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: Юрков Евгений Юрьевич

Группа: М8О–212Б–22

Вариант: 36

Преподаватель: Соколов Андрей Алексеевич

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 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 №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
*  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, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

**Вариант 36:** Перевод числа x из десятичной системы в другую (в двоичную и в троичную). Отсортировать целочисленный массив (пузырьковая сортировка и сортировка Хоара).

**Общие сведения о программе**

Основной файл программы 1 - main1.cpp, основной файл программы 2 – main2.cpp; simple.cpp и difficult.cpp – два способа реализации заданных функций. Также используется заголовочные файлы: iostream, dlfcn.h, functional. В программе используются следующие системные вызовы:

1. **dlopen** – открытие динамической библиотеки.
2. **dlerror** – получение данных об ошибке при использовании динамических библиотек.
3. **dlsym** – получение адреса символа из динамической библиотеки.
4. **dlclose –** закрытие динамической библиотеки.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы с динамическими библиотеками в ОС UNIX.
2. Написать файлы из которых будет состоять программа.
3. Написать CMakeLists.txt и указать в нем создание динамической библиотеки.

**Основные файлы программы**

**main1.cpp:**

#include <iostream>

#include "functions.hpp"

int main(int argc, char\*\* argv) {

int query;

std::cout << "0 - сменить реализацию\n";

std::cout << "1 {N} - перевести число в систему счисления\n";

std::cout << "2 {SIZE} {ARRAY} - отсортировать массив\n";

std::cout << "-1 - выход" << std::endl;

do {

std::cout << "\033[1m";

std::cin >> query;

std::cout << "\033[0m";

switch(query) {

case (0):

std::cout << "Библиотека была подключена на этапе линковки" << std::endl;

break;

case (1):

{

long num;

std::cin >> num;

char\* result = translation(num);

std::cout << "\033[36m" << result << "\033[0m" << std::endl;

delete [] result;

}

break;

case (2):

{

unsigned long size;

std::cin >> size;

int \*array = new int[size];

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

std::cin >> array[i];

}

int\* result = sort(array, size);

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

std::cout << "\033[36m" << result[i] << ' ';

}

std::cout << "\033[0m" << std::endl;

delete [] array;

delete [] result;

}

break;

}

} while (query != -1);

}

**main2.cpp:**

#include <iostream>

#include <dlfcn.h>

#include <functional>

template<class T> std::function<T> make\_func(void\* ptr) {

return \*reinterpret\_cast<T\*>(ptr);

}

int main(int argc, char\*\* argv) {

short current\_lib = 1;

void\* handle = dlopen("./libsimple.so", RTLD\_LAZY);

char\* error;

if ((error = dlerror()) != NULL) {

std::cerr << error << std::endl;

return -1;

}

auto translation = make\_func<char\* (long)>(dlsym(handle, "translation"));

if ((error = dlerror()) != NULL) {

std::cerr << error << std::endl;

return -1;

}

auto sort = make\_func<int\* (int\*, const unsigned long&)>(dlsym(handle, "sort"));

if ((error = dlerror()) != NULL) {

std::cerr << error << std::endl;

return -1;

}

int query;

std::cout << "0 - сменить реализацию\n";

std::cout << "1 {N} - перевести число в систему счисления\n";

std::cout << "2 {SIZE} {ARRAY} - отсортировать массив\n";

std::cout << "-1 - выход" << std::endl;

do {

std::cout << "\033[1m";

std::cin >> query;

std::cout << "\033[0m";

switch(query) {

case (0):

if (current\_lib == 2) {

dlclose(handle);

handle = dlopen("./libsimple.so", RTLD\_LAZY);

current\_lib = 1;

std::cout << "Перевод в двоичную и сортировака пузырьком\n";

}

else {

dlclose(handle);

handle = dlopen("./libdifficult.so", RTLD\_LAZY);

current\_lib = 2;

std::cout << "Перевод в троичную и сортировка Хоара\n";

}

translation = make\_func<char\* (long)>(dlsym(handle, "translation"));

sort = make\_func<int\* (int\*, const unsigned long&)>(dlsym(handle, "sort"));

break;

case (1):

{

long num;

std::cin >> num;

char\* result = translation(num);

std::cout << "\033[36m" << result << "\033[0m" << std::endl;

delete [] result;

}

break;

case (2):

{

unsigned long size;

std::cin >> size;

int \*array = new int[size];

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

std::cin >> array[i];

}

int\* result = sort(array, size);

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

std::cout << "\033[36m" << result[i] << ' ';

}

std::cout << "\033[0m" << std::endl;

delete [] array;

delete [] result;

}

break;

}

} while (query != -1);

dlclose(handle);

}

**functions.hpp**

#pragma once

extern "C" {

char\* translation(long x);

}

extern "C" {

int\* sort(int\* array, const unsigned long& size);

}

**simple.hpp**:

#include "functions.hpp"

extern "C" {

char\* translation(long x) {

char\* arr = new char[100];

unsigned long i = 0;

while (x > 0) {

arr[i++] = (x % 2) + '0';

x /= 2;

}

const unsigned long size = i + 1;

char\* result = new char[size];

for (int j = 0; j < size - 1; ++j) {

result[j] = arr[--i];

}

result[size - 1] = '\0';

delete [] arr;

return result;

}

}

extern "C" {

int\* sort(int\* array, const unsigned long& size) {

int\* result = new int[size];

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

result[i] = array[i];

}

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

for (int j = 1; j < size; ++j) {

if (result[j] < result[j - 1]) {

int tmp = result[j];

result[j] = result[j - 1];

result[j - 1] = tmp;

}

}

}

return result;

}

}  
**difficult.hpp**:

#include "functions.hpp"

#include <iostream>

extern "C" {

char\* translation(long x) {

char\* arr = new char[100];

unsigned long i = 0;

while (x > 0) {

arr[i++] = (x % 3) + '0';

x /= 3;

}

const unsigned long size = i + 1;

char\* result = new char[size];

for (int j = 0; j < size - 1; ++j) {

result[j] = arr[--i];

}

result[size - 1] = '\0';

delete [] arr;

return result;

}

}

extern "C" {

void \_sort(int\* arr, int first, int last) {

int i = first, j = last, pivot = arr[(first + last) / 2];

do {

while (arr[i] < pivot)

i++;

while (arr[j] > pivot)

j--;

if(i <= j) {

if (arr[i] > arr[j]) {

int tmp = arr[i];

arr[i] = arr[j];

arr[j] = tmp;

}

i++;

j--;

}

} while (i <= j);

if (i < last)

\_sort(arr, i, last);

if (first < j)

\_sort(arr, first, j);

}

int\* sort(int\* array, const unsigned long& size) {

int\* result = new int[size];

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

result[i] = array[i];

}

\_sort(result, 0, size - 1);

return result;

}

}

**Тестирование:**

[kruyneg@matebook14build]$ ./first  
0                 - сменить реализацию   
1 {N}             - перевести число в систему счисления   
2 {SIZE} {ARRAY}  - отсортировать массив   
-1                - выход   
**1 2**   
10   
**2 3**   
3 2 1   
1 2 3    
**0**    
Библиотека была подключена на этапе линковки   
**-1**  
  
[kruyneg@matebook14build]$ ./second  
0                 - сменить реализацию   
1 {N}             - перевести число в систему счисления   
2 {SIZE} {ARRAY}  - отсортировать массив   
-1                - выход   
**1 2**   
10   
**2 3**   
3 2 1   
1 2 3    
**0**   
Перевод в троичную и сортировка Хоара   
**1 2**   
2   
**1 3**   
10   
**2 3**   
10 1 5   
1 5 10    
**0**   
Перевод в двоичную и сортировака пузырьком   
**-1**

**Вывод**

Проделав работу, я успешно написал программу, которая использовала динамические библиотеки, и могла менять реализацию используемых функций, в зависимости от подключенной библиотеки. Я получил практические навыки работы с динамическими библиотеками в ОС UNIX, были освоены навыки компиляции динамических библиотек. Одной из основных проблем являлось нахождение символа в библиотеке (необходимо было использовать extern “C” {...} ).

Динамические библиотеки используются для экономии ресурсов, так как динамическое подключение уменьшает размер выполняемого файла. Они позволяют программе легко изменять или дополнять функциональность программы без необходимости перекомпиляции исходного кода. Преимуществом динамических библиотек также является удобство обновления и исправления ошибок в функциях, содержащихся в библиотеке.