Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

Студент: Рамалданов Рустамхан Ражудинович

Группа: М8О-208Б-22

Вариант: 15

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**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/pepelulka/OS_labs>

**Постановка задачи**

Требуется создать динамические библиотеки, которые реализуют определенный функционал.

Далее использовать данные библиотеки 2-мя способами:

* Во время компиляции (на этапе «линковки»/linking)
* Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

**Общие сведения о программе**

Программа состоит из одного заголовочного файла – lib.h и из двух его реализаций – lib1.c и lib2.c, которые будут линковаться разными способами в файлы main1.cpp и main2.cpp соответственно. Программа написана под ОС Linux.

Были использованы следующие системные вызовы:

1. dlopen()
2. dlsym()

**Общий метод и алгоритм решения**

В файлах lib1.c и lib2.c воплощены разные реализации функций с одним и тем же интерфейсом, которые по разному будут линковаться в файлах main1.cpp и main2.cpp

**Исходный код**

**========================== lib.h ==========================**

**#ifndef \_\_LIB\_\_**

**#define \_\_LIB\_\_**

**#include "stdio.h"**

**float Derivative(float a, float deltaX);**

**int\* Sort(int \*array, size\_t size);**

**#endif**

**========================== lib1.c ==========================**

**#include "lib.h"**

**#include <math.h>**

**float Derivative(float a, float deltaX) {**

**return (cos(a + deltaX) - cos(a)) / deltaX;**

**}**

**int\* Sort(int \* array, size\_t size) {**

**if (size == 1 || size == 0) {**

**return array;**

**}**

**for (size\_t i = 0; i < size; i++) {**

**for (int j = size - 1; j >= 1; j--) {**

**if (array[j] < array[j - 1]) {**

**int c = array[j];**

**array[j] = array[j - 1];**

**array[j - 1] = c;**

**}**

**}**

**}**

**return array;**

**}**

**========================== lib2.c ==========================**

**#include "lib.h"**

**#include <math.h>**

**float Derivative(float a, float deltaX) {**

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

**}**

**static void swap(int\* lhs, int\* rhs) {**

**int c = \*lhs;**

**\*lhs = \*rhs;**

**\*rhs = c;**

**}**

**static void QuickSort(int\* array, int first, int last) {**

**int i = first, j = last, x = array[(first + last) / 2];**

**do {**

**while (array[i] < x) i++;**

**while (array[j] > x) j--;**

**if(i <= j) {**

**if (array[i] > array[j]) swap(&array[i], &array[j]);**

**i++;**

**j--;**

**}**

**} while (i <= j);**

**if (i < last) {**

**QuickSort(array, i, last);**

**}**

**if (first < j) {**

**QuickSort(array, first, j);**

**}**

**}**

**int\* Sort(int \* array, size\_t size) {**

**if (size == 1 || size == 0) {**

**return array;**

**}**

**QuickSort(array, 0, size - 1);**

**return array;**

**}**

**========================== main1.cpp ==========================**

**extern "C" {**

**#include "lib.h"**

**}**

**#include <iostream>**

**#include <sstream>**

**#include <vector>**

**// MainRoutine pipeline :**

**// | 1. Start() runs event loop and get line.**

**// | 2. ProceedLine(line) parse string and decide what function to call.**

**// | It uses Tokenize(line) to divide string into words.**

**// | 3. Call handlers (Help(), Exit() ...)**

**// \./**

**class TMainRoutine {**

**private:**

**// Handlers =========================================================**

**static void InvalidArgs() {**

**std::cout << "Invalid arguments\n";**

**}**

**static void MissingArgs() {**

**std::cout << "Missing arguments\n";**

**}**

**static void Function1(float a, float deltaX) {**

**std::cout << Derivative(a, deltaX) << std::endl;**

**}**

**static void Function2(int \* array, size\_t size) {**

**Sort(array, size);**

**for (size\_t i = 0; i < size; i++) {**

**std::cout << array[i] << " ";**

**}**

**std::cout << std::endl;**

**}**

**static void Help() {**

**std::cout << "I can do:\n";**

**std::cout << " 1. help\n";**

**std::cout << " 2. exit - exits program\n";**

**std::cout << " 3. 1 <a> <deltaX> - calculates derivative of cos(x) of a with specific deltaX\n";**

**std::cout << " 4. 2 <size> <a0> <a1> ... <an> - sort array a\n";**

**}**

**// Handlers end =====================================================**

**static std::vector<std::string> Tokenize(const std::string &line) {**

**std::vector<std::string> result;**

**std::stringstream ss(line);**

**while (ss) {**

**std::string token;**

**ss >> token;**

**if (token != "") {**

**result.push\_back(token);**

**}**

**}**

**return result;**

**}**

**// Return false if user exited**

**static bool ProceedLine(const std::string &line) {**

**std::vector<std::string> tokens = Tokenize(line);**

**if (tokens[0] == "help") {**

**Help();**

**} else if (tokens[0] == "exit") {**

**return false;**

**} else if (tokens[0] == "1") {**

**try {**

**if (tokens.size() < 3) {**

**MissingArgs();**

**return true;**

**}**

**Function1(std::atof(tokens[1].c\_str()), std::atof(tokens[2].c\_str()));**

**} catch (...) {**

**InvalidArgs();**

**}**

**} else if (tokens[0] == "2") {**

**try {**

**if (tokens.size() < 2) {**

**MissingArgs();**

**return true;**

**}**

**int size = std::atoi(tokens[1].c\_str());**

**if (size < 0) {**

**InvalidArgs();**

**return true;**

**}**

**if (tokens.size() < static\_cast<size\_t>(2 + size)) {**

**MissingArgs();**

**return true;**

**}**

**int\* ptr = new int[size];**

**for (int i = 0; i < size; i++) {**

**ptr[i] = std::atoi(tokens[i + 2].c\_str());**

**}**

**Function2(ptr, size);**

**delete[] ptr;**

**} catch (...) {**

**InvalidArgs();**

**}**

**} else {**

**InvalidArgs();**

**}**

**return true;**

**}**

**public:**

**static void Start() {**

**bool terminated = false;**

**do {**

**std::cout << "> ";**

**std::string line;**

**std::getline(std::cin, line);**

**terminated = !ProceedLine(line);**

**} while (!terminated);**

**}**

**};**

**int main() {**

**TMainRoutine::Start();**

**}**

**========================== main2.cpp ==========================**

**#ifdef \_\_unix\_\_**

**extern "C" {**

**#include "lib.h"**

**}**

**#include <dlfcn.h>**

**#include <iostream>**

**#include <vector>**

**#include <array>**

**#include <sstream>**

**// MainRoutine pipeline :**

**// | 1. Start() runs event loop and get line.**

**// | 2. ProceedLine(line) parse string and decide what function to call.**

**// | It uses Tokenize(line) to divide string into words.**

**// | 3. Call handlers (Help(), Exit() ...)**

**// \./**

**class TMainRoutine {**

**private:**

**// Contracts:**

**using Function1Ptr = float(\*)(float, float);**

**using Function2Ptr = int\*(\*)(int\*, size\_t);**

**const char \* Function1Name = "Derivative";**

**const char \* Function2Name = "Sort";**

**//**

**Function1Ptr func1;**

**Function2Ptr func2;**

**size\_t currentMode;**

**std::array<void\*, 2> dls;**

**// Handlers =========================================================**

**void SwitchMode() {**

**currentMode = 1 - currentMode;**

**func1 = (Function1Ptr)(dlsym(dls[currentMode], Function1Name));**

**func2 = (Function2Ptr)(dlsym(dls[currentMode], Function2Name));**

**}**

**void InvalidArgs() {**

**std::cout << "Invalid arguments\n";**

**}**

**void MissingArgs() {**

**std::cout << "Missing arguments\n";**

**}**

**void Function1(float a, float deltaX) {**

**std::cout << func1(a, deltaX) << std::endl;**

**}**

**void Function2(int \* array, size\_t size) {**

**func2(array, size);**

**for (size\_t i = 0; i < size; i++) {**

**std::cout << array[i] << " ";**

**}**

**std::cout << std::endl;**

**}**

**void Help() {**

**std::cout << "I can do:\n";**

**std::cout << " 1. help\n";**

**std::cout << " 2. exit - exits program\n";**

**std::cout << " 3. 1 <a> <deltaX> - calculates derivative of cos(x) of a with specific deltaX\n";**

**std::cout << " 4. 2 <size> <a0> <a1> ... <an> - sort array a\n";**

**}**

**// Handlers end =====================================================**

**std::vector<std::string> Tokenize(const std::string &line) {**

**std::vector<std::string> result;**

**std::stringstream ss(line);**

**while (ss) {**

**std::string token;**

**ss >> token;**

**if (token != "") {**

**result.push\_back(token);**

**}**

**}**

**return result;**

**}**

**// Return false if user exited**

**bool ProceedLine(const std::string &line) {**

**std::vector<std::string> tokens = Tokenize(line);**

**if (tokens[0] == "help") {**

**Help();**

**} else if (tokens[0] == "exit") {**

**return false;**

**} else if (tokens[0] == "1") {**

**try {**

**if (tokens.size() < 3) {**

**MissingArgs();**

**return true;**

**}**

**Function1(std::atof(tokens[1].c\_str()), std::atof(tokens[2].c\_str()));**

**} catch (...) {**

**InvalidArgs();**

**}**

**} else if (tokens[0] == "2") {**

**try {**

**if (tokens.size() < 2) {**

**MissingArgs();**

**return true;**

**}**

**int size = std::atoi(tokens[1].c\_str());**

**if (size < 0) {**

**InvalidArgs();**

**return true;**

**}**

**if (tokens.size() < static\_cast<size\_t>(2 + size)) {**

**MissingArgs();**

**return true;**

**}**

**int\* ptr = new int[size];**

**for (int i = 0; i < size; i++) {**

**ptr[i] = std::atoi(tokens[i + 2].c\_str());**

**}**

**Function2(ptr, size);**

**delete[] ptr;**

**} catch (...) {**

**InvalidArgs();**

**}**

**} else if (tokens[0] == "0") {**

**SwitchMode();**

**std::cout << "Switched\n";**

**} else {**

**InvalidArgs();**

**}**

**return true;**

**}**

**public:**

**TMainRoutine() = delete;**

**TMainRoutine(const char \* lib1Path, const char \* lib2Path) {**

**dls[0] = dlopen(lib1Path, RTLD\_LOCAL | RTLD\_LAZY);**

**if (dls[0] == nullptr) {**

**throw std::invalid\_argument("Can't open first lib.");**

**}**

**dls[1] = dlopen(lib2Path, RTLD\_LOCAL | RTLD\_LAZY);**

**if (dls[1] == nullptr) {**

**throw std::invalid\_argument("Can't open second lib.");**

**}**

**currentMode = 1;**

**SwitchMode();**

**}**

**void Start() {**

**bool terminated = false;**

**do {**

**std::cout << "> ";**

**std::string line;**

**std::getline(std::cin, line);**

**terminated = !ProceedLine(line);**

**} while (!terminated);**

**}**

**};**

**#endif**

**int main() {**

**if (getenv("PATH\_TO\_LIB1") == nullptr) {**

**std::cerr << "PATH\_TO\_LIB1 is not specified\n";**

**exit(1);**

**}**

**if (getenv("PATH\_TO\_LIB2") == nullptr) {**

**std::cerr << "PATH\_TO\_LIB2 is not specified\n";**

**exit(1);**

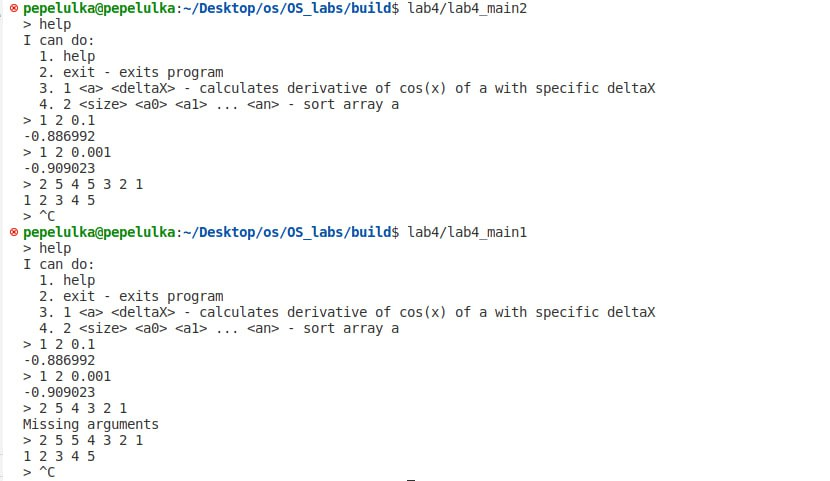
**}**

**TMainRoutine routine(getenv("PATH\_TO\_LIB1"), getenv("PATH\_TO\_LIB2"));**

**routine.Start();**

**}**

**Демонстрация работы программы**



**Выводы**

В ходе выполнения лабораторной работе были получены глубокие знания о процессе сборки и компиляции программ на языке Си и C++. Была изучена разница между динамической линковкой на этапе компиляции и на этапе выполнения программы.