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: Волков Евгений Андреевич
Группа: М8О-207Б-21
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023

Содержание

Репозиторий

Постановка задачи

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. 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:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для *программы №2*).
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;
3. «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 5:

Контракт 1:

Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e.

Float SinIntegral(float A, float B, float e);

Реализация 1:

Метод прямоугольников

Реализация 2:

Метод трапеций

Контракт 2:

Рассчет значения числа e (основание натурального логарифма)

Float E(int x)

Реализация 1:

(1 + 1/x) ^ x

Реализация 2:

Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!))

Общие сведения о программе

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.

Общий метод и алгоритм решения

Объявим необходимые функции внутри файла lib.h. Используем спецификатор хранения extern, который сообщает компилятору, что находящиеся за ним типы и имена переменных объявляются где-то в другом месте.

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.

Используемые алгоритмы:

- Синус — сумма ряда Тейлора;
- Факториал — наивная реализация;
- Возведение в степень — алгоритм «бинарного» возведения в степень.

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.

RTLD_LAZY - это флаг, используемый при загрузке динамических библиотек с помощью функции dlopen() в операционных системах Linux и других, поддерживающих стандарт POSIX.

Когда вы загружаете динамическую библиотеку с помощью dlopen(), вы можете использовать флаг RTLD_LAZY, чтобы отложить разрешение всех

символов до тех пор, пока они не будут запрошены в вашей программе. Это означает, что функция dlopen() вернет управление сразу после загрузки библиотеки, не разрешая все ее символы.

Когда вы затем вызываете функцию dlsym() для поиска символа в загруженной библиотеке, символ будет разрешен только в момент вызова этого символа. Это может помочь ускорить загрузку библиотеки и уменьшить потребление памяти, если в вашей программе не используются все символы в библиотеке. Однако этот подход также может привести к задержкам при первом вызове функции, когда происходит разрешение символа.

Исходный код

lib.h

```
#ifndef __LIB_H__
#define __LIB_H__

extern float SinIntegral(float A, float B, float e);    // extern значит что функция
будет определена в другом месте
extern float E(int x);

#endif
```

lib1.c

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

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

float Sin(float x)
{
    float result = 0.0;
    int n = 0;
    while (n <= 10)
    {
        float numerator = binPow(-1, n) * binPow(x, 2 * n + 1);
        float denominator = 1.0;
        for (int i = 1; i <= 2 * n + 1; i++)
        {
            denominator *= i;
        }
        result += numerator / denominator;
        n++;
    }
```

```
    return result;
}

float SinIntegral(float A, float B, float e)
{
    float sum = 0.0;
    float x = A;
    while (x <= B)
    {
        sum += Sin(x) * e;
        x += e;
    }
    return sum;
}

float E(int x)
{
    printf("\nCalculation value of number e (base of natural logarithm)\n");
    printf("with approximation %d\n", x);
    printf("by formula e(x) = (1 + 1/x) ^ x\n");
    float mant = (float)1 + ((float)1 / (float)x);
    float e = binPow(mant, x);
    return e;
}
```

lib2.c

```
#include <stdio.h>

float binPow(float x, int y)
{
    float z = 1.0;
```

```
    while (y > 0)
    {
        if (y % 2 != 0)
        {
            z *= x;
        }
        x *= x;
        y /= 2;
    }
    return z;
}

float Sin(float x)
{
    float result = 0.0;
    int n = 0;
    while (n <= 10)
    {
        float numerator = binPow(-1, n) * binPow(x, 2 * n + 1);
        float denominator = 1.0;
        for (int i = 1; i <= 2 * n + 1; i++)
        {
            denominator *= i;
        }
        result += numerator / denominator;
        n++;
    }
    return result;
}
```

```
float SinIntegral(float A, float B, float e)
{
   float sum = (Sin(A) + Sin(B)) / 2.0;
   float x = A + e;
   while (x < B)
   {
      sum += Sin(x);
      x += e;
   }
   return sum * e;
}

int fact(int x)
{
   int res = 1;
   for (int i = 2; i <= x; i++)
   {
      res *= i;
   }
   return res;
}

float E(int x)
{
   float sum = 0;
   for (int i = 0; i <= x; i++)
   {
      sum += (1 / (float)fact(i));
   }
   return sum;
```

```
}
```

main1.c

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

int main(int argc, char const *argv[])
{
   while (!feof(stdin))
   {
      printf("\nWrite:\n [command] [arg1] ... [argN]\n");
      printf("\nIf you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]\n");
      printf("\nIf you want to calculate number e (base of natural logarithm), write
2 [approximation]\n\n");

      int target;
      scanf("%d", &target);

      if (target == 1)
      {
         float a, b, e;
         scanf("%f %f %f", &a, &b, &e);
         printf("SinIntegral: %f\n", SinIntegral(a, b, e));
      }
      else if (target == 2)
      {
         int x;
         scanf("%d", &x);
         printf("E: %f\n", E(x));
      }
   }
```

```
    return 0;
}
```

main2.c

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

// проверяет на ошибки
#define CHECK_ERROR(expr, message) \
    do                             \
    {                              \
        void *res = (expr);        \
        if (res == NULL)           \
        {                          \
            perror(message);       \
            return -1;             \
        }                          \
    } while (0)

const char *names[] = {"./liblib1.so", "./liblib2.so"};

int main()
{
    int n = 0;
    void *handle;
    float (*E)(int);
    float (*SinIntegral)(float, float, float);
    CHECK_ERROR(handle = dlopen(names[n], RTLD_LAZY), "dlopen error"); //
dlopen загружает боблиотеку в оперативную память
    CHECK_ERROR(E = dlsym(handle, "E"), "dlsym error (E)"); // dlsym
определяет функцию из определенной библиотеки
```

```
   CHECK_ERROR(SinIntegral = dlsym(handle, "SinIntegral"), "dlsym error
(SinIntegral)");
   while (!feof(stdin))
   {
      printf("\nWrite:\n [command] [arg1] ... [argN]\n");
      printf("\nIf you want to change methods of calculation, write 0\n");
      printf("\nIf you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]\n");
      printf("\nIf you want to calculate number e (base of natural logarithm), write
2 [approximation]\n\n");
      printf("Current lib is %d\n\n", n);
      int t;
      scanf("%d", &t);

      if (t == 0)
      {
         n = (n + 1) % 2; // меняет 0 на 1, 1 на 0

         if (dlclose(handle) != 0)   // dlclose освобождает память от библиотеки
         {
            perror("dlclose error");
            return -1;
         };

         CHECK_ERROR(handle = dlopen(names[n], RTLD_LAZY), "dlopen
error");  // загружаем другую библиотеку
         CHECK_ERROR(E = dlsym(handle, "E"), "dlsym error (E)");
         CHECK_ERROR(SinIntegral = dlsym(handle, "SinIntegral"), "dlsym error
(SinIntegral)");
      }
```

```
        if (t == 1)
        {
            int a, b, e;
            scanf("%d %d %d", &a, &b, &e);
            printf("SinIntegral: %f\n", (*SinIntegral)(a, b, e));
        }

        if (t == 2)
        {
            int x;
            scanf("%d", &x);
            printf("E: %f\n", (*E)(x));
        }
    }
    return 0;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.8)

project(main)
include_directories(include)

add_library(lib1 SHARED src/lib1.c)
add_library(lib2 SHARED src/lib2.c)

add_executable(main1 src/main1.c include/lib.h)
add_executable(main2 src/main1.c include/lib.h)
target_link_libraries(main1 lib1)
target_link_libraries(main2 lib2)
```

```
add_executable(main src/main2.c)
target_link_libraries(main ${CMAKE_DL_LIBS})
```

Демонстрация работы программы

```
ggame@ggame:~/hubs/newos/evgeny/OC-labs/lab5/build$ ./main

Write:
 [command] [arg1] ... [argN]

If you want to change methods of calculation, write 0

If you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]

If you want to calculate number e (base of natural logarithm), write 2 [approximation]

Current lib is 0

0

Write:
 [command] [arg1] ... [argN]

If you want to change methods of calculation, write 0

If you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]

If you want to calculate number e (base of natural logarithm), write 2 [approximation]
```

Current lib is 1

1 0 3.14 0.0001
SinIntegral: 2.000399

Write:
[command] [arg1] ... [argN]

If you want to change methods of calculation, write 0

If you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]

If you want to calculate number e (base of natural logarithm), write 2 [approximation]

Current lib is 1

2 4
E: 2.708333

Write:
[command] [arg1] ... [argN]

If you want to change methods of calculation, write 0

If you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]

If you want to calculate number e (base of natural logarithm), write 2 [approximation]

Current lib is 1

0

Write:
[command] [arg1] ... [argN]

If you want to change methods of calculation, write 0

If you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]

If you want to calculate number e (base of natural logarithm), write 2 [approximation]

Current lib is 0

2 4

Calculation value of number e (base of natural logarithm)
with approximation 4
by formula e(x) = (1 + 1/x) ^ x
E: 2.441406

Write:
[command] [arg1] ... [argN]

If you want to change methods of calculation, write 0

If you want to take integral of f(x) = sin(x), write 1 [A] [B] [C]

If you want to calculate number e (base of natural logarithm), write 2

[approximation]

Current lib is 0

Calculation value of number e (base of natural logarithm)

with approximation 4

by formula e(x) = (1 + 1/x) ^ x

E: 2.441406

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.