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.