Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

Студент: Шубин Г. С.

Группа: М8О–308Б–19

Вариант: 8

Преподаватель: Миронов Е. С.

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2021.

**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 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 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.

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 2: Подсчет площади плоской геометрической фигуры по двум сторонам. float Square(float A, float B). Фигура прямоугольник. Фигура прямоугольный треугольник.

**Общие сведения о программе**

Программа компилируется при помощи Makefile в 2 исполняемых файла main\_01, main\_02 и 2 библиотеки libimp0.so, libimp1.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 способами:

1. на этапе компиляции (стадия линковки) при помощи #include в программе main\_0.
2. при помощи загрузки библиотек при помощи dlopen в программе main\_1.

**Листинг программы**

**Makefile:**

all: main\_0 main\_1

    rm -f \*.o

main\_0: main\_0.o libimp0.so

    gcc -o main\_0 main\_0.o -L. -limp0 -lm -Wl,-rpath,.

main\_0.o: ./src/main\_0.c

    gcc -c ./src/main\_0.c -lm

libimp0.so: sinintegral.o square.o

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

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

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

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

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

main\_1: main\_1.o libimp0.so libimp1.so

    gcc -o main\_1 main\_1.o -L. -limp0 -limp1 -lm -ldl -Wl,-rpath,.

main\_1.o: ./src/main\_1.c

    gcc -c ./src/main\_1.c -lm

libimp1.so: sinintegral1.o square1.o

    gcc -shared -o libimp1.so sinintegral1.o square1.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

square1.o: ./src/lib/imp\_1/src/square.c

    gcc -c -fPIC ./src/lib/imp\_1/src/square.c -o square1.o -lm

clean:

    rm -f \*.o \*.so main\_0 main\_1

**/src/main\_0.c:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include "./lib/imp\_0/sinintegral.h"

#include "./lib/imp\_0/square.h"

int main()

{

    char cmd = '0';

    float a = 0.0;

    float b = 0.0;

    float e = 0.0;

    printf("Enter command:\n");

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

    printf("2 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float\n\n");

    cmd = getchar();

    if (cmd == '1')

    {

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

        {

            printf("SCANF ERROR\n");

            exit(-1);

        }

        printf("square = %f\n", square(a, b));

    }

    else if(cmd == '2')

    {

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

        {

            printf("SCANF ERROR\n");

            exit(-1);

        }

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

    }

    else

    {

        printf("INVALID COMMAND\n");

        exit(-1);

    }

}

**/src/main\_1.c:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <math.h>

#include <dlfcn.h>

int main()

{

    char cmd = 'a';

    char changer = '0';

    float a = 0.0;

    float b = 0.0;

    float e = 0.0;

    void\* library\_handler\_0 = NULL;

    void\* library\_handler\_1 = NULL;

    float (\*squarefunc)(float,float);

    float (\*sinintfunc)(float,float,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 -> return value - float; arg1,arg2 - float\n");

    printf("2 arg1 arg2 arg3 -> return value - float; arg1,arg2,arg3 - float\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", &a, &b) != 2)

            {

                printf("SCANF ERROR\n");

                exit(-1);

            }

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

            else { squarefunc = dlsym(library\_handler\_1, "square"); }

            printf("square = %f\n", (\*squarefunc)(a, b));

        }

        else if(cmd == '2')

        {

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

            {

                printf("SCANF ERROR\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, e));

        }

    }

    dlclose(library\_handler\_0);

    dlclose(library\_handler\_1);

    return 0;

}

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

#include <stdio.h>

#include <math.h>

#include "../sinintegral.h"

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

{

    int n = (b-a)/e;

    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;

    }

    return integral;

}

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

#include <stdio.h>

#include "../square.h"

float square(float A, float B) { return A\*B; }

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

#ifndef \_\_SININTEGRAL\_\_

#define \_\_SININTEGRAL\_\_

float sinintegral(float, float, float);

#endif

**/src/lib/imp\_0/square.h:**

#ifndef \_\_SQUARE\_\_

#define \_\_SQUARE\_\_

float square(float, float);

#endif

**/src/lib/imp\_1/src/sinintegral.c:**

#include <stdio.h>

#include <math.h>

#include "../sinintegral.h"

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

{

    int n = (b-a)/e;

    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;

    }

    return integral;

}

**/src/lib/imp\_1/src/square.c:**

#include <stdio.h>

#include "../square.h"

float square(float A, float B) { return 0.5\*A\*B; }

**/src/lib/imp\_1/sinintegral.h:**

#ifndef \_\_SININTEGRAL1\_\_

#define \_\_SININTEGRAL1\_\_

float sinintegral(float, float, float);

#endif

**/src/lib/imp\_1/square.h:**

#ifndef \_\_SQUARE1\_\_

#define \_\_SQUARE1\_\_

float square(float, float);

#endif

**Пример работы**

gregory@DESKTOP-7L8SUG4:~/labsOS/lab5$ make

gcc -c ./src/main\_0.c -lm

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

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

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

gcc -o main\_0 main\_0.o -L. -limp0 -lm -Wl,-rpath,.

gcc -c ./src/main\_1.c -lm

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

gcc -c -fPIC ./src/lib/imp\_1/src/square.c -o square1.o -lm

gcc -shared -o libimp1.so sinintegral1.o square1.o -lm

gcc -o main\_1 main\_1.o -L. -limp0 -limp1 -lm -ldl -Wl,-rpath,.

rm -f \*.o

gregory@DESKTOP-7L8SUG4:~/labsOS/lab5$ ./main\_0

Enter command:

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

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

1 2.0 4.0

square = 8.000000

gregory@DESKTOP-7L8SUG4:~/labsOS/lab5$ ./main\_0

Enter command:

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

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

2 1.0 3.0 0.1

integral = 1.530933

gregory@DESKTOP-7L8SUG4:~/labsOS/lab5$ ./main\_1

Enter command:

0 -> change implementation

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

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

3 -> exit

1 2.0 3.0

square = 6.000000

0

1 2.0 3.0

square = 3.000000

2 1.0 3.0 0.1

integral = 1.529019

0

2 1.0 3.0 0.1

integral = 1.530933

3

**Вывод**

На СИ можно удобно писать статические и динамические библиотеки, причем существует несколько механизмов работы с ними, используя знания полученные во время компиляции (этап линковки) или при помощи загрузки библиотек при помощи их местоположения и контракта. При помощи библиотек мы можем писать более сложные вещи, которые используют простые функции, структуры и т.п., написанные ранее и сохраненные в различных библиотеках.