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: Куценко Максим Дмитриевич

Группа: М8О–212Б-22

Вариант: 28

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

1. Создание динамических библиотек
2. 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:

1. Динамические библиотеки, реализующие контракты, которые заданы вариантом;
2. Тестовая программа (программа №1), которая используют одну из библиотек, используя
3. знания полученные на этапе компиляции;
4. 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,
4. предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

*Задание 5.* Рассчёт значения числа Пи при заданной длине ряда (K):

1. Ряд Лейбница
2. Формула Валлиса

*Задание 7.* Подсчёт площади плоской геометрической фигуры по двум сторонам:

1. Прямоугольник
2. Прямоугольный треугольник

**Общие сведения о программе**

При помощи команды add\_library в CMake создаются общие библиотеки с именем формата «lib» + <имя библиотеки> + «.so». К одной из библиотек подключаем стандартную математическую библиотеку через target\_link\_libraries.

Для исполняемого файла static мы подключаем созданную библиотеку на этапе компиляции через target\_link\_libraries в Cmake и #include в файле static.c.

Для исполняемого файла dynamic мы подключаем библиотеки во время исполнения и используем функции из них при помощи следующих функций:

1. **dlopen** — загрузка библиотеки в память и перемещение в адресное пространство процесса.
2. **dlsym —** поиск данного символа в символьной таблице библиотеки.
3. **dlclose —** выгрузка библиотеки.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить создание динамических и статических библиотек

**Основные файлы программы**

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.0)

project(Dynamic\_library)

set(CMAKE\_CXX\_STANDARD 20)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

add\_library(impl1 SHARED impl1.c)

target\_link\_libraries(impl1 m)

add\_library(impl2 SHARED impl2.c)

add\_executable(static static.c)

add\_executable(dynamic dynamic.c)

target\_link\_libraries(static impl1)

**lib.h:**

#include "stdio.h"

float Pi(int K);

float Square(float A, float B);

**impl1.c**

#include "lib.h"

#include "math.h"

float Square(float A, float B) {

return A\*B;

}

float Pi(int K) {

double pi = 0.0;

for(int i = 0; i < K; i++) {

pi += 4\*pow(-1,i)/(2\*i+1);

}

return pi;

}

**impl2.c**

#include "lib.h"

float Square(float A, float B) {

return 0.5\*A\*B;

}

float Pi(int K) {

double pi = 2;

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

float n = i;

pi \*= (4\*n\*n)/(4\*n\*n-1);

}

return pi;

}

**static.c**

#include "lib.h"

int main() {

int option = 0;

while (option != 3) {

printf("Your options: 1 - pi(1 arg), 2 - square(2 args), 3 - exit\n");

scanf("%d", &option);

switch(option) {

case 1:

int k;

scanf("%d", &k);

printf("Pi: %f\n", Pi(k));

break;

case 2:

float side1, side2;

scanf("%f%f", &side1, &side2);

printf("%f\n", Square(side1, side2));

break;

case 3:

printf("Exiting!\n");

break;

default:

printf("Wrong option!\n");

break;

}

}

}

**dynamic.c**

#include "stdio.h"

#include "dlfcn.h"

int main() {

int current = 1;

void\* LBhandle = dlopen("./libimpl1.so", RTLD\_LAZY);

if (!LBhandle) {

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

return -1;

}

float (\*Pi)(int) = dlsym(LBhandle, "Pi");

if (!Pi) {

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

dlclose(LBhandle);

return 1;

}

float (\*Square)(float, float) = dlsym(LBhandle, "Square");

if (!Square) {

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

dlclose(LBhandle);

return 2;

}

int option = 0;

while (option != 3) {

printf("You're using implementation %d.\nOptions: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit\n", current);

scanf("%d", &option);

switch(option) {

case 0:

dlclose(LBhandle);

if (current == 1) {

LBhandle = dlopen("./libimpl2.so", RTLD\_LAZY);

printf("Moved from implementation %d to %d\n", current, current++);

} else {

LBhandle = dlopen("./libimpl1.so", RTLD\_LAZY);

printf("Moved from implementation %d to %d\n", current, current--);

}

Pi = dlsym(LBhandle, "Pi");

if (!Pi) {

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

dlclose(LBhandle);

return 1;

}

Square = dlsym(LBhandle, "Square");

if (!Square) {

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

dlclose(LBhandle);

return 2;

}

break;

case 1:

int k;

scanf("%d", &k);

printf("Pi: %f\n", Pi(k));

break;

case 2:

float side1, side2;

scanf("%f%f", &side1, &side2);

printf("%f\n", Square(side1, side2));

break;

case 3:

printf("Exiting!\n");

dlclose(LBhandle);

break;

default:

printf("Wrong option!\n");

break;

}

}

}

**Примеры работы**

**[Maxim@HONOR-MB15 build]$ ./static**

Your options: 1 - pi(1 arg), 2 - square(2 args), 3 - exit

1 10

Pi: 3.041840

Your options: 1 - pi(1 arg), 2 - square(2 args), 3 - exit

2 2 5

10.000000

Your options: 1 - pi(1 arg), 2 - square(2 args), 3 - exit

3

Exiting!

**[Maxim@HONOR-MB15 build]$ ./dynamic**

You're using implementation 1.

Options: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit

1 10

Pi: 3.041840

You're using implementation 1.

Options: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit

2 2 5

10.000000

You're using implementation 1.

Options: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit

0

Moved from implementation 2 to 1

You're using implementation 2.

Options: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit

1 10

Pi: 3.067705

You're using implementation 2.

Options: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit

2 2 5

5.000000

You're using implementation 2.

Options: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit

0

Moved from implementation 1 to 2

You're using implementation 1.

Options: 0 - change implementation, 1 - pi(1 arg), 2 - square(2 args), 3 - exit

3

Exiting!

**Вывод**

Изучил статические и динамические библиотеки в языке C, процесс их создания в CMake и разницу в форматах. Подробнее познакомился с процессом линковки.

Статические библиотеки более удобны для использования их создателем, а также для применения небольших проектах. С другой стороны, динамические библиотеки более удобны, когда их код используется сразу в нескольких программах некоторого проекта, поскольку при этом системные ресурсы используются более эффективно — не происходит линковка при компиляции для каждой программы соответственно.

В процессе выполнения лабораторной работы также пришлось линковать стандартную математическую библиотеку с одной из собственных.