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: Кудряшова Т.И.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 02.10.25

Москва, 2025

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:

● 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.

Вариант 15

Первая функция:

Расчет производной функции cos(x) в точке a с приращением dx:

Вторая функция:

Отсортировать целочисленный массив:

Общий метод и алгоритм решения.

Немного теории:

Быстрая сортировка Хаора: **Разделяй и властвуй!!!**

Алгоритм:

1) Выбираем опорный элемент(pivot)
2) Разбиваем массив на 2 части: слева все, что меньше, справа все что больше опорного элемента
3) Рекурсивно сортируем левую и правую части как показано выше

Системные вызовы:

- dlopen(path, RTLD_LAZY) – загружает динамическую библиотеку .so в память процесса во время выполнения.
- dlsym(handle, "symbol") – получает адрес функции или переменной внутри ранее загруженной библиотеки.
- dlclose(handle) – выгружает динамическую библиотеку из процесса и освобождает связанные ресурсы.
- dlerror() – возвращает текст последней ошибки механизма динамической загрузки библиотек.
- malloc(sizeof(type) * n) – выделяет динамическую память в куче размером n элементов указанного типа.
- free(ptr) – освобождает ранее выделенную динамическую память по указателю ptr.
- cosf(x) – вычисляет значение косинуса от аргумента x (используется при вычислении производной)
- open(path, flags) – открывает ELF-файл динамической библиотеки при работе dlopen.
- read(fd, buffer, size) – читает данные из файла библиотеки при её загрузке.
- mmap(addr, size, …) – отображает секции динамической библиотеки в адресное пространство процесса.
- close(fd) – закрывает файловый дескриптор библиотеки после загрузки.
- write(fd, buffer, size) – выполняет фактический вывод на экран внутри printf

Код программы.

Funcs.h

```h
#ifndef FUNCS_H
#define FUNCS_H

#include <stddef.h>

#ifdef __cplusplus
extern "C" {
#endif

float cos_derivative(float a, float dx);
int* sort(int* array, size_t n);

#ifdef __cplusplus
}
#endif

#endif
```

Impl1.c

```c
#include "funcs.h"
#include <math.h>

// f'(x) = (f(a + dx) - f(a)) / dx
float cos_derivative(float a, float dx) {
    return (cosf(a + dx) - cosf(a)) / dx;
}

// Пузырьковая сортировка
int* sort(int* array, size_t n) {
    for (size_t i = 0; i < n; i++) {
        for (size_t j = 0; j < n - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
            }
        }
    }
    return array;
}
```

Impl2.c

```c
#include "funcs.h"
#include <math.h>

// f'(x) = (f(a + dx) - f(a - dx)) / (2dx)
float cos_derivative(float a, float dx) {
    return (cosf(a + dx) - cosf(a - dx)) / (2 * dx);
}

// Быстрая сортировка Хоара
```

```c
static void quicksort(int* arr, int left, int right) {
    if (left >= right) return;

    int pivot = arr[(left + right) / 2];
    int i = left, j = right;

    while (i <= j) {
        while (arr[i] < pivot) i++;
        while (arr[j] > pivot) j--;

        if (i <= j) {
            int tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            i++;
            j--;
        }
    }

    if (left < j) quicksort(arr, left, j);
    if (i < right) quicksort(arr, i, right);
}

int* sort(int* array, size_t n) {
    quicksort(array, 0, (int)n - 1);
    return array;
}
```

Prog1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "funcs.h"

void print_menu() {
    printf("ДОСТУПНЫЕ КОМАНДЫ\n");
    printf("1 a dx\n");
    printf("    Вычислить производную cos(x) в точке a с шагом dx\n");
    printf("    Пример: 1 3.14 0.001\n");
    printf("2 n arr...\n");
    printf("    Отсортировать массив из n целых чисел\n");
    printf("    Пример: 2 5 3 9 1 6 2\n\n");
    printf("любая другая команда — выход\n");
}

int main() {
    print_menu();

    while (1) {
        int command;
        printf("Введите команду: ");

        if (scanf("%d", &command) != 1) {
            printf("Ошибка: ожидалось число команды.\n");
            scanf("%*s");
            continue;
        }
        if (command == 1) {
            float a, dx;
            if (scanf("%f %f", &a, &dx) != 2) {
                printf("Ошибка: формат команды '1 a dx'\n");
                scanf("%*s");
```

```
                continue;
            }
            printf("Результат: cos'(a) = %f\n", cos_derivative(a, dx));
        }
        else if (command == 2) {
            size_t n;
            if (scanf("%zu", &n) != 1) {
                printf("Ошибка: ожидалось число n.\n");
                scanf("%*s");
                continue;
            }
            int* arr = malloc(n * sizeof(int));
            if (!arr) {
                printf("Ошибка: недостаточно памяти.\n");
                continue;
            }
            for (size_t i = 0; i < n; i++) {
                if (scanf("%d", &arr[i]) != 1) {
                    printf("Ошибка: ожидалось целое число массива.\n");
                    free(arr);
                    scanf("%*s");
                    continue;
                }
            }
            sort(arr, n);
            printf("Отсортированный массив: ");
            for (size_t i = 0; i < n; i++) printf("%d ", arr[i]);
            printf("\n");
            free(arr);
        }
        else {
            printf("Выход.\n");
            break;
        }
    }
    return 0;
}
```

Prog2.c

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

typedef float (*cos_d_t)(float, float);
typedef int*  (*sort_t)(int*, size_t);

void* lib = NULL;
cos_d_t cos_derivative_ptr = NULL;
sort_t  sort_ptr = NULL;

void print_menu() {
    printf("ДОСТУПНЫЕ КОМАНДЫ\n");
    printf("0 - Переключить реализацию (impl1 <-> impl2)\n");
    printf("1 a dx\n");
    printf("    Вычислить производную cos(x)\n");
    printf("    Пример: 1 0 0.01\n");
    printf("2 n arr...\n");
    printf("    Отсортировать массив\n");
    printf("    Пример: 2 4 9 2 3 1\n\n");
```

```
    printf("любая другая команда — выход\n");
}

void load_lib(const char* path) {
    if (lib) dlclose(lib);
    lib = dlopen(path, RTLD_LAZY);
    if (!lib) {
        printf("Ошибка загрузки библиотеки: %s\n", dlerror());
        exit(1);
    }
    cos_derivative_ptr = (cos_d_t)dlsym(lib, "cos_derivative");
    sort_ptr           = (sort_t)dlsym(lib, "sort");
    printf("Используется библиотека: %s\n", path);
}

int main() {
    print_menu();
    load_lib("./libimpl1.so");
    while (1) {
        int cmd;
        printf("Введите команду: ");
        if (scanf("%d", &cmd) != 1) {
            printf("Ошибка: ожидалось число.\n");
            scanf("%*s");
            continue;
        }
        if (cmd == 0) {
            static int toggle = 0;
            toggle = !toggle;
            if (toggle) load_lib("./libimpl2.so");
            else        load_lib("./libimpl1.so");
            printf("Библиотека переключена.\n");
        }
        else if (cmd == 1) {
            float a, dx;
            if (scanf("%f %f", &a, &dx) != 2) {
                printf("Ошибка: формат команды '1 a dx'\n");
                scanf("%*s");
                continue;
            }
            printf("Результат: cos'(a) = %f\n", cos_derivative_ptr(a, dx));
        }
        else if (cmd == 2) {
            size_t n;
            if (scanf("%zu", &n) != 1) {
                printf("Ошибка: ожидалось число n.\n");
                scanf("%*s");
                continue;
            }
            int* arr = malloc(n * sizeof(int));
            if (!arr) {
                printf("Ошибка: недостаточно памяти.\n");
                continue;
            }
            for (size_t i = 0; i < n; i++) {
                if (scanf("%d", &arr[i]) != 1) {
                    printf("Ошибка: ожидалось число массива.\n");
                    free(arr);
                    scanf("%*s");
                    continue;
                }
            }
            sort_ptr(arr, n);
```

```
            printf("Отсортированный массив: ");
            for (size_t i = 0; i < n; i++) printf("%d ", arr[i]);
            printf("\n");

            free(arr);
        }
        else {
            printf("Выход.\n");
            break;
        }
    }
    if (lib) dlclose(lib);
    return 0;
}
```

Протокол работы программы.

Сборка:

gcc -fPIC -shared impl1.c -o libimpl1.so –lm

gcc -fPIC -shared impl2.c -o libimpl2.so –lm

gcc prog1.c -L. -limpl1 -o prog1_1 –lm // для 1 версии

gcc prog1.c -L. –limpl2 -o prog1_2 –lm // для 2 версии

gcc prog2.c -ldl -o prog2

Запуск и тестирование программа 1:

./prog1_ //аналогично для 2разница лишь в скоростях сортировки и точности производной

ДОСТУПНЫЕ КОМАНДЫ

1 a dx

Вычислить производную cos(x) в точке a с шагом dx

Пример: 1 3.14 0.001

2 n arr...

Отсортировать массив из n целых чисел

Пример: 2 5 3 9 1 6 2

любая другая команда — выход

Введите команду: n

Ошибка: ожидалось число команды.

Введите команду: 1 3 4

Результат: cos'(a) = 0.435974

Введите команду: 2 2 22 22

Отсортированный массив: 22 22

Введите команду: 0

Выход.

Программа 2:

./prog2

====== ДОСТУПНЫЕ КОМАНДЫ ======

0 - Переключить реализацию (impl1 <-> impl2)

1 a dx

   Вычислить производную cos(x)

   Пример: 1 0 0.01

2 n arr...

   Отсортировать массив

   Пример: 2 4 9 2 3 1

любая другая команда — выход

Используется библиотека: ./libimpl1.so

Введите команду: 0

Используется библиотека: ./libimpl2.so

Библиотека переключена.

Введите команду: 1 0 0.1

Результат: cos'(a) = 0.000000

Введите команду: 2 4 65 54 43 32

Отсортированный массив: 32 43 54 65

Введите команду: 3

Выход.

Вывод.

Сама возможность создания динамических библиотек да и библиотек в принципе интересна. Прикольный опыт.