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: Меджидли Исмаил Ибрагим оглы
Группа: М8О-201Б-21
Вариант: 34
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023

Содержание

Репозиторий

https://github.com/imedzhidli/Operational-Systems

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.

7	Подсчет площади плоской геометрической фигуры по двум сторонам	Float Square(float A, float B)	Фигура прямоугольник	Фигура прямоугольный треугольник
8	Перевод числа x из десятичной системы счисления в другую	Char* translation(long x)	Другая система счисления двоичная	Другая система счисления троичная

Общие сведения о программе

Для выполнения данной лабораторной работы я предварительно создал 4 файла: первые два – lib1.cpp и lib2.cpp являются исходным кодом для наших динамических библиотек. Файлы first.cpp и second.cpp являются двумя программами, которые нужно было реализовать по заданию. first.cpp является программой, к которой библиотека подгружается на этапе компиляции, а second.cpp является программой, к которой библиотека подключается непосредственно в самом коде.

1) add_library(d1 SHARED lib1.cpp) - создает целевую библиотеку d1, которая собирается из исходного файла lib1.cpp. Ключевое слово SHARED указывает, что это динамическая библиотека (shared library).
2) target_link_libraries(main1 d1 -Wl,-rpath,.) - добавляет библиотеку d1 в целевой файл main1. -Wl,-rpath,. задает путь к директории, где будут искаться зависимости во время исполнения.

3) target_link_libraries(main2 d1) - добавляет библиотеку d1 в целевой файл main2. D1 - это библиотека для загрузки динамических библиотек во время исполнения программы.
4) set_target_properties(d1 PROPERTIES OUTPUT_NAME "d1") - задает имя выходного файла для библиотеки d1.
5) set_target_properties(d1 PROPERTIES PREFIX "") - удаляет префикс lib из имени выходного файла библиотеки d1.
6) set_target_properties(d1 PROPERTIES SUFFIX ".so") - задает суффикс .so для выходного файла библиотеки d1. Для Unix-подобных систем это означает, что это динамическая библиотека.

void* dlopen(...) - загружает нашу библиотеку;

void* dlsym(...) - присваивает указателю на функцию ее адрес в библиотеке

int dlclose(...) - освобождает указатель на библиотеку

lab5_test.cpp - тесты к лабораторной работе

Общий метод и алгоритм решения

В самом начале выполнения лабораторной работы я реализовал две библиотеки: lib1.cpp и lib2.cpp. Там реализовал простейший подсчет площади квадрата и прямоугольного треугольника, а также алгоритмы перевода числа из десятичной системы в двоичную и троичную. В first.cpp через конструкцию if обрабатывал команды, которые вводит пользователь и выдавал ожидаемый результат. В second.cpp подгружал библиотеки через

void* dlopen(...), через void* dlsym(...) делал функции из динамических библиотек видными для second.cpp.

Исходный код

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.16.3)

project(lab5 LANGUAGES CXX)

# Добавляем библиотеку d1
add_library(d1 SHARED lib1.cpp)
set_target_properties(d1 PROPERTIES OUTPUT_NAME "d1")
set_target_properties(d1 PROPERTIES PREFIX "")
set_target_properties(d1 PROPERTIES SUFFIX ".so")

# Добавляем библиотеку d2
add_library(d2 SHARED lib2.cpp)
set_target_properties(d2 PROPERTIES OUTPUT_NAME "d2")
set_target_properties(d2 PROPERTIES PREFIX "")
set_target_properties(d2 PROPERTIES SUFFIX ".so")

# Добавляем исполняемый файл main1 и линкуем с библиотекой d1
add_executable(main1 first.cpp)
target_link_libraries(main1 d1)
target_link_options(main1 PRIVATE -Wl,-rpath=$ORIGIN)

# Добавляем исполняемый файл main2 и линкуем с библиотекой dl
add_executable(main2 second.cpp)
target_link_libraries(main2 dl)
# target_link_libraries(main2 dl)
```

first.cpp

```
#include <iostream>

using namespace std;

extern "C" float Square(float A, float B);
extern "C" char* translation(long x);

int main(){
   int command;
   while((cout << "Введите команду: ") && (cin >> command)){
       if(command == 1){
           cout << "Введите длину A и B: ";
           float A, B;
           cin >> A >> B;
           cout << "Площадь -  " << Square(A, B) << endl;
       }
```

```
        else if(command == 2){
            long x;
            cout << "Введите десятичное число: ";
            cin >> x;
            char* memory = translation(x); /*преобразование числа в
бинарное\троичное и суем в переменную*/
            cout << "Двоичное число - " << memory << endl;
            free(memory);
        }
        else
            cout << "Команды могут быть 1 и 2 ";
    }
}
```

second.cpp

```
#include <cstdlib>
#include <iostream>
#include <dlfcn.h>

using namespace std;

int main(){
    int num_of_lib; /*чекаем правильность введенный библиотеки*/
    cout << "Введите номер библиотеки: ";
    cin >> num_of_lib;
    if(num_of_lib != 1 && num_of_lib != 2){
        cout << "Ошибочная библиотека\n";
        exit(1);
    }

    --num_of_lib; /*уменьшаем число нашей либы*/
    int command;
    const char* libs[] = {"/usr/lib/d1.so", "/usr/lib/d2.so"}; /*выбираем
нужную либу*/
    // void* library_handle;
    void* library_handle = dlopen(libs[num_of_lib], RTLD_LAZY);
    /*флаг - разрешение символов будет производиться только при первом
обращении к соответствующей функции в библиотеке
    Загружает динамическую библиотеку с указанным именем и возвращает
указатель на обработчик библиотеки*/
    if(!library_handle){ /*проверим была ли загружена ф-ия с помощью
dlopen()*/
        cout << "Ошибка при вызове функции dlopen\n";
        exit(1);
```

```
    }

    float (*Square)(float A, float B); /*указатель на функцию*/
    char* (*translation)(long x);

    Square = (float(*)(float, float))dlsym(library_handle, "Square");
/*передаем функцию из либы с помощью dlsym() -

получить адрес функции в динамической библиотеке по её имени*/
    translation = (char*(*)(long x))dlsym(library_handle, "translation");

    cout << "Введите команду 0, 1 или 2\n";
    while(cin >> command) {
        switch (command) { /*проверяем что значение корректное*/
            default:
                cout << "Неверная команда\n";
                break;
            case 0: /*закрываем текущую либу,переменная num_of_lib
обновляется с помощью операции остатка от деления, чтобы
            переключаться между двумя библиотеками по кругу. Далее
вызывается dlopen с новой библиотекой.
            функции Square и translation обновляются, чтобы они указывали
на функции из новой библиотеки.*/
                dlclose(library_handle);
                num_of_lib = (num_of_lib + 1) % 2;
                library_handle = dlopen(libs[num_of_lib], RTLD_LAZY);
                if(!library_handle){
                    cout << "Ошибка при вызове функции dlopen\n";
                    exit(1);
                }
                Square = (float(*)(float, float))dlsym(library_handle,
"Square");
                translation = (char*(*)(long x))dlsym(library_handle,
"translation");
                cout << "Смена функций библиотеки\n";
                break;
            case 1: /*тут находим площадь*/
                cout << "Введите длину A и B: ";
                float A, B;
                cin >> A >> B;
                cout << "Площадь - " << Square(A, B) << endl;
                break;
            case 2: /*тут делаем перевод в другую сс, в зависимости от
выбранной либы*/
                long x;
```

```
                cout << "Введите десятичное число: ";
                cin >> x;
                char* memory = translation(x);
                if(num_of_lib + 1 == 1) {
                    cout << "Двоичное число - " << memory << endl;
                }
                else if (num_of_lib + 1 == 2){
                    cout << "Троичное число - " << memory << endl;
                }
                free(memory);
                break;
        }
    }
    dlclose(library_handle);
}
```

lib1.cpp

```
#include <iostream>
#include <algorithm>
#include <cstring>

using namespace std;

extern "C" float Square(float A, float B);
extern "C" char* translation(long x);

float Square(float A, float B){
    return A * B;
}

char* translation(long x){
    string bin;
    if(x == 0) bin += "0";
    while(x > 0){
        bin += to_string(x % 2);
        x /= 2;
    }
    string number = bin;
    reverse(number.begin(), number.end());
    char* answer = (char*) malloc((number.size() + 1) * sizeof(char));
    strcpy(answer, number.c_str());
    return answer;
}
```

lib2.cpp

```cpp
#include <iostream>
#include <algorithm>
#include <cstring>

extern "C" float Square(float A, float B);
extern "C" char* translation(long x);

float Square(float A, float B){
    return 0.5 * (A * B);
}

char* translation(long x) {
    std::string bin;
    if(x == 0) bin += "0";
    while (x > 0) {
        bin += std::to_string(x % 3);
        x /= 3;
    }
    std::string number = bin;
    std::reverse(number.begin(), number.end());
    char *answer = (char *) malloc((number.size() + 1) * sizeof(char));
    strcpy(answer, number.c_str());
    return answer;
}
```

lab5_test.cpp

```cpp
#include <cstdlib>

#include <iostream>

#include <dlfcn.h>

#include <gtest/gtest.h>


TEST(Lab5Test, d1){

    float (*Square)(float A, float B); /*указатель на функцию*/

    char* (*translation)(long x);


    void* library_handle = dlopen("/usr/lib/d1.so", RTLD_LAZY);

    ASSERT_NE(library_handle, nullptr);
```

```
    Square = (float(*)(float, float))dlsym(library_handle, "Square");
/*передаем функцию из либы с помощью dlsym() -

получить адрес функции в динамической библиотеке по её имени*/
    translation = (char*(*)(long x))dlsym(library_handle, "translation");
    float A = 3, B = 5;
    float expected = 15;
    float result = Square(A, B);
    EXPECT_EQ(expected, result);
    long number = 5;
    char *resnum = translation(number);


    int sum = atoi(resnum);


    int expectednum = 101;
    EXPECT_EQ(sum, expectednum);
    dlclose(library_handle);
}


TEST(Lab5Test, d2){
    float (*Square)(float A, float B); /*указатель на функцию*/
    char* (*translation)(long x);


    void* library_handle = dlopen("/usr/lib/d2.so", RTLD_LAZY);
    ASSERT_NE(library_handle, nullptr);
    Square = (float(*)(float, float))dlsym(library_handle, "Square");
/*передаем функцию из либы с помощью dlsym() -

получить адрес функции в динамической библиотеке по её имени*/
    translation = (char*(*)(long x))dlsym(library_handle, "translation");
    float A = 3, B = 5;
```

```
    float expected = 7.5;
    float result = Square(A, B);
    EXPECT_EQ(expected, result);
    long number = 5;
    char *resnum = translation(number);

    int sum = atoi(resnum);

    int expectednum = 12;
    EXPECT_EQ(sum, expectednum);
    dlclose(library_handle);
}

extern "C" float Square(float A, float B);
extern "C" char* translation(long x);

TEST(Lab5Test, d1link){
    float A = 3, B = 5;
    float expected = 15;
    float result = Square(A, B);
    EXPECT_EQ(expected, result);
    long number = 5;
    char *resnum = translation(number);

    int sum = atoi(resnum);

    int expectednum = 101;
    EXPECT_EQ(sum, expectednum);

}
```

```
// TEST(Lab5Test, d2link){
//     float A = 3, B = 5;
//     float expected = 7.5;
//     float result = Square(A, B);
//     EXPECT_EQ(expected, result);
//     long number = 5;
//     char *resnum = translation(number);

//     int sum = atoi(resnum);

//     int expectednum = 12;
//     EXPECT_EQ(sum, expectednum);

// }
```

Демонстрация работы программы

imedzhidli@imedzhidli:~/Desktop/OS/5Laba/build$ ./main1

Введите команду: 0

Команды могут быть 1 и 2 Введите команду: 1

Введите длину A и B: 3 5

Площадь - 15

Введите команду: 2

Введите десятичное число: 5

Двоичное число - 101

Введите команду:

imedzhidli@imedzhidli:~/Desktop/OS/5Laba/build$ ./main2

Введите номер библиотеки: 2

Введите команду 0, 1 или 2

1

Введите длину A и B: 3 5

Площадь - 7.5

2

Введите десятичное число: 5

Троичное число - 12

0

Смена функций библиотеки

1

Введите длину A и B: 3 5

Площадь - 15

2

Введите десятичное число: 5

Двоичное число - 101

imedzhidli@imedzhidli:~/Desktop/OS/5Laba/build$

Выводы:

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