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: Эсмедляев Федор Романович

Группа: М8О–212Б–22

Вариант: 35

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 динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов

В конечном итоге, программа должна состоять из следующих частей:

* Динамическая библиотека, реализующая заданных вариантом интерфейс;
* Тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции;
* Тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс.

Провести анализ между обоими типами использования библиотеки.

Вариант 35:

7) Подсчет площади плоской геометрической фигуры подвум сторонам

Float Square(float A, float B)

Фигура прямоугольник

Фигура прямоугольный треугольник

9) Перевод числа x из десятичной системы счисления в другую

Int \* Sort(int \* array)

Пузырьковая сортировка

Сортировка Хоара

**Общие сведения о программе**

Программа компилируется из файлов compile.cpp и runtime.cpp, также используется заголовочный файл functions.hpp. В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамический общий объект (общую библиотеку) из файла, имя которого указано в строке filename (завершается null) и возвращает непрозрачный описатель на загруженный объект.
2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти(указывается одним из аргументов).
3. **dlclose** – уменьшает счётчик ссылок на динамически загружаемый общий объект, на который ссылается handle. Если счётчик ссылок достигает нуля, то объект выгружается. Все общие объекты, которые были автоматически загружены при вызове dlopen() для объекта, на который ссылается handle, рекурсивно закрываются таким же способом.

**Общий метод и алгоритм решения**.

Первый вариант исполнения реализуется в программе fast.cpp. При помощи cmake создаются 2 библиотеки с разными версиями имплементации функций. Далее одна из библиотек линкуется к исполняемому файлу.

Второй вариант исполнения реализуется в программе runtime.cpp. В нем поключение библиотек происходит при помощи системных вызовов. Также реализована возможность переключения между разными вариантами имплементации.

Различные имплементации представлены в файлах impl1.cpp и impl2.cpp.

**Основные файлы программы**

**fast.cpp:**

#include <iostream>

#include "func.h"

int main()

{

std::cout << "Программа подгружает библиотеки во время выполнения. Инструкции: " << '\n'

<< "\"1\" - вычислить площадь фигуры" << '\n'

<< "\"2\" - отсортировать массив" << '\n'

<< "\"-1\" - выход" << '\n'

<< "Введите инструкцию: ";

int option;

std::cin >> option;

while (option != -1)

{

switch (option)

{

case -1:

break;

case 1:

float A, B;

std::cin >> A >> B;

std::cout << "Площадь фигуры: " << S(A, B) << '\n'

<< "Введите инструкцию: ";

std::cin >> option;

break;

case 2:

int size;

std::cin >> size;

int \*m = new int[size];

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

{

std::cin >> m[i];

}

int\* mas = sort(m, size);

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

{

std::cout << mas[i] << " ";

}

std::cout << '\n' << "Введите инструкцию: ";

std::cin >> option;

//delete [] mas;

//delete [] m;

break;

}

}

return 0;

}

**runtime.cpp:**

#include <dlfcn.h>

#include <iostream>

int main()

{

std::cout << "Программа подгружает библиотеки во время выполнения. Инструкции: " << '\n'

<< "\"0\" - сменить реализацию" << '\n'

<< "\"1\" - вычислить площадь фигуры" << '\n'

<< "\"2\" - отсортировать массив" << '\n'

<< "\"-1\" - выход" << '\n'

<< "Введите инструкцию: ";

int option;

std::cin >> option;

int impl = 1;

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

char\* error;

error = dlerror();

if (error) {

std::cerr << "Error loading libimpl1.so: " << error << '\n';

return 1;

}

double (\*S)(double, double);

S = reinterpret\_cast<decltype(S)>(dlsym(handle, "S"));

error = dlerror();

if (error) {

std::cerr << "Error loading symbol 'S': " << error << '\n';

return 1;

}

int\* (\*sort)(int\*, int);

sort = reinterpret\_cast<decltype(sort)>(dlsym(handle, "sort"));

error = dlerror();

if (error) {

std::cerr << "Error loading symbol 'sort': " << error << '\n';

return 1;

}

while (option != -1)

{

switch (option)

{

case -1:

break;

case 0:

dlclose(handle);

error = dlerror();

if (error)

{

std::cerr << "Error closing library: " << error << '\n';

return 1;

}

switch (impl)

{

case 1:

impl = 2;

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

std::cout << "Switched to libimpl2.so\n";

break;

case 2:

impl = 1;

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

std::cout << "Switched to libimpl1.so\n";

break;

}

error = dlerror();

if (error)

{

std::cerr << "Error loading new library: " << error << '\n';

return 1;

}

S = reinterpret\_cast<decltype(S)>(dlsym(handle, "S"));

error = dlerror();

if (error)

{

std::cerr << "Error loading symbol 'S': " << error << '\n';

return 1;

}

sort = reinterpret\_cast<decltype(sort)>(dlsym(handle, "sort"));

error = dlerror();

if (error)

{

std::cerr << "Error loading symbol 'sort': " << error << '\n';

return 1;

}

std::cout << "Реализация изменена" << '\n'

<< "Введите инструкцию: ";

std::cin >> option;

break;

case 1:

double A, B;

std::cin >> A >> B;

std::cout << "Площадь фигуры: " << S(A, B) << '\n'

<< "Введите инструкцию: ";

std::cin >> option;

break;

case 2:

int size;

std::cin >> size;

int \*m = new int[size];

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

{

std::cin >> m[i];

}

int\* mas = sort(m, size);

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

{

std::cout << mas[i] << " ";

}

std::cout << '\n' << "Введите инструкцию: ";

std::cin >> option;

//delete [] mas;

//delete [] m;

break;

}

}

dlclose(handle);

error = dlerror();

if (error) {

std::cerr << "Error closing library: " << error << '\n';

return 1;

}

return 0;

}

**func.h:**

#pragma once

extern "C"

{

double S(double stor1, double stor2);

}

extern "C"

{

int\* sort(int\* mas, int size);

}

**impl1.cpp:**

#include <bits/stdc++.h>

#include "func.h"

extern "C"

{

double S(double stor1, double stor2)

{

//std::cout << "2312312313\n";

return stor1 \* stor2;

}

}

extern "C"

{

int\* sort(int\* mas, int size)

{

int\* v = new int[size];

v = mas;

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

{

v[i] = mas[i];

}

bool f = true;

while (f)

{

f = false;

for (int i = 1; i < size; i++)

{

if (v[i - 1] > v[i])

{

std::swap(v[i-1], v[i]);

f = true;

}

}

}

return v;

}

}

**impl2.cpp:**

#include <bits/stdc++.h>

#include "func.h"

extern "C"

{

double S(double stor1, double stor2)

{

//std::cout << "2312312313\n";

return stor1 \* stor2 / 2;

}

}

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, int size)

{

int\* result = new int[size];

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

{

result[i] = array[i];

}

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

return result;

}

}

**CMakeLists.txt:**

cmake\_minimum\_required(VERSION 3.10)

project(Dynamic\_Libraries)

set(CMAKE\_CXX\_STANDARD\_REQUIRED ON)

set(CMAKE\_CXX\_STANDARD 17)

add\_library(impl1 SHARED impl1.cpp)

add\_library(impl2 SHARED impl2.cpp)

add\_executable(runtime runtime.cpp)

target\_link\_libraries(runtime impl2)

**Пример работы**

**Test 1:**

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab4/build$ "/home/fedorubuntu/Документы/OS/lab4/build/runtime"

Программа подгружает библиотеки во время выполнения. Инструкции:

"0" - сменить реализацию

"1" - вычислить площадь фигуры

"2" - отсортировать массив

"-1" - выход

Введите инструкцию: 1

2 3

Площадь фигуры: 6

Введите инструкцию: 1

4 6

Площадь фигуры: 24

Введите инструкцию: 0

Switched to libimpl2.so

Реализация изменена

Введите инструкцию: 1

2 3

Площадь фигуры: 3

Введите инструкцию: 1

4 6

Площадь фигуры: 12

Введите инструкцию: -1

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab4/build$

**Test 2:**

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab4/build$ "/home/fedorubuntu/Документы/OS/lab4/build/runtime"

Программа подгружает библиотеки во время выполнения. Инструкции:

"0" - сменить реализацию

"1" - вычислить площадь фигуры

"2" - отсортировать массив

"-1" - выход

Введите инструкцию: 2

3

3 2 1

1 2 3

Введите инструкцию: 2

3

2 1 3

1 2 3

Введите инструкцию: 0

Switched to libimpl2.so

Реализация изменена

Введите инструкцию: 2

3

2 1 3

1 2 3

Введите инструкцию: 2

5

5 4 3 2 1

1 2 3 4 5

Введите инструкцию: -1

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab4/build$

**Test 3:**

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab4/build$ "/home/fedorubuntu/Документы/OS/lab4/build/runtime"

Программа подгружает библиотеки во время выполнения. Инструкции:

"0" - сменить реализацию

"1" - вычислить площадь фигуры

"2" - отсортировать массив

"-1" - выход

Введите инструкцию: 0

Switched to libimpl2.so

Реализация изменена

Введите инструкцию: 1

4 6

Площадь фигуры: 12

Введите инструкцию: 0

Switched to libimpl1.so

Реализация изменена

Введите инструкцию: 1

4 6

Площадь фигуры: 24

Введите инструкцию: 2

4

7 12 19 4

4 7 12 19

Введите инструкцию: 0

Switched to libimpl2.so

Реализация изменена

Введите инструкцию: 2

10

10 9 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 9 10

Введите инструкцию: -1

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/OS/lab4/build$

**Вывод**

Во время выполнения данной работы я познакомился с динамическими библиотеками. Я изучил создание библиотек при помощи CMake, а также мною было изучено подключение библиотек 2 способами: на этапе компиляции и на этапе исполнения программы. Подключение библиотек на этапе компиляции позволяет динамически изменять функции в зависимости от подключенной библиотеки, но замедляет время выполнения, так как требуется время на подключение библиотек.