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: Москвин Артём Артурович

Группа: М8О-208Б-20

Вариант: 29

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

Репозиторий

Постановка задачи

Общие сведения о программе

Общий метод и алгоритм решения

Исходный код

Демонстрация работы программы

Выводы

**Репозиторий**

https://github.com/Pert002

**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 5 файлов: первые два - realization1.cpp и realization2.cpp являются исходным кодом для наших динамических библиотек. Файлы main1.cpp и main2.cpp являются двумя программами, которые нужно было реализовать по заданию. main1.cpp является программой, к которой библиотека подгружается на этапе компиляции, а main2.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 realization1.cpp -o d1.o

g++ -fPIC -c realization2.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++ main1.cpp -L. -ld1 -o main1 -Wl,-rpath -Wl,.

Этой строчкой мы делаем исполняемый файл из нашей программы main1.cpp, при этом компилируем мы только с одной библиотекой (то есть компиляция может проходить либо с ключом -ld1, либо с ключом -ld2).

4) g++ main2.cpp -L. -ldl -o main2 -Wl,-rpath -Wl,.

Этой строчкой мы делаем исполняемый файл из нашей программы main2.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 файлов, вследствие чего в папке остаются исходные 5 объектов.

**Общий метод и алгоритм решения**

В самом начале выполнения лабораторной работы я реализовал две библиотеки: realization1.cpp и realization2.cpp. В библиотеке realization1.cpp реализовано вычисление числа пи при помощи ряда Лейбница и перевод числа x в двоичную систему счисления. В библиотеке main2.cpp реализовано вычисление числа пи при помощи формулы Валисса и перевод числа x в троичную систему счисления. Далее в файле main1.cpp я реализовал обычное считывание команды при помощи проверки равенства функции scanf на -1 (вводится EOF - Ctrl+D на Ubuntu) и конструкции switch-case. Если вводится команда, отличная от 1 или 2, вылезает сообщение о том, что ввод был осуществлен неправильно. Если вводится 1, то считается число пи при помощи ряда Лейбница. Если вводится 2, то число x переводится в двоичную систему счисления.  
Что же касается main2.cpp, то там суть почти та же. В начале создаю необходимые указатели, позже загружаю какую-либо дин. библиотеку в зависимости от ввода пользователя. При помощи известного нам считывания до EOF я считываю команду. Если это не 0, не 1 и не 2, то прошу ввести правильную команду. Если эта команда 0, то программа меняет библиотеки (то есть, если раньше мне был доступен подсчет числа пи при помощи ряда Лейбница и перевод числа x в двоичную систему счисления , то теперь мне будет доступен подсчет числа пи при помощи формулы Валисса и перевод числа x в троичную систему счисления). Если команда 1, я считаю число пи. Если 2, то перевожу число x в нужную систему счисления. В конце освобождается указатель на библиотеку в целях избежания утечек памяти, программа завершается.

**Исходный код**

**realization1.cpp**

|  |
| --- |
| #include "realizations.h" |

|  |
| --- |
| #include <string> |

|  |
| --- |
| #include <algorithm> |

|  |
| --- |
| #include <cstring> |

|  |
| --- |
|  |

|  |
| --- |
| float Pi(int K) { |

|  |
| --- |
| float pi = 0; |

|  |
| --- |
| float n = 1; |

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

|  |
| --- |
| if (i % 2 == 0) pi += 4/n; |

|  |
| --- |
| else pi -= 4/n; |

|  |
| --- |
| n += 2; |

|  |
| --- |
| } |

|  |
| --- |
| return pi; |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| char\* translation(long x) { |

|  |
| --- |
| std::string s; |

|  |
| --- |
| if(x==0) s += "0"; |

|  |
| --- |
| while (x > 0) { |

|  |
| --- |
| if (x % 2 == 0) { |

|  |
| --- |
| x /= 2; |

|  |
| --- |
| s += "0"; |

|  |
| --- |
| } |

|  |
| --- |
| else { |

|  |
| --- |
| x /= 2; |

|  |
| --- |
| s += "1"; |

|  |
| --- |
| } |

|  |
| --- |
| } |

|  |
| --- |
| std::reverse(s.begin(), s.end()); |

|  |
| --- |
| char\* ans = (char\*) malloc((s.size() + 1) \* sizeof(char)); |

|  |
| --- |
| strcpy(ans, s.c\_str());//копирует string в char\* |

|  |
| --- |
| return ans; |

}

**realization2.cpp**

|  |
| --- |
| #include "realizations.h" |

|  |
| --- |
| #include <string> |

|  |
| --- |
| #include <algorithm> |

|  |
| --- |
| #include <cstring> |

|  |
| --- |
| #include <cmath> |

|  |
| --- |
|  |

|  |
| --- |
| float Pi(int K){ |

|  |
| --- |
| float pi = 1; |

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

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

|  |
| --- |
| } |

|  |
| --- |
| return pi \* 2; |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| char\* translation(long x) { |

|  |
| --- |
| std::string s; |

|  |
| --- |
| if(x==0) s += "0"; |

|  |
| --- |
| while (x > 0) { |

|  |
| --- |
| s += std::to\_string(x % 3); |

|  |
| --- |
| x /= 3; |

|  |
| --- |
| } |

|  |
| --- |
| std::reverse(s.begin(), s.end()); |

|  |
| --- |
| char\* ans = (char\*) malloc((s.size() + 1) \* sizeof(char)); |

|  |
| --- |
| strcpy(ans, s.c\_str());//копирует string в char\* |

|  |
| --- |
| return ans; |

**}**

**main1.cpp**

|  |
| --- |
| #include <iostream> |

|  |
| --- |
| #include "realizations.h" |

|  |
| --- |
|  |

|  |
| --- |
| int main(){ |

|  |
| --- |
| int command; |

|  |
| --- |
| std::cout << "Insert a command\n 1 - employ Pi function,\n 2 - employ function of conversion to binary notation\n"; |

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

|  |
| --- |
| switch (command) { |

|  |
| --- |
| case 1: { |

|  |
| --- |
| int K; |

|  |
| --- |
| std::cout << "Insert K\n"; |

|  |
| --- |
| std::cin >> K; |

|  |
| --- |
| float pi = Pi(K); |

|  |
| --- |
| std::cout << "Result is " << pi << "\n"; |

|  |
| --- |
| break; |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| case 2: { |

|  |
| --- |
| long x; |

|  |
| --- |
| std::cout << "Insert x that u want to convert to binary\n"; |

|  |
| --- |
| std::cin >> x; |

|  |
| --- |
| char\* result = translation(x); |

|  |
| --- |
| std::cout << "Result is " << result << "\n"; |

|  |
| --- |
| free(result); |

|  |
| --- |
| break; |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| default: { |

|  |
| --- |
| std::cout << "Wrong command\n"; |

|  |
| --- |
| std::cout << "Insert a command\n 1 - employ Pi function,\n 2 - employ function of conversion to binary notation\n"; |

|  |
| --- |
| } |

|  |
| --- |
| } |

|  |
| --- |
| } |

|  |
| --- |
| return 0; |

**}**

**main2.cpp**

|  |
| --- |
| #include <iostream> |

|  |
| --- |
| #include <cstdlib> |

|  |
| --- |
| #include <dlfcn.h> |

|  |
| --- |
|  |

|  |
| --- |
| int main() { |

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

|  |
| --- |
| float (\*Pi)(int K);//объявление указателей на функции |

|  |
| --- |
| char\* (\*translation)(long x);//объявление указателей на функции |

|  |
| --- |
| const char\* libs[] = {"libd1.so", "libd2.so"}; |

|  |
| --- |
| int cur\_lib; |

|  |
| --- |
| int start\_lib; |

|  |
| --- |
| std::cout << "Enter start library: \n"; |

|  |
| --- |
| std::cout << "1 for using first library\n"; |

|  |
| --- |
| std::cout << "2 for using second library\n"; |

|  |
| --- |
| std::cin >> start\_lib; |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
| bool flag = true; |

|  |
| --- |
| while (flag) { |

|  |
| --- |
| if (start\_lib == 1) { |

|  |
| --- |
| cur\_lib = 0; |

|  |
| --- |
| flag = false; |

|  |
| --- |
| } |

|  |
| --- |
| else if (start\_lib == 2) { |

|  |
| --- |
| cur\_lib = 1; |

|  |
| --- |
| flag = false; |

|  |
| --- |
| } |

|  |
| --- |
| else { |

|  |
| --- |
| std::cout << "Error\nYou should enter only 1 or 2\n"; |

|  |
| --- |
| std::cin >> start\_lib; |

|  |
| --- |
| } |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
| handle = dlopen(libs[cur\_lib], RTLD\_LAZY); //rtld lazy выполняется поиск только тех символов, на которые есть ссылки из кода |

|  |
| --- |
| if (!handle) { |

|  |
| --- |
| std::cout << "Error\nCan not open library\n"; |

|  |
| --- |
| exit(EXIT\_FAILURE); |

|  |
| --- |
| } |

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

|  |
| --- |
| translation = (char\*(\*)(long x))dlsym(handle, "translation");//dlsym присваивает указателю на функцию, объявленному в начале, ее адрес в библиотеке |

|  |
| --- |
| int command; |

|  |
| --- |
| std::cout << "Insert a command\n 0 - change the contract\n 1 - employ Pi function,\n 2 - employ function of conversion to binary notation\n"; |

|  |
| --- |
|  |

|  |
| --- |
|  |

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

|  |
| --- |
| switch (command) { |

|  |
| --- |
| case 0: { |

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

|  |
| --- |
| cur\_lib = 1 - cur\_lib; |

|  |
| --- |
| handle = dlopen(libs[cur\_lib], RTLD\_LAZY); |

|  |
| --- |
| if (!handle) { |

|  |
| --- |
| std::cout << "Error\nCan not open library\n"; |

|  |
| --- |
| exit(EXIT\_FAILURE); |

|  |
| --- |
| } |

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

|  |
| --- |
| translation = (char\*(\*)(long x))dlsym(handle, "translation"); |

|  |
| --- |
| std::cout << "You have changed contracts!\n"; |

|  |
| --- |
| break; |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| case 1: { |

|  |
| --- |
| int K; |

|  |
| --- |
| std::cout << "Insert K\n"; |

|  |
| --- |
| std::cin >> K; |

|  |
| --- |
| float pi = Pi(K); |

|  |
| --- |
| std::cout << "Result is " << pi << "\n"; |

|  |
| --- |
| break; |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| case 2: { |

|  |
| --- |
| long x; |

|  |
| --- |
| std::cout << "Insert x that u want to convert to binary or tenary\n"; |

|  |
| --- |
| std::cin >> x; |

|  |
| --- |
| char\* result = translation(x); |

|  |
| --- |
| std::cout << "Result is " << result << "\n"; |

|  |
| --- |
| free(result); |

|  |
| --- |
| break; |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
| default: { |

|  |
| --- |
| std::cout << "Wrong command\n"; |

|  |
| --- |
| std::cout << "Insert a command\n 1 - employ Pi function,\n 2 - employ function of conversion to binary notation\n"; |

|  |
| --- |
| } |

|  |
| --- |
| } |

|  |
| --- |
| } |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
| dlclose(handle); |

|  |
| --- |
| return 0; |

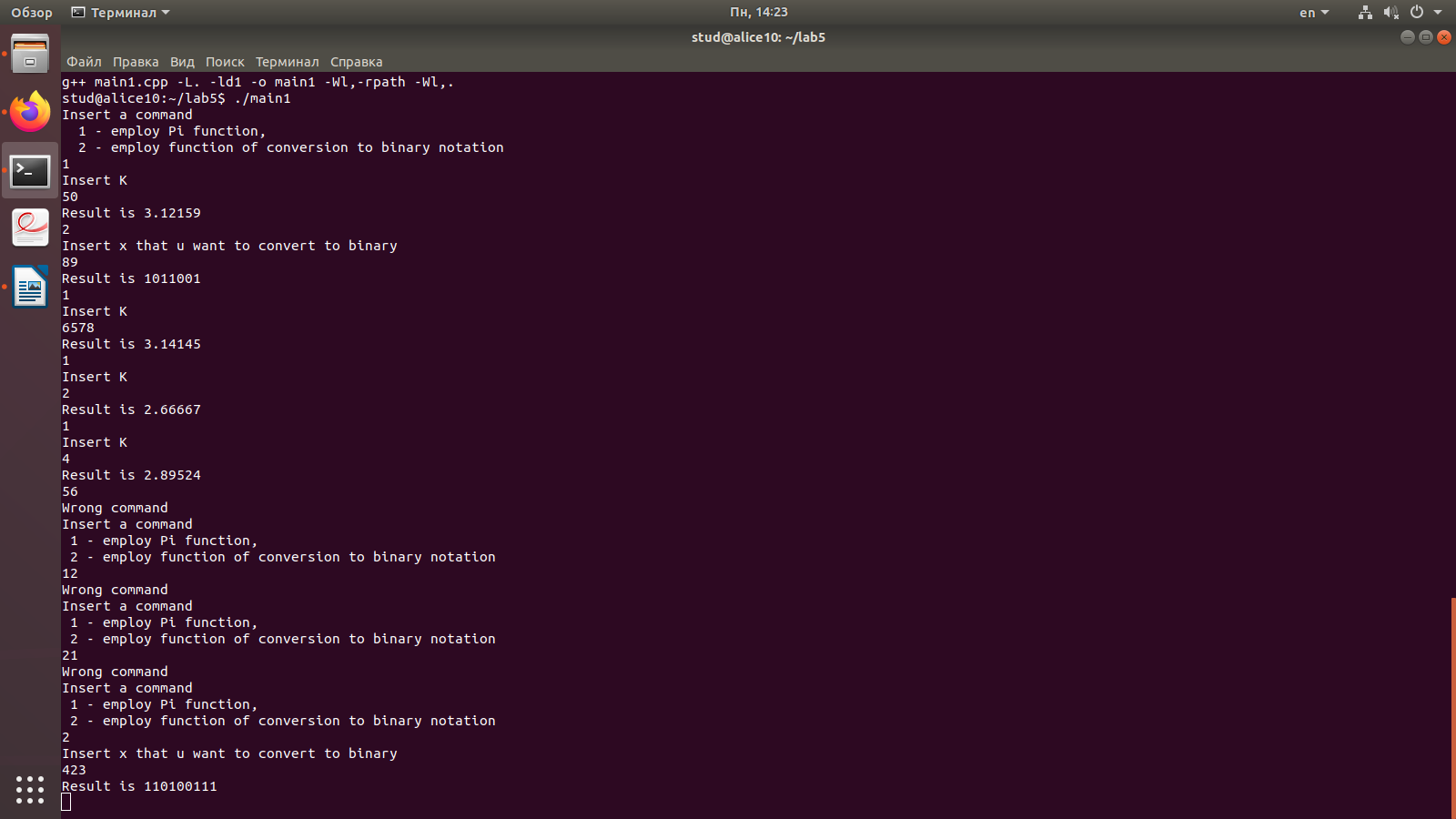
**}**

**realizations.h**

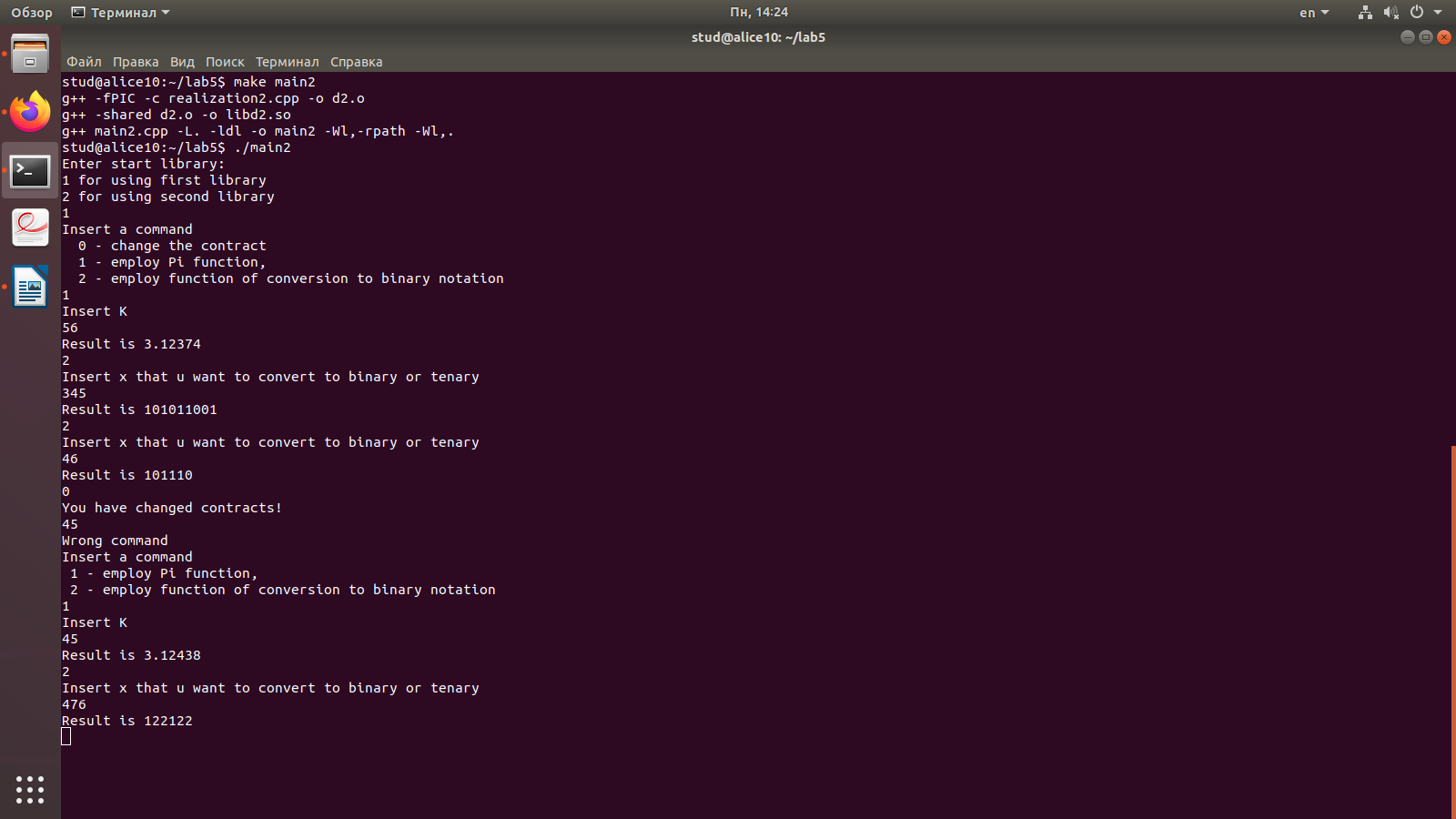
|  |
| --- |
| extern "C" float Pi(int K); |

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

**Демонстрация работы программы**



**Выводы**

Данная лабораторная работа научила меня пользоваться dl-функциями, благодаря реализации исполняемых файлов по заданию, я закрепил навык работы с динамическими библиотеками и полностью осознал их отличие от статических библиотек.