Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу

«Операционные системы»

Тема работы

«Динамические библиотеки»

Студент: Лютоев Илья Александрович
Группа: М8О-207Б-21
Вариант: 32
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

Постановка задачи

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами:
Во время компиляции (на этапе «линковки»)
Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками
В конечном итоге, в лабораторной работе необходимо получить следующие части:
Динамические библиотеки, реализующие контракты, которые заданы вариантом;
Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.
Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:
Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2).
«1 arg1 arg2 … argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;
«2 arg1 arg2 … argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

Вариант 32

1. Рассчет значения числа e(основание натурального логарифма)
    1. Float E(int x)
    2. (1 + 1/x) ^ x
    3. Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!))
2. Перевод числа x из десятичной системы счисления в другую
1. Char* translation(long x)
    2. Другая система счисления двоичная
    3. Другая система счисления троичная
3. Сборочная система
1. CMake

2. Возможность сборки всех таргетов с ASAN без переопределения CMake флагов, если указана соответствующая переменная и ОС имеет поддержку ASAN.

Общий метод и алгоритм решения

Программа состоит из двух интерфейсов (main1.c и main2.c), каждый из них реализован по-разному, в соответствии с заданием. Также каждая реализация контрактов представляет из себя отдельный файл: lib1.c и lib2.c. Для объявления необходимых функций также используется заголовочный файл lib.h. Так как все собирается с помощью CMake, то в проекте присутствует CMakeLists.txt.
Объявим необходимые функции внутри файла lib.h. Используем спецификатор хранения extern.
Так как по заданию необходимо подключать библиотеки на этапе линковки, то подключать lib.h в реализации lib1.c и lib2.c не следует. В этих файлах просто напишем логику работы необходимых функций. Важно, чтобы они назывались также, как и те, что объявлены в lib.h.
Интерфейс 1:
Подключаем lib.h и пользуемся функциями так, как будто библиотека обычная. Различия наступают в сборке программы. Если бы мы собирали такой код в терминале, то прописали бы gcc -c -fpic lib1.c. Опция -fpic - требует от компилятора, при создании объектных файлов, порождать позиционно-независимый код. Формат позиционно-независимого кода позволяет подключать исполняемые модули к коду основной программы в момент её загрузки. Далее gcc -shared -o liblib1.so lib1.o -lm. Опция -shared - указывает gcc, что в результате должен быть собран не исполняемый файл, а разделяемый объект — динамическая библиотека.
Интерфейс 2:
Воспользуемся системными вызовами из библиотеки <dlfcn.h>.
Функция dlopen открывает динамическую библиотеку (объект .so) по названию.
Функция dlsym - обоаботчик динамически загруженного объекта вызовом dlopen.
Функция dlclose, соответственно, закрывает динамическую библиотеку.
Собираем с помощью gcc -L. -Wall -o main.out main2.c -llib2 -llib1. Флаг -L. Означает, что поиск файлов библиотек будет начинаться с текущей директории.
Система сборки:
ASAN — это Address Sanitizer, инструмент, с помощью которого можно ловить RE связанные с неправильным обращением к памяти.
Наиболее логичный способ их интеграции в CMake — интегрировать

их как типы сборки CMake, чтобы программы были созданы оптимально для санитайзеров. Для получения оптимальных результатов эти типы сборки игнорируют все другие флаги компилятора.

Исходный код

lib.h

```
#ifndef __LIB_H__
#define __LIB_H__

extern float E(int x);
extern char* translation(long x);

#endif
```

lib1.c

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

float bin_pow(float x, int y)
{
    float z = 1.0;
    while (y > 0) {
        if (y % 2 != 0) {
            z *= x;
        }
        x *= x;
        y /= 2;
    }
    return z;
}

float E(int x)
{
    printf("Вычисление числа e с апроксимацией %d\n", x);
    printf("Используя (1 + 1/x) ^ x\n");
    float base = (float) 1.0 + ((float) 1 / (float) x);
    float E = bin_pow(base, x);
    return E;
}

char* translation(long x)
{
    printf("Перевод %d в двоичную систему счисления\n", x); // binary
    int flag = 0;
    if (x < 0)
        flag = 1, x = -x;
    int longsize = 32;
    int cnt = 0;
    char *binary = (char *) malloc(longsize * sizeof(char));
    for (int i = 0; i < longsize; i++) {
        binary[i] = 's';
    }
    while(x > 0) {
```

```
        if (x % 2 == 1) {
            binary[longsize - cnt - 1] = '1';
        } else {
            binary[longsize - cnt - 1] = '0';
        }
        x = x / 2;
        cnt ++;
    }
    if (flag)
        binary[longsize - cnt - 1] = '-';

    return binary;
}
```

lib2.c

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

long fact(int x)
{
    if (x == 0)
        return 1;
    long n = 1;
    for (int i = 2; i <= x; i ++) {
        n *= i;
    }
    return n;
}

float machineEps(void)
{
    float e = 1.0f;
    while (1.0f + e / 2.0f > 1.0f)
        e /= 2.0f;
    return e;
}

float E(int x)
{
    printf("Вычисление числа e с апроксимацией %d\n", x);
    printf("используя сумму ряда по n от 0 до x, где элементы ряда равны: (1/(n!))\n");

    float maceps = machineEps();

    float E = 0;
    for (int n = 0; n <= x; n ++) {
        float sol = 1.0f / fact(n);
        float fsol = sol > 0 ? sol : (float) (-1) * sol;
        if (fsol <= maceps) {
            printf("Апроксимация сломалась из-за машинного нуля %.8f\n", maceps);
            break;
        }
        E += sol;
    }
```

```
    return E;
}

char* translation(long x)
{
    printf("Перевод %d в троичную систему счисления\n", x); // ternary
    int flag = 0;
    if (x < 0)
        flag = 1, x = -x;
    int longsize = 32;
    int cnt = 0;
    char *ternary = (char *) malloc(longsize * sizeof(char));
    for (int i = 0; i < longsize; i++) {
        ternary[i] = 's';
    }
    while(x > 0) {
        if (x % 3 == 1) {
            ternary[longsize - cnt - 1] = '1';
        } else if (x % 3 == 2) {
            ternary[longsize - cnt - 1] = '2';
        } else {
            ternary[longsize - cnt - 1] = '0';
        }
        x = x / 3;
        cnt ++;
    }
    if (flag)
        ternary[longsize - cnt - 1] = '-';

    return ternary;

}
```

main1.c

```
#include <stdio.h>
#include <stdlib.h>
#include "../include/lib.h"

int main()
{
    int com = 0;
    while (scanf("%d", &com) != EOF) {
        switch (com)
        {
        case 1:
            int a;
            scanf("%d", &a);
            printf("Ответ: %f\n", E(a));
            break;
        case 2:
            long b;
            scanf("%ld", &b);
            char* s = translation(b);
            printf("Ответ: ");
```

```
            for (int i = 0; i < 32; i ++) {
                if (s[i] == 's') {
                    continue;
                }
                printf("%c", s[i]);
            }
            printf("\n");
            free(s);
            break;
        default:
            printf("Неправильная команда\n");
            break;
        }
    }
}
```

main2.c

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

const char* lib1 = "./liblib1.so";
const char* lib2 = "./liblib2.so";

int main(int argc, char const *argv[])
{
    int command = 0;
    int link = 0;

    void *current_lib = dlopen(lib1, RTLD_LAZY);
    printf("\nТекущая библиотека - %d\n", link);

    char* (*translation)(long x);
    float (*E)(int x);

    translation = dlsym(current_lib, "translation");
    E = dlsym(current_lib, "E");

    while (scanf("%d", &command) != EOF) {
        switch (command) {
        case 0:
            dlclose(current_lib);
            if (link == 0) {
                current_lib = dlopen(lib2, RTLD_LAZY);
            } else {
                current_lib = dlopen(lib1, RTLD_LAZY);
            }
            link = !link;
            translation = dlsym(current_lib, "translation");
            E = dlsym(current_lib, "E");
            break;

        case 1:
            int x;
```

```
            scanf("%d", &x);
            printf("Ответ: %f\n", E(x));
            break;

        case 2:
            long b;
            scanf("%ld", &b);
            char* s = translation(b);
            printf("Ответ: ");

            for (int i = 0; i < 32; i ++) {
                if (s[i] == 's') {
                    continue;
                }
                printf("%c", s[i]);
            }
            printf("\n");
            free(s);
            break;

        default:
            printf("Неправильная команда\n");
            break;
        }
        printf("\nТекущая библиотека - %d\n", link);
    }
    return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.8 FATAL_ERROR)
project(main LANGUAGES C)

set(BUILD_WITH_ASAN 1)

add_library(
        lib1 SHARED
        ./include/lib.h
        ./src/lib1.c
)
add_library(
        lib2 SHARED
        ./include/lib.h
        ./src/lib2.c
)

add_executable(main1 ./src/main1.c)
target_include_directories(main1 PRIVATE ./include)
target_link_libraries(main1 PRIVATE lib1 m)

add_executable(main2 ./src/main1.c)
target_include_directories(main2 PRIVATE ./include)
target_link_libraries(main2 PRIVATE lib2 m)

add_executable(main ./src/main2.c)
```

```
target_include_directories(main PRIVATE ./include m)

if (${BUILD_WITH_ASAN})
   message("-- Adding sanitizers")
   target_compile_options(main PRIVATE -fsanitize=address)
   target_link_options(main PRIVATE -fsanitize=address)
   target_compile_options(main1 PRIVATE -fsanitize=address)
   target_link_options(main1 PRIVATE -fsanitize=address)
   target_compile_options(main2 PRIVATE -fsanitize=address)
   target_link_options(main2 PRIVATE -fsanitize=address)
endif()
```

Демонстрация работы программы

```
 lyutoev@lyutoev Ξ ~/workshop/os/OS/lab5/build Ξ  main ± Ξ ./main1
1
100
Вычисление числа e с апроксимацией 100
Используя (1 + 1/x) ^ x
Ответ: 2.704813
2
100
Перевод 100 в двоичную систему счисления
Ответ: 1100100
^C
 ✘ lyutoev@lyutoev Ξ ~/workshop/os/OS/lab5/build Ξ  main ± Ξ ./main2
1
100
Вычисление числа e с апроксимацией 100
используя сумму ряда по n от 0 до x, где элементы ряда равны: (1/(n!))
Апроксимация сломалась из-за машинного нуля 0.00000012
Ответ: 2.718282
2
100
Перевод 100 в троичную систему счисления
Ответ: 10201
^C
 ✘ lyutoev@lyutoev Ξ ~/workshop/os/OS/lab5/build Ξ  main ± Ξ ./main

Текущая библиотека - 0
1
100
Вычисление числа e с апроксимацией 100
Используя (1 + 1/x) ^ x
Ответ: 2.704813


Текущая библиотека - 0
2
100
Перевод 100 в двоичную систему счисления
Ответ: 1100100


Текущая библиотека - 0
0


Текущая библиотека - 1
1
100
```

```
Вычисление числа e с апроксимацией 100
используя сумму ряда по n от 0 до x, где элементы ряда равны: (1/(n!))
Апроксимация сломалась из-за машинного нуля 0.00000012
Ответ: 2.718282


Текущая библиотека - 1
2
100
Перевод 100 в троичную систему счисления
Ответ: 10201


Текущая библиотека - 1
```

Выводы

В ходе лабораторной работы я познакомился с созданием динамических библиотек в ОС Linux, а также с возможностью загружать эти библиотеки в ходе выполнения программы.