Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Тема работы**

**“Динамические библиотеки”**

Студент: Зинин Владислав Владимирович

Группа: М8О-208Б-20

Вариант: 10

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/frankeloff

**Постановка задачи**

Задача: реализовать 2 динамические библиотеки и 2 программы для работы с ними. Первая программа будет загружать библиотеку (одну) на этапе компиляции при помощи ключа -lmylib, а вторая программа будет подключать две динамические библиотеки при помощи dl-функций в самом коде.

**Общие сведения о программе**

Для выполнения данной лабораторной работы я предварительно создал 5 файлов: первые два - first.cpp и second.cpp являются исходным кодом для наших динамических библиотек. Файлы compilation.cpp и launch.cpp являются двумя программами, которые нужно было реализовать по заданию. compilation.cpp является программой, к которой библиотека подгружается на этапе компиляции, а launch.cpp является программой, к которой библиотека подключается непосредственно в самом коде.

Помимо этого, для удобства компиляции всех программ я создал MakeFile со следующим набором команд:  
1) g++ -fPIC -c first.cpp -o d1.o

g++ -fPIC -c second.cpp -o d2.o

При помощи этих команд наши cpp-библиотеки превращаются в объектные файлы. Это, так называемый, “промежуточный этап” создания динамических библиотек.

2) g++ -shared d1.o -o libd1.so

g++ -shared d2.o -o libd2.so

При помощи флага -shared мы создаем наши нужные по заданию динамические библиотеки.

3) g++ compilation.cpp -L. -ld1 -o main1 -Wl,-rpath -Wl,.

Этой строчкой мы делаем исполняемый файл из нашей программы compilation.cpp, при этом компилируем мы только с одной библиотекой (то есть компиляция может проходить либо с ключом -ld1, либо с ключом -ld2).

4) g++ launch.cpp -L. -ldl -o main2 -Wl,-rpath -Wl,.

Этой строчкой мы делаем исполняемый файл из нашей программы launch.cpp, только теперь с флагом -ldl. Далее в нашей программе main2 будут доступны 2 динамические библиотеки, действия над которыми будут обрабатываться при помощи следующих функций:  
void\* dlopen(...) - вгружает нашу библиотеку;

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

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

5) rm -r \*.so \*.o main1 main2

При помощи команды make clean происходит удаление всех созданных файлов, вследствие чего в папке остаются исходные 5 объектов.

**Общий метод и алгоритм решения**

В самом начале выполнения лабораторной работы я реализовал две библиотеки: first.cpp и second.cpp. В библиотеке first.cpp реализован расчет производной функции cos(x) в точке A с приращением deltaX первым способом и реализован подсчёт наибольшего общего делителя для двух натуральных чисел алгоритмом Евклида. В библиотеке second.cpp реализован расчет производной функции cos(x) в точке A с приращением deltaX вторым способом и реализован подсчёт наибольшего общего делителя для двух натуральных чисел наивным алгоритмом. Далее в файле compilation.cpp я реализовал обычное считывание команды при помощи проверки равенства функции scanf на -1 (вводится EOF - Ctrl+D на Ubuntu) и конструкции switch-case. Если вводится команда, отличная от 1 или 2, вылезает сообщение о том, что ввод был осуществлен неправильно. Если вводится 1, то рассчитывается производная первым способом. Если вводится 2, то подсчитывается наибольший общий делитель алгоритмом Евклада.  
Что же касается launch.cpp, то там суть почти та же. В начале создаю необходимые указатели, позже загружаю какую-либо дин. библиотеку в зависимости от ввода пользователя. При помощи известного нам считывания до EOF я считываю команду. Если это не 0, не 1 и не 2, то прошу ввести правильную команду. Если эта команда 0, то программа меняет библиотеки (то есть, если раньше мне был доступен подсчет наибольшего общего делителя алгоритмом Евклида и расчет производной первым способом, то теперь мне будет доступен доступен подсчет наибольшего общего делителя наивным алгоритмом и расчет производной вторым способом Если команда 1, я считаю наибольший общий делитель. Если 2, то производную. В конце освобождается указатель на библиотеку в целях избежания утечек памяти, программа завершается.

**Исходный код**

**first.cpp**

#include <math.h>

#include <iostream>

extern "C" int GCF (int A, int B); // Глобальная функция, которую можно использовать в других файлах программы

extern "C" float Derivative(float A, float deltaX);

int GCF(int A, int B){

if(A % B == 0)

{

return B;

}

else if (B % A == 0){

return A;

}

else{

if(A > B){

return GCF(A%B, B);

}

else{

return GCF(A, B%A);

}

}

}

float Derivative(float A, float deltaX){

std::cout << "You are on the first. Your result - ";

return (cos(A + deltaX) - cos(A))/deltaX;

}

**second.cpp**

#include <math.h>

#include <iostream>

extern "C" int GCF (int A, int B); // Глобальная функция, которую можно использовать в других файлах программы

extern "C" float Derivative(float A, float deltaX);

int GCF(int A, int B){

std::cout << "You are on the second. Your result - ";

int a;

if(A == B) return A;

else if(A > B){

a = B;

}

else{

a = A;

}

while(A%a != 0 || B%a != 0){

a--;

}

return a;

}

float Derivative(float A, float deltaX){

std::cout << "You are on the second. Your result - ";

return (cos(A + deltaX) - cos(A-deltaX))/(2\*deltaX);

}

**launch.cpp**

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <dlfcn.h>

#include <math.h>

int main(){

void\* handle = NULL;

int (\*GCF)(int A, int B);

float (\*Derivative)(float A, float deltaX);

int start\_lib;

const char\* lib\_arr[] = {"libd1.so", "libd2.so"};

std:: cout << "Enter start library: " << std:: endl;

std:: cout << "1 for using first library" << std:: endl;

std:: cout << "2 for using second library" << std:: endl;

std:: cin >> start\_lib;

int curlib;

while(true){

if(start\_lib == 1){

curlib = 0;

break;

}

else if(start\_lib == 2){

curlib = 1;

break;

}

else{

std:: cout << "You should enter only 1 or 2!" << std:: endl;

std:: cin >> start\_lib;

}

}

handle = dlopen(lib\_arr[curlib], RTLD\_LAZY); //rtld lazy выполняется поиск только тех символов, на которые есть ссылки из кода

//RTLD\_LAZY включает режим отложенного связывания, в котором поиск функции в библиотеке выполняется в момент первого обращения к ней

if (!handle) {

std:: cout << "An error while opening library has been detected" << std:: endl;

exit(EXIT\_FAILURE);

}

GCF = (int(\*)(int, int))dlsym(handle, "GCF"); //возвращаем адрес функции из памяти библиотеки (если адрес не найден, вовзращается null)

if (!GCF){

std::cout << "The function is missing" << std::endl;

dlclose(handle);

exit(EXIT\_FAILURE);

}

Derivative = (float(\*)(float, float))dlsym(handle, "Derivative"); //dlsym присваивает указателю на функцию, объявленному в начале, ее адрес в библиотеке

if (!Derivative){

std::cout << "Derivative is missing" << std::endl;

dlclose(handle);

exit(EXIT\_FAILURE);

}

int command;

std:: cout << "Hello there! Please enter your command according to next rules: " << std:: endl;

std:: cout << '\t' << "0 for changing the contract;" << std:: endl;

std:: cout << '\t' << "1 for calculating greatest common divisor; " << std:: endl;

std:: cout << '\t' << "2 for calculating the Derivative; " << std:: endl;

std:: cout << '\t' << "3 for exit; " << std:: endl;

while (printf("Please enter your command: ") && (scanf("%d", &command)) != EOF) {

if (command == 0) {

dlclose(handle); //освобождает указатель на библиотеку и программа перестает ей пользоваться

if (curlib == 0) {

curlib = 1 - curlib;

handle = dlopen(lib\_arr[curlib], RTLD\_LAZY);

if (!handle) {

std:: cout << "An error while opening library has been detected" << std:: endl;

exit(EXIT\_FAILURE);

}

GCF = (int(\*)(int, int))dlsym(handle, "GCF");

if (!GCF){

std::cout << "GCF is missing" << std::endl;

dlclose(handle);

exit(EXIT\_FAILURE);

}

Derivative = (float(\*)(float, float))dlsym(handle, "Derivative");

if (!Derivative){

std::cout << "Derivative is missing" << std::endl;

dlclose(handle);

exit(EXIT\_FAILURE);

}

}

else if (curlib == 1) {

curlib = 1 - curlib;

handle = dlopen(lib\_arr[curlib], RTLD\_LAZY);

if (!handle) {

std:: cout << "An error while opening library has been detected" << std:: endl;

exit(EXIT\_FAILURE);

}

GCF = (int(\*)(int, int))dlsym(handle, "GCF");

if (!GCF){

std::cout << "GCF is missing" << std::endl;

dlclose(handle);

exit(EXIT\_FAILURE);

}

Derivative = (float(\*)(float, float))dlsym(handle, "Derivative");

if (!Derivative){

std::cout << "Derivative is missing" << std::endl;

dlclose(handle);

exit(EXIT\_FAILURE);

}

}

std:: cout << "You have changed contracts!" << std:: endl;

}

else if (command == 1) {

int a,b;

std:: cout << "Please enter numbers x, y ";

std:: cin >> a >> b;

if (a <= 0 || b <= 0) {

std:: cout << "Please enter a positive numbers!" << std:: endl;

}

else {

std:: cout << "Result " << GCF(a, b) << std:: endl;

}

}

else if (command == 2) {

float A, B;

std:: cout << "Please enter A and delta X increment: ";

std:: cin >> A >> B;

std::cout << Derivative(A, B) << std:: endl;

}

else if (command == 3){

dlclose(handle);

return 0;

}

else {

std:: cout << "You had to enter only 0, 1, 2 or 3!" << std:: endl;

}

}

return 0;

}

**compilation.cpp**

#include <iostream>

extern "C" int GCF (int A, int B); // Глобальная функция, которую можно использовать в других файлах программы

extern "C" float Derivative(float A, float deltaX);

int main(){

int com;

while(printf("Please, enter your command:") && scanf("%d", &com) != EOF)

{

switch(com){

case 1:

int first;

int second;

std::cin >> first >> second;

if(first <= 0 || second <= 0){

std::cout << "Please, enter only positive numbers!" << std::endl;

}

else{

std::cout << "Result: " << GCF(first, second) << std::endl;

}

break;

case 2:

float A;

float deltaX;

std::cin >> A >> deltaX;

std::cout << "Result: " << Derivative(A, deltaX) << std::endl;

break;

case 3:

exit(0);

default:

std::cout << "You must enter only 1 or 2!" << std::endl;

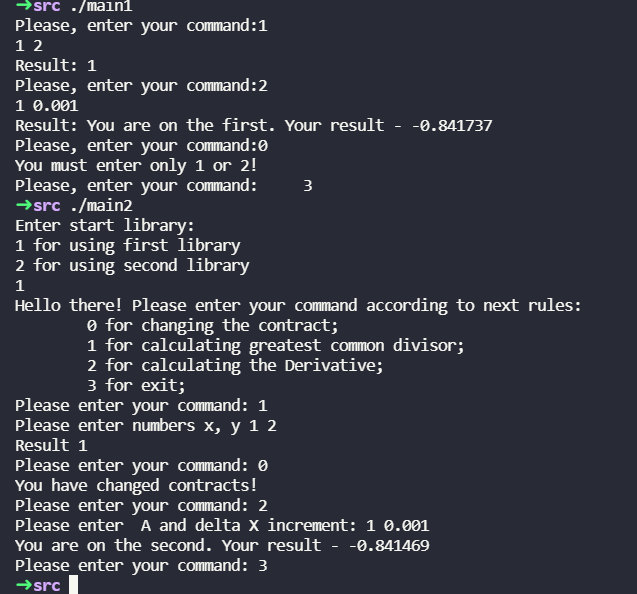
break;

}

}

return 0;

}

**Демонстрация работы программы**

**Выводы**Данная лабораторная работа научила меня пользоваться dl-функциями, благодаря реализации исполняемых файлов по заданию, я закрепил навык работы с динамическими библиотеками и полностью осознал их отличие от статических библиотек.