Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт № 8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

Студент: Орусский Вячеслав Русланович

Группа: М8О-206Б-21

Вариант: №18

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Содержание**

**[Репозиторий](#_Toc133018239)** [3](#_Toc133018239)

[**Цель работы** 3](#_Toc133018240)

[**Постановка задачи** 3](#_Toc133018241)

[**Общие сведения о программе** 4](#_Toc133018242)

[**Общий метод и алгоритм решения** 5](#_Toc133018243)

[**Исходный код** 6](#_Toc133018244)

[lib.h 6](#_Toc133018245)

[lib\_first.cpp 6](#_Toc133018246)

[Lib\_second.cpp 7](#_Toc133018247)

[main.cpp 8](#_Toc133018248)

[dynamic\_main.cpp 9](#_Toc133018249)

[**Демонстрация работы программы** 11](#_Toc133018250)

[**Выводы** 13](#_Toc133018251)

**Репозиторий**

[Ссылка](https://github.com/RoKivals/OperatingSystems/tree/master/Lab%235/src)

**Цель работы**

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

**Постановка задачи**

Требуется создать динамические библиотеки, которые реализуют определенный 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

В конечном итоге, в лабораторной работе необходимо получить следующие части:

* Динамические библиотеки, реализующие контракты, которые заданы вариантом;
* Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.

*Провести анализ двух типов использования библиотек.*

Пользовательский ввод для обоих программ должен быть организован следующим образом:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
2. «1 arg1 arg2 … argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;
3. «2 arg1 arg2 … argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

**Вариант №18:**

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Общие сведения о программе**

Проект состоит из пяти исходных файлов main.c, dynamic\_main.c, lib.h, first\_lib.c, second\_lib.c. Первые два — это программы, в который тестируются функции из библиотек. Третий файл – интерфейс библиотек. И, наконец, 4 и 5 — это файлы, в которых хранятся реализации функции библиотек. В first\_lib.c хранятся функции для Реализации 1, а в second\_lib.c для Реализации 2.

Собирается программа при помощи команд:

slava@DESKTOP-9JJF73M MINGW64 gcc -fPIC -lm -c first\_lib.c -o first.o

slava@DESKTOP-9JJF73M MINGW64 gcc -fPIC -lm -c second\_lib.c -o second.o

slava@DESKTOP-9JJF73M MINGW64 gcc -shared -o lib1.so first.o

slava@DESKTOP-9JJF73M MINGW64 gcc -shared -o lib2.so second.o

slava@DESKTOP-9JJF73M MINGW64 sudo cp lib1.so /usr/lib

slava@DESKTOP-9JJF73M MINGW64 sudo cp lib2.so /usr/lib

slava@DESKTOP-9JJF73M MINGW64 gcc main.c -l1 -lm -o first\_static

slava@DESKTOP-9JJF73M MINGW64 gcc main.c -l2 -lm -o second\_static

slava@DESKTOP-9JJF73M MINGW64 gcc dynamic\_main.c -ldl -lm -o dynamic

В программе используются такие команды, как:

**dlopen(const char\* filename, int flag)** - загружает динамическую библиотеку, имя которой мы передаем, и возвращает прямой указатель на начало динамической библиотеки.

**dlclose(void\* handle) -** уменьшает на единицу счетчик ссылок на указатель динамической библиотеки handle. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок handle принимает нулевое значение, то динамическая библиотека выгружается.

**dlsym(void\* handle, char\* symbol) -**  использует указатель на динамическую библиотеку, возвращаемую dlopen, и оканчивающееся нулем символьное имя, а затем возвращает адрес, указывающий, откуда загружается этот символ.

**Общий метод и алгоритм решения**

Для начала я реализовал две библиотеки: first\_lib и second\_lib. В первой из них реализовано вычисление экспоненты с помощью 1 замечательного предела и подсчет количества простых чисел на отрезке наивным алгоритмом, а во второй соответственно вычисление экспоненты с помощью ряда Тейлора и подсчет количества простых чисел на отрезке с помощью решета Эратосфена. Далее были реализованы еще две программы: main и dynamic\_main, в которых использовались наши библиотеки, но в первой программе библиотека подключалась на этапе линковке, поэтому для проверки 2 библиотек используются 2 разные программы. А во второй мы использовали динамическую библиотеку, которая использовалась непосредственно во время исполнения программы, причем я также реализовал возможность изменения подключаемой библиотеки, поэтому для проверки двух библиотек необходима лишь одна программа.

**Исходный код**

lib.h

#ifndef OPERATINGSYSTEMS\_LAB\_5\_LIB\_H\_  
#define OPERATINGSYSTEMS\_LAB\_5\_LIB\_H\_  
  
*// Подсчёт кол-ва простых чисел на отрезке  
int* PrimeCount(*int* start, *int* finish);  
  
*// Нахождение значения экспоненты  
float* E(*int* argument);  
  
#endif *//OPERATINGSYSTEMS\_LAB\_5\_LIB\_H\_*

lib\_first.cpp

#include "lib.h"  
#include <cmath>  
#include <bits/stdc++.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
*// Проверяем, является ли число простым  
bool* IsPrime(*int* digit) {  
 *for* (*int* curr\_div(2); curr\_div <= sqrt(digit); ++curr\_div) {  
 *if* (digit % curr\_div == 0) *return false*;  
 }  
 *return true*;  
}  
  
*int* PrimeCount(*int* start, *int* finish) {  
 printf("Count prime digits by native algorithm\n");  
  
 *// Единица - не простое число, скипаем его  
 if* (start == 1) ++start;  
  
 *// Подсчёт результата (кол-ва простых чисел)  
 int* result = 0;  
  
 *for* (; start <= finish; ++start) {  
 *if* (IsPrime(start)) ++result;  
 }  
 *return* result;  
}  
  
*// Вычисление значения экспоненты через 2-ой замечательный предел  
float* E(*int* argument) {  
 printf("Calculating the E by second wonderful limit\n");  
  
 *if* (argument <= 0) *return* -1;  
  
 *float* e = 1.0;  
  
 *for* (*int* index = 0; index < argument; ++index) {  
 e \*= 1 + 1 / *static\_cast*<*float*>(argument);  
 }  
  
 *return* e;  
}

Lib\_second.cpp

#include "lib.h"  
#include <cmath>  
#include <bits/stdc++.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <malloc.h>  
  
  
*int* PrimeCount(*int* start, *int* finish) {  
 printf("Count prime digits by sieve of Eratosthenes\n");  
  
 *int* curent\_number;  
 *int* \*prime\_number\_list;  
 *int* is\_prime;  
 finish = finish - 1;  
 prime\_number\_list = (*int* \*) malloc(finish \* *sizeof*(*int*));  
  
 *for* (*int* index = 0; index < finish; ++index) {  
 prime\_number\_list[index] = index + 2;  
 }  
  
 *for* (*int* index = 0; index < finish; ++index) {  
 curent\_number = prime\_number\_list[index];  
 is\_prime = 0;  
  
 *for* (*int* second\_index = index + 1; second\_index < finish; ++second\_index) {  
 *if* (!(prime\_number\_list[second\_index] % curent\_number)) {  
 *for* (*int* third\_index = second\_index; third\_index < finish - 1; ++third\_index) {  
 prime\_number\_list[third\_index] = prime\_number\_list[third\_index + 1];  
 }  
  
 is\_prime = 1;  
 --finish;  
 --second\_index;  
 }  
 }  
  
 *if* (is\_prime == 0) {  
 *break*;  
 }  
 }  
  
 *int* prime\_count = 0;  
 *for* (*int* index = 0; index < finish; ++index) {  
 *if* (prime\_number\_list[index] >= start) {  
 ++prime\_count;  
 }  
 }  
  
 free(prime\_number\_list);  
  
 *return* prime\_count;  
}  
  
*float* E(*int* argument) {  
 printf("Calculating a E by Taylor series\n");  
  
 *if* (argument <= 0) {  
 *return* -1;  
 }  
  
 *float* e = 1.0;  
 *float* term = 1.0;  
  
 *for* (*int* index = 1; index <= argument; ++index) {  
 term /= *static\_cast*<*float*>(index);  
 e += term;  
 }  
  
 *return* e;  
}

main.cpp

#include "lib.h"  
#include <stdio.h>  
  
*int* main() {  
 *int* command = 0;  
 printf("To compute count of prime numbers on [A, B] enter -- 1.Args: start finish\n");  
 printf("To compute E enter -- 2.Args: argument\n");  
  
 *//std::cin >> command  
 while* (scanf("%d", &command) != EOF) {  
 *switch* (command) {  
 *case* 1: {  
 *// Границы отрезка  
 int* start, finish;  
  
 *if* (scanf("%d %d", &start, &finish) == 2) {  
 printf("Count: %d\n", PrimeCount(start, finish));  
 }  
 *break*;  
 }  
 *case* 2: {  
 *int* argument;  
  
 *if* (scanf("%d", &argument) == 1) {  
 printf("E is: %f\n", E(argument));  
 }  
  
 *break*;  
 }  
 *default*: {  
 printf("This command is not supported, enter 1 or 2\n");  
 }  
 }  
 }  
  
 *return* 0;  
}

dynamic\_main.cpp

#include <stdio.h>  
#include <stdlib.h>  
#include <fcntl.h>  
#include <math.h>  
#include <string>  
#include <dlfcn.h>  
  
*typedef enum* {  
 first\_contract,  
 second\_contract,  
} contracts;  
  
contracts contract = first\_contract;  
  
*const* std::string first\_library\_name = "lib1.so";  
*const* std::string second\_library\_name = "lib2.so";  
  
*int* (\*PrimeCount)(*int*, *int*) = NULL;  
*float* (\*E)(*int*) = NULL;  
  
*void* \*library\_handle = NULL;  
  
*// Загрузка динамической библиотеки  
void* load\_library(contracts contract) {  
 std::string library\_name;  
  
 *switch* (contract) {  
 *case* first\_contract: {  
 library\_name = first\_library\_name;  
 *break*;  
 }  
 *case* second\_contract: {  
 library\_name = second\_library\_name;  
 *break*;  
 }  
 }  
  
 *// Загружаем общий объект  
 // RTLD\_LAZY - флаг, который пропускает неразрешимые символы при загрузке объекта* library\_handle = dlopen(library\_name, RTLD\_LAZY);  
  
 *if* (library\_handle == NULL) {  
 perror("The library was not open");  
 exit(EXIT\_FAILURE);  
 }  
}  
  
*void* load\_contract() {  
 load\_library(contract);  
 PrimeCount = dlsym(library\_handle, "PrimeCount");  
 E = dlsym(library\_handle, "E");  
}  
  
*void* change\_contract() {  
 dlclose(library\_handle);  
  
 *switch* (contract) {  
 *case* first\_contract: {  
 contract = second\_contract;  
 *break*;  
 }  
 *case* second\_contract: {  
 contract = first\_contract;  
 *break*;  
 }  
 }  
  
 load\_contract();  
}  
  
*int* main() {  
 load\_contract();  
 *int* command = 0;  
 printf("To compute count of prime number on [A, B] enter -- 1.Args: start finish\n");  
 printf("To compute E enter -- 2.Args: argument\n");  
 printf("To change contract enter -- 0\n");  
  
 *while* (scanf("%d", &command) != EOF) {  
 *switch* (command) {  
 *case* 0: {  
 change\_contract();  
 printf("Contract has been changed\n");  
  
 *switch* (contract) {  
 *case* first\_contract: {  
 printf("Contract is first\n");  
 *break*;  
 }  
 *case* second\_contract: {  
 printf("Contract is second\n");  
 *break*;  
 }  
 }  
  
 *break*;  
 }  
 *case* 1: {  
 *int* start, finish;  
 *if* (scanf("%d %d", &start, &finish) == 2) {  
 printf("Count: %d\n", PrimeCount(start, finish));  
 }  
  
 *break*;  
 }  
 *case* 2: {  
 *int* argument;  
 *if* (scanf("%d", &argument) == 1) {  
 printf("E is: %f\n", E(argument));  
 }  
  
 *break*;  
 }  
 *default*: {  
 printf("This command is not supported, enter 1 or 2 or 0\n");  
 }  
 }  
 }  
  
 *return* 0;  
}

**Демонстрация работы программы**

**Тест:**

slava@DESKTOP-9JJF73M MINGW64 gcc -fPIC -lm -c first\_lib.c -o first.o

slava@DESKTOP-9JJF73M MINGW64 gcc -fPIC -lm -c second\_lib.c -o second.o

slava@DESKTOP-9JJF73M MINGW64 gcc -shared -o lib1.so first.o

slava@DESKTOP-9JJF73M MINGW64 gcc -shared -o lib2.so second.o

slava@DESKTOP-9JJF73M MINGW64 sudo cp lib1.so /usr/lib

slava@DESKTOP-9JJF73M MINGW64 sudo cp lib2.so /usr/lib

slava@DESKTOP-9JJF73M MINGW64 gcc main.c -l1 -lm -o first\_static

slava@DESKTOP-9JJF73M MINGW64 gcc main.c -l2 -lm -o second\_static

slava@DESKTOP-9JJF73M MINGW64 gcc dynamic\_main.c -ldl -lm -o dynamic

slava@DESKTOP-9JJF73M MINGW64./first\_static

*To compute count of prime number on [A, B] enter -- 1.Args: start finish*

*To compute E enter -- 2.Args: argument*

*1 6 29*

*PrimeCount with naive algoritm*

*Count: 7*

*2 4000*

*Calculating a E with first wonderful limit*

*E is: 2.717744*

slava@DESKTOP-9JJF73M MINGW64./second\_static

*To compute count of prime number on [A, B] enter -- 1.Args: start finish*

*To compute E enter -- 2.Args: argument*

*1 6 29*

*PrimeCount with sieve of Eratosthenes*

*Count: 7*

*2 4000*

*Calculating a E by Taylor series*

*E is: 2.718282*

slava@DESKTOP-9JJF73M MINGW64./dynamic

*To compute count of prime number on [A, B] enter -- 1.Args: start finish*

*To compute E enter -- 2.Args: argument*

*To change contract enter -- 0*

*1 1 20*

*PrimeCount with naive algoritm*

*Count: 8*

*2 4000*

*Calculating a E with first wonderful limit*

*E is: 2.717744*

*0*

*Contract has been changed*

*Contract is second*

*1 1 20*

*PrimeCount with sieve of Eratosthenes*

*Count: 8*

*2 4000*

*Calculating a E by Taylor series*

*E is: 2.718282*

**Выводы**

Проделав лабораторную работу, я научился создавать динамические библиотеки в OC Linux, а также узнал, что их можно использовать двумя способами: во время работы программы и во время компиляции. Вообще библиотеки позволяют использовать разработанный ранее программный код в различных программах. У каждой библиотеки должен быть свой заголовочный файл, в котором должен быть описан ее интерфейс, то есть должны быть объявлены все функции, содержащиеся в библиотеке. Библиотеки бывают двух видов – статические и динамические. Код первых при компиляции полностью входит в состав исполняемого файла, что делает программу легко переносимой. Код динамических библиотек не входит в исполняемый файл, последний содержит лишь ссылку на библиотеку. Если динамическая библиотека будет удалена или перемещена в другое место, то программа работать не будет. С другой стороны, использование динамических библиотек позволяет сократить размер исполняемого файла. Также если в памяти находится две программы, использующие одну и туже динамическую библиотеку, то последняя будет загружена в память лишь единожды. Но при работе с динамическими библиотеками нельзя забывать, что они компилируются особым образом. Они должны содержать так называемый позиционно-независимый код (position independent code). Наличие такого кода позволяет библиотеке подключаться к программе, когда последняя загружается в память. Это связано с тем, что библиотека и программа не являются единой программой, а значит, как угодно, могут располагаться в памяти относительно друг друга.