

**Московский авиационный институт (национальный
исследовательский университет)**

Институт информационных технологий и прикладной математики
«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету «Операционные
системы» №4**

Студент: Брюханов З. Д.
Преподаватель: Миронов Е. С.
Группа: М8О-207Б-22
Дата: 10.11.2023
Оценка:
Подпись:

Оглавление

Цель работы	3
Постановка задачи	3
Общий алгоритм решения	4
Реализация	4
Пример работы	9
Вывод	11

Цель работы

Целью является приобретение практических навыков в:

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

Постановка задачи

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 №1), которая использует одну из библиотек, используя знания полученные на этапе компиляции;
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод для обеих программ должен быть организован следующим образом:

1. 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»;
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.

Вариант 15:

- Расчет производной функции $\cos(x)$ в точке A с приращением δx ;
- Отсортировать целочисленный массив.

Общий алгоритм решения

В данной лабораторной есть два 3 исполняемых файла - 2 с различными реализациями статических библиотек и 1 с динамическими библиотеками. Статическая библиотека подключается к программе на этапе компиляции и мы работаем с ней почти также, как с любой другой библиотекой. При использовании динамической библиотеки я использую средства языка, чтобы получить функции из нее. У меня имеется общий заголовочный файл и 2 файла с библиотеками, в которых написаны разные реализации двух функций. В программе, использующей динамические библиотеки, есть переменная, регулирующая какая библиотека должна быть подключена в текущий момент. По команде пользователя мы меняем динамические библиотеки.

Реализация

lib.h

```
#pragma once
#include <stdio.h>

float derivative(float point, float delta_x);

void sort(int* array, int size);
```

lib_first.c

```
#include "lib.h"
#include <math.h>

float derivative(float point, float delta_x) {
    return (cosf(point + delta_x) - cosf(point)) / delta_x;
}

void sort(int* array, int size) {
    for (int first_index = 0; first_index < size - 1; ++first_index) {
        for (int second_index = 0; second_index < size - first_index - 1; ++second_index) {
            if (array[second_index] > array[second_index + 1]) {
                int temporary = array[second_index];
                array[second_index] = array[second_index + 1];
```

```

        array[second_index + 1] = temporary;
    }
}
}

```

lib_second.c

```

#include "lib.h"
#include <math.h>

float derivative(float point, float delta_x) {
    return (cosf(point + delta_x) - cosf(point - delta_x)) / (2 * delta_x);
}

void sort(int* array, int size) {
    if (size < 2) return;

    int pivot = array[size / 2];
    int left_pointer, right_pointer;

    for (left_pointer = 0, right_pointer = size - 1;; ++left_pointer, --right_pointer)
    {
        while (array[left_pointer] < pivot) {
            left_pointer++;
        }

        while (array[right_pointer] > pivot) {
            right_pointer--;
        }

        if (left_pointer >= right_pointer) {
            break;
        }

        int temporary = array[left_pointer];
        array[left_pointer] = array[right_pointer];
        array[right_pointer] = temporary;
    }
    sort(array, left_pointer);
    sort(array + left_pointer, size - left_pointer);
}

```

static_lib.c

```
#include "lib.h"
#include <stdio.h>

int main() {
    int command;

    while (scanf("%d", &command) != EOF) {

        if (command == 1) {

            float point, delta_x;
            printf("Enter 2 numbers (point and delta_x): ");
            scanf("%f %f", &point, &delta_x);
            printf("The result of the first function: %f\n", derivative(point, delta_x));

        } else if (command == 2) {

            int size;
            printf("Enter the size of the array: ");
            scanf("%d", &size);

            int array[size];
            printf("Enter the array elements:\n");
            for (int index_array = 0; index_array < size; ++index_array) {
                printf("Element %d: ", index_array + 1);
                scanf("%d", &array[index_array]);
            }

            sort(array, size);

            printf("The result of the second function:\n");
            for (int index_array = 0; index_array < size; ++index_array) {
                printf("%d ", array[index_array]);
            }
            printf("\n");

        } else if (command == -1) {

            printf("The program is completed.\n");
            break;

        } else {
```

```

        printf("You entered the wrong command.\n");
    }
}

return 0;
}

```

dynamic_lib.c

```

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

const char* FIRST_LIBRARY_PATH = "trash/liblib_first.so";
const char* SECOND_LIBRARY_PATH = "trash/liblib_second.so";
void* library_descriptor = NULL;
int current_library = 1;

float (* derivative_func)(float point, float delta_x) = NULL;

void (* sort_func)(int* array, int size) = NULL;

void error_processing(bool exception, char* bug_report) {
    if (exception) {
        fprintf(stderr, "%s", bug_report);
        exit(-1);
    }
}

void open_library(const char* path_to_library) {
    if (library_descriptor != NULL) {
        dlclose(library_descriptor);
    }

    library_descriptor = dlopen(path_to_library, RTLD_LAZY);
    error_processing(library_descriptor == NULL, "Library opening error.\n");

    derivative_func = dlsym(library_descriptor, "derivative");
    error_processing(derivative_func == NULL, "Error in finding the method
derivative.\n");

    sort_func = dlsym(library_descriptor, "sort");
}

```

```

    error_processing(sort_func == NULL, "Error in finding the method sort.\n");
}

void swap_library() {
    if (current_library == 1) {

        open_library(SECOND_LIBRARY_PATH);
        current_library = 2;

    } else {

        open_library(FIRST_LIBRARY_PATH);
        current_library = 1;

    }

    printf("Current library - %d\n", current_library);
}

int main() {
    open_library(FIRST_LIBRARY_PATH);
    int command;

    while (scanf("%d", &command) != EOF) {
        if (command == 0) {

            swap_library();

        } else if (command == 1) {

            float point, delta_x;
            printf("Enter 2 numbers (point and delta_x): ");
            scanf("%f %f", &point, &delta_x);
            printf("The result of the first function: %f\n", (*derivative_func)(point,
delta_x));

        } else if (command == 2) {

            int size;
            printf("Enter the size of the array: ");
            scanf("%d", &size);

            int array[size];
            printf("Enter the array elements:\n");
            for (int index_array = 0; index_array < size; ++index_array) {

```



```

        printf("Element %d: ", index_array + 1);
        scanf("%d", &array[index_array]);
    }

    (*sort_func)(array, size);

    printf("The result of the second function:\n");
    for (int index_array = 0; index_array < size; ++index_array) {
        printf("%d ", array[index_array]);
    }
    printf("\n");

} else if (command == -1) {

    printf("The program is completed.\n");
    break;

} else {

    printf("You entered the wrong command.\n");

}

}

dlclose(library_descriptor);
return 0;
}

```

Пример работы

Test 1.

```

br_zahar@mbp-zahar laba04 % ./static_lib_first
1
Enter 2 numbers (point and delta_x): 1.12 0.23
The result of the first function: -0.942069
2
Enter the size of the array: 10
Enter the array elements:
Element 1: 4
Element 2: 3
Element 3: 2
Element 4: 1

```

Element 5: 9
Element 6: -1
Element 7: 23
Element 8: 43
Element 9: -123
Element 10: 0
The result of the second function:
-123 -1 0 1 2 3 4 9 23 43
-1
The program is completed.

Test 2.

```
br_zahar@mbp-zahar laba04 % ./dynamic_lib
1
Enter 2 numbers (point and delta_x): 10000 0.2
The result of the first function: 0.398957
0
Current library - 2
1
Enter 2 numbers (point and delta_x): 10000 0.2
The result of the first function: 0.303874
2
Enter the size of the array: 5
Enter the array elements:
Element 1: 9
Element 2: 23414
Element 3: -1234
Element 4: 0
Element 5: 9
The result of the second function:
-1234 0 9 9 23414
-1
The program is completed.
```

Вывод

В ходе данной работы я познакомился со статическими и динамическими библиотеками. Их основное различие в компиляции: у статической библиотеки код полностью входит в исполняемый файл при компиляции. Благодаря этому ими проще делиться. В случае динамической библиотеки, код программы содержит только ссылку на библиотеку. Динамической библиотекой сложнее делиться зато проще вносить изменения: не придется перекомпилировать исходные файлы программы, потому что библиотека передается по ссылке и все изменения уже будут доступны в момент выполнения программы. Для меня наиболее сложным было не написание данных библиотек, а их компиляция в `makefile`.