Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работ №5 по курсу

«Операционные системы»

Группа: М80 – 207Б-18
Студент: Цапков Александр Максимович
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____

Москва, 2019.

Содержание

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Демонстрация работы программы
6. Вывод

Постановка задачи.

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: вектор целочисленных значений

Общие сведения о программе

Программа состоит из одного исполняемого файла и 2х файлов для библиотеки vector.cpp и Vector.hpp. В лабораторной работе были использованы следующие системные вызовы.

1. **dlopen** — открывает библиотку
2. **dlsym** – ищет адрес того что мы передали (к примеру функцию) в ранее открытой нами библиотеке.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Реализовать библиотеку с функциями для работы с вектором (C Style, не классами). Также для работы с dlsym нужно объявлять прототипы функций через extern “C”.
2. Написать меню для теста этой библиотеки.
3. В задании с подключением через линковку мы настраиваем мэйкфайл для компилирования библиотеки в динамическую и подключаем ее (На этапе же линковки)
4. В задание с «загрузкой функций на ходу» мы подключаем специальную библиотеку для работы с динамической библиотекой. Также переделываем главный исполняемый файл, где мы должны открывать динамическую библиотеку и искать там нужные нам функции, используем их и снова закрываем библиотеку.

Основные файлы программы.

Файл main.cpp

```
//#include "vector.cpp"
#include "Vector.hpp"
#include <iostream>
#include <dlfcn.h>

vector* vector_create_method()
{
	void *dl_handle;
	vector* (*func)();
	char *error;

	dl_handle = dlopen("libvector.so", RTLD_LAZY);
	if (!dl_handle) {
		printf("#%s\n", dlerror());
		return nullptr;
	}

	func = (vector* (*)())dlsym(dl_handle, "vector_create");
	error = dlerror();
	if (error != NULL) {
		printf("#%s\n", error);
		return nullptr;
	}

	vector* out = func();

	dlclose( dl_handle );
	return out;
}

void vector_push_back_method(vector* argument1, int argument2)
{
	void *dl_handle;
	void (*func)(vector*, const int&);
	char *error;
```

```
	dl_handle = dlopen("libvector.so", RTLD_LAZY);
	if (!dl_handle) {
		printf("# %s\n", dlerror());
		return;
	}

	func = (void (*)(vector*, const int&))dlsym(dl_handle,
"vector_push_back");
	error = dlerror();
	if (error != NULL) {
		printf("# %s\n", error);
		return;
	}

	func(argument1, argument2);

	dlclose( dl_handle );
	return;
}

int& vector_at_method(vector* argument1, int argument2)
{
	void *dl_handle;
	int& (*func)(vector*, int);
	char *error;

	dl_handle = dlopen("libvector.so", RTLD_LAZY);
	if (!dl_handle) {
		printf("# %s\n", dlerror());
		exit(-1);
	}

	func = (int& (*)(vector*, int))dlsym(dl_handle, "vector_at");
	error = dlerror();
	if (error != NULL) {
		printf("# %s\n", error);
		exit(-1);
```

```
        }

        int& out = func(argument1, argument2);

        dlclose( dl_handle );
        return out;
}

void vector_destroy_method(vector** argument1)
{
        void *dl_handle;
        void (*func)(vector**);
        char *error;

        dl_handle = dlopen("libvector.so", RTLD_LAZY);
        if (!dl_handle) {
                printf("# %s\n", dlerror());
                return;
        }

        func = (void (*)(vector**))dlsym(dl_handle, "vector_destroy");
        error = dlerror();
        if (error != NULL) {
                printf("# %s\n", error);
                return;
        }

        func(argument1);

        dlclose( dl_handle );
        return;
}

int main() {
   vector* vec = vector_create_method();
   char com = '0';
   int pos, val;
```

```
    while(1) {
        std::cin >> com;
        switch(com) {
            case 'p':
                std::cin >> val;
                vector_push_back_method(vec, val);
                break;
            case 'c':
                std::cin >> pos >> val;
                vector_at_method(vec, pos) = val;
                break;
            case 'o':
                std::cin >> pos;
                std::cout << vector_at_method(vec, pos) << '\n';
                break;
            case 'q':
                vector_destroy_method(&vec);
                return 0;
                break;
            default:
                std::cout << "err\n";
                break;
        }
    }
}
```

Пример работы программы

Я приведу пример работы с загрузкой функций в коде, но пример работы с линковкой ни чем не отличается кроме мэйкфайла.

```
iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab5/src/DLoad#
make
g++ -Wall -pedantic -c main.cpp
g++ -Wall -pedantic -c -fPIC vector.cpp
g++ -shared -o libvector.so vector.o
g++  -rdynamic -o main.out main.o -ldl -L. -lvector -Wl,-
rpath,.
iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab5/src/
DLoad# ./main.o
main.o    main.out
iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab5/src/
DLoad# ./main.out
p 1
p 2
p 228
p 322
o 0
1
o 1
2
o 2
228
o 3
322
c 3 137
o 3
137
p 99
o 4
99
c 0 0
o 0
0
q
iPad-Alex:~/Documents/Study/Labs/OS/sem3os/lab5/src/DLoad#
```

Вывод

В данной лабораторной работе я познакомился с динамическими библиотеками. До этого я честно не знал что это такое, но все же использовал их, не зная что я делаю. Сейчас же мало того что я знаю что это, но еще и могу ее сам реализовать. Но увы, в моих лабораторных работах в институте они пока не очень нужны, так как вся прелесть динамических библиотек в том что их могут одновременно использовать несколько процессов не расходуя лишнюю память и ресурсы и подгружать только нужные функции, в то время как мои библиотеки достаточно малы и гораздо легче использовать статические библиотеки.