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: Шандрюк Пётр Николаевич

Группа: М8О-208Б-20

Вариант: 28

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

* Репозиторий
* Постановка задачи
* Общие сведения о программе
* Общий метод и алгоритм решения
* Исходный код
* Сборка программы
* Демонстрация работы программы
* Выводы

**Репозиторий**

https://github.com/Peter1811/OS/tree/main/lab5

**Постановка задачи**

**Цель работы**

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 No1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (программа No2), 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 No2). 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

| **N** | **Описание** | **Сигнатура** | **Реализация 1** | **Реализация 2** |
| --- | --- | --- | --- | --- |
| 2 | Рассчет значения числа Пи при заданной длине ряда (K) | float Pi(int K) | Ряд Лейбница | Формула Валлис |
| 8 | Подсчет площади плоской геометрической фигуры по двум сторона | Float Square(float A, float B | Фигура прямоугольни | Фигура прямоугольный треугольник |

**Общие сведения о программе**

Программа компилируется в двух файлах: main\_1.cpp и main\_2.cpp

Используемые библиотечные вызовы:

| void \*dlopen(const char \*filename, int flag); | Загружает динамическую библиотеку, имя которой указано в строке filename и возвращает прямой указатель на начало загруженной библиотеки. |
| --- | --- |
| const char \*dlerror(void); | Возвращает указатель на начало строки, описывающей ошибку, полученную на предыдущем вызове. |
| void \*dlsym(void \*handle, char \*symbol); | Получает параметр handle, который является выходом вызова dlopen и параметр symbol, который является строкой, в которой содержится название символа, который необходимо загрузить из библиотеки. Возвращает указатль на область памяти, в которой содержится необходимый символ. |
| int dlclose(void \*handle); | Уменьшает счетчик ссылок на указатель handle и если он равен нулю, то освобождает библиотеку. |

**Общий метод и алгоритм решения**

Для реализации поставленной задачи необходимо:

1. Изучить работу с библиотеками.
2. Реализовать две библиотеки согласно заданию.
3. Реализовать две программы (для работы с динамическими и статическими библиотеками).

**Исходный код**

**first.cpp**

#include <cmath>

// extern "C" float Pi (int K);

// extern "C" float Square (float A, float B);

float Pi(int K) {

if (K < 0) return -1;

float pi = 0;

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

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

}

return pi;

}

float Square(float A, float B) {

if ((A < 0) or (B < 0)) return -1;

return A \* B;

}

float Square(int A, int B) {

// if ((A < 0) or (B < 0)) return -1;

return 2;

}

**second.cpp**

#include <cmath>

float Pi(int K) {

if (K < 0) return -1;

float pi = 1.0;

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

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

}

return pi \* 2;

}

float Square(float A, float B) {

//if (A > B) {

// float C = pow(A \* A - B \* B, 1/2);

// return 0.5 \* B \* C;

//} else {

// float C = pow(B \* B - A \* A, 1/2);

// return 0.5 \* A \* C;

// }

// if ((A < 0) or (B < 0)) return -1;

// return 0.5 \* A \* B;

return 1;

}

**main\_1.cpp**

#include <iostream>

float Pi (int x);

float Square (float A, float B);

float Square(int A, int B);

int main () {

int choice;

std::cout << "Enter the number of function: 1 - pi, 2 - square" << std::endl;

while (std::cin >> choice) {

if (choice == 1) {

std::cout << "The calculation of Pi" << std::endl;

int x;

std::cin >> x;

float pi = Pi(x);

if (pi == -1) {

std::cout << "Wrong input" << std::endl;

exit(-1);

}

std::cout << pi << std::endl;

} else if (choice == 2) {

std::cout << "The calculation of square" << std::endl;

float a, b;

std::cin >> a >> b;

float s = Square((int)a, (int)b);

if (s == -1) {

std::cout << "Wrong input" << std::endl;

exit(-1);

}

std::cout << s << std::endl;

} else {

std::cout << "No such choice" << std::endl;

exit(-1);

}

std::cout << "Enter the number of function: 1 - pi, 2 - square" << std::endl;

}

return 0;

}

main\_2.cpp

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <dlfcn.h>

int main () {

void\* handle = NULL; //адрес, в будущем нужный нам для получения доступа к библиотеке

float (\*Pi)(int x); //объявление указателей на функции

float (\*Square)(int A, int B);

float (\*Square1)(float A, float B); // объявление указателей на функции

const char\* lib\_array[] = {"libd1.so", "libd2.so"};

int start\_library;

int current = start\_library - 1;

std:: cout << "Enter start library: " << std:: endl;

std:: cout << '\t' << "1 - first realization" << std:: endl;

std:: cout << '\t' << "2 - second realization" << std:: endl;

std:: cin >> start\_library;

if (start\_library != 1 && start\_library != 2) {

std:: cout << "Wrong input" << std:: endl;

exit(-1);

}

handle = dlopen(lib\_array[current], RTLD\_LAZY); //rtld lazy выполняется поиск только тех символов, на которые есть ссылки из кода

if (!handle) {

std:: cout << "An error while opening library" << std:: endl;

exit(EXIT\_FAILURE);

}

Pi = (float(\*)(int))dlsym(handle, "Pi"); //возвращаем адрес функции из памяти библиотеки

Square1 = (float(\*)(float, float))dlsym(handle, "Square"); //dlsym присваивает указателю на функцию, объявленному в начале, ее адрес в библиотеке

Square = (float(\*)(int, int))dlsym(handle, "Square");

int command;

std:: cout << '\t' << "0 - change the contract" << std:: endl;

std:: cout << '\t' << "1 - calculate the pi " << std:: endl;

std:: cout << '\t' << "2 - calculate the square " << std:: endl;

while (std::cin >> command){

if (command == 0) {

dlclose(handle); //освобождает указатель на библиотеку и программа перестает ей пользоваться

current = 1 - current;

handle = dlopen(lib\_array[current], RTLD\_LAZY);

if (!handle) {

std:: cout << "An error while opening library has been detected" << std:: endl;

exit(EXIT\_FAILURE);

}

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

Square = (float(\*)(int, int))dlsym(handle, "Square");

std:: cout << "You have changed contracts!" << std:: endl;

}

else if (command == 1) {

int x;

std:: cin >> x;

float exp = Pi(x);

if (exp == -1) {

std:: cout << "Wrong input" << std:: endl;

dlclose(handle);

exit(-1);

}

else {

std:: cout << "Result: " << exp << std:: endl;

}

}

else if (command == 2) {

float A, B, square;

std:: cin >> A >> B;

square = Square(A, B);

if (square == -1) {

std:: cout << "Wrong input" << std:: endl;

dlclose(handle);

exit(-1);

}

else {

std:: cout << "Result: " << square << std:: endl;

}

}

else {

std:: cout << "Wrong input2" << std:: endl;

dlclose(handle);

exit(-1);

}

}

dlclose(handle);

return 0;

}

**Makefile**

**files: main1 main2**

**main1: libd1.so main\_1.cpp**

**g++ main\_1.cpp -L. -ld1 -o main1 -Wl,-rpath -Wl,.**

**main2: libd1.so libd2.so main\_2.cpp**

**g++ main\_2.cpp -L. -ldl -o main2 -Wl,-rpath -Wl,.**

**libd1.so: d1.o**

**g++ -shared d1.o -o libd1.so**

**libd2.so: d2.o**

**g++ -shared d2.o -o libd2.so**

**d1.o: first.cpp**

**g++ -fPIC -c first.cpp -o d1.o**

**d2.o: second.cpp**

**g++ -fPIC -c second.cpp -o d2.o**

**clean:**

**rm -r \*.so \*.o main1 main2**

**Демонстрация работы программы**

**peter@DESKTOP-V53N291:$ make**

**peter@DESKTOP-V53N291:$ ./main1**

Enter the number of function: 1 - pi, 2 - square

1

The calculation of Pi

4

2.89524

**peter@DESKTOP-V53N291:$ ./main2**

Enter start library:

1 - first realization

2 - second realization

2

The calculation of Pi

4

2.89524

**Выводы**

В ходе лабораторной работы я познакомился с созданием динамических библиотек в ОС Linux, а также с возможностью загружать эти библиотеки в ходе выполнения программы. Динамические библиотеки помогают уменьшить размер исполняемых файлов. Загрузка динамических библиотек во время выполнения также упрощает компиляцию. Однако также можно подключить библиотеку к программе на этапе линковки. Она все равно загрузится при выполнении, но теперь программа будет изначально знать что и где искать. Если библиотека находится не в стандартной для динамических библиотек директории, необходимо также сообщить линкеру, чтобы тот передал необходимый путь в исполняемый файл. При помощи библиотек мы можем писать более сложные вещи, которые используют простые функции, структуры и т.п., написанные ранее и сохраненные в различных библиотеках.