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: Епанешников В. С.

Группа: М8О–206Б–19

Вариант: 31

Преподаватель: Соколов Андрей Алексеевич

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-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 31:

Функция 1: Рассчет значения числа е(основание натурального логарифма).Float E(int x)). Подсчет формулой и суммой ряда.

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

**Общие сведения о программе**

Программа компилируется при помощи Makefile в 2 исполняемых файла main1, main2 и 2 библиотеки liblab1.so, liblab2.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 header для них 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 main\_0.
      2. при помощи загрузки библиотек при помощи dlopen в программе main\_1.

**Основные файлы программы**

**Makefile:**

**# Makefile for lab work 5**

**CC = gcc**

**CFLAGS = -std=c99**

**LDFLAGS = -L.**

**CLIBFLAGS = -c -fPIC**

**LDLIBFLAGS = -shared**

**EXECUTABLE1 = task1**

**EXECUTABLE2 = task2**

**SRC1 = main1.c**

**SRC2 = main2.c**

**OBJ1 = $(SRC1:.c=.o)**

**OBJ2 = $(SRC2:.c=.o)**

**SRCLIB1 = ./libs/liblab1.c**

**ND\_SRCLIB1 = $(notdir $(SRCLIB1))**

**OBJLIB1 = $(ND\_SRCLIB1:.c=.o)**

**SOLIB1 = $(ND\_SRCLIB1:.c=.so)**

**SRCLIB2 = ./libs/liblab2.c**

**ND\_SRCLIB2 = $(notdir $(SRCLIB2))**

**OBJLIB2 = $(ND\_SRCLIB2:.c=.o)**

**SOLIB2 = $(ND\_SRCLIB2:.c=.so)**

**all: $(SOLIB1) $(SOLIB2) $(EXECUTABLE1) $(EXECUTABLE2)**

**$(EXECUTABLE1): $(OBJ1)**

**$(CC) $(LDFLAGS) $< $(SOLIB1) -o $@**

**$(EXECUTABLE2): $(OBJ2)**

**$(CC) $< -o $@**

**$(SOLIB1): $(OBJLIB1)**

**$(CC) $(LDLIBFLAGS) $< -o $@**

**$(OBJLIB1): $(SRCLIB1)**

**$(CC) $(CLIBFLAGS) $<**

**$(SOLIB2): $(OBJLIB2)**

**$(CC) $(LDLIBFLAGS) $< -o $@**

**$(OBJLIB2): $(SRCLIB2)**

**$(CC) $(CLIBFLAGS) $<**

**clean:**

**rm \*.o $(SOLIB1) $(SOLIB2) $(EXECUTABLE1) $(EXECUTABLE2)**

**main1.c:**

**#include <stdio.h>**

**#include "./libs/libs.h"**

**int main() {**

**int cmd = 0;**

**int x = 0;**

**float a = 0.f, b = 0.f;**

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

**switch(cmd) {**

**case 0:**

**printf("This is not program 2\n");**

**break;**

**case 1:**

**scanf("%d", &x);**

**printf("Result: %f\n", e(x));**

**break;**

**case 2:**

**scanf("%f %f", &a, &b);**

**printf("Result: %f\n", square(a, b));**

**break;**

**}**

**}**

**return 0;**

**}**

**main2.c:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <dlfcn.h>**

**#define LIBNAME\_FIRST "liblab1.so"**

**#define LIBNAME\_SECOND "liblab2.so"**

**int main() {**

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

**float (\*square) (float, float) = NULL;**

**void \*handle1 = dlopen(LIBNAME\_FIRST, RTLD\_LAZY);**

**if (!handle1) {**

**printf("%s\n", dlerror());**

**exit(1);**

**}**

**void \*handle2 = dlopen(LIBNAME\_SECOND, RTLD\_LAZY);**

**if (!handle2) {**

**printf("%s\n", dlerror());**

**exit(2);**

**}**

**int lib = 1;**

**int cmd = 0;**

**int x = 0;**

**float a = 0.f, b = 0.f;**

**char \*error;**

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

**switch(cmd) {**

**case 0:**

**lib = (lib == 1) ? 2 : 1;**

**break;**

**case 1:**

**if (lib == 1) {**

**e = dlsym(handle1, "e");**

**}**

**else {**

**e = dlsym(handle2, "e");**

**}**

**if ((error = dlerror()) != NULL) {**

**printf("Dlsym error\n");**

**exit(3);**

**}**

**scanf("%d", &x);**

**printf("Result: %f\n", e(x));**

**break;**

**case 2:**

**if (lib == 1) {**

**square = dlsym(handle1, "square");**

**}**

**else {**

**square = dlsym(handle2, "square");**

**}**

**if ((error = dlerror()) != NULL) {**

**printf("Dlsym error\n");**

**exit(4);**

**}**

**scanf("%f %f", &a, &b);**

**printf("Result: %f\n", square(a, b));**

**break;**

**}**

**}**

**if (dlclose(handle1) != 0) {**

**perror("Dlclose error");**

**exit(5);**

**}**

**if (dlclose(handle2) != 0) {**

**perror("Dlclose error");**

**exit(6);**

**}**

**return 0;**

**}**

**liblab1.c:**

**#include <math.h>**

**float e(int x) {**

**return pow((1 + 1. / x), x);**

**}**

**float square(float a, float b) {**

**return a \* b;**

**}**

**liblab2.c:**

**int factorial(int x) {**

**if (x == 0) {**

**return 1;**

**}**

**return x \* factorial(x - 1);**

**}**

**float e(int x) {**

**float e = 0;**

**for (int i = 0; i <= x; ++i) {**

**e += 1. / factorial(i);**

**}**

**return e;**

**}**

**float square(float a, float b) {**

**return 0.5 \* a \* b;**

**}**

**libs.h:**

**// Рассчет значения числа е(основание натурального логарифма)**

**float e(int x);**

**// Подсчет площади плоской геометрической фигуры по двум сторонам**

**float square(float a, float b);**

**}**

**return integral;**

**}**

**Пример работы**

MacBook:src vladislove$ cat ../tests/test\_main1

1 10

1 25

1 100

1 100000

2 10 20

2 -5 10

2 100 20

2 -50 0

MacBook:src vladislove$ ./task1 < ../tests/test\_main1

Result: 2.593742

Result: 2.665836

Result: 2.704814

Result: 2.718268

Result: 200.000000

Result: -50.000000

Result: 2000.000000

Result: -0.000000

MacBook:src vladislove$ cat ../tests/main2

1 5

1 10

1 15

0

1 15

2 100 20

0

2 10 20

2 -13 5

1 30

0

1 30

2 -13 5

MacBook:src vladislove$ ./task2 < ../tests/main2

Result: 2.488320

Result: 2.593742

Result: 2.632879

Result: 2.718282

Result: 1000.000000

Result: 200.000000

Result: -65.000000

Result: 2.674319

Result: 2.718282

Result: -32.50000

**Вывод**

Данная лабораторная работа направлена на изучение динамических библиотек. 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.

Динамические библиотеки используются в большинстве современных программ. Преимущества использования динамических библиотек превелируют над недостатками. Сейчас компьютеры обладают достаточной мощностью для быстрого выполнения системных вызовов. Важнее сэкономить объём памяти, используемой программой.