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: Мирошников Дмитрий Евгеньевич
Группа: М8О-210Б-22
Вариант: 15
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 определённый 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 её интерфейс.

Провести анализ между обоими типами использования библиотеки.

Вариант 15: контракты 2 и 9

2	Рассчет производной функции	Float	f'(x) = (f(A + deltaX)	f'(x) = (f(A + deltaX)
	cos(x) в точке A с приращением deltaX	Derivative(float A, float deltaX)	– f(A))/deltaX	– f(A-deltaX))/(2*deltaX)

9	Отсортировать целочисленный массив	Int * Sort(int * array)	Пузырьковая сортировка	Сортировка Хоара

Общие сведения о программе

Программа компилируется при помощи утилиты CMake. Также используется заголовочные файлы: stdio.h, math.h, stdlib.h, string.h, dlfcn.h. 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. Принимаемые параметры: **const char* filename** – путь до файла с динамической библиотекой (.so), **int flag** – определенное условие подключение библиотеки.
2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти(указывается одним из аргументов). Принимаемые параметры: **void* handle** – возвращаемое значение выполняемой функции dlopen, **char* symbol** – является строкой, в которой содержится название символа, который необходимо загрузить из библиотеки.
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. Принимаемые параметры: **void* handle** – возвращаемое значение выполняемой функции dlopen.
4. **dlerror –** возвращает указатель на начало строки, описывающей ошибку, полученную на пердыдущем вызове.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы dlsym, dlopen, dlclose.
2. Написать библиотеку realization.h, для работы с двумя реализациями контрактов realization1.c и realization2.c.
3. Организовать простейший командный интерфейс в файлах dynMain.c и statMain.c.
4. В файле statMain.c подключить библиотеку на этапе компиляции.
5. В файле dynMain.c загрузить библиотечные функции в runtime, с помощью dlsym, dlopen, dlclose.

Основные файлы программы

realization.hpp

```
#pragma once

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

```
int ArraySize;
```

```
extern "C" {
    float Derivative(float A, float deltaX);
    int* Sort(int* array);
}
```

realization_1.cpp

```
#include "realization.hpp"

float Derivative(float A, float deltaX) {
    return (cosf(A + deltaX) - cosf(A)) / deltaX;
}
```

```
int* Sort(int* array) {
    int i, j, temp;
```

```
    for (i = 0; i < ArraySize - 1; ++i) {
        for (j = 0; j < ArraySize - i - 1; ++j) {
            if (array[j] > array[j + 1]) {
                temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }

    return array;
}
```

realization_2.cpp

```
#include "realization.hpp"

float Derivative(float A, float deltaX) {
    return (cosf(A + deltaX) - cosf(A - deltaX)) / (2 * deltaX);
}
```

```
int* Sort(int* array) {
    int base, left, right, i, j;
    base = left = right = i = j = 0;
    std::stack<int> st;
    st.push(ArraySize - 1);
    st.push(0);
    while (!st.empty()) {
        left = st.top();
        st.pop();
        right = st.top();
        st.pop();
        if (((right - left) == 1) && (array[left] > array[right])) {
            std::swap(array[left], array[right]);
```

```
        } else {
            base = array[(left + right) / 2];
            i = left;
            j = right;
            while (i <= j) {
                while (base > array[i]) {
                    ++i;
                }
                while (base < array[j]) {
                    --j;
                }
                if (i <= j) {
                    std::swap(array[i], array[j]);
                    ++i;
                    --j;
                }
            }
        }
        if (left < j) {
            st.push(j);
            st.push(left);
        }
        if (right > i) {
            st.push(right);
            st.push(i);
        }
    }
    return array;
}
```

Dynamic_Main.cpp

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include <malloc.h>

typedef enum {
    FIRST,
    SECOND,
} Contract;
```

```
Contract current = FIRST;
```

```
const char* lib_name1 = "./lib_first.so";
const char* lib_name2 = "./lib_second.so";
```

```
float (*Derivative)(float A, float deltaX) = NULL;
int* (*Sort)(int* array) = NULL;
int* ArraySizePtr;
```

```
void* libHandle = NULL;
```

```
void LoadDynamicLib(Contract con) {
    if (con == FIRST) {
        libHandle = dlopen(lib_name1, RTLD_LAZY);
    } else if (con == SECOND) {
        libHandle = dlopen(lib_name2, RTLD_LAZY);
```

```
    } else {
        std::cerr << "Contract if error\n";
        exit(EXIT_FAILURE);
    }
```

```
    if (!libHandle) {
        std::cerr << dlerror();
        exit(EXIT_FAILURE);
    }
}
```

```
void LoadContract() {
    LoadDynamicLib(current);
    Derivative = (float (*)(float, float))dlsym(libHandle, "Derivative");
    Sort = (int* (*)(int*))dlsym(libHandle, "Sort");
    ArraySizePtr = (int*)dlsym(libHandle, "ArraySize");

    if (Derivative == NULL || Sort == NULL || ArraySizePtr == NULL) {
        std::cerr << "dlsym error\n";
    }
}
```

```
void UnloadDynamicLib() {
    dlclose(libHandle);
}
```

```
void ChangeContract() {
    UnloadDynamicLib();
```

```
    if (current == FIRST) {
        current = SECOND;
    } else {
        current = FIRST;
    }
```

```
    LoadContract();
}
```

```
void GuideToUse() {
    std::cout << "Hello user, there are list of commands you can use\n";
    std::cout << "0 - change current contract\n";
    std::cout << "1 - get first function result\n";
    std::cout << "2 - get second function result\n";
}
```

```
int main() {
    LoadContract();
```

```
    int cmd = 0;
```

```
    GuideToUse();
    while(std::cin >> cmd) {
        switch (cmd) {
            case 0:
               ChangeContract();
               if (current == FIRST) {
                std::cout << "Contract was changed to first\n";
               } else {
                std::cout << "Contract was changed to second\n";
               }
```

```
            break;
        case 1:
            float A, deltaX;
            std::cin >> A >> deltaX;
            std::cout << "The value of Derivative is: " << Derivative(A, deltaX) << '\n';
            break;
        case 2:
            int* array;
            scanf("%d", ArraySizePtr);
            array = (int*)malloc(*ArraySizePtr * sizeof(int));
            for (int i = 0; i < *ArraySizePtr; ++i) {
                std::cin >> array[i];
            }
            array = Sort(array);
            std::cout << "Sorted array: ";
            for (int i = 0; i < *ArraySizePtr; ++i) {
                std::cout << array[i] << ' ';
            }
            std::cout << '\n';
            free(array);
            array = nullptr;
            break;
        default:
            std::cout << "There are no commands with that id\n";
            break;
    }
}
UnloadDynamicLib();
```

```
    return 0;
}
```

Static_Main.cpp

```
#include "realization.hpp"

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

```
void GuideToUse() {
    std::cout << "Hello user, there are list of commands you can use\n";
    std::cout << "1 - get first function result\n";
    std::cout << "2 - get second function result\n";
}
```

```
int main() {
    int cmd = 0;
```

```
    GuideToUse();
    while(std::cin >> cmd) {
        switch (cmd) {
        case 1:
            float A, deltaX;
            std::cin >> A >> deltaX;
            std::cout << "The value of Derivative is: " << Derivative(A, deltaX) << '\n';
            break;
        case 2:
            int* array;
```

```
            std::cin >> ArraySize;
            array = (int*)malloc(sizeof(int) * ArraySize);
            for (int i = 0; i < ArraySize; ++i) {
                std::cin >> array[i];
            }
            array = Sort(array);
            std::cout << "Sorted array: ";
            for (int i = 0; i < ArraySize; ++i) {
                std::cout << array[i] << ' ';
            }
            std::cout << '\n';
            free(array);
            array = nullptr;
            break;
        default:
            std::cout << "There are no commands with that id\n";
            break;
        }
    }
```

```
    return 0;
}
```

Пример работы программы

dmitrijmrsh@LAPTOP-7SMT8REA:~/Labs-git/os_lab_4/src/build$ ./dynamic_main

Hello user, there are list of commands you can use

0 - change current contract

1 - get first function result

2 - get second function result

1 2 2

The value of Derivative is: -0.118748

2 4 4 2 1 3

Sorted array: 1 2 3 4

0

Contract was changed to second

1 2 2

The value of Derivative is: -0.413411

2 4 4 2 1 3

Sorted array: 1 2 3 4

```
dmitrijmrsh@LAPTOP-7SMT8REA:~/Labs-
git/os_lab_4/src/build$ ./static_main
Hello user, there are list of commands you can use
1 - get first function result
2 - get second function result
1 2 2
The value of Derivative is: -0.118748
2 4 4 2 1 3
Sorted array: 1 2 3 4
```

Вывод

Изучив работу динамических библиотек, я научился различать и работать с библиотеками, которые подключаются на этапе компиляции и в „runtime“. Прочитав мануал по библиотеке dlfcn.h я разобрался в нюансах и тонкостях использования ее функций. В будущем мне поможет навык работы с динамическими библиотеками, ведь зачастую использования „тонны“ include’ов и import’ов приводит к огромному нагромождению всевозможных функций и объектов. Грамотное и своевременное подключение библиотек позволит простой работе и меньшей затрате по памяти.