Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

**ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ**

Студент: Козырев Пётр Андреевич

Группа: М8О–212Б–22

Вариант: 24

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**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:

* 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»;
* «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;
* «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:**

Описание: Подсчёт наибольшего общего делителя для двух натуральных чисел

Сигнатура: Int GCF(int A, int B)

Реализация 1: Алгоритм Евклида

Реализация 2: Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.

**Функция 2:**

Описание: Подсчет площади плоской геометрической фигуры по двум сторонам

Сигнатура: Float Square(float A, float B)

Реализация 1: Фигура прямоугольник

Реализация 2: Фигура прямоугольный треугольник

**Общие сведения о программе**

*func.h* – Заголовочный файл для функций;

*first\_realization.c* – Содержит реализацию 1, данных функций;

*second\_realization.c* – Содержит реализацию 2, данных функций;

*main\_1.c* - Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;

*main\_2.c* - Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты;

*run.sh* – Компиляция и запуск программы main\_1.c и main\_2.c;

**Общий метод и алгоритм решения**.

Для реализации, поставленной задачи необходимо:

1. Реализовать заданные функции двумя способами, в соответствии с заданием
2. Написать тестовую программу №1 использовать данные библиотеки во время компиляции
3. Написать тестовую программу №2 использовать данные библиотеки Во время исполнения программы.
4. Скомпилировать и запустить программу.

**Основные файлы программы**

**funcs.h**

#pragma once

int GCF(int A,int B);

float Square(float A, float B);

**first\_realization.c**

#include <stdio.h>

#include "funcs.h"

int GCF(int A, int B){

    return B ? GCF(B, A % B) : A;

}

float Square(float A,float B){

    return A \* B;

}

**second\_realization.c**

#include <stdio.h>

#include "funcs.h"

int GCF(int A, int B){

    int res = 1;

    int n;

    if(A <= B) n = A;

    else n = B;

    for(int i = 2; i < n; ++i){

        if(A % i == 0 && B % i == 0){

            return i;

        }

    }

}

float Square(float A,float B){

    return 0.5 \* A \* B;

}

**main\_1.c**

#include <stdio.h>

#include <stdlib.h>

#include "funcs.h"

int main(){

    printf("1 arg1 arg2 - GCD function\n");

    printf("2 arg1 arg2 - Square function\n");

    printf("3 - exit\n");

    int command;

    printf("Input command: ");

    scanf("%d",&command);

    while(command != 3){

        switch (command){

        case 1:

            int x,y;

            printf("\nInput args: ");

            scanf("%d %d",&x,&y);

            printf("-----------------\n");

            printf("GCD(%d, %d) = %d\n",x,y,GCF(x,y));

            printf("-----------------\n");

            printf("\nInput command: ");

            scanf("%d",&command);

            break;

        case 2:

            float a,b;

            printf("\nInput args: ");

            scanf("%f %f",&a,&b);

            printf("-----------------\n");

            printf("Square(%.2f, %.2f) = %.2f\n",a,b,Square(a,b));

            printf("-----------------\n");

            printf("\nInput command: ");

            scanf("%d",&command);

            break;

        case 3:

            break;

        default:

            printf("\nIncorrect Input\n");

            exit(1);

            break;

        }

    }

}

**main\_2.c**

#include <stdio.h>

#include <dlfcn.h>

#include <string.h>

#include <stdlib.h>

int main(){

    printf("0 - Change realization\n");

    printf("1 arg1 arg2 - GCD function\n");

    printf("2 arg1 arg2 - Square function\n");

    printf("3 - exit\n");

    int flag\_lib = 1;      // номер текущей библиотеки

    void \*library\_handler; // прямой указатель на начало динамической библиотеки

    library\_handler = dlopen("libFirst.so",RTLD\_LAZY);

    if (!library\_handler){

        // если ошибка, то вывести ее на экран

        fprintf(stderr,"dlopen() error: %s\n", dlerror());

        exit(1); // в случае ошибки закончить работу программу

    };

    int command;

    printf("Input command: ");

    scanf("%d",&command);

    while(command != 3){

        switch (command){

        case 0:

            dlclose(library\_handler); // закрываю текущую библиотеку

            if(flag\_lib == 1){

                library\_handler = dlopen("libSecond.so",RTLD\_LAZY);

                flag\_lib = 2;

            }

            else{

                library\_handler = dlopen("libFirst.so",RTLD\_LAZY);

                flag\_lib = 1;

            }

            printf("\nREALIZATION CHANGED\n");

            printf("Input command: ");

            scanf("%d",&command);

            break;

        case 1:

            int x,y;

            printf("\nInput args: ");

            scanf("%d %d",&x,&y);

            printf("-----------------\n");

            int (\*gcdfunc)(int, int); // указатель на функцию GCF

            char name1[] = "GCF";

            gcdfunc = dlsym(library\_handler,name1);

            if (!gcdfunc){

                fprintf(stderr,"dlopen() error: %s\n", dlerror());

                exit(1);

            };

            if(flag\_lib == 1) printf("\nGCD realization №1\n");

            else printf("\nGCD realization №2\n");

            printf("GCD(%d, %d) = %d\n",x,y,(\*gcdfunc)(x, y));

            printf("-----------------\n");

            printf("\nInput command: ");

            scanf("%d",&command);

            break;

        case 2:

            float a,b;

            printf("\nInput args: ");

            scanf("%f %f",&a,&b);

            printf("-----------------\n");

            float (\*squarefunc)(float, float); // указатель на функцию Square

            char name2[] = "Square";

            if(flag\_lib == 1) printf("\nSquare realization №1\n");

            else printf("\nSquare realization №2\n");

            squarefunc = dlsym(library\_handler,name2);

            printf("Square(%.2f, %.2f) = %.2f\n",a,b,(\*squarefunc)(a, b));

            printf("-----------------\n");

            printf("\nInput command: ");

            scanf("%d",&command);

            break;

        case 3:

            break;

        default:

            printf("\nIncorrect Input\n");

            exit(1);

            break;

        }

    }

    dlclose(library\_handler);

}

**run.sh**

echo -e "Запуск main\_1 с одной реализацией функции\n"

gcc -c main\_1.c first\_realization.c

gcc main\_1.o first\_realization.o -o res1

./res1

echo -e "\n------------------------------------------------\n"

echo -e "Запуск main\_2\n"

gcc -o libFirst.so -shared -fPIC first\_realization.c

gcc -o libSecond.so -shared -fPIC second\_realization.c

gcc main\_2.c -L. -lFirst -lSecond -o res2

export LD\_LIBRARY\_PATH=.

./res2

**Пример работы**

**petrkozyrev@LAPTOP-F4OC0VPH:~/osi/os\_lab\_4$ ./run.sh**

**Запуск main\_1 с одной реализацией функции**

**1 arg1 arg2 - GCD function**

**2 arg1 arg2 - Square function**

**3 - exit**

**Input command: 1**

**Input args: 15 5**

**-----------------**

**GCD(15, 5) = 5**

**-----------------**

**Input command: 2**

**Input args: 10 10**

**-----------------**

**Square(10.00, 10.00) = 100.00**

**-----------------**

**Input command: 3**

**------------------------------------------------**

**Запуск main\_2**

**0 - Change realization**

**1 arg1 arg2 - GCD function**

**2 arg1 arg2 - Square function**

**3 - exit**

**Input command: 1**

**Input args: 30 3**

**-----------------**

**GCD realization №1**

**GCD(30, 3) = 3**

**-----------------**

**Input command: 2**

**Input args: 100 100**

**-----------------**

**Square realization №1**

**Square(100.00, 100.00) = 10000.00**

**-----------------**

**Input command: 0**

**REALIZATION CHANGED**

**Input command: 1**

**Input args: 20 20**

**-----------------**

**GCD realization №2**

**GCD(20, 20) = 20**

**-----------------**

**Input command: 2**

**Input args: 100 100**

**-----------------**

**Square realization №2**

**Square(100.00, 100.00) = 5000.00**

**-----------------**

**Input command: 3**

**Вывод**

В результате выполнения лабораторной работы мной были приобретены практические навыки в создании динамических библиотек и программ, которые используют функции этих библиотек.

В рамках задания было создано несколько динамических библиотек, реализующих определенный функционал. После этого были использованы данные библиотеки двумя способами: на этапе компиляции, при линковке программы, и во время исполнения, загружая библиотеки в память с помощью интерфейса ОС.

В результате лабораторной работы мной были получены следующие основные части: динамические библиотеки, реализующие заданные контракты, тестовая программа №1, которая использует одну из библиотек на этапе компиляции, и тестовая программа №2, которая загружает библиотеки только по их местоположению и контрактам.

Использование динамической библиотеки во время компиляции означает, что код библиотеки подключается к программе во время ее компиляции. Это позволяет компилятору проверить правильность использования функций и переменных из библиотеки и включить необходимые зависимости в исполняемый файл.

С другой стороны, использование динамической библиотеки во время исполнения программы означает, что код библиотеки загружается и подключается к программе только во время ее выполнения. Это позволяет сократить размер исходного кода программы и обеспечить более гибкую работу с библиотеками, поскольку можно обновлять их без перекомпиляции программы.

Таким образом, выбор между использованием библиотек на этапе компиляции или на этапе исполнения зависит от требований конкретного проекта и наличия нужного функционала в библиотеках.