

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-210БВ-24

Студент: Телепнева А.В.

Преподаватель: Бахарев В.Д.

Оценка: \_\_\_\_\_

Дата: 30.11.25

Москва, 2025

# Постановка задачи

## Вариант 10.

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

- Во время компиляции (на этапе линковки/linking)
- 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 использует одну из библиотек, используя информацию, полученную на этапе компиляции.
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты.

### Контракты и реализации функций:

1. Расчет производной функции  $\cos(x)$  в точке  $a$  с приращением  $dx$ :

Сигнатура функции: `float cos_derivative(float a, float dx);`

- Реализация №1:  $f'(x) = (f(a + dx) - f(a)) / dx$
- Реализация №2:  $f'(x) = (f(a + dx) - f(a - dx)) / (2dx)$

2. Подсчет наибольшего общего делителя для двух натуральных чисел:

Сигнатура функции: `int gcd(int a, int b);`

- Реализация №1: Алгоритм Евклида
- Реализация №2: Наивный алгоритм: пытаться разделить числа на все числа, что меньше  $a$  и  $b$

## Общий метод и алгоритм решения

### Использованные системные вызовы:

- `dlopen` - загружает динамическую библиотеку в адресное пространство процесса во время выполнения программы. Используется для реализации динамической подгрузки библиотек.
- `dlsym` - получает адрес функции по её имени из загруженной динамической библиотеки. Позволяет вызывать функции, не зная их адресов на этапе компиляции.
- `dlclose` - освобождает ресурсы и выгружает динамическую библиотеку из памяти процесса.
- `dlerror` - используется для получения диагностической информации в случае ошибки загрузки библиотеки или поиска символа.

## Описание работы программы:

### Программа №1 (использование библиотеки при линковке)

Данная программа использует динамическую библиотеку, подключённую на этапе компиляции.

Адреса функций известны заранее, и реализация не может быть изменена во время выполнения.

#### Алгоритм работы программы:

1. Программа запускается и ожидает ввод команд пользователя.
2. В зависимости от введённой команды:
  - команда 1 вызывает функцию численного дифференцирования  $\cos(x)$ ;
  - команда 2 вызывает функцию вычисления наибольшего общего делителя.
3. После выполнения функции результат выводится на экран.
4. Программа продолжает работу до завершения ввода.

### Программа №2 (динамическая загрузка библиотек)

Во второй программе динамические библиотеки загружаются во время выполнения с использованием функций `dlopen` и `dlsym`.

#### Алгоритм работы программы:

1. При запуске загружается первая динамическая библиотека.
2. Пользователь вводит команды:
  - 1 — вызов первой функции контракта;
  - 2 — вызов второй функции контракта;
  - 0 — переключение между реализациями (выгрузка текущей библиотеки и загрузка другой).
3. После каждой операции результат выполнения функции выводится на экран.
4. При переключении реализаций программа продолжает работу без перезапуска.

Использование динамической загрузки библиотек позволяет гибко управлять функциональностью программы и изменять поведение приложения во время выполнения.

## Код программы

### dynamic\_test.c

```
#include <stdio.h>

#include <dlfcn.h>

typedef float (*cos_deriv_t)(float, float);

typedef int (*gcd_t)(int, int);

int main() {

    void *handle = dlopen("./libimpl1.so", RTLD_LAZY);

    if (!handle) {

        printf("dlopen error\n");

        return 1;

    }

    cos_deriv_t cos_derivative = dlsym(handle, "cos_derivative");

    gcd_t gcd = dlsym(handle, "gcd");

    int cmd;

    while (scanf("%d", &cmd) == 1) {

        if (cmd == 0) {

            dlclose(handle);

            static int toggle = 0;

            toggle = !toggle;
```

```

        handle = dlopen(toggle ? "./libimpl2.so" : "./libimpl1.so",
RTLD_LAZY);

        cos_derivative = dlsym(handle, "cos_derivative");

        gcd = dlsym(handle, "gcd");

        printf("Library switched\n");

    } else if (cmd == 1) {

        float a, dx;

        scanf("%f %f", &a, &dx);

        printf("Result: %f\n", cos_derivative(a, dx));

    } else if (cmd == 2) {

        int a, b;

        scanf("%d %d", &a, &b);

        printf("Result: %d\n", gcd(a, b));

    }

}

dlclose(handle);

return 0;

}

```

### impl1.c

```

#include <math.h>

#include "include/contracts.h"

float cos_derivative(float a, float dx) {

    return (cos(a + dx) - cos(a)) / dx;

}

int gcd(int a, int b) {

```

```
while (b != 0) {  
  
    int t = b;  
  
    b = a % b;  
  
    a = t;  
  
}  
  
return a;  
  
}
```

### impl2.c

```
#include <math.h>  
  
#include "include/contracts.h"  
  
float cos_derivative(float a, float dx) {  
  
    return (cos(a + dx) - cos(a - dx)) / (2 * dx);  
  
}  
  
int gcd(int a, int b) {  
  
    int res = 1;  
  
    int min = (a < b) ? a : b;  
  
    for (int i = 1; i <= min; i++) {  
  
        if (a % i == 0 && b % i == 0)  
  
            res = i;  
  
    }  
  
    return res;  
  
}
```

### static\_test.c

```
#include <stdio.h>  
  
#include "include/contracts.h"
```

```

int main() {

    int cmd;

    while (scanf("%d", &cmd) == 1) {

        if (cmd == 1) {

            float a, dx;

            scanf("%f %f", &a, &dx);

            printf("Result: %f\n", cos_derivative(a, dx));

        } else if (cmd == 2) {

            int a, b;

            scanf("%d %d", &a, &b);

            printf("Result: %d\n", gcd(a, b));

        }

    }

    return 0;

}

```

## Протокол работы программы

### Тестирование:

```

• merkuriiii@FordFocus2006:~/OS/4$ export LD_LIBRARY_PATH=./:$LD_LIBRARY_PATH
• merkuriiii@FordFocus2006:~/OS/4$ ./static_test
1 0 0.001
Result: -0.000500
2 48 18
Result: 6
• merkuriiii@FordFocus2006:~/OS/4$ ./dynamic_test
1 0 0.01
Result: -0.005000
0
Library switched
1 0 0.001
Result: 0.000000
2 81 27
Result: 27

```

## **Вывод**

В ходе лабораторной работы были изучены принципы создания и использования динамических библиотек в ОС Linux. Были реализованы два подхода к подключению библиотек, каждый из которых имеет свои преимущества и ограничения. На практике продемонстрировано использование системных вызовов для динамической загрузки и управления библиотеками.