Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Тема работы**

**“Динамические библиотеки”**

Студент: Борисов Ян Артурович

Группа: М8О-208Б-20

Вариант: 34

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/Yannikupy/OS

**Постановка задачи**

Задача: реализовать 2 динамические библиотеки и 2 программы для работы с ними. Первая программа будет загружать библиотеку (одну) на этапе компиляции при помощи ключа -lmylib, а вторая программа будет подключать две динамические библиотеки при помощи dl-функций в самом коде.

**Общие сведения о программе**

Для выполнения данной лабораторной работы я предварительно создал 4 файла: первые два – lib1.cpp и lib2.cpp являются исходным кодом для наших динамических библиотек. Файлы first.cpp и second.cpp являются двумя программами, которые нужно было реализовать по заданию. first.cpp является программой, к которой библиотека подгружается на этапе компиляции, а second.cpp является программой, к которой библиотека подключается непосредственно в самом коде.

Помимо этого, для удобства компиляции всех программ я создал MakeFile со следующим набором команд:  
1) g++ -fPIC -c lib1.cpp -o d1.o

g++ -fPIC -c lib2.cpp -o d2.o

При помощи этих команд наши cpp-библиотеки превращаются в объектные файлы. Это, так называемый, “промежуточный этап” создания динамических библиотек.

2) g++ -shared d1.o -o libd1.so

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

При помощи флага -shared мы создаем наши нужные по заданию динамические библиотеки.

3) g++ -fsanitize=address first.cpp -L. -ld1 -o main1 -Wl,-rpath -Wl,.

Этой строчкой мы делаем исполняемый файл из нашей программы first.cpp, при этом компилируем мы только с одной библиотекой (то есть компиляция может проходить либо с ключом -ld1, либо с ключом -ld2).

4) g++ -fsanitize=address second.cpp -L. -ldl -o main2 -Wl,-rpath -Wl,.

Этой строчкой мы делаем исполняемый файл из нашей программы launch.cpp, только теперь с флагом -ldl. Далее в нашей программе main2 будут доступны 2 динамические библиотеки, действия над которыми будут обрабатываться при помощи следующих функций:  
void\* dlopen(...) - вгружает нашу библиотеку;

void\* dlsym(...) - присваивает указателю на функцию ее адрес в библиотеке

int dlclose(...) - освобождает указатель на библиотеку

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

При помощи команды make clean происходит удаление всех созданных при работе программы файлов.

**Общий метод и алгоритм решения**

В самом начале выполнения лабораторной работы я реализовал две библиотеки: lib1.cpp и lib2.cpp. Там реализовал простейший подсчет площади квадрата и прямоугольного треугольника, а также алгоритмы перевода числа из десятичной системы в двоичную и троичную. В first.cpp через конструкцию if обрабатывал команды, которые вводит пользователь и выдавал ожидаемый результат. В second.cpp подгружал библиотеки через

void\* dlopen(...), через void\* dlsym(...) делал функции из динамических библиотек видными для second.cpp.

**Исходный код**

**first.cpp**

#include <iostream>

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

extern "C" char\* translation(long x);

int main(){

int command;

while((std::cout << "Enter command: ") && (std::cin >> command)){

if(command == 1){

std::cout << "Enter lengths A and B: ";

float A, B;

std::cin >> A >> B;

std::cout << "Square is " << Square(A, B) << std::endl;

}

else if(command == 2){

long x;

std::cout << "Enter decimal number: ";

std::cin >> x;

char\* memory = translation(x);

std::cout << "Binary number is " << memory << std::endl;

free(memory);

}

else

std::cout << "Commands can be 1 and 2 ";

}

}

**second.cpp**

extern "C" float E (int x);

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

float E (int x) {

if (x < 0) {

return -1;

}

float e = 1.0;

float term = 1.0;

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

term /= i;

e += term;

}

return e;

}

float Square (float A, float B) {

if (A <= 0 || B <= 0) {

return -1;

}

float square = 0.5 \* A \* B;

return square;

}

**lib1.cpp**#include <iostream>

#include <algorithm>

#include <cstring>

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

extern "C" char\* translation(long x);

float Square(float A, float B){

return A \* B;

}

char\* translation(long x){

std::string bin;

if(x == 0) bin += "0";

while(x != 0){

if(x % 2 ==0)

{

x/=2;

bin += "0";

}

else

if(x % 2 != 0)

{

x /= 2;

bin += "1";

}

}

std::string number = bin;

std::reverse(number.begin(), number.end());

char\* answer = (char\*) malloc((number.size() + 1) \* sizeof(char));

strcpy(answer, number.c\_str());

return answer;

}

**lib2.cpp**#include <iostream>

#include <algorithm>

#include <cstring>

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

extern "C" char\* translation(long x);

float Square(float A, float B){

return 0.5 \* (A \* B);

}

char\* translation(long x) {

std::string bin;

if(x == 0) bin += "0";

while (x > 0) {

bin += std::to\_string(x % 3);

x /= 3;

}

std::string number = bin;

std::reverse(number.begin(), number.end());

char \*answer = (char \*) malloc((number.size() + 1) \* sizeof(char));

strcpy(answer, number.c\_str());

return answer;

}

**Демонстрация работы программы**

./main1

Enter command: 1

Enter lengths A and B: 5 6

Square is 30

Enter command: 2

Enter decimal number: 10

Binary number is 1010  
./main2

Enter num of library: 1

Enter command 0, 1 or 2

1

Enter lengths A and B: 5 6

Square is 30

2

Enter decimal number: 101

Binary number is 1100101

0

Change contract

2

Enter decimal number: 10

Ternary number is 101

1

Enter lengths A and B: 5 6

Square is 15 **Выводы**В данной работе я научился работать с динамическими библиотеками, изучил теорию их использования, осознал их отличия от статических библиотек и успешно реализовал динамические библиотеки, требуемые по заданию.