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-212БВ-24

Студент: Сучков В.С.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 13.12.25

Москва, 2025

Постановка задачи

Вариант 31.

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.

Общий метод и алгоритм решения

Использованные новые функции:

`dlopen()` - Загружает указанную .so библиотеку в память и возвращает её дескриптор.

`dlsym()` - Используя дескриптор, находит адрес функции или переменной по её имени (строке) и возвращает указатель.

`dlclose()` - Выгружает библиотеку, если она больше не используется другими частями программы.

Алгоритм решения:

Программа №1 (static.c) — Статическая Компоновка:
Эта программа использует функции из динамических библиотек, но связывает их на этапе компиляции.

- Чтение ввода: Программа в бесконечном цикле читает команды из стандартного ввода (stdin) с помощью низкоуровневого системного вызова read().
- Парсинг команды: Используется функция sscanf для быстрого определения, какую команду (1 или 2) и с какими аргументами ввёл пользователь.
- Обработка команды:

а. Если команда 1, программа напрямую вызывает функцию e(x).
б. Если команда 2, программа напрямую вызывает функцию area(a, b).

- Выход: При вводе exit цикл прерывается.

Программа №2 (dynamic.c) - Динамическая Компоновка:

- Чтение ввода: Аналогично, программа читает команды из stdin через read().
- Инициализация: В начале main вызываются dlopen(), который загружает библиотеку реализаций 1 (libvar1.so), и dlsym(), который получает указатели на функции (e_func, area_func).
- Обработка команды 0 (Переключение): При вводе 0 программа сначала выгружает текущие библиотеки (dlclose()). Затем она определяет новую версию и загружает соответствующие ей .so файлы через dlopen(). А потом получает новые указатели на функции через dlsym().
- Обработка команд 1 и 2 (Вычисление): Команды парсятся с помощью sscanf. Вместо прямого вызова, программа использует указатели на функции (e_func(x) или area_func(a, b)), которые указывают на версию, загруженную в данный момент.
- Выход: При выходе программа обязательно вызывает dlclose() для освобождения всех загруженных библиотек из памяти.

Код программы

static.c:

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


int main() {
    char buffer[256];
```

```
    int n;

    while (1) {
        n = read(STDIN_FILENO, buffer, sizeof(buffer) - 1);
        if (n <= 0) break;

        buffer[n] = '\0';

        if (buffer[n-1] == '\n') {
            buffer[n-1] = '\0';
        }

        if (strcmp(buffer, "exit") == 0) {
            break;
        }

        if (buffer[0] == '1') {
            int x;
            if (sscanf(buffer + 1, "%d", &x) == 1) {
                if (x > 0) {
                    float result = e(x);
                    char output[256];
                    int len = snprintf(output, sizeof(output),
"e(%d) = %f\n", x, result);
                    write(STDOUT_FILENO, output, len);
                } else {
                    const char msg[] = "Ошибка. x должен быть
положительным\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                }
            } else {
                const char msg[] = "Неверный формат команды.
Используйте: 1 x\n";
```

```
                write(STDERR_FILENO, msg, sizeof(msg));
            }
        } else if (buffer[0] == '2') {
            float a, b;
            if (sscanf(buffer + 1, "%f %f", &a, &b) == 2) {
                if (a > 0 && b > 0) {
                    float result = area(a, b);
                    char output[256];
                    int len = snprintf(output, sizeof(output),
"area(%.2f, %.2f) = %.2f\n", a, b, result);
                    write(STDOUT_FILENO, output, len);
                } else {
                    const char msg[] = "Ошибка. a и b должны
быть положительными\n";
                    write(STDERR_FILENO, msg, sizeof(msg));
                }
            } else {
                const char msg[] = "Неверный формат команды.
Используйте: 2 a b\n";
                write(STDERR_FILENO, msg, sizeof(msg));
            }
        } else {
            const char msg[] = "Неизвестная команда\n";
            write(STDERR_FILENO, msg, sizeof(msg));
        }
    }

    return 0;
}
```

dynamic.c:

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

```c
#include <unistd.h>
#include <string.h>

typedef float e_func(int x);
typedef float area_func(float a, float b);

int main() {
    void *lib;
    e_func *e;
    area_func *area;

    int current_lib = 1;

    if (current_lib == 1) {
        lib = dlopen("./libvar1.so", RTLD_NOW);
    } else {
        lib = dlopen("./libvar2.so", RTLD_NOW);
    }

    if (!lib) {
        const char msg[] = "Ошибка загрузки библиотеки\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        return 1;
    }

    e = (e_func *)dlsym(lib, "e");
    area = (area_func *)dlsym(lib, "area");

    if (!e || !area) {
        const char msg[] = "Ошибка загрузки функций\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        dlclose(lib);
```

```
        return 1;
    }

    char buffer[256];
    int n;

    const char msg[] = "Текущая библиотека - 1\n";
    write(STDOUT_FILENO, msg, sizeof(msg));

    while (1) {
        n = read(STDIN_FILENO, buffer, sizeof(buffer) - 1);
        if (n <= 0) break;

        buffer[n] = '\0';

        if (buffer[n-1] == '\n') {
            buffer[n-1] = '\0';
        }

        if (strcmp(buffer, "exit") == 0) {
            break;
        }

        if (strcmp(buffer, "0") == 0) {
            dlclose(lib);
            current_lib = (current_lib == 1) ? 2 : 1;

            if (current_lib == 1) {
                lib = dlopen("./libvar1.so", RTLD_NOW);
            } else {
                lib = dlopen("./libvar2.so", RTLD_NOW);
            }
```

```
            if (!lib) {
                const char msg[] = "Ошибка загрузки
библиотеки\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return 1;
            }

            e = (e_func *)dlsym(lib, "e");
            area = (area_func *)dlsym(lib, "area");

            if (!e || !area) {
                const char msg[] = "Ошибка загрузки функций\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                dlclose(lib);
                return 1;
            }

            char output[256];
            int len = snprintf(output, sizeof(output),
"Переключение на библиотеку %d\n", current_lib);
            write(STDOUT_FILENO, output, len);

        } else if (buffer[0] == '1') {
            int x;
            if (sscanf(buffer + 1, "%d", &x) == 1) {
                if (x > 0) {
                    float result = e(x);
                    char output[256];
                    int len = snprintf(output, sizeof(output),
"e(%d) = %f\n", x, result);
                    write(STDOUT_FILENO, output, len);
                } else {
```

```
                    const char msg[] = "Ошибка. x должен быть
положительным\n";

                    write(STDERR_FILENO, msg, sizeof(msg));

                }

            } else {

                const char msg[] = "Неверный формат команды.
Используйте: 1 x\n";

                write(STDERR_FILENO, msg, sizeof(msg));

            }

        } else if (buffer[0] == '2') {

            float a, b;

            if (sscanf(buffer + 1, "%f %f", &a, &b) == 2) {

                if (a > 0 && b > 0) {

                    float result = area(a, b);

                    char output[256];

                    int len = snprintf(output, sizeof(output),
"area(%.2f, %.2f) = %.2f\n", a, b, result);

                    write(STDOUT_FILENO, output, len);

                } else {

                    const char msg[] = "Ошибка. a и b должны
быть положительными\n";

                    write(STDERR_FILENO, msg, sizeof(msg));

                }

            } else {

                const char msg[] = "Неверный формат команды.
Используйте: 2 a b\n";

                write(STDERR_FILENO, msg, sizeof(msg));

            }

        } else {

            const char msg[] = "Неизвестная команда\n";

            write(STDERR_FILENO, msg, sizeof(msg));

        }

    }
```

```
    if (lib) {
        dlclose(lib);
    }
    return 0;
}
```

var1.c:

```
#include <math.h>


//(1 + 1/x)^x
float e(int x) {
    return powf(1.0 + (1.0 / (float)x), (float)x);
}


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

var2.c:

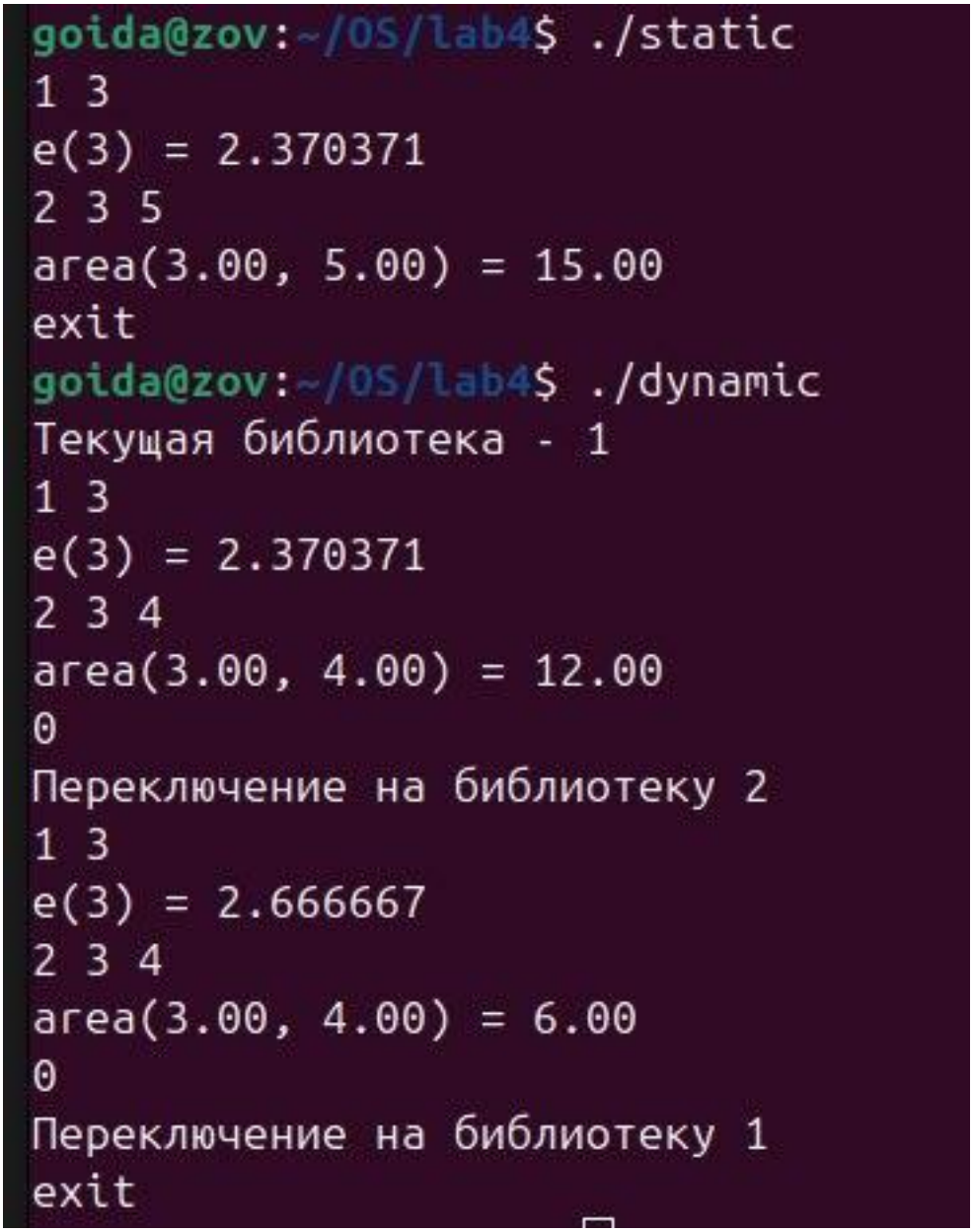
```
//Сумма ряда 1/n!
float e(int x) {
    float result = 0.0;
    float factorial = 1.0;

    for (int n = 0; n <= x; n++) {
        if (n > 0) {
            factorial *= n;
        }
```

```
        result += 1.0 / factorial;

    }

    return result;

}



//Площадь прямоугольного треугольника

float area(float a, float b) {

    return 0.5 * a * b;

}
```

func.h:

```
#ifndef FUNC_H

#define FUNC_H



float e(int x);



float area(float a, float b);



#endif
```

Протокол работы программы



Вывод

В процессе выполнения лабораторной работы я составил программу, демонстрирующую использование статических и динамических библиотек. Я приобрел базовые навыки создания библиотек и их правильного использования в коде. Одной из основных сложностей была в импорте динамических библиотек.

Таким образом, можно сделать выводы: статическая компоновка хоть и может быть автономнее динамической, но явно уступает в расходе памяти и размере. Также нужно учитывать сложность обновления при статической компоновке. Динамическая компоновка имеет больше преимуществ, по сравнению со статической.