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: Стрыгин Д.Д.

Группа: М8О–306Б–19

Вариант: 10

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 10:

Функция 1: Подсчёт наибольшего общего делителя для двух натуральных чисел. Int GCF(int A, int B). Алгоритм Евклида. Наивный алгоритм - пытаться разделить числа на все числа, что меньше A и B.

Функция 2: Рассчет производной функции cos(x) в точке A с приращением deltaX. float Derivative(float A, float deltaX). f'(x) = (f(A + deltaX)-f(A))/deltaX. f'(x) = (f(A + deltaX) – f(A-deltaX))/(2\*deltaX).

**Общие сведения о программе**

Программа компилируется при помощи Makefile в 2 исполняемых файла lab5\_1, lab5\_2 и 2 библиотеки libdyn1.so, libdyn2.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 heder с реализациями и объявлениями для двух функций, собираем из них библиотеку и используем 2-мя способами:

* + - 1. на этапе компиляции (стадия линковки) при помощи #include в программе lab5\_1.
      2. при помощи загрузки библиотек при помощи dlopen в программе lab5\_2.

**Основные файлы программы**

**Makefile:**

FLAGS = -g3 -Wall -pedantic

all: lab5\_1 lab5\_2

libdyn1.so: der\_1.c gcf\_simple.c lib1.c

    gcc $(FLAGS) -fPIC -c der\_1.c gcf\_simple.c lib1.c

    gcc -shared der\_1.o gcf\_simple.o lib1.o -o libdyn1.so -lm

libdyn2.so: der\_2.c gcf\_evclid.c lib2.c

    gcc $(FLAGS) -fPIC -c der\_2.c gcf\_evclid.c lib2.c

    gcc -shared der\_2.o gcf\_evclid.o lib2.o -o libdyn2.so -lm

lab5\_1: lab5\_1.c lab5\_1.h libdyn1.so

    gcc $(FLAGS) -c lab5\_1.c

    gcc  lab5\_1.o -L. libdyn1.so -Wl,-rpath,. -o lab5\_1

lab5\_2: lab5\_2.c libdyn2.so

    gcc $(FLAGS) -c lab5\_2.c

    gcc -o lab5\_2  lab5\_2.o -l dl

clear:

    rm \*.o \*.so lab5\_1 lab5\_2

**lab5\_1.c:**

#include <stdio.h>

#include <stdlib.h>

#include "lab5\_1.h"

int main() {

    char comand;

    while (scanf("%c", &comand) > 0) {

        if (comand == '1') {

            int A, B;

            scanf("%d%d", &A, &B);

            printf("GCF = %d\n", GCF(A, B));

        }

        else if (comand == '2') {

            float A, deltax;

            scanf("%f%f", &A, &deltax);

            printf("der = %f\n", der(A, deltax));

        }

        else if (comand == EOF) exit(0);

        else if (comand == '\n' || comand == ' ') continue;

        else printf("Wrong format\n");

    }

}

**lab5\_1.h**

#pragma once

float der(float A, float deltax);

int GCF (int A, int B);

**lab5\_2.c:**

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

#define FIRST 1

#define SECOND 2

int curRealisation = SECOND;

void \*libFD;

float (\*der)(float, float);

int (\*GCF)(int, int);

void (\*whos\_now)();

char \*err = NULL;

void swapRealisaton(){

    if (dlclose(libFD) != 0) {

        perror("Can't close dl");

        exit(1);

    }

    char \*nextName;

    if (curRealisation == SECOND) {

        curRealisation = FIRST;

        nextName = "./libdyn1.so";

    }

    else {

        curRealisation = SECOND;

        nextName = "./libdyn2.so";

    }

    libFD = dlopen (nextName, RTLD\_LAZY);

    if (!libFD) {

        perror("can't swap lib");

        exit(1);

    }

    der = dlsym(libFD, "der");

    if ((err = dlerror()) != NULL) {

        fprintf(stderr, "%s\n", err);

        dlclose(libFD);

        exit(1);

    }

    GCF = dlsym(libFD, "GCF");

    if ((err = dlerror()) != NULL) {

        fprintf(stderr, "%s\n", err);

        dlclose(libFD);

        exit(1);

    }

    whos\_now = dlsym(libFD, "whos\_now");

    if ((err = dlerror()) != NULL) {

        fprintf(stderr, "%s\n", err);

        dlclose(libFD);

        exit(1);

    }

}

int main() {

    libFD = dlopen("./libdyn1.so", RTLD\_LAZY);

    if (!libFD) {

        perror("can't swap lib");

        exit(1);

    }

    der = dlsym(libFD, "der");

    if ((err = dlerror()) != NULL) {

        fprintf(stderr, "%s\n", err);

        dlclose(libFD);

        exit(1);

    }

    GCF = dlsym(libFD, "GCF");

    if ((err = dlerror()) != NULL) {

        fprintf(stderr, "%s\n", err);

        dlclose(libFD);

        exit(1);

    }

    whos\_now = dlsym(libFD, "whos\_now");

    if ((err = dlerror()) != NULL) {

        fprintf(stderr, "%s\n", err);

        dlclose(libFD);

        exit(1);

    }

    char comand;

    while(scanf("%c", &comand) > 0) {

        if (comand == '1') {

            int A, B;

            scanf("%d%d", &A, &B);

            printf("GCF = %d\n", GCF(A, B));

        }

        else if (comand == '2') {

            float A, deltax;

            scanf("%f%f", &A, &deltax);

            printf("der = %f\n", der(A, deltax));

        }

        else if (comand == '?') whos\_now();

        else if (comand == '~') swapRealisaton();

        else if (comand == EOF) {

            dlclose(libFD);

            exit(0);

        }

        else if (comand == '\n' || comand == ' ') continue;

        else printf("Wrong format\n");

    }

}

**der\_1.c:**

#include <math.h>

#include <stdio.h>

float der(float A, float deltax) {

    if (deltax == 0.0) {

        return A;

    }

    return (cos(A + deltax) - cos(A)) / deltax;

}

**der\_2.c:**

#include <math.h>

#include <stdio.h>

float der(float A, float deltax) {

    if (deltax == 0.0) {

        return A;

    }

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

}

**gcf\_evclid.c:**

int GCF (int A, int B) {

    while(B) {

        A %= B;

        int temp = A;

        A = B;

        B = temp;

    }

    return A;

}

**gcf\_simple.c:**

#define MAX(a, b) ((a) > (b) ? (a) : (b))

#define MIN(a, b) ((a) < (b) ? (a) : (b))

int GCF (int A, int B) {

    for (int i = MIN (A, B); i > 1; --i) {

        if (A % i == 0 && B % i == 0) {

            return i;

        }

    }

    return 1;

}

**lib1.c:**

#include <stdio.h>

void whos\_now() {

    printf("lib 1\n");

}

**lib2.c:**

#include <stdio.h>

void whos\_now() {

    printf("lib 2\n");

}

**Пример работы**

**en@DESKTOP-1B5EV3F:/mnt/c/Users/danst/OneDrive/Документы/GitHub/OC/lab5$ make**

**gcc -g3 -Wall -pedantic -fPIC -c der\_1.c gcf\_simple.c lib1.c**

**gcc -shared der\_1.o gcf\_simple.o lib1.o -o libdyn1.so -lm**

**gcc -g3 -Wall -pedantic -c lab5\_1.c**

**gcc lab5\_1.o -L. libdyn1.so -Wl,-rpath,. -o lab5\_1**

**gcc -g3 -Wall -pedantic -fPIC -c der\_2.c gcf\_evclid.c lib2.c**

**gcc -shared der\_2.o gcf\_evclid.o lib2.o -o libdyn2.so -lm**

**gcc -g3 -Wall -pedantic -c lab5\_2.c | ^**

**gcc -o lab5\_2 lab5\_2.o -l dl**

**den@DESKTOP-1B5EV3F:/mnt/c/Users/danst/OneDrive/Документы/GitHub/OC/lab5$ ./lab5\_1**

**1**

**28 12**

**GCF = 4**

**2**

**3.14 0.0001**

**der = -0.001541**

**den@DESKTOP-1B5EV3F:/mnt/c/Users/danst/OneDrive/Документы/GitHub/OC/lab5$ ./lab5\_2**

**1**

**28 12**

**GCF = 4**

**2 1.7 0.0001**

**der = -0.991823**

**~**

**?**

**lib 1**

**~**

**?**

**lib 2**

**1 28 12**

**GCF = 4**

**2 1.7 0.001**

**der = -914.418030**

**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.