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: Гордовой Д.С.

Группа: М8о–201Б–19

Вариант: 5

Преподаватель: Миронов Евгений Сергеевич

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 5:

Функция 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.

Функция 6: Расчет значения числа е (основание натурального логарифма). float E(int x). (1 + 1/x) ^ x. Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!)).

**Общие сведения о программе**

Скомпилируем прорамму с Makefile. У нас выйдет два исполняемых файла static и dynamic. 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 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. 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.

3. 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.

**Листинг программы**

**Makefile:**all: static dynamic

rm -f \*.o

static: static.o libimp0.so

gcc -o static static.o -L. -limp0 -lm -Wl,-rpath,.

static.o: ./src/static.c

gcc -c ./src/static.c -lm

libimp0.so: sinintegral.o e.o

gcc -shared -o libimp0.so sinintegral.o e.o -lm

sinintegral.o: ./src/lib/imp\_0/src/sinintegral.c

gcc -c -fPIC ./src/lib/imp\_0/src/sinintegral.c -lm

e.o: ./src/lib/imp\_0/src/e.c

gcc -c -fPIC ./src/lib/imp\_0/src/e.c -lm

dynamic: dynamic.o libimp0.so libimp1.so

gcc -o dynamic dynamic.o -L. -limp0 -limp1 -lm -ldl -Wl,-rpath,.

dynamic.o: ./src/dynamic.c

gcc -c ./src/dynamic.c -lm

libimp1.so: sinintegral1.o e1.o

gcc -shared -o libimp1.so sinintegral1.o e1.o -lm

sinintegral1.o: ./src/lib/imp\_1/src/sinintegral.c

gcc -c -fPIC ./src/lib/imp\_1/src/sinintegral.c -o sinintegral1.o -lm

e1.o: ./src/lib/imp\_1/src/e.c

gcc -c -fPIC ./src/lib/imp\_1/src/e.c -o e1.o -lm

clean:

rm -f \*.o \*.so static dynamic

**src/static.c:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include "./lib/imp\_0/sinintegral.h"

#include "./lib/imp\_0/e.h"

int main()

{

char cmd = '0';

int x;

float a = 0.0;

float b = 0.0;

float c = 0.0;

printf("Enter command:\n");

printf("1 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float\n\n");

printf("2 arg1 -> return value - float; arg1 - int\n");

cmd = getchar();

if(cmd == '1')

{

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

{

printf("INVALID INPUT\n");

exit(-1);

}

printf("integral = %f\n", sinintegral(a, b, c));

}

else if (cmd == '2')

{

if(scanf("%d", &x) != 1)

{

printf("INVALID INPUT\n");

exit(-1);

}

printf("E = %f\n", E(x));

}

else

{

printf("INVALID COMMAND\n");

exit(-1);

}

}

**src/dynamic.c:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <math.h>

#include <dlfcn.h>

int main()

{

char cmd = 'a';

char changer = '0';

int x;

float a = 0.0;

float b = 0.0;

float c = 0.0;

void\* library\_handler\_0 = NULL;

void\* library\_handler\_1 = NULL;

float (\*sinintfunc)(float,float,float);

float (\*efunc)(float);

if((library\_handler\_0 = dlopen("libimp0.so", RTLD\_LAZY)) == 0)

{

printf("OPEN LIBRARY ERROR\n");

exit(-1);

}

if((library\_handler\_1 = dlopen("libimp1.so", RTLD\_LAZY)) == 0)

{

printf("OPEN LIBRARY ERROR\n");

exit(-1);

}

printf("Enter command:\n");

printf("0 -> change implementation\n");

printf("1 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float\n");

printf("2 arg1 -> return value - float; arg1 - int\n");

printf("3 -> exit\n\n");

while(cmd != '3')

{

if(scanf("%c", &cmd) != 1)

{

printf("SCANF ERROR\n");

exit(-1);

}

if(cmd == '0')

{

if(changer == '0') { changer = '1'; }

else { changer = '0'; }

}

else if(cmd == '1')

{

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

{

printf("INVALID INPUT\n");

exit(-1);

}

if(changer == '0') { sinintfunc = dlsym(library\_handler\_0, "sinintegral"); }

else { sinintfunc = dlsym(library\_handler\_1, "sinintegral"); }

printf("integral = %f\n", (\*sinintfunc)(a, b, c));

}

else if (cmd == '2')

{

if(scanf("%d", &x) != 1)

{

printf("INVALID INPUT\n");

exit(-1);

}

if(changer == '0') { efunc = dlsym(library\_handler\_0, "e"); }

else { efunc = dlsym(library\_handler\_1, "e"); }

printf("e = %f\n", (\*efunc)(x));

}

}

dlclose(library\_handler\_0);

dlclose(library\_handler\_1);

return 0;

}

**/src/lib/imp\_0/sinintegral .h**#ifndef \_\_SININTEGRAL\_\_

#define \_\_SININTEGRAL\_\_

float sinintegral(float, float, float);

#endif

**/src/lib/imp\_0/src/sinintegral.c:**

#include <stdio.h>

#include <math.h>

#include "../sinintegral.h"

float sinintegral(float a, float b, float e)

{ // а-нпи b-впи e-шаг

int n = (b-a)/e; //шаг (разбиение отрезка на n равных отрезков)

float integral = 0.0;

float c = 0.0;

if (n < 0) { n = -n; }

if(a < b) { c = a; }

else { c = b; }

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

{

integral += sinf((2.0\*c+e)/2.0)\*e;

c += e;

}

printf("1 реализация методом прямоугольников\n");

return integral;

}

**/src/lib/imp\_0/e.h**#ifndef \_\_E\_\_

#define \_\_E\_\_

float E(int);

#endif **/src/lib/imp\_0/src/e.c:**

#include <stdio.h>

#include <math.h>

#include "../e.h"

float E(int x) {

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

}

**/src/lib/imp\_1/sinintegral .h**#include <stdio.h>

#include <math.h>

#include "../sinintegral.h"

float sinintegral(float a, float b, float e)

{ // а-нпи b-впи e-шаг

int n = (b-a)/e; //шаг (разбиение отрезка на n равных отрезков)

float integral = 0.0;

float c = 0.0;

if (n < 0) { n = -n; }

if(a < b) { c = a; }

else { c = b; }

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

{

integral += (sinf(c)+sinf(c+e))/2.0\*e;

c += e;

}

printf("2 реализация методом трапеции\n");

return integral;

}

}

**/src/lib/imp\_1/e.h**#ifndef \_\_E1\_\_

#define \_\_E1\_\_

float E(int);

#endif **/src/lib/imp\_1/src/e.c:**

#include <stdio.h>

#include "../e.h"

int factorial(int x) {

int result = 1;

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

result \*= i;

}

return result;

}

float E(int x) {

float result = 0;

for (int n = 0; n <= x; ++n) {

result += 1.0 / factorial(n);

}

return result;

}

**Пример работы**

kitanosa@kitanosa:~/labs/os/lab5$ ./static

Enter command:

1 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float

2 arg1 -> return value - float; arg1 - int

1 1 3 0.1

1 реализация методом прямоугольников

integral = 1.530933

kitanosa@kitanosa:~/labs/os/lab5$ ./static

Enter command:

1 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float

2 arg1 -> return value - float; arg1 - int

2 7

1 реализация x = 7, result = 2.546500

E = 2.546500

kitanosa@kitanosa:~/labs/os/lab5$ ./static

Enter command:

1 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float

2 arg1 -> return value - float; arg1 - int

2 100

1 реализация x = 100, result = 2.704811

E = 2.704811

kitanosa@kitanosa:~/labs/os/lab5$ ./static

Enter command:

1 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float

2 arg1 -> return value - float; arg1 - int

2 1000

1 реализация x = 1000, result = 2.717051

E = 2.717051

kitanosa@kitanosa:~/labs/os/lab5$ ./dynamic

Enter command:

0 -> change implementation

1 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float

2 arg1 -> return value - float; arg1 - int

3 -> exit

1 1 3 0.1

1 реализация методом прямоугольников

integral = 1.530933

0

2 100

2 реализация x = 100, result = inf

e = inf

3

**Вывод**

Выполняя лабораторную работу я изучил основы работы с динамическими библиотеками на операционной системе Linux. Динамические библиотеке позволяют более легко работать с имполняемыми файлами, уменьшая их [вес.Использование](http://вес.Использование) библиотек добавляет модульность программе, упрощая тем самым поддежку кода в дальнейшем. Загрузка динамических библиотек во время выполнения также упрощает компиляцию