

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу
«Операционные системы»**

Динамические библиотеки

Студент: Пермяков Никита Александрович

Группа: М8О –208Б-19

Вариант: 33

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2020

Содержание

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Тестирование
6. Демонстрация работы программы
7. 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 33

1. Расчет значений числа e
2. Отсортировать целочисленный массив

Общие сведения о программе

Программа состоит из следующих файлов:

- 1) `main.cpp` - программа, которая использует библиотеку на этапе компиляции
- 2) `main_dynamic.cpp` - программа, которая подгружает необходимые данные из библиотеки во время исполнения программы.
- 3) `library.h`, `operation.cpp`, `translation.cpp`. – реализация библиотеки.

Статическая библиотека - коллекция объектных файлов, которые присоединяются к программе во время линковки программы. Таким образом статические библиотеки используются только при создании программы. Потом в работе самой программы они не принимают участие, в отличие от динамических библиотек.

Динамическая библиотека - созданная специальным образом библиотека, которая присоединяется к результирующей программе в два этапа.

- 1) Этап компиляции - линковщик встраивает в программу описания требуемых функций и переменных, которые присутствуют в библиотеке. Сами объектные файлы из библиотеки не присоединяются к программе.
- 2) Присоединение этих объектных файлов (кодов функций) осуществляет системный динамический загрузчик во время запуска программы. Загрузчик проверяет все библиотеки прилинкованные с программой на наличие требуемых объектных файлов, затем загружает их в память и присоединяет их в копии запущенной программы, находящейся в памяти.

Основные системные вызовы:

- 1) `void *dlopen(const char *filename, int flag)` - загружает динамическую библиотеку, имя которой указано в строке *filename*, и возвращает прямой указатель на начало динамической библиотеки. Если *filename* не является полным именем файла (т.е. не начинается с "/"), то файл ищется в следующих местах:
 - в разделенном двоеточием списке каталогов, в переменной окружения пользователя `LD_LIBRARY_PATH`.
 - в списке библиотек, кэшированных в файле `/etc/ld.so.cache`.
 - в `usr/lib` и далее в `/lib`

flag=RTLD_LAZY, подразумевает разрешение неопределенных символов в виде кода, содержащегося в исполняемой динамической библиотеке.

- 2) void *dlsym(void *handle, char *symbol) - передается указатель на объект, возвращаемый вызовом dlopen(3) и имя символа (с null в конце). В результате функция возвращает адрес, по которому символ расположен в памяти. Если символ не найден в указанном объекте или во всех общих объектах, которые были автоматически загружены dlopen(3) на момент загрузки объекта, то dlsym() возвращает NULL (поиск, выполняемый dlsym(), охватывает всё дерево зависимостей этих общих объектов).
- 3) int dlclose(void *handle) - уменьшает на единицу счетчик ссылок на указатель динамической библиотеки handle. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается. При успешном завершении возвращает 0.
- 4) dlerror — возвращает читабельную строку, описывающую последнюю возникшую ошибку, возникшую при взаимодействии с динамической библиотекой.

Необходимые ключи g++ для создания и использования динамической библиотеки:

- 1) -shared — ключ, необходимый для создания shared object файла, т.е. самой динамической библиотеки
- 2) -L. — ключ указывает ранее прописанный путь для динамической библиотеки

```
LD_LIBRARY_PATH=/root:${LD_LIBRARY_PATH}
export LD_LIBRARY_PATH
```

указывается путь, где лежат все пользовательские библиотеки

- 3) -fPIC — генерируемый компилятором код становится независимым от адресов, связано это с тем, что все объектные файлы создаваемые обычным образом не имеют представления о том в какие адреса памяти

будет загружена использующая их программа. Несколько различных программ могут использовать одну библиотеку, и каждая из них располагается в различном адресном пространстве. Поэтому требуется, чтобы переходы в функциях библиотеки (операции **goto** на ассемблере) использовали не абсолютную адресацию, а относительную.

Общий метод и алгоритм решения

Линковка

1. Файлы компилируются с ключом -fPIC в объектные файлы.
2. Из объектных файлов собирается динамическая библиотекаso.
3. Главный файл компилируется и линкуется с динамической библиотекой dl.

Системные вызовы

1. Динамическая библиотека загружается при помощи dlopen.
2. Функции из динамической библиотеки подгружаются при помощи dlsym.

Листинг программы

main.cpp

```
#include<iostream>
```

```
#include<dlfcn.h>
```

```
void *handleLib = nullptr;
```

```
static bool mode = false;
```

```
float (* calcE)(int x) = nullptr;
```

```
int* (* Sort)(int * array, uint64_t&& n) = nullptr;
```

```
char *error;
```

```

void loadDLibs(){
    const char *name;

    if(mode){
        name = "operation.so";
    } else {
        name = "translation.so";
    }

    handleLib = dlopen(name, RTLD_LAZY);
    if(!handleLib){
        fprintf(stderr, "%s\n", dlerror());
        exit(EXIT_FAILURE);
    }
}

void closeLib(){
    dlclose(handleLib);
}

void openLib(){
    loadDLibs();

    float* calcE = (float*) dlsym(handleLib, "calcE");
    int* Sort = (int*) dlsym(handleLib, "Sort");

    if(error = dlerror()) {
        fprintf(stderr, "%s\n", error);
        exit(EXIT_FAILURE);
    }
}

```

```
    }  
}
```

```
void changeContext(){  
    closeLib();  
  
    mode = (bool(mode)) ? false : true;  
    std::cout << "Contract was changed" << std::endl;  
  
    openLib();  
}
```

```
inline void menu(){  
    std::cout << "1. Calc (1 + 1/x) ^ x" << std::endl;  
    std::cout << "2. Sort Hoarry" << std::endl;  
}
```

```
int main(){  
    mode = false;  
    openLib();  
  
    uint16_t cmd = 0;  
    bool inloop;  
    char c;  
    int tmp;  
  
    int x;  
  
    int* array;  
    uint64_t size, capacity;
```



```

do{
    menu();
    std::cin >> cmd;
    std::cin.get();

    switch(cmd){
        case 0:{
            changeContext();
            break;
        }
        case 1:{
            std::cout << std::endl << "x: ";
            std::cin >> x;
            std::cout << "Result calcE(x): " << calcE(x) << std::endl << std::endl;
            break;
        }
        case 2:{
            inloop = true;

            capacity = 1;
            size = 0;
            array = nullptr;
            std::cout << "array: ";

            while (inloop) {
                tmp = 0;
                while(true){
                    c = std::cin.get();
                    if (c == '\n') {

```

```

        inloop = false;
        break;
    } else if (c == ' ') {
        break;
    }
    tmp = tmp * 10 + (int)c - '0';
}

```

```

if (size == capacity - 1){
    capacity = capacity * 2;
    int * new_array = new int[capacity];
    for(uint64_t i = 0; i < size; ++i)
        new_array[i] = array[i];
    delete[] array;
    array = new_array;
    new_array = nullptr;
}

```

```

array[size] = (int)tmp;
++size;
capacity = sizeof(array) / sizeof(array[0]);
}

```

```

Sort(array, size - 1);
std::cout << "Result Sort(array): ";
for (uint64_t i = 0; i < size; ++i)
    std::cout << array[i] << " ";
std::cout << std::endl << std::endl;

```

```

array = nullptr;
break;

```

```

    }
    default:{
        std::cout << std::endl << "[ERROR] Key is not defined" << std::endl;
    }
}
} while(true);
closeLib();
}

```

main_dynamic.cpp

```
#include<iostream>
```

```
#include<dlfcn.h>
```

```
void *handleLib = nullptr;
```

```
static bool mode = false;
```

```
float (* calcE)(int x) = nullptr;
```

```
int* (* Sort)(int * array, uint64_t&& n) = nullptr;
```

```
char *error;
```

```
void loadDLibs(){
```

```
    const char *name;
```

```
    if(mode){
```

```
        name = "operation.so";
```

```
    } else {
```

```

    name = "translation.so";
}

handleLib = dlopen(name, RTLD_LAZY);
if(!handleLib){
    fprintf(stderr, "%s\n", dlerror());
    exit(EXIT_FAILURE);
}

}

void closeLib(){
    dlclose(handleLib);
}

void openLib(){
    loadDLibs();

    float* calcE = (float*) dlsym(handleLib, "calcE");
    int* Sort = (int*) dlsym(handleLib, "Sort");

    if(error = dlerror()) {
        fprintf(stderr, "%s\n", error);
        exit(EXIT_FAILURE);
    }
}

void changeContext(){
    closeLib();

    mode = (bool(mode)) ? false : true;

```

```
std::cout << "Contract was changed" << std::endl;
```

```
openLib();
```

```
}
```

```
inline void menu(){
```

```
    std::cout << "1. Calc  $(1 + 1/x)^x$ " << std::endl;
```

```
    std::cout << "2. Sort Hoarry" << std::endl;
```

```
}
```

```
int main(){
```

```
    mode = false;
```

```
    openLib();
```

```
    uint16_t cmd = 0;
```

```
    bool inloop;
```

```
    char c;
```

```
    int tmp;
```

```
    int x;
```

```
    int* array;
```

```
    uint64_t size, capacity;
```

```
    do{
```

```
        menu();
```

```
        std::cin >> cmd;
```

```
        std::cin.get();
```

```
        switch(cmd){
```

```

case 0:{
    changeContext();
    break;
}
case 1:{
    std::cout << std::endl << "x: ";
    std::cin >> x;
    std::cout << "Result calcE(x): " << calcE(x) << std::endl << std::endl;
    break;
}
case 2:{
    inloop = true;

    capacity = 1;
    size = 0;
    array = nullptr;
    std::cout << "array: ";

    while (inloop) {
        tmp = 0;
        while(true){
            c = std::cin.get();
            if (c == '\n') {
                inloop = false;
                break;
            } else if (c == ' ') {
                break;
            }
            tmp = tmp * 10 + (int)c - '0';
        }
    }
}

```

```

        if (size == capacity - 1){
            capacity = capacity * 2;
            int * new_array = new int[capacity];
            for(uint64_t i = 0; i < size; ++i)
                new_array[i] = array[i];
            delete[] array;
            array = new_array;
            new_array = nullptr;
        }

        array[size] = (int)tmp;
        ++size;
        capacity = sizeof(array) / sizeof(array[0]);
    }
    Sort(array, size - 1);
    std::cout << "Result Sort(array): ";
    for (uint64_t i = 0; i < size; ++i)
        std::cout << array[i] << " ";
    std::cout << std::endl << std::endl;

    array = nullptr;
    break;
}
default:{
    std::cout << std::endl << "[ERROR] Key is not defined" << std::endl;
}
}
} while(true);
closeLib();

```

```
}
```

library.h

```
#pragma once
```

```
#include<iostream>
```

```
#include <math.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

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

```
extern "C" int * Sort(int * array, uint64_t&& n);
```

LibTrans.cpp

```
char* translation(long x){
    char *res = (char*)malloc(sizeof(char));
    res[0] = '\0';
    if(!x){
        char *tmp = (char*)malloc((strlen(res) + 2) * sizeof(char));
        strcpy(tmp+1, res);
        free(res);
        res = tmp;
        res[0] = '0';
    }

    while(x > 0){
        char ch = (x % 3) + '0';
        x = x / 3;
        char *tmp = (char*)malloc((strlen(res) + 2) * sizeof(char));
        strcpy(tmp+1, res);
        free(res);
        res = tmp;
        res[0] = ch;
    }
    return res;
}
```


LibSort.cpp

```
#include "library.h"
```

```
void sortHoary(int* array, uint64_t& first, uint64_t& last) {
    uint64_t i = first;
    uint64_t j = last;
    int tmp;
    int middle = array[(first + last) / 2];
    do {
        while (array[i] < middle)
            ++i;
        while (array[j] > middle)
            --j;
        if (i <= j) {
            if (i < j) {
                tmp = array[i];
                array[i] = array[j];
                array[j] = tmp;
            }
            ++i;
            --j;
        }
    } while (i <= j);

    if (i < last)
        sortHoary(array, i, last);
    if (first < j)
        sortHoary(array, first, j);
}

int * Sort(int * array, uint64_t&& n){
    uint64_t s = 0;
    if (n < 2)
        throw std::runtime_error("[ERROR] count element of array must be great 2");
    sortHoary(array, s, n);
    return array;
}
```

CmakeList.txt

```
cmake_minimum_required(VERSION 3.16)
project(5_server_msg LANGUAGES CXX VERSION 0.1.0)
```

```
set(CMAKE_CXX_STANDARD 17)
```

```
add_executable(main main.cpp)
```

```
add_executable(main_dynamic main_dynamic.cpp)
```

```
add_library(operation SHARED operation.cpp)
```

```
set_target_properties(operation PROPERTIES
```

VERSION

```
${PROJECT_VERSION} PUBLIC_HEADER library.h)
```

```
add_library(translation SHARED translation.cpp)
```

```
set_target_properties(translation PROPERTIES
```

VERSION

```
${PROJECT_VERSION} PUBLIC_HEADER translation.h)
```

```
target_link_libraries(operation m)
```

```
target_link_libraries(translation m)
```

```
set(custom_targets)
```

```
list(APPEND custom_targets operation)
```

```
list(APPEND custom_targets translation)
```

```
target_link_libraries(main_dynamic ${CMAKE_DL_LIBS})
```

```
add_dependencies(main_dynamic operation translation)
```

Тестирование

0. Change library

1. Calc to bin

1

x: 65

Result translation(x): 2102

0. Change library

1. Calc to bin

1

x: 5

Result translation(x): 12

0. Change library

1. Calc to bin

0

Contract was changed

0. Change library

2. Sort Hoarry

2

array: 45 56 76 43

Result Sort(array): 0 43 45 56 76

Выводы

Файлы, использующие динамические библиотеки на практике в больших проектах занимают гораздо меньший размер, чем файлы, использующие статические. Необходимости перекомпилировать библиотеку в случае изменений основного файла программы и использовать одну библиотеку для нескольких различных проектов.