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: Ткаченко Егор Юрьевич
Группа: М8О-207Б-21
Вариант: 24
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/Tnirpps/OS_lab

Постановка задачи

Цель работы

Изучить создание и использование динамических библиотек.

Задание

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки двумя 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 следующее:

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа No1, которая использует одну из библиотек, используя знания, полученные на этапе компиляции;
- Тестовая программа No2, которая загружает библиотеки, используя 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 на другую (необходимо только для программы No2);
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.

Задание варианта

№	Описание	Сигнатура	Реализация 1	Реализация 2
1	Подсчёт наибольшего общего делителя для двух натуральных чисел	Int GCD(int A, int B)	Алгоритм Евклида	Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.
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 из файлов main.c, main_dyn.c, child.c, realisation1.c, realisation2.c. Также используется заголовочные файлы: fcntl.h, stdio.h, realisation.h, dlfcn.h. В программе используются следующие системные вызовы:

1. dlopen() – загружает общий динамический объект и возвращает «handle» на него.
2. dlsym() – указывает адресс в объекте, откуда загружать символ.
3. dlerror() – возвращает строку ошибки, связанную с работой динамического объекта.
4. dlclose() – уменьшает на единицу счетчик ссылок на указатель динамической библиотеки «handle».

Общий метод и алгоритм решения

Описываем решения в библиотечных файлах, создаём общий заголовочный файл. Нам не потребуется два, так как в обеих реализациях одни и те же функции, соответственно, между двумя заголовочными файлами не было бы различия. Далее собираем всё в исполняемый файл.

Исходный код

```
============================== main_dyn.c ================================
#include <dlfcn.h>
#include <stdio.h>

#define check(VALUE, OKVAL, MSG) if (VALUE != OKVAL) { printf("%s", MSG); return 1; }
#define check_wrong(VALUE, WRONG_VAL, MSG) if (VALUE == WRONG_VAL) { printf("%s",
MSG); return 1; }
```

```
// it is very important to make prefix "lib" and set extension ".so"
const char* DYN_LIB_1 = "./libDyn1.so";
const char* DYN_LIB_2 = "./libDyn2.so";

const char* GCD_NAME = "GCD";
const char* SQUARE_NAME = "Square";

int main(int argc, const char** argv) {
    int dynLibNum = 1;
    void* handle = dlopen(DYN_LIB_1, RTLD_LAZY);
    check_wrong(handle, NULL, "Error opening dynamic library!\n")
    int (*GCD)(int, int);
    float (*Square)(float, float);
    *(void**) (&GCD) = dlsym(handle, GCD_NAME);
    *(void**) (&Square) = dlsym(handle, SQUARE_NAME);
    char* error = dlerror();
    check(error, NULL, error)
    int q;
    int x, y;
    float A, B;
    while (scanf("%d", &q) > 0) {
        switch (q) {
            case 0:
                check(dlclose(handle), 0, "Error closing dynamic library!\n")
                if (dynLibNum) {
                    handle = dlopen(DYN_LIB_2, RTLD_LAZY);
                } else {
                    handle = dlopen(DYN_LIB_1, RTLD_LAZY);
                }
                check_wrong(handle, NULL, "Error opening dynamic library!\n")
                *(void**) (&GCD) = dlsym(handle, GCD_NAME);
                *(void**) (&Square) = dlsym(handle, SQUARE_NAME);
                error = dlerror();
                check(error, NULL, error)
                /* switch between 0 and 1 */
                dynLibNum = dynLibNum ^ 1;
                break;
            case 1:
```

```
            check(scanf("%d%d", &x, &y), 2, "Error reading integer!\n");
            printf("GCD(%d, %d) = %d\n", x, y, GCD(x, y));
            break;
        case 2:
            check(scanf("%f %f", &A, &B), 2, "Error reading floats!\n");
            printf("Area is: %f\n", Square(A, B));
            break;
        default:
            printf("End.\n");
            check(dlclose(handle), 0, "Error closing dynamic library!\n")
            return 0;
      }
   }
}
================================  main.c ==================================
#include "../headers/realisation.h"
#include <stdio.h>

#define check(VALUE, OKVAL, MSG) if (VALUE != OKVAL) { printf("%s", MSG); return 1; }

int main(int argc, const char** argv) {
   int q;
   while (scanf("%d", &q) > 0) {
      if (q == 1) {
         int x, y;
         check(scanf("%d%d", &x, &y), 2, "Error reading integer!\n");
         printf("GCD(%d, %d) = %d\n", x, y, GCD(x, y));
      } else if (q == 2) {
         float A, B;
         check(scanf("%f %f", &A, &B), 2, "Error reading floats!\n");
         printf("Area is: %f\n", Square(A, B));
      } else {
         printf("End.\n");
         return 0;
      }
   }
}
============================  realisation1.c ================================
```

```
#include "../headers/realisation.h"

void swap_int(int* x, int* y) {
    int tmp = *x;
    *x = *y;
    *y = tmp;
}

int GCD(int x, int y) {
    while (y > 0) {
        if (x >= y) {
            x = x % y;
        }
        swap_int(&x, &y);
    }
    return x;
}

float Square(float A, float B) {
    return A * B;
}
```

============================== realisation2.c ==============================

```
#include "../headers/realisation.h"

void swap_int(int* x, int* y) {
    int tmp = *x;
    *x = *y;
    *y = tmp;
}

int GCD(int x, int y) {
    if (x > y) {
        swap_int(&x, &y);
    }
    for (int i = x; i > 1; --i) {
        if (x % i == 0 && y % i == 0) {
            return i;
        }
```

```
    }
    return 1;
}

float Square(float A, float B) {
    return A/2.f * B;
}
```

```
================================ realisation.h ================================
#ifndef LAB_5_REALISATION_H
#define LAB_5_REALISATION_H

int GCD(int x, int y);
float Square(float A, float B);

#endif //LAB_5_REALISATION_H
```

Демонстрация работы программы

```
hplp739@user:~/Desktop/OS/lab_5$ ./a.out
1
12 4
GCD(12, 4) = 4
2
2 2
Area is: 4.000000
0
2
2 2
Area is: 2.000000
1
33 11
GCD(33, 11) = 11
8
End.
hplp739@user:~/Desktop/OS/lab_5$
```

Выводы

Во время выполнения работы я изучил основы работы с динамическими библиотеками на операционных системах Linux, реализовал программу, которая использует созданные динамические библиотек. Выяснил некоторые различия в механизмах работы динамических и статических библиотек. Осознал что, использование библиотек добавляет модульность программе, что упрощает дальнейшую поддержку кода.