Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

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

Студент: Попов Илья Павлович

Группа: М80-206Б-20

Вариант: 2

Преподаватель: Соколов Андрей Алексеевич

Дата: 11.12.2021

Оценка: 5

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

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 №2.

Контракты и реализации функций

1 Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e Float SinIntegral(float A, float B, float e)

Подсчет интеграла методом прямоугольников.

Подсчет интеграла методом трапеций.

3 Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные) Int PrimeCount(int A, int B)

Наивный алгоритм.

Проверить делимость текущего числа на все предыдущие числа. Решето Эратосфена

Листинг программы

functions.h

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

float SinIntegral(float A, float B, float e);

int PrimeCount(int a, int b);

#endif

lib1.c

#include <math.h>

#include <stdbool.h>

#include <stdio.h>

//Подсчет интеграла методом прямоугольников.

float SinIntegral(float A, float B, float e) {

float dx = (B - A) / e;

int steps = (B - A) / dx;

float cur = A;

float res = 0;

for (int i = 0; i < steps; ++i){

res += dx \* sin(cur + dx / 2);

cur += dx;

}

res += (B - cur) \* sin((B + cur) / 2);

return res;

}

//Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.

int PrimeCount(int a, int b){

int count = 0;

bool find\_div = false;

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

if (i == 0 || i == 1){

find\_div = true;

}

for(int j = i - 1; j > 1; --j){

if(i % j == 0){

find\_div = true;

break;

}

}

if(!find\_div) ++count;

find\_div = false;

}

return count;

}

lib2.c

#include <math.h>

//Подсчет интеграла методом трапеций.

float SinIntegral(float A, float B, float e) {

float dx = (B - A) / e;

int steps = (B - A) / dx;

float cur = A;

float res = 0;

for (int i = 0; i < steps; ++i){

res += dx \* sin(sin(cur) + sin(cur + dx)) / 2;

cur += dx;

}

res += (B - cur) \* (sin(B) + sin(cur)) / 2;

return res;

}

//Решето Эратосфена

int PrimeCount(int a, int b){

int count = 0;

int sieve[b + 1];

for(int i = 0; i < b + 1; ++i){

sieve[i] = 0;

}

sieve[0] = 1;

sieve[1] = 1;

for(int i = 2; i <= b; ++i){

if(sieve[i] != 0){

continue;

}

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

sieve[j] = 1;

}

++count;

}

return count;

}

makefile

ADRES="/home/lunidep/Desktop/OS/lab5"

done: prog1 prog2

lib1.so: lib1.c

gcc -shared lib1.c -o lib1.so -lm -Wall

lib2.so: lib2.c

gcc -shared lib2.c -o lib2.so -lm -Wall

prog2: lib1.so lib2.so prog2.c

gcc prog2.c -ldl -o prog2 -Wall

prog1: lib1.so prog1.c

gcc prog1.c -L$(ADRES) -Wl,-R. -l1 -o prog1 -Wall

prog1.c

#include <stdio.h>

#include "functions.h"

void usage(){

printf("1. Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e\n");

printf("USAGE: Float SinIntegral(float A, float B, float e)\n\n");

printf("2. Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные)\n");

printf("USAGE: Int PrimeCount(int A, int B)\n\n");

}

int main(){

usage();

int command;

while(scanf("%d", &command) != EOF){

switch(command){

case 1:{

float a, b, e;

if(scanf("%f%f%f", &a, &b, &e) != 3){

printf("Wrong arguements!\n");

continue;

}

printf("%f\n", SinIntegral(a, b, e));

break;

}

case 2:{

int a, b;

if(scanf("%d%d", &a, &b) != 2){

printf("Wrong arguements!\n");

continue;

}

printf("%d\n", PrimeCount(a, b));

break;

}

default:{

printf("Wrong command!\n");

}

}

}

}

prog2.c

#include <stdio.h>

#include <dlfcn.h>

void usage(){

printf("0. Переключение реализации контрактов\n\n");

printf("1. Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e\n");

printf("USAGE: Float SinIntegral(float A, float B, float e)\n\n");

printf("2. Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные)\n");

printf("USAGE: Int PrimeCount(int A, int B)\n\n");

}

int main(){

usage();

int command;

int version = 0;

float (\*SinIntegral)(float, float, float);

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

void \*lib1\_handler = dlopen("./lib1.so", RTLD\_LAZY);

void \*lib2\_handler = dlopen("./lib2.so", RTLD\_LAZY);

if (!lib1\_handler || !lib2\_handler){

fprintf(stderr,"dlopen() error: %s\n", dlerror());

return -1;

}

SinIntegral = dlsym(lib1\_handler,"SinIntegral");

PrimeCount = dlsym(lib1\_handler,"PrimeCount");

while(scanf("%d", &command) != EOF){

switch(command){

case 0:{

version ^= 1;

if(!version){

SinIntegral = dlsym(lib1\_handler,"SinIntegral");

PrimeCount = dlsym(lib1\_handler,"PrimeCount");

} else{

SinIntegral = dlsym(lib2\_handler,"SinIntegral");

PrimeCount = dlsym(lib2\_handler,"PrimeCount");

}

printf("Switched to realization %d\n", version + 1);

break;

}

case 1:{

float a, b, e;

if(scanf("%f%f%f", &a, &b, &e) != 3){

printf("Wrong arguements!\n");

continue;

}

printf("%f\n", SinIntegral(a, b, e));

break;

}

case 2:{

int a, b;

if(scanf("%d%d", &a, &b) != 2){

printf("Wrong arguements!\n");

continue;

}

printf("%d\n", PrimeCount(a, b));

break;

}

default:{

printf("Wrong command!\n");

}

}

}

dlclose(lib1\_handler);

dlclose(lib2\_handler);

}

Примеры работы

**Тест №1**

lunidep@lunidep-VirtualBox:~/Desktop/OS/lab5$ ./static

1. Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e

USAGE: Float SinIntegral(float A, float B, float e)

2. Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные)

USAGE: Int PrimeCount(int A, int B)

1 0 1 0.1

0.479426

2 1 10000

1229

lunidep@lunidep-VirtualBox:~/Desktop/OS/lab5$ ./dynamic

0. Переключение реализации контрактов

1. Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e

USAGE: Float SinIntegral(float A, float B, float e)

2. Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные)

USAGE: Int PrimeCount(int A, int B)

1 0 1 0.1

0.479426

2 1 10000

1229

0

Switched to realization 2

1 0 1 0.1

0.420735

2 1 10000

1229

0

Switched to realization 1

Вывод

В ходе выполнения данной лабораторной работы я познакомился с тем, как создавать и использовать динамические библиотеки.

Динамические библиотеки используются во всех крупных проектах, чтобы при внесении изменений надо было перекомпилировать только одну библиотеку, а не весь проект. Также они удобны тем, что достаточно один раз выгрузить динамическую библиотеку в память и ей смогут пользоваться все нуждающиеся программы.