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: Модин-Глазков Богдан Арсеньевич

Группа: М8О–212Б–22

Вариант:32

Преподаватель:

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 динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов

В конечном итоге, программа должна состоять из следующих частей:

* Динамическая библиотека, реализующая заданных вариантом интерфейс;
* Тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции;
* Тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс.

Провести анализ между обоими типами использования библиотеки.

# Общие сведения о программе

Программа компилируется при помощи утилиты CMake. Также используется заголовочные файлы: stdio.h, math.h, stdlib.h, string.h, dlfcn.h. В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамический общий объект (общую библиотеку) из файла, имя которого указано в строке filename (завершается null) и возвращает непрозрачный описатель на загруженный объект. Принимаемые параметры: **const char\* filename** – путь до файла с динамической библиотекой (.so), **int flag** – определенное условие подключение библиотеки.
2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти(указывается одним из аргументов). Принимаемые параметры: **void\* handle** – возвращаемое значение выполняемой функции dlopen, **char\* symbol** – является строкой, в которой содержится название символа, который необходимо загрузить из библиотеки.
3. **dlclose** – уменьшает счётчик ссылок на динамически загружаемый общий объект, на который ссылается handle. Если счётчик ссылок достигает нуля, то объект выгружается. Все общие объекты, которые были автоматически загружены при вызове dlopen() для объекта, на который ссылается handle, рекурсивно закрываются таким же способом. Принимаемые параметры: **void\* handle** – возвращаемое значение выполняемой функции dlopen.
4. **dlerror –** возвращает указатель на начало строки, описывающей ошибку, полученную на пердыдущем вызов

# Основные файлы программы

**first\_functions.h**

#pragma once

#include "support\_functions.h"

int GCF(int, int);

float calculating\_e(int);

**second\_functions.h**

#pragma once

#include "support\_functions.h"

double calculating\_e(int);

int GCF(int, int );

**support\_functions.h**

#pragma once

#include <math.h>

#include <stdio.h>

int abs(int);

int factorial(int);

void swap(int\*, int\*);

**first\_functions.c**

#include "../include/first\_functions.h"

int GCF(int first\_number, int second\_number){

first\_number = abs(first\_number);

second\_number = abs(second\_number);

if(first\_number < second\_number){

swap(&first\_number, &second\_number);

}

while(second\_number != 0){

first\_number %= second\_number;

swap(&first\_number, &second\_number);

}

return first\_number;

}

float calculating\_e(int x){

return -pow((1 + 1.0/x), x);

}

second\_functions.c

#include "../include/second\_functions.h"

int GCF(int first\_number, int second\_number){

first\_number = abs(first\_number);

second\_number = abs(second\_number);

int min\_number = (first\_number < second\_number) ? first\_number : second\_number;

for(int current\_number = min\_number; current\_number > 0; --current\_number){

if(((first\_number % current\_number) + (second\_number % current\_number)) == 0){

return current\_number;

}

}

return 1;

}

double calculating\_e(int x){

double number\_e = 1.0;

for(int i = 1;i <= x; ++i){

number\_e += 1.0/(factorial(i));

}

return (-number\_e);

}

support\_functions.c

#include "../include/support\_functions.h"

int abs(int x){

return (x > 0) ? x : -x;

}

int factorial(int n)

{

int rezult = 1;

for(int i = 1; i <= n; ++i){

rezult \*= i;

}

return rezult;

}

void swap(int \*first\_number, int \*second\_number)

{

int buffer = \*first\_number;

\*first\_number = \*second\_number;

\*second\_number = buffer;

}

test1.c

#include <stdio.h>

#include <dlfcn.h>

#include <stddef.h>

#include <stdbool.h>

const char\* list\_library\_paths[2] = {"./libs/libfirstLib.so", "./libs/libsecondLib.so"};

void\* init\_library(const char\* name\_library) {

return dlopen(name\_library, RTLD\_LAZY);

}

int current\_library = 1;

void\* list\_hdl[2];

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

double (\*calculating\_e) (int);

void init\_funtions(){

GCF = (int(\*)(int,int))dlsym(list\_hdl[current\_library - 1], "GCF");

calculating\_e = (double(\*)(int))dlsym(list\_hdl[current\_library - 1], "calculating\_e");

}

void library\_swap(){

if(current\_library == 1){

current\_library = 2;

init\_funtions();

return;

}

current\_library = 1;

init\_funtions();

}

int main(){

list\_hdl[0] = init\_library(list\_library\_paths[0]);

list\_hdl[1] = init\_library(list\_library\_paths[1]);

init\_funtions();

int command;

printf("Введи 0 для смены библиотеки, 1 - для запуска первой функции, 2 - для запуска второй, -1 для завершения программы.\n");

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

if(command == 0){

library\_swap();

printf("%s %d \n", "Текущая библиотека: ", current\_library);

}

else if(command == 1){

printf("%s \n", "Введи через пробел 2 числа, для которых ты хочешь найти НОД.");

int a, b;

int result;

scanf("%d", &a);

scanf("%d", &b);

result = GCF(a, b);

printf("НОД = %d \n", result);

}

else if(command == 2){

printf("%s \n", "Введи целое число для вычисления числа 'e', чем больше число, тем больше точность.");

int x;

double result;

scanf("%d", &x);

result = calculating\_e(x);

printf("Число 'e' = %f \n", result);

}

else if(command == -1){

printf("%s \n", "Программа завершена.");

break;

}

else{

printf("Введи 0 для смены библиотеки, 1 - для запуска первой функции, 2 - для запуска второй, -1 для завершения программы.\n");

}

}

}

**strace**

**write(1, "\320\222\320\262\320\265\320\264\320\270 0 \320\264\320\273\321\217 \321\201\320\274\320\265\320\275\321\213 \320"..., 199Введи 0 для смены библиотеки, 1 - для запуска первой функции, 2 - для запуска второй, -1 для завершения программы.**

**) = 199**

**newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0**

**read(0, 0x56194f5dac20, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)**

**--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---**

**read(0, 0x56194f5dac20, 1024) = ? ERESTARTSYS (To be restarted if SA\_RESTART is set)**

**--- SIGWINCH {si\_signo=SIGWINCH, si\_code=SI\_KERNEL} ---**

**read(0, 1**

**"1\n", 1024) = 2**

**write(1, "\320\222\320\262\320\265\320\264\320\270 \321\207\320\265\321\200\320\265\320\267 \320\277\321\200\320\276\320\261\320\265"..., 109Введи через пробел 2 числа, для которых ты хочешь найти НОД.**

**) = 109**

**read(0, 34**

**"34\n", 1024) = 3**

**read(0, 45**

**"45\n", 1024) = 3**

**write(1, "\320\235\320\236\320\224 = 1 \n", 12НОД = 1**

**) = 12**

**read(0, 4**

**"4\n", 1024) = 2**

**write(1, "\320\222\320\262\320\265\320\264\320\270 0 \320\264\320\273\321\217 \321\201\320\274\320\265\320\275\321\213 \320"..., 199Введи 0 для смены библиотеки, 1 - для запуска первой функции, 2 - для запуска второй, -1 для завершения программы.**

**) = 199**

**read(0, 2**

**"2\n", 1024) = 2**

**write(1, "\320\222\320\262\320\265\320\264\320\270 \321\206\320\265\320\273\320\276\320\265 \321\207\320\270\321\201\320\273\320\276"..., 148Введи целое число для вычисления числа 'e', чем больше число, тем больше точность.**

**) = 148**

**read(0, 234**

**"234\n", 1024) = 4**

**write(1, "\320\247\320\270\321\201\320\273\320\276 'e' = 2.712497 \n", 27Число 'e' = 2.712497**

**) = 27**

**read(0,**

**"\n", 1024) = 1**

**read(0, 3**

**"3\n", 1024) = 2**

**write(1, "\320\222\320\262\320\265\320\264\320\270 0 \320\264\320\273\321\217 \321\201\320\274\320\265\320\275\321\213 \320"..., 199Введи 0 для смены библиотеки, 1 - для запуска первой функции, 2 - для запуска второй, -1 для завершения программы.**

**) = 199**

**read(0, -1**

**"-1\n", 1024) = 3**

**write(1, "\320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260 \320\267\320\260\320\262\320\265\321\200\321\210\320"..., 40Программа завершена.**

**) = 40**

# Вывод

Изучив работу динамических библиотек, я научился различать и работать с библиотеками, которые подключаются на этапе компиляции и в „runtime“. Прочитав мануал по библиотеке dlfcn.h я разобрался в нюансах и тонкостях использования ее функций. В будущем мне поможет навык работы с динамическими библиотеками, ведь зачастую использования „тонны“

include’ов и import’ов приводит к огромному нагромождению всевозможных функций и объектов. Грамотное и своевременное подключение библиотек позволит простой работе и меньшей затрате по памяти.