. $ ./program1
2. Commands: 1 a b | 2 a b | q
3. 1 1 10
4. 4
5. 2 3.0 4.0
6. 12.0000
7. q

Тест 2: Program2 (динамическая загрузка)

8. $ ./program2
9. Commands: 0 | 1 a b | 2 a b | q
10. Lib 1> 1 1 20
11. 8
12. Lib 1> 2 5.0 6.0
13. 30.0000
14. Lib 1> 0
15. Lib 2> 1 1 20
16. 8
17. Lib 2> 2 5.0 6.0
18. 15.0000
19. Lib 2> q

Strace:

Ниже представлен фрагмент вывода `strace` с выделенными системными вызовами, используемыми в нашей программе. Написанные ниже строки соответствуют системным вызовам, вызванным непосредственно из нашего кода:

```
execve("./program1", ["./program1"], 0x7ffe9453fad0 /* 26 vars */) = 0

brk(NULL)                               = 0x5d0ec20dd000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffeae416120) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7caa1347c000

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

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/glibc-hwcaps/x86-64-
v3/libfunctions1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/glibc-hwcaps/x86-64-v3",
0x7ffeae415340, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/glibc-hwcaps/x86-64-
v2/libfunctions1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/glibc-hwcaps/x86-64-v2",
0x7ffeae415340, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD,
"/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls/haswell/x86_64/libfunctions1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls/haswell/x86_64",
0x7ffeae415340, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls/haswell/libfunctions1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls/haswell",
0x7ffeae415340, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls/x86_64/libfunctions1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls/x86_64",
0x7ffeae415340, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls/libfunctions1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/tls", 0x7ffeae415340, 0) =
-1 ENOENT (No such file or directory)

openat(AT_FDCWD,
"/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/haswell/x86_64/libfunctions1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/haswell/x86_64",
0x7ffeae415340, 0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/haswell/libfunctions1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/haswell", 0x7ffeae415340,
0) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/x86_64/libfunctions1.so",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
```

```
newfstatat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/x86_64", 0x7ffeae415340, 0)
= -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/libfunctions1.so",
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|0777, st_size=15208, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7caa13477000

mmap(0x7caa13478000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7caa13478000

mmap(0x7caa13479000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7caa13479000

mmap(0x7caa1347a000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x7caa1347a000

close(3)                                = 0

openat(AT_FDCWD, "/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build/libc.so.6", O_RDONLY|O_CLOEXEC)
= -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=21124, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 21124, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7caa13471000

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\0O{\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) = 0x7caa13200000

mprotect(0x7caa13228000, 2023424, PROT_NONE) = 0

mmap(0x7caa13228000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x7caa13228000

mmap(0x7caa133bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7caa133bd000

mmap(0x7caa13416000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000) = 0x7caa13416000

mmap(0x7caa1341c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0)
= 0x7caa1341c000

close(3)                                = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7caa1346e000
```

```
arch_prctl(ARCH_SET_FS, 0x7caa1346e740) = 0
set_tid_address(0x7caa1346ea10)         = 397
set_robust_list(0x7caa1346ea20, 24)     = 0
rseq(0x7caa1346f0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7caa13416000, 16384, PROT_READ) = 0
mprotect(0x7caa1347a000, 4096, PROT_READ) = 0
mprotect(0x5d0eb2780000, 4096, PROT_READ) = 0
mprotect(0x7caa134b6000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7caa13471000, 21124)           = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
getrandom("\xa3\x1b\xc8\xb8\x19\x9c\x3c\x05", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5d0ec20dd000
brk(0x5d0ec20fe000)                     = 0x5d0ec20fe000
write(1, "Commands: 1 a b | 2 a b | q\n", 28) = 28
write(1, "> ", 2)                       = 2
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
read(0, "1 2 3\n", 1024)                = 6
write(1, "2\n", 2)                      = 2
write(1, "> ", 2)                       = 2
read(0, "2 2 4\n", 1024)                = 6
write(1, "8.0000\n", 7)                 = 7
write(1, "> ", 2)                       = 2
read(0, "\320\271\n", 1024)             = 3
write(1, "Error\n", 6)                  = 6
write(1, "> ", 2)                       = 2
read(0, "q\n", 1024)                    = 2
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
execve("./program2", ["./program2"], 0x7fffe9abc150 /* 26 vars */) = 0
brk(NULL)                               = 0x56e7e2ec7000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc96ef8960) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x71841ff0e000
```

```
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=21124, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 21124, PROT_READ, MAP_PRIVATE, 3, 0) = 0x71841ff08000
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\0O{\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) = 0x71841fc00000
mprotect(0x71841fc28000, 2023424, PROT_NONE) = 0
mmap(0x71841fc28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x28000) = 0x71841fc28000
mmap(0x71841fdbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x71841fdbd000
mmap(0x71841fe16000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x215000) = 0x71841fe16000
mmap(0x71841fe1c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0)
= 0x71841fe1c000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x71841ff05000
arch_prctl(ARCH_SET_FS, 0x71841ff05740) = 0
set_tid_address(0x71841ff05a10)         = 401
set_robust_list(0x71841ff05a20, 24)     = 0
rseq(0x71841ff060e0, 0x20, 0, 0x53053053) = 0
mprotect(0x71841fe16000, 16384, PROT_READ) = 0
mprotect(0x56e7b8c2d000, 4096, PROT_READ) = 0
mprotect(0x71841ff48000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x71841ff08000, 21124)           = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
getrandom("\x77\xa5\xdf\x92\x26\x4a\x81\xf1", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x56e7e2ec7000
brk(0x56e7e2ee8000)                     = 0x56e7e2ee8000
```

```
write(1, "Commands: 0 | 1 a b | 2 a b | q\n", 32) = 32

openat(AT_FDCWD, "./libfunctions1.so", 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|0777, st_size=15208, ...}, AT_EMPTY_PATH) = 0

getcwd("/mnt/c/MAI/3_sem/OS/OS_LAB/OS_LAB_No4/build", 128) = 44

mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x71841ff09000

mmap(0x71841ff0a000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x71841ff0a000

mmap(0x71841ff0b000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x71841ff0b000

mmap(0x71841ff0c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x71841ff0c000

close(3)                                = 0

mprotect(0x71841ff0c000, 4096, PROT_READ) = 0

write(1, "Lib 1> ", 7)                  = 7

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

read(0, "\n", 1024)                     = 1

write(1, "Error\n", 6)                  = 6

write(1, "Lib 1> ", 7)                  = 7

read(0, "1 2 3\n", 1024)                = 6

write(1, "2\n", 2)                      = 2

write(1, "Lib 1> ", 7)                  = 7

read(0, "2 2 4\n", 1024)                = 6

write(1, "8.0000\n", 7)                 = 7

write(1, "Lib 1> ", 7)                  = 7

read(0, "q\n", 1024)                    = 2

munmap(0x71841ff09000, 16424)           = 0

exit_group(0)                           = ?

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

Ключевые системные вызовы из нашего кода:

1. openat(AT_FDCWD, "./libfunctions1.so", O_RDONLY|O_CLOEXEC) – открытие файла динамической библиотеки

2. mmap(NULL, size, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, fd, 0) – отображение библиотеки в память

3. mprotect(addr, size, PROT_READ) – установка защиты памяти для загруженной библиотеки

4. munmap(addr, size) – удаление отображения библиотеки из памяти

5. read(fd, ...) – чтение данных из stdin

6. write(fd, ...) – запись данных в stdout

7. exit_group(0) – завершение процесса

Все системные вызовы проверяются на ошибки: возвращаемые значения сравниваются с -1 (или 0 для некоторых вызовов), и при ошибке программа выводит сообщение и завершается.

Вывод

В ходе выполнения лабораторной работы были созданы динамические библиотеки с двумя различными реализациями функций подсчета простых чисел и вычисления площади геометрических фигур. Были реализованы две программы, демонстрирующие различные способы использования динамических библиотек.

- Реализованы две динамические библиотеки (libfunctions1.so и libfunctions2.so) с различными алгоритмами:
  - Наивный алгоритм и решето Эратосфена для подсчета простых чисел
  - Вычисление площади прямоугольника и прямоугольного треугольника
- Создана программа program1 с линковкой библиотек на этапе компиляции
- Создана программа program2 с динамической загрузкой библиотек во время выполнения
- Реализован интерфейс для переключения между реализациями в program2
- Все программы корректно обрабатывают входные данные и выводят результаты

При выполнении работы были изучены механизмы создания и использования динамических библиотек в Linux, включая статическую и динамическую линковку, функции dlopen/dlsym/dlclose для работы с библиотеками, а также принципы разделения интерфейса и реализации. Программы успешно прошли тестирование и демонстрируют оба подхода к использованию динамических библиотек.