Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу
«Операционные системы»

Студент: Скрипачев Фёдор Михайлович
Группа: М8О-209Б-23
Вариант: 16
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2025

Содержание

Репозиторий

https://github.com/gthcbr25/osi/tree/main/oslab4

Постановка задачи

Цель работы

Приобретение практических навыков в:

- Создание динамических библиотек
- Создание программ, которые используют функции динамических библиотек

Задание

Требуется создать динамические библиотеки, которые реализуют определенный 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 No1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
- Тестовая программа (программа No2), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ двух типов использования библиотек.

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 No2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
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

N	Описание	Сигнатура	Реализация 1	Реализация 2
3	Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные)	Int PrimeCount(int A, int B)	Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.	Решето Эратосфена
4	Подсчёт наибольшего общего делителя для двух натуральных чисел	Int GCF (int A, int B)	Алгоритм Евклида	Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.

Общие сведения о программе

Программа компилируется в двух файлах: static_main.c и dynamic_main.c

Используемые библиотечные вызовы:

void *dlopen(const char *filename, int flag);	Загружает динамическую библиотеку, имя которой указано в строке filename и возвращает прямой указатель на начало загруженной библиотеки.
const char *dlerror(void);	Возвращает указатель на начало строки, описывающей ошибку, полученную на предыдущем вызове.
void *dlsym(void *handle, char *symbol);	Получает параметр handle, который является выходом вызова dlopen и параметр symbol, который является строкой, в которой содержится название символа, который необходимо загрузить из библиотеки. Возвращает указатль на область памяти, в которой содержится необходимый символ.
int dlclose(void *handle);	Уменьшает счетчик ссылок на указатель handle и если он равен нулю, то освобождает библиотеку.

Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Изучить работу с библиотеками.
2. Реализовать две библиотеки согласно заданию.
3. Реализовать две программы (для работы с динамическими и статическими библиотеками).

Исходный код

Lib1.c

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

int PrimeCount(int A, int B) {
    int count = 0;
    for (int i = A; i <= B; i++) {
        if (i < 2) continue;
        bool is_prime = true;
        for (int j = 2; j * j <= i; j++) {
            if (i % j == 0) {
                is_prime = false;
                break;
            }
        }
        if (is_prime) count++;
    }
    return count;
}

int GCF(int a, int b) {
    int c;
    while (b) {
        c = a % b;
        a = b;
        b = c;
    }
```

```
    return abs(a);
}
```

Lib2.c

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

int PrimeCount(int A, int B) {
    if (B < 2) return 0;
    bool *is_prime = (bool *)malloc((B + 1) * sizeof(bool));
    for (int i = 0; i <= B; i++) is_prime[i] = true;
    is_prime[0] = is_prime[1] = false;

    for (int i = 2; i * i <= B; i++) {
        if (is_prime[i]) {
            for (int j = i * i; j <= B; j += i) {
                is_prime[j] = false;
            }
        }
    }
    int count = 0;
    for (int i=A; i<=B; ++i){
        if (is_prime[i]) count++;
    }
    free(is_prime);
    return count;
}

int GCF(int a, int b){
```

```
    int minimal = 0;
    if (abs(a) < abs(b)){
        minimal = abs(a);
    }else{
        minimal = abs(b);
    }
    for (int i = minimal; i > 0; i--){
        if (a % i == 0 && b % i == 0){
            return i;
        }
    }
    return 1;
}
```

Dynamic.c

```
// dynamic.c: Динамическая загрузка библиотек
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

typedef int (*PrimeCountFunc)(int, int);
typedef int (*GcfFunc)(int, int);

int main() {
    void *lib1_handle;
    void *lib2_handle;
    PrimeCountFunc PrimeCount;
    GcfFunc GCF;
    char *error;
```

```
lib1_handle = dlopen("./lib1.so", RTLD_LAZY);
lib2_handle = dlopen("./lib2.so", RTLD_LAZY);
if (!lib1_handle || !lib2_handle) {
    fprintf(stderr, "Error loading library: %s\n", dlerror());
    exit(1);
}

void *lib_handle;
int var;
scanf("%d", &var);
if (var == 1) lib_handle = lib1_handle;
else lib_handle = lib2_handle;

PrimeCount = (PrimeCountFunc)dlsym(lib_handle, "PrimeCount");
GCF = (GcfFunc)dlsym(lib_handle, "GCF");

if ((error = dlerror()) != NULL) {
    fprintf(stderr, "Error loading symbols: %s\n", error);
    exit(1);
}

int choice;
printf("1: Count primes\n2: GCF\nChoose option: ");
scanf("%d", &choice);
int A, B;
printf("Enter A and B: ");
scanf("%d %d", &A, &B);
if (choice == 1) {
    printf("Prime count: %d\n", PrimeCount(A, B));
```

```
    } else if (choice == 2) {
        printf("GCF: %d\n", GCF(A, B));
    }

    dlclose(lib_handle);
    return 0;
}
```

Static.c

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

extern int PrimeCount(int A, int B);
extern int GCF(int A, int B);

int main() {
    int choice;
    printf("1: Count primes\n2: GCF\nChoose option: ");
    scanf("%d", &choice);

    int A, B;
    printf("Enter A and B: ");
    scanf("%d %d", &A, &B);
    if (choice == 1) {
        printf("Prime count: %d\n", PrimeCount(A, B));
    } else if (choice == 2) {
        printf("GCF: %d\n", GCF(A, B));
    }
    return 0;
}
```

Демонстрация работы программы

./static
1: Count primes
2: GCF Choose option:
2
Enter A and B: 15 5
GCF: 5
./dynamic
1: Count primes
2: GCF Choose option:
2
Enter A and B: 15 5
GCF: 5

Выводы

В лабораторной работе я изучил создание динамических библиотек в Linux и их загрузку во время выполнения программы. Такие библиотеки сокращают размер исполняемых файлов и упрощают компиляцию. Библиотеку можно подключить на этапе линковки: программа заранее узнает расположение функций, но загрузка всё равно произойдет при запуске. Если библиотека находится вне стандартных директорий, путь к ней нужно указать линкеру. Использование библиотек позволяет повторно применять готовые структуры и функции, упрощая разработку сложных проектов.