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: Чирикова П.С.

Группа: М8О-201Б-21

Вариант: №18

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Цель работы
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы
   * + 1. **Цель работы**

Целью является приобретение практических навыков в:

* Создание динамических библиотек
* Создание программ, которые используют функции динамических библиотек
  + - 1. **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

* + - 1. **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.

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.

* + - 1. **Общий метод и алгоритм решения**

Для начала я реализовала 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 реализовала возможность изменения подключаемой библиотеки, поэтому для проверки двух библиотек необходима лишь одна программа.

* + - 1. **Исходный код**

**main.cpp**

*#include <stdio.h>*

*#include <stdlib.h>*

*#include <fcntl.h>*

*#include <math.h>*

*#include <dlfcn.h>*

*typedef enum {*

*first\_contract,*

*second\_contract,*

*} contracts;*

*contracts contract = first\_contract;*

*const char\* first\_library\_name = "lib1.so";*

*const char\* second\_library\_name = "lib2.so";*

*int (\*PrimeCount)(int, int) = NULL;*

*float (\*E)(int) = NULL;*

*void\* library\_handle = NULL;*

*void load\_library(contracts contract) {*

*const char\* library\_name;*

*switch (contract) {*

*case first\_contract: {*

*library\_name = first\_library\_name;*

*break;*

*}*

*case second\_contract: {*

*library\_name = second\_library\_name;*

*break;*

*}*

*}*

*library\_handle = dlopen(library\_name, RTLD\_LAZY);*

*if (library\_handle == NULL) {*

*perror("dlopen error");*

*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;*

*}*

***mathematical-analysis1.cpp***

#include "mathematical-analysis.hpp"#include <iostream>float E(int x) { return pow((1.0 + 1.0 / (float) x), x);}float Derivative(float A, float deltaX) { float deltaY = cos(A + deltaX) - cos(A); return deltaY / deltaX;}

***mathematical-analysis2.cpp***

#include "mathematical-analysis.hpp"#include <iostream>unsigned long long Factorial(int n) { unsigned long long result = 1; while (n > 0) { result \*= n; n--; } return result;}float E(int x) { float result = 0; for (int n = 0; n <= x; n++) { result += 1.0 / ((float)Factorial(n)); } return result;}float Derivative(float A, float deltaX) { float deltaY = cos(A + deltaX) - cos(A - deltaX); return deltaY / 2.0 / deltaX;}

* + - 1. **Демонстрация работы программы**

**Тест:**

polina@polina-Vostro-3400::~/OS/cmake-build-debug/lab5$ ./mathematical-analysis1.cpp

*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*

polina@polina-Vostro-3400::~/OS/cmake-build-debug/lab5$ ./mathematical-analysis2.cpp

*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*

**7. Выводы**

Проделав лабораторную работу, я научилась создавать динамические библиотеки в OC Linux, а также узнала, 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.