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

Лабораторная работа № 5 по курсу

«Операционные системы»

Студент: Постнов Александр Вячеславович
Группа: М8О-201Б-21
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/kappaprideonly/mai-os-labs

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 используют 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 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 №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.

1	Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e	Float SinIntegral(float A, float B, float e)	Подсчет интеграла методом прямоугольников.	Подсчет интеграла методом трапеций.
6	Рассчет значения числа e(основание натурального логарифма)	Float E(int x)	(1 + 1/x) ^ x	Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!))

Общие сведения о программе

CMakeLists.txt - описание процесса сборки проекта

main_dynamic.cpp - main, который подгружает библиотеку во время runtime

main_static.cpp - main, который подключает библиотеку во время компиляции

signature.h - объявления функций

implem1.cpp - 1 реализация функций

implem2.cpp - 2 реализация функций

lab5_test.cpp - тесты

Общий метод и алгоритм решения

Для статической реализации необходимо скомпилировать исполняемый файл с библиотекой. Для динамической реализации, небходимо использовать системный вызов dlopen, чтобы подгрузить библиотеку во время runtime.

Исходный код

CMakeLists.txt

```
# Generate shared / static librarry for hello.cpp
# STATIC generates `libhello.a`
# SHARED geneartes `libhello.so`

add_library(d1_static STATIC src/implem1.cpp include/signature.h)
add_library(d2_static STATIC src/implem2.cpp include/signature.h)
add_library(d1_dynamic SHARED src/implem1.cpp include/signature.h)
add_library(d2_dynamic SHARED src/implem2.cpp include/signature.h)

target_include_directories(d1_static PRIVATE include)
target_include_directories(d2_static PRIVATE include)
target_include_directories(d1_dynamic PRIVATE include)
target_include_directories(d2_dynamic PRIVATE include)


# Genarete `main` executable from `main.cpp`
add_executable(main_static_1 main_static.cpp)
add_executable(main_static_2 main_static.cpp)
add_executable(main_dynamic main_dynamic.cpp)

target_include_directories(main_static_1 PRIVATE include)
target_include_directories(main_static_2 PRIVATE include)
target_include_directories(main_dynamic PRIVATE include)

# Link `main` program with library
target_link_libraries(main_static_1 PRIVATE d1_static)
target_link_libraries(main_static_2 PRIVATE d2_static)
```

signature.h

```
#ifndef SIGNATURE_H
#define SIGNATURE_H

#include <cstddef>
#include <algorithm>
#include <cmath>

extern "C" {
    float E(int x);
    float SinIntegral(float a, float b, float e);
}

#endif /* SIGNATURE */
```

implem1.cpp

```
#include "signature.h"

float E(int x) {
    if (x < 0) {
        return -1;
    }
    float e = 1.0;
    for (int i = 0; i < x; ++i) {
        e *= 1 + 1 / static_cast<float>(x);
    }
    return e;
}

float SinIntegral(float a, float b, float e) {
    if (a > b || e > (b - a)) {
        return -1;
```

```
    }
    float h = e;
    float w = 0;
    for (float i = a; i <= b; i += e) {
        w += std::sin(i);
    }
    return h * w;
}
```

implem2.cpp

```
#include "signature.h"
#include <iostream>

float E(int x) {
    if (x < 0) {
        return -1;
    }
    float e = 1.0;
    float term = 1.0;
    for (int i = 1; i <= x; ++i) {
        term /= i;
        e += term;
    }
    return e;
}

float SinIntegral(float a, float b, float e) {
    float value = 0;
    for (float i = a; i <= b - e; i += e) {
        value += (std::sin(i) + std::sin(i + e)) / 2;
    }
    return value * e;
}
```

main_dynamic.cpp

```
#include <array>
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <dlfcn.h>
#include <string>
#include <vector>
```

```
int main() {
    const std::vector <std::string> LIB = {"./libd1_dynamic.so",
"./libd2_dynamic.so"};
    const std::vector <std::string> FUNC_NAME = {"E", "SinIntegral"};
    int curlib = 0;
    float (*e)(int x);
    float (*sinIntegral)(float a, float b, float e);
    void* handle = dlopen(LIB[curlib].c_str(), RTLD_LAZY);
    if (handle == nullptr) {
        std::cout << "Fail dlopen\n";
        return EXIT_FAILURE;
    }
    e = (float(*)(int))dlsym(handle, FUNC_NAME[0].c_str());
    sinIntegral = (float(*)(float, float, float))dlsym(handle,
FUNC_NAME[1].c_str());
    int command;
    while (std::cin >> command) {
        if (command == 1) {
            int num;
            std::cin >> num;
            float value = e(num);
            if (value == -1) {
                std::cout << "incorrect value of variable" << std::endl;
                continue;
            }
            std::cout << value << std::endl;
        } else if (command == 2) {
            float a;
            float b;
            float e;
            std::cin >> a >> b >> e;
            float value = sinIntegral(a, b, e);
            if (value == -1) {
                std::cout << "incorrect values of variables" << std::endl;
                continue;
            }
            std::cout << value << std::endl;
        } else if (command == 0) {
            dlclose(handle);
            curlib ^= 1;
            void* handle = dlopen(LIB[curlib].c_str(), RTLD_LAZY);
            if (handle == nullptr) {
                std::cout << "Fail dlopen\n";
                return EXIT_FAILURE;
```

```
            }
            e = (float(*)(int))dlsym(handle, FUNC_NAME[0].c_str());
            sinIntegral = (float(*)(float, float, float))dlsym(handle,
FUNC_NAME[1].c_str());
        } else {
            std::cout << "you have to enter 1 or 2" << std::endl;
        }
    }
    dlclose(handle);
}
```

main_static.cpp

```
#include <iostream>

#include "signature.h"

int main() {
    int command;
    while (std::cin >> command) {
        if (command == 1) {
            int num;
            std::cin >> num;
            float value = E(num);
            if (value == -1) {
                std::cout << "incorrect value of variable" << std::endl;
                continue;
            }
            std::cout << value << std::endl;
        } else if (command == 2) {
            float a;
            float b;
            float e;
            std::cin >> a >> b >> e;
            float value = SinIntegral(a, b, e);
            if (value == -1) {
                std::cout << "incorrect values of variables" << std::endl;
                continue;
            }
            std::cout << value << std::endl;
        } else {
            std::cout << "you have to enter 1 or 2" << std::endl;
        }
    }
}
```

lab5_test.cpp

```
#include <cstddef>
#include <cstdio>
#include <cstdlib>
#include <fstream>
#include <gtest/gtest.h>
#include <string>
#include <dlfcn.h>


#include "signature.h"


TEST(Lab5Test, DynamicTest) {
    const std::vector <std::string> FUNC_NAME = {"E", "SinIntegral"};
    const float expectE = 2.71;
    const auto inputParameterE = {500, 300, 200};

    const std::vector <std::vector <float>> inputParameterSin = {
        {1, 5, 0.001},
        {5, 6, 0.001},
        {1, 100, 0.001}
    };
    const std::vector <float> answerSin = {0.2566, -0.676, -0.322};

    float (*eOne)(int x);
    float (*sinIntegralOne)(float a, float b, float e);
    void* handleOne = dlopen(getenv("PATH_TO_LIB1"), RTLD_LAZY);
    std::cout << getenv("PATH_TO_LIB1");
    ASSERT_NE(handleOne, nullptr);


    float (*eTwo)(int x);
    float (*sinIntegralTwo)(float a, float b, float e);
    void* handleTwo = dlopen(getenv("PATH_TO_LIB2"), RTLD_LAZY);
    ASSERT_NE(handleTwo, nullptr);

    eOne = (float(*)(int))dlsym(handleOne, FUNC_NAME[0].c_str());
    eTwo = (float(*)(int))dlsym(handleTwo, FUNC_NAME[0].c_str());
    sinIntegralOne = (float(*)(float, float, float))dlsym(handleOne,
FUNC_NAME[1].c_str());
```

```
    sinIntegralTwo = (float(*)(float, float, float))dlsym(handleTwo,
FUNC_NAME[1].c_str());
    for (const auto& elem : inputParameterE) {
        auto eOutOne = eOne(elem);
        auto eOutTwo = eTwo(elem);
        EXPECT_NEAR(eOutOne, expectE, 0.1);
        EXPECT_NEAR(eOutTwo, expectE, 0.1);
    }

    for (size_t i = 0; i < inputParameterSin.size(); ++i) {
        float a = inputParameterSin[i][0];
        float b = inputParameterSin[i][1];
        float e = inputParameterSin[i][2];
        auto sqOne = sinIntegralOne(a, b, e);
        auto sqTwo = sinIntegralTwo(a, b, e);
        EXPECT_NEAR(sqOne, answerSin[i], 0.1);
        EXPECT_NEAR(sqTwo, answerSin[i], 0.1);
    }
}

TEST(Lab5Test, StaticOneTest) {
    const float expectE = 2.71;
    const auto inputParameterE = {500, 300, 200};

    const std::vector <std::vector <float>> inputParameterSin = {
        {1, 5, 0.001},
        {5, 6, 0.001},
        {1, 100, 0.001}
    };
    const std::vector <float> answerSin = {0.2566, -0.676, -0.322};

    for (const auto& elem : inputParameterE) {
        auto eOut = E(elem);
        EXPECT_NEAR(eOut, expectE, 0.1);
    }

    for (size_t i = 0; i < inputParameterSin.size(); ++i) {
        float a = inputParameterSin[i][0];
        float b = inputParameterSin[i][1];
        float e = inputParameterSin[i][2];
        auto sq = SinIntegral(a, b, e);
        EXPECT_NEAR(sq, answerSin[i], 0.1);
    }
}
```

Демонстрация работы программы

```
➜  lab5 git:(main) ✗ ./main_dynamic
1 100
2.70481
3
you have to enter 1 or 2
0
1 100
2.71828
2 1 100 0.1
-0.321262
0
2 1 100 0.1
-0.304547
➜  lab5 git:(main) ✗ ./main_static_1
```

1 100

2.70481

2 1 100 0.1

-0.304547

➜ lab5 git:(main) ✗ ./main_static_2

1 100

2.71828

2 1 100 0.1

-0.321262

Выводы:

Понял различие динамических и статических библиотек. Создал динамическую библиотеку. Создал программы, которые используют функции динамических библиотек. Разобрался как собирается программа(более подробно этапы сборки), узнал различие extern и inline.