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: Железнов Илья Васильевич
Группа: М8О–210Б–22
Вариант: 7
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 7: контракты 1 и 8.

1	Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e	Float SinIntegral(float A, float B, float e)	Подсчет интеграла методом прямоугольников.	Подсчет интеграла методом трапеций.
8	Перевод числа x из десятичной системы счисления в другую	Char* translation(long x)	Другая система счисления двоичная	Другая система счисления троичная

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

dynMain.c:

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

typedef enum {
    FIRST,
    SECOND,
} CONTEXT;

CONTEXT r = FIRST;

const char* libName1 = "../cmake-build-debug/libfirst.so";
const char* libName2 = "../cmake-build-debug/libsecond.so";

float (*sinInt)(float, float, float) = NULL; // Pointer to sinus
integral function
char* (*translation)(long x) = NULL; // Pointer to translation
function
char* err;

void* libHandle = NULL;

// function to loaded dynamic library
void loadDynLib(CONTEXT con)
{
    const char* name;

    if (con == FIRST) {
        name = libName1;
    } else if (con == SECOND) {
        name = libName2;
    } else {
        puts("Error enter! exit 1");
        exit(EXIT_FAILURE);
    }

    libHandle = dlopen(name, RTLD_LAZY);

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

// completion function from dynamic library
void unloadDynLub()
{
    dlclose(libHandle);
}

// functions for loading into a dynamic library
```

```
void loadContext()
{
    loadDynLib(r);
    sinInt = dlsym(libHandle, "SinIntegral");
    translation = dlsym(libHandle, "translation");

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

// context change function
void changeContext()
{
    unloadDynLub();

    if (r == FIRST) {
        r = SECOND;
    } else {
        r = FIRST;
    }

    loadContext();
}

void doc2use()
{
    printf("\tWelcome to programm!\n");
    printf("To work with the program you can use 3
commands:\n");
    printf("\t|0| change the contract\n");
    printf("\t|1| use first function (and more arguments)\n");
    printf("\t|2| use second function (and more arguments)\n");
}

int main()
{
    r = FIRST;
    loadContext();

    int cmd = 0;

    doc2use();
    while(scanf("%d", &cmd) != EOF) {
        switch (cmd)
        {
        case 0:
            changeContext();
            puts("Contract was changed!");
            if (r == FIRST) {
                puts("\tNOW CONTEXT IS FIRST");
            } else {
```

```
                puts("\tNOW CONTEXT IS SECOND");
            }
            break;
        case 1:
            float A, B, e;
            if (scanf("%f %f %f", &A, &B, &e) == EOF) {
                break;
            }
            printf("%f\n", sinInt(A, B, e));
            break;
        case 2:
            long x;
            if (scanf("%ld", &x) == EOF) {
                break;
            }

            char* str;
            printf("Translate decimal number '%ld' to ", x);

            if (r == FIRST) {
                printf("binary\n");
            } else {
                printf("ternary\n");
            }

            str = translation(x);
            printf("\t Result: %s\n", str);
            free(str);
        default:
            printf("Unknown command\n");
            break;
        }
    }

    unloadDynLub();

    return 0;
}
```

statMain.c

```
#include "realization.h"


#include <stdio.h>
#include <stdlib.h>

int main()
{
    int cmd = 0;

    while(scanf("%d", &cmd) != EOF) {
        switch (cmd)
```

```
        {
        case 1:
            float A, B, e;
            if (scanf("%f %f %f", &A, &B, &e) == EOF) {
                break;
            }

            puts("Calculate sin(x)");
            printf("%f\n", SinIntegral(A, B, e));
            break;
        case 2:
            long x;
            if (scanf("%ld", &x) == EOF) {
                break;
            }
            char* str;
            printf("Translation x to binary\n");
            printf("%s\n", str = translation(x));
            free(str);
        default:
            break;
        }
    }

    return 0;
}
```

realization.h

```
#ifndef __REALIZATION_H__
#define __REALIZATION_H__

extern float SinIntegral(float A, float B, float e);
extern char* translation(long x);

#endif // __REALIZATION_H__
```

realization1.c

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

#include "realization.h"

// rectangles method
float SinIntegral(float A, float B, float e)
{
    float step = e;
    int count = (int)((B - A) / step);
```

```
    float ans = 0;

    for (int i = 0; i <= count; ++i) {
        ans += step * sinf(A + (float)(i - 1) * step);
    }

    return ans;
}

// convert to binary
char* translation(long x)
{
    char* res = calloc(1, sizeof(char));
    res[0] = '\0';

    if (x == 0) {
        char* tmp = calloc(strlen(res) + 2, sizeof(char));
        strcpy(tmp + 1, res);
        free(res);
        res = tmp;
        res[0] = '0';
    }

    while (x > 0) {
        char c = (x & 1u) + '0';
        x >>= 1u;
        char* tmp = calloc(strlen(res) + 2, sizeof(char));
        strcpy(tmp + 1, res);
        free(res);
        res = tmp;
        res[0] = c;
    }

    return res;
}
```

realization2.c

```
#include <math.h>

#include <stdlib.h>
#include <string.h>

#include "realization.h"

// trapezoid method
float SinIntegral(float A, float B, float e)
{
    int n = (int)((B - A) / e);
    float range = (B - A) / (float)n;
    float sinDiap = sinf(A) + sinf(B);// sinus diapason
function's
```

```
    for (int i = 1; i < n; ++i) {
        sinDiap += 2 * sinf(A + ((float)i * range));
    }

    return (range * sinDiap) / 2;
}

// convert to ternary
char* translation(long x)
{
    char* res = calloc(1, sizeof(char));
    res[0] = '\0';

    if (x == 0) {
        char* tmp = calloc(strlen(res) + 2, sizeof(char));
        strcpy(tmp + 1, res);
        free(res);
        res = tmp;
        res[0] = '0';
    }

    while (x > 0) {
        char c = (x % 3) + '0';
        x /= 3;

        char* tmp = calloc(strlen(res) + 2, sizeof(char));
        strcpy(tmp + 1, res);
        free(res);
        res = tmp;
        res[0] = c;
    }

    return res;
}
```

Пример работы

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/oc_lab4/source/cmake-build-debug$
make statMain

[ 50%] Built target first

[ 75%] Building C object
CMakeFiles/statMain.dir/src/statMain.c.o

[100%] Linking C executable statMain

[100%] Built target statMain
```

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/oc_lab4/source/cmake-build-debug$ ./statMain
1 2 3 1
Calculate sin(x)
1.750768
2 123
Translation x to binary
1111011
keinpop@DESKTOP-T6SLHUS:/mnt/c/oc_lab4/source/cmake-build-debug$ make dynMain
[ 33%] Built target second
[ 66%] Built target first
Consolidate compiler generated dependencies of target dynMain
[100%] Built target dynMain
keinpop@DESKTOP-T6SLHUS:/mnt/c/oc_lab4/source/cmake-build-debug$ ./dynMain
        Welcome to programm!
To work with the program you can use 3 commands:
        |0| change the contract
        |1| use first function (and more arguments)
        |2| use second function (and more arguments)
0
Contract was changed!
        NOW CONTEXT IS SECOND
0
Contract was changed!
        NOW CONTEXT IS FIRST
1
1 2 1
0.841471
2
123
Translate decimal number '123' to binary
         Result: 1111011
0
```

```
Contract was changed!
        NOW CONTEXT IS SECOND
1
1 2 1
0.875384
2
123
Translate decimal number '123' to ternary
         Result: 11120
```

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.