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: Бурдинский Владислав Дмитриевич

Группа: М8О–203Б–23

Вариант: 18

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2024.

**Постановка задачи**

Цель работы

Ц елью является приобретение практических навыков в:

Создание динамических библиотек

Создание программ, которые используют функции динамических библиотек

Задание

Требуется создать динамические библиотеки, которые реализуют заданный вариантом 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

**Код программы**

**e\_formula.cpp**

#include <cmath>

#include <iostream>

extern "C" float E(int x) {

if (x == 0) return 1.0f;

float result = pow(1.0f + 1.0f / x, x);

return result;

}

e\_series.cpp

extern "C" float E(int x) {

float sum = 0.0f;

int factorial = 1;

for (int n = 0; n <= x; ++n) {

if (n > 0){

factorial \*= n;

}

sum += 1.0f / factorial;

}

return sum;

}

prime\_naive.cpp

#include <cmath>

#include <iostream>

extern "C" int PrimeCount(int A, int B) {

int count = 0;

for (int num = A; num <= B; ++num) {

if (num < 2) continue;

bool isPrime = true;

int stop\_point = sqrt(num);

for (int j = 2; j <= stop\_point; ++j) {

if (num % j == 0) {

isPrime = false;

break;

}

}

if (isPrime) {

++count;

}

}

return count;

}

prime\_sieve.cpp

#include <vector>

extern "C" int PrimeCount(int A, int B) {

if (B < 2 || A > B) return 0;

std::vector<bool> isPrime(B + 1, true);

isPrime[0] = isPrime[1] = false;

for (int i = 2; i \* i <= B; ++i) {

if (isPrime[i]) {

for (int j = i \* i; j <= B; j += i) {

isPrime[j] = false;

}

}

}

int count = 0;

for (int i = A; i <= B; ++i) {

if (isPrime[i]) count++;

}

return count;

}

test\_dynamic.cpp

#include <iostream>

#include <dlfcn.h> // Для macos

#include <sstream>

#include <string>

typedef int (\*PrimeCountFunc)(int, int);

typedef float (\*EFunc)(int);

int main() {

void\* handlePrime = dlopen("./prime\_naive.dylib", RTLD\_LAZY);

void\* handleE = dlopen("./e\_formula.dylib", RTLD\_LAZY);

if (!handlePrime || !handleE) {

std::cerr << "Ошибка загрузки библиотек" << std::endl;

return 1;

}

PrimeCountFunc PrimeCount = (PrimeCountFunc)dlsym(handlePrime, "PrimeCount");

EFunc E = (EFunc)dlsym(handleE, "E");

std::string command;

while (std::getline(std::cin, command)) {

if (command.empty()) continue;

std::istringstream iss(command);

int cmd;

iss >> cmd;

if (cmd == 0) {

dlclose(handlePrime);

dlclose(handleE);

static bool toggle = false;

toggle = !toggle;

if (toggle) {

handlePrime = dlopen("./prime\_sieve.dylib", RTLD\_LAZY);

handleE = dlopen("./e\_series.dylib", RTLD\_LAZY);

} else {

handlePrime = dlopen("./prime\_naive.dylib", RTLD\_LAZY);

handleE = dlopen("./e\_formula.dylib", RTLD\_LAZY);

}

if (!handlePrime || !handleE) {

std::cerr << "Ошибка загрузки библиотек" << std::endl;

return 1;

}

PrimeCount = (PrimeCountFunc)dlsym(handlePrime, "PrimeCount");

E = (EFunc)dlsym(handleE, "E");

std::cout << "Реализации переключены" << std::endl;

} else if (cmd == 1) {

int A, B;

iss >> A >> B;

int result = PrimeCount(A, B);

std::cout << "Количество простых чисел: " << result << std::endl;

} else if (cmd == 2) {

int x;

iss >> x;

float result = E(x);

std::cout << "Значение числа e: " << result << std::endl;

} else {

std::cout << "Неизвестная команда" << std::endl;

}

}

dlclose(handlePrime);

dlclose(handleE);

return 0;

}

test\_static.cpp

#include <iostream>

#include <sstream>

#include <string>

extern "C" {

int PrimeCount(int A, int B);

float E(int x);

}

int main() {

std::string command;

while (std::getline(std::cin, command)) {

if (command.empty()) continue;

std::istringstream iss(command);

int cmd;

iss >> cmd;

if (cmd == 1) {

int A, B;

iss >> A >> B;

int result = PrimeCount(A, B);

std::cout << "Количество простых чисел: " << result << std::endl;

} else if (cmd == 2) {

int x;

iss >> x;

float result = E(x);

std::cout << "Значение числа e: " << result << std::endl;

} else {

std::cout << "Неизвестная команда" << std::endl;

}

}

return 0;

}

unit\_tests.cpp

#include <gtest/gtest.h>

// Объявления функций из библиотек

extern "C" {

int PrimeCount(int A, int B);

float E(int x);

}

TEST(PrimeCountTest, NaiveImplementation) {

EXPECT\_EQ(PrimeCount(1, 10), 4); // Простые числа: 2, 3, 5, 7

EXPECT\_EQ(PrimeCount(10, 20), 4); // Простые числа: 11, 13, 17, 19

EXPECT\_EQ(PrimeCount(1, 1), 0); // Нет простых чисел

}

TEST(ETest, FormulaImplementation) {

EXPECT\_NEAR(E(1), 2.0f, 0.01f);

EXPECT\_NEAR(E(5), 2.48832f, 0.01f);

EXPECT\_NEAR(E(1000), 2.71692f, 0.0001f);

}

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.10)

project(Lab4DynamicLibraries)

# Устанавливаем стандарт C++

set(CMAKE\_CXX\_STANDARD 17)

set(CMAKE\_CXX\_STANDARD\_REQUIRED True)

# Включаем позиционно-независимый код для создания динамических библиотек

set(CMAKE\_POSITION\_INDEPENDENT\_CODE ON)

# Добавляем флаг для генерации отладочной информации (опционально)

#set(CMAKE\_BUILD\_TYPE Debug)

# Добавляем все исходные файлы в переменную SOURCES

set(SOURCES

prime\_naive.cpp

prime\_sieve.cpp

e\_formula.cpp

e\_series.cpp

test\_static.cpp

test\_dynamic.cpp

unit\_tests.cpp

)

# Создаём динамические библиотеки

add\_library(prime\_naive SHARED prime\_naive.cpp)

add\_library(prime\_sieve SHARED prime\_sieve.cpp)

add\_library(e\_formula SHARED e\_formula.cpp)

add\_library(e\_series SHARED e\_series.cpp)

# Устанавливаем имена выходных файлов библиотек без префикса "lib"

set\_target\_properties(prime\_naive PROPERTIES PREFIX "")

set\_target\_properties(prime\_sieve PROPERTIES PREFIX "")

set\_target\_properties(e\_formula PROPERTIES PREFIX "")

set\_target\_properties(e\_series PROPERTIES PREFIX "")

# Добавляем исполняемые файлы

add\_executable(test\_static test\_static.cpp)

add\_executable(test\_dynamic test\_dynamic.cpp)

# Связываем библиотеки с исполняемым файлом test\_static

target\_link\_libraries(test\_static

PRIVATE

prime\_naive

e\_formula

)

# Для test\_dynamic необходимо добавить библиотеку dl (для Unix-подобных систем)

if(UNIX)

target\_link\_libraries(test\_dynamic PRIVATE dl)

endif()

# Настройка Google Test

# Добавляем FetchContent для загрузки Google Test

include(FetchContent)

FetchContent\_Declare(

googletest

URL https://github.com/google/googletest/archive/release-1.12.1.zip

)

# Загружаем и создаём Google Test

FetchContent\_MakeAvailable(googletest)

# Включаем тестирование

enable\_testing()

# Создаём тестовый исполняемый файл

add\_executable(unit\_tests unit\_tests.cpp)

# Связываем тесты с библиотеками и Google Test

target\_link\_libraries(unit\_tests

PRIVATE

gtest\_main

prime\_naive

prime\_sieve

e\_formula

e\_series

)

# Добавляем тесты в CTest

include(GoogleTest)

gtest\_discover\_tests(unit\_tests)

**Пример работы**

(base) vladislavburdinskij@MacBook-Pro-Vladislav build % ./test\_dynamic

2 4000

Значение числа e: 2.71775

1 1

Количество простых чисел: 0

^C

(base) vladislavburdinskij@MacBook-Pro-Vladislav build % **Вывод**

В данной лабораторной я познакомился с тем, как работают динамические и статические библиотеки. В ходе данной ЛР я создал две реализации заданных функций, одну в виде динамических библиотек, а другую в виде статических.