Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

**ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ**

Студент: Сектименко Ирина Владимировна

Группа: М8О–206Б–20

Вариант: 18

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

* Создание динамических библиотек
* Создание программ, которые используют функции динамических библиотек

## Задание

Требуется создать динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов

В конечном итоге, программа должна состоять из следующих частей:

* Динамическая библиотека, реализующая заданных вариантом интерфейс;
* Тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции;
* Тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс.

Провести анализ между обоими типами использования библиотеки.

В моем варианте надо реализовать функцию нахождения количества простых чисел от A до B наивным способом или с помощью решета Эратосфена и функцию вычисления числа e по простой формуле или по формуле Тейлора.

## Общие сведения о программе

Программа со статической библиотекой компилируется из файла main\_static.cpp и использует библиотеку first.h. Программа с динамической библиотекой компилируется из файла main\_dynamic.cpp, в ней можно переключать библиотеки с first на second и обратно. Заголовочный файл first.h представляет собой интерфейс библиотеки, ее реализация находится в файле first.cpp. Вторая библиотека находится в файле second.cpp. В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамический общий объект (общую библиотеку) из файла, имя которого указано в строке filename (завершается null) и возвращает непрозрачный описатель на загруженный объект.
2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти(указывается одним из аргументов).
3. **dlclose** – уменьшает счётчик ссылок на динамически загружаемый общий объект, на который ссылается handle. Если счётчик ссылок достигает нуля, то объект выгружается. Все общие объекты, которые были автоматически загружены при вызове dlopen() для объекта, на который ссылается handle, рекурсивно закрываются таким же способом.

## Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы dlsym, dlopen, dlclose.
2. Написать библиотеку first, для нахождения простых чисел наивных способом и нахождения числа e по простой формуле.
3. Написать библиотеку second, для нахождения простых чисел с помощью решета Эратосфена и нахождения числа e по формуле Тейлора.
4. В файле main\_static.cpp подключить библиотеку на этапе компиляции.
5. В файле main\_dynamic.cpp подключить библиотеку во время исполнения.

## Основные файлы программы

**first.h:**

#pragma once

#include <cmath>

extern "C" {

double E(int x); // formula

}

extern "C" {

int primeCount(int A, int B); // naive algorithm

}

**first.cpp**

#include "first.h"

double E(int x)

{

return pow((1 + 1.0 / x), x);

}

bool isPrime(int x) {

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

{

if (x % i == 0)

{

return false;

}

}

return true;

}

int primeCount(int a, int b)

{

int count = 0;

for (int i = a; i <= b; ++i)

{

if (isPrime(i))

{

++count;

}

}

return count;

}

**second.cpp**

#include <vector>

extern "C" {

int fact(int n)

{

int res = 1;

for (int i = 1; i <= n; ++i) {

res \*= i;

}

return res;

}

}

extern "C" {

double E(int x) // formula Teylora

{

double sum = 1;

for (int i = 1; i <= x; ++i)

{

sum += (1.0 / fact(i));

}

return sum;

}

}

extern "C" {

int sum(std::vector<int> & v)

{

int n = v.size();

int count = 0;

for (int i = 0; i < n; ++i)

{

count += v[i];

}

return count;

}

}

extern "C" {

int primeCount(int a, int b) // sieve of Eratosthenes

{

std::vector<int> numbers(b + 1, 1);

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

{

for (int j = i + 1; j <= b; ++j)

{

if (j % i == 0)

{

numbers[j] = 0;

}

}

}

return sum(numbers) - 1;

}

}

**main\_static.cpp**

#include <iostream>

#include <string>

#include <cmath>

#include <typeinfo>

#include "first.h"

int StringToInt(std::string s)

{

int res = 0;

int n = s.size();

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

{

res += ((s[i] - '0') \* pow(10, n - 1 - i));

}

return res;

}

int main(int argc, char \*argv[])

{

std::string mode = argv[1];

if (mode == "1")

{

int x = StringToInt(std::string(argv[2]));

std::cout << E(x) << std::endl;

} else if (mode == "2")

{

int a = StringToInt(std::string(argv[2])), b = StringToInt(std::string(argv[3]));

std::cout << primeCount(a, b) << std::endl;

} else

{

std::cout << typeid(argv[1]).name() << " nothing\n";

}

return 0;

}

**main\_dynamic.cpp**

#include <iostream>

#include <dlfcn.h>

// g++ -shared -o second.so second.cpp

// g++ -rdynamic -o main main\_dynamic.cpp -ldl

typedef double (\*E)(int x);

typedef int (\*primeCount)(int a, int b);

int main() {

void \*libHandle;

primeCount prime;

libHandle = dlopen("./second.so", RTLD\_LAZY);

int lib;

int r = 1;

while(std::cin >> lib) {

if (lib == 0 && r == 1) {

dlclose(libHandle);

libHandle = dlopen("./first.so", RTLD\_LAZY);

r = 2;

} else if (lib == 0 && r == 2) {

dlclose(libHandle);

libHandle = dlopen("./second.so", RTLD\_LAZY);

r = 1;

} else {

break;

}

}

if (lib == 1) {

E e = (E)dlsym(libHandle, "E");

int x;

std::cin >> x;

double result = e(x);

std::cout << result << std::endl;

} else if (lib == 2) {

primeCount prime = (primeCount)dlsym(libHandle, "primeCount");

int a, b;

std::cin >> a >> b;

int result = prime(a, b);

std::cout << result << std::endl;

}

dlclose(libHandle);

return 0;

}

**Пример работы**

main\_dynamic.cpp

1 10 -> 2.71828

0 1 10 -> 2.59374

2 1 15 -> 7

0 2 1 15 -> 7

main\_static.cpp

1 10 -> 2.59374

0 2 1 15 -> 7

**Вывод**

Самым сложным в лабораторной работе было понять условие. Вторым по сложности – понять, как компилировать динамические библиотеки.

Динамические и статические библиотеки различаются по времени их подключения к программе: динамические – во время исполнения, когда системный вызов к ней обращается, статические – во время компиляции. Кроме того, во время компиляции исполняемый файл получает ссылку, указывающую на динамическую библиотеку в то время, как статические библиотеки добавляются к коду программы во время компиляции, что расходует больше памяти.