Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

Студент: Садаков А.А.

Группа: М8О–206Б–19

Вариант: 9

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

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

**Задание**

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 6:

Функция 1: Рассчет производной функции cos(x) в точке A с приращением delta. Реализация1: f'(x) = (f(A + deltaX) Реализация2: f'(x) = (f(A + deltaX)

Функция 2: Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные). Реализация1: наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа. Реализация2: решето Эратосфена.

**Общие сведения о программе**

Программа компилируется при помощи CMake в 2 исполняемых файла mainDyn, mainStat и 2 библиотеки lib1.so, lib2.so. В первом случае мы используем библиотеку, которая использует знания полученные во время компиляции (на этапе линковки). Во втором случае программа загружает библиотеки и взаимодействует с ними при помощи следующих системных вызовов:

1. **dlopen** – загружает динамическую библиотеку, имя которой указано первым аргументом, и возвращает прямой указатель на начало динамической библиотеки. Второй аргумент отвечает за разрешение неопределенных символов, возвращает 0 при успешном завершении и значение != 0 в случае ошибки.
2. **exit** – завершение работы программы с кодом, указанным в качестве аргумента.
3. **dlsym** – использует указатель на динамическую библиотеку – первый аргумент, возвращаемую dlopen, и оканчивающееся нулем символьное имя – второй аргумент, а затем возвращает адрес, указывающий, откуда загружается этот символ. Если символ не найден, то возвращаемым значением dlsym является NULL.
4. **dlclose –** уменьшает на единицу счетчик ссылок на указатель динамической библиотеки, передаваемый в качестве аргумента. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается.

**Общий метод и алгоритм решения**.

Создаем по два исполняемых файла и два heder-a с реализациями и объявлениями для каждой из двух функций, собираем из них библиотеку и используем 2-мя способами:

* + - 1. на этапе компиляции (стадия линковки) при помощи #include в программе mainStat.
      2. при помощи загрузки библиотек при помощи dlopen в программе mainDyn.

**Основные файлы программы**

**CMakeLists.txt:**

**cmake\_minimum\_required(VERSION 3.0.2)**

**project(lab5 VERSION 1.0 LANGUAGES C)**

**set(STATIC\_EXE mainStat)**

**set(SHARED\_EXE mainDyn)**

**set(SOURCE\_STAT\_EXE mainStat.c)**

**set(SOURCE\_SHAR\_EXE mainDyn.c)**

**add\_library(1S STATIC lib1.h lib1.c)**

**add\_library(1D SHARED lib1.h lib1.c)**

**add\_library(2S STATIC lib2.h lib2.c)**

**add\_library(2D SHARED lib2.h lib2.c)**

**target\_link\_libraries(1D PRIVATE m)**

**target\_link\_libraries(2D PRIVATE m)**

**target\_link\_libraries(1S PRIVATE m)**

**target\_link\_libraries(2S PRIVATE m)**

**add\_executable(${STATIC\_EXE} ${SOURCE\_STAT\_EXE} lib2.h lib2.c)**

**add\_executable(${SHARED\_EXE} ${SOURCE\_SHAR\_EXE})**

**target\_link\_libraries(${STATIC\_EXE} ${LIB2S} m 2S)**

**target\_link\_libraries(${SHARED\_EXE} PRIVATE m dl)**

**add\_definitions(-std=c99 -Wextra -Wall -Werror -Wno-sign-compare -Wno-unused-result -lm -ldl)**

**mainDyn.c:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <stdbool.h>**

**#include <math.h>**

**#include <dlfcn.h>**

**int main() {**

**float arg1, arg2;**

**int arg3, arg4;**

**void\* handle = NULL;**

**float (\*Derivative)(float,float);**

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

**handle = dlopen("./lib2D.so", RTLD\_LAZY);**

**if (handle == NULL) {**

**printf("OPEN LIBRARY ERROR\n");**

**exit(-1);**

**}**

**Derivative = dlsym(handle, "Derivative");**

**PrimeCount = dlsym(handle, "PrimeCount");**

**int cmd = 0, libNum = 0;**

**printf("Enter command:\n");**

**printf("0 change implementation\n");**

**printf("1 Derivative(float, float)\n");**

**printf("2 PrimeCount(int, int)\n");**

**printf("3 exit\n\n");**

**while (cmd != 3) {**

**if(scanf("%d", &cmd) != 1) {**

**printf("SCANF ERROR\n");**

**exit(-1);**

**}**

**switch (cmd) {**

**case 0:**

**dlclose(handle);**

**if (libNum == 0) {**

**if ((handle = dlopen("./lib1D.so", RTLD\_LAZY)) == 0) {**

**printf("OPEN LIBRARY ERROR\n");**

**exit(-1);**

**}**

**libNum = 1;**

**} else {**

**if ((handle = dlopen("./lib2D.so", RTLD\_LAZY)) == 0) {**

**printf("OPEN LIBRARY ERROR\n");**

**exit(-1);**

**}**

**libNum = 0;**

**}**

**Derivative = dlsym(handle, "Derivative");**

**PrimeCount = dlsym(handle, "PrimeCount");**

**break;**

**case 1:**

**if(scanf("%f %f", &arg1, &arg2) != 2) {**

**printf("SCANF ERROR\n");**

**exit(-1);**

**}**

**printf("%f\n", (\*Derivative)(arg1, arg2));**

**break;**

**case 2:**

**if(scanf("%d %d", &arg3, &arg4) != 2) {**

**printf("SCANF ERROR\n");**

**exit(-1);**

**}**

**printf("%d\n", (\*PrimeCount)(arg3, arg4));**

**break;**

**case 3:**

**{break;}**

**default:**

**fprintf(stderr, "incorrect input, try again\n");**

**}**

**}**

**dlclose(handle);**

**return 0;**

**}**

**mainStat.c:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <stdbool.h>**

**#include <math.h>**

**#include "lib2.h"**

**int main() {**

**int arg1, arg2;**

**float arg3, arg4;**

**int cmd = 0;**

**printf("Enter command:\n");**

**printf("1 Derivative(float, float)\n");**

**printf("2 PrimeCount(int, int)\n");**

**printf("3 exit\n\n");**

**while (cmd != 3) {**

**if(scanf("%d", &cmd) != 1) {**

**printf("SCANF ERROR\n");**

**exit(-1);**

**}**

**switch (cmd) {**

**case 1:**

**if(scanf("%f %f", &arg3, &arg4) != 2) {**

**printf("SCANF ERROR\n");**

**exit(-1);**

**}**

**printf("%f\n", Derivative(arg3, arg4));**

**break;**

**case 2:**

**if(scanf("%d %d", &arg1, &arg2) != 2) {**

**printf("SCANF ERROR\n");**

**exit(-1);**

**}**

**printf("%d\n", PrimeCount(arg1, arg2));**

**break;**

**case 3:**

**{break;}**

**default:**

**fprintf(stderr, "incorrect input, try again\n");**

**}**

**}**

**return 0;**

**}**

**lib1.h:**

**#ifndef LIB1\_H**

**#define LIB1\_H**

**#include <stdio.h>**

**#include <math.h>**

**float Derivative (float A, float deltaX);**

**int PrimeCount (int A, int B);**

**#endif**

**lib2.h:**

**#ifndef LIB2\_H**

**#define LIB2\_H**

**#include <stdio.h>**

**#include <math.h>**

**#include <stdbool.h>**

**float Derivative (float A, float deltaX);**

**int PrimeCount (int A, int B);**

**#endif**

**lib1.c:**

**#include "lib1.h"**

**float Derivative (float A, float deltaX) {**

**printf("Реализация функции Derivative из первой библиотеки\n");**

**return (cos(A + deltaX) - cos(A)) / deltaX;**

**}**

**int PrimeCount (int A, int B) {**

**printf("Реализация функции PrimeCount из первой библиотеки\n");**

**int ans = 0;**

**if (A == 1) {**

**++A;**

**}**

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

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

**if (i % j == 0) {**

**--ans;**

**break;**

**}**

**}**

**++ans;**

**}**

**return ans;**

**}**

**lib2.c:**

**#include "lib2.h"**

**float Derivative (float A, float deltaX) {**

**printf("Реализация функции Derivative из второй библиотеки\n");**

**return (cos(A + deltaX) - cos(A - deltaX)) / (2 \* deltaX);**

**}**

**int PrimeCount (int A, int B) {**

**printf("Реализация функции PrimeCount из второй библиотеки\n");**

**if (A == 1) {**

**++A;**

**}**

**bool ERA[B + 1];**

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

**ERA[i] = true;**

**}**

**int ans = 0;**

**int P = 2;**

**while (P \* P <= B) {**

**int i = P \* P;**

**if (ERA[i]) {**

**while (i <= B) {**

**ERA[i] = false;**

**i += P;**

**}**

**}**

**P++;**

**}**

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

**if (ERA[i]) {**

**ans++;**

**}**

**}**

**return ans;**

**}**

**Пример работы**

**aleksandr201@LAPTOP-M1VJISD6:~/OS/lab5$ ./mainDyn**

**Enter command:**

**0 change implementation**

**1 Derivative(float, float)**

**2 PrimeCount(int, int)**

**3 exit**

**0**

**1**

**1.0 0.5**

**Реализация функции Derivative из первой библиотеки**

**-0.939130**

**0**

**1**

**1.0 0.5**

**Реализация функции Derivative из второй библиотеки**

**-0.806845**

**3**

**Вывод**

Одним из преимуществ динамических библиотек является то, что разные программы могут совместно использовать одну копию динамической библиотеки, что значительно экономит используемое пространство. Еще одним преимуществом динамической библиотеки является то, что её можно обновить ​​до более новой версии без необходимости перекомпиляции всех исполняемых файлов, которые её используют.