Московский авиационный институт
(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Операционные системы»

Лабораторная работа № 5

Студент: Марочкин И.А.
Группа: М8О-206Б-19
Преподаватель: Соколов А.А.
Дата: 24.04.2021
Оценка:

Москва, 2020

Цель работы

Приобретение практических навыков в:

- Создание динамических библиотек
- Создание программ, которые используют функции динамических библиотек.

Задание

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). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
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 ее выполнения.

Вариант:

Функция 1: Расчет 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.

Функция 2: Подсчет количества простых чисел на отрезке [A, B] (A, B - натуральные).

`int PrimeCount(int A, int B)`

Наивный алгоритм. Решето Эратосфена.

Общие сведения о программе

Программа компилируется при помощи Makefile в 3-х исполняемых файла `program_1_rel_1` `program_1_rel_2` `program_2` и 2 библиотеки `realization_1.so`, `realization_2.so`. В первом случае мы используем библиотеку, которая использует знания полученные во время компиляции (на этапе линковки). Во втором случае программа загружает библиотеки и взаимодействует с ними при помощи следующих системных вызовов:

dlopen - загружает динамическую библиотеку, имя которой указано первым аргументом, и возвращает прямой указатель на начало динамической библиотеки. Второй аргумент отвечает за разрешение неопределенных символов, возвращает 0 при успешном завершении и значение != 0 в случае ошибки.

dlsym - использует указатель на динамическую библиотеку – первый аргумент, возвращаемую dlopen, и оканчивающееся нулем символьное имя – второй аргумент, а затем возвращает адрес, указывающий, откуда загружается этот символ. Если символ не найден, то возвращаемым значением dlsym является NULL.

dlclose - уменьшает на единицу счетчик ссылок на указатель динамической библиотеки, передаваемый в качестве аргумента. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается.

Листинг программы

Makefile

```
CC = gcc
CFLAGS = -pedantic -Wall
CLIBFLAGS = -fPIC -shared

all: program_1_rel_1 program_1_rel_2 program_2

# program_1

program_1_rel_1: program_1.o realization_1.o
	$(CC) $(CFLAGS) $^ -o $@

program_1_rel_2: program_1.o realization_2.o
	$(CC) $(CFLAGS) $^ -o $@

program_1.o: ./Programs/program_1.c
	$(CC) $(CFLAGS) -c $^

realization_1.o: ./Libraries/realization_1.c ./Libraries/library.h
	$(CC) $(CFLAGS) $< -c

realization_2.o: ./Libraries/realization_2.c ./Libraries/library.h
	$(CC) $(CFLAGS) $< -c

#program 2

program_2: program_2.o realization_1.so realization_2.so
	$(CC) $(CFLAGS) $< -o $@

program_2.o: ./Programs/program_2.c
	$(CC) $(CFLAGS) -c $^

realization_1.so: ./Libraries/realization_1.c ./Libraries/library.h
	$(CC) $(CFLAGS) $(CLIBFLAGS) $< -o $@

realization_2.so: ./Libraries/realization_2.c ./Libraries/library.h
	$(CC) $(CFLAGS) $(CLIBFLAGS) $< -o $@

clean:
	rm -rf program_1_rel_1 program_1_rel_2 program_2 *.o *.so
```

Libraries/library.h

```
#ifndef LIBRARY
#define LIBRARY

int PrimeCount(int, int);
float SinIntegral(float, float, float);

#endif
```

Libraries/realization_1.c

```
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include "library.h"

float SinIntegral(float A, float B, float e) {
    printf("In 1-st realization\n");
    int n = (B - A) / e;
    float s = 0;

    for (int i = 0; i < n - 1; ++i){
        s += sinf(A + i*e);
    }
```

```
    s *= e;
    return s;
}

int PrimeCount(int A, int B) {
    printf("In 1-st realization\n");
    int res = 0;
    bool prime = true;

    if (A == 1 || A == 2){
        res = 1;
        A = 3;
    }

    for (int i = A; i <= B; ++i){
        for (int j = 2; j < i - 1; ++j){
            if (i % j == 0){
                prime = false;
                break;
            }
        }
        if (prime == true){
            res += 1;
        }
        prime = true;
    }
    return res;
}
```

Libraries/realization_2.c

```
#include <math.h>
#include <stdio.h>
#include "library.h"
float SinIntegral(float A, float B, float e) {
    printf("In 2-nd realization\n");
    int n = (B - A) / e;
    float s = 0;

    for (int i = 0; i < n - 1; ++i){
        s += 2*(sinf(A + i*e));
    }

    s *= (B - A) / (2*n);
    return s;
}
int PrimeCount(int A, int B){
    printf("In 2-nd realization\n");
    int res = 0;
    int matrix[B];

    for (int i = 0; i < B; ++i){
        matrix[i] = i;
    }
    matrix[1] = 0;

    for (int i = 2; i < B; ++i){
        if (matrix[i] != 0){
            if (matrix[i] >= A){
                res += 1;
            }
            for (int j = 2*i; j < B; j += i){
                matrix[j] = 0;
            }
        }
    }
    return res;
}
```

Programs/program_1.c

```
#include <stdio.h>
#include <stdlib.h>
#include "../Libraries/library.h"
void menu (){
       printf("=====================================================================\n");
       printf("|| 1 float float float - count integral sin(x)                   ||\n");
       printf("|| 2 int int - count number of primes                            ||\n");
       printf("|| -1 - Exit                                                      ||\n");
       printf("=====================================================================\n");
       printf("\n");
}
int main(){
    menu();
       int cmd = 0;
       while (cmd != -1){
        printf("Enter command: ");
        scanf("%d", &cmd);
        if (cmd == 1){
            float a, b, e;
            if (scanf("%f %f %f", &a, &b, &e) != 3){
                printf("Invalid count of arguments\n");
                menu();
            } else {
                printf("integral sin(x) = %f\n", SinIntegral(a, b, e));
            }
            continue;
        }
        else if (cmd == 2){
            int a, b;
            if (scanf("%d %d", &a, &b) != 2) {
                printf("Invalid count of arguments\n");
                menu();
            } else {
                printf("primes count = %d\n", PrimeCount(a, b));
            }
            continue;
        }
        else if (cmd == -1){
            break;
        }
        else{
            printf("Unknown command\n");
            menu();
        }
    }
    return 0;}
```

Programs/program_2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
void menu (){
       printf("=====================================================================\n");
    printf("|| 0 - change realization                                        ||\n");
       printf("|| 1 float float float - count integral sin(x)                   ||\n");
       printf("|| 2 int int - count number of primes                            ||\n");
       printf("|| -1 - Exit                                                      ||\n");
       printf("=====================================================================\n");
       printf("\n");}
int ChangeMode(int realization){
    if (realization == 1){
        return 2;}
    else {
        return 1;
    }
}
```

```
int main(){
    menu();
    int cmd = -2;
    int realization = 1;
    void* library_handler_1 = NULL;
       void* library_handler_2 = NULL;
       float (*sinintegral)(float, float, float);
    int (*primecount)(int, int);

    if((library_handler_1 = dlopen("realization_1.so", RTLD_LAZY)) == 0){
               printf("Can't open library\n");
               exit(-1);
       }
       if((library_handler_2 = dlopen("realization_2.so", RTLD_LAZY)) == 0){
               printf("Can't open library\n");
               exit(-2);
       }
       while (cmd != -1){
        printf("Enter command: ");
        scanf("%d", &cmd);
        if (cmd == 0){
            printf("Realization was changed from %d to %d\n", realization,
ChangeMode(realization));
            realization = ChangeMode(realization);
        }
        else if (cmd == 1){
            float a, b, e;
            if (scanf("%f %f %f", &a, &b, &e) != 3){
                printf("Invalid count of arguments\n");
                menu();
            } else if (realization == 1){
                sinintegral = (float (*)(float, float, float))dlsym(library_handler_1,
"SinIntegral");
                printf("integral sin(x) = %f\n", (*sinintegral)(a, b, e));
            }
            else if (realization == 2){
                sinintegral = (float (*)(float, float, float))dlsym(library_handler_2,
"SinIntegral");
                printf("integral sin(x) = %f\n", (*sinintegral)(a, b, e));
            }
        }
        else if (cmd == 2){
            int a, b;
            if (scanf("%d %d", &a, &b) != 2) {
                printf("Invalid count of arguments\n");
                menu();
            } else if (realization == 1){
                primecount = (int (*)(int, int))dlsym(library_handler_1, "PrimeCount");
                printf("primes count = %d\n", (*primecount)(a, b));
            }
            else if (realization == 2){
                primecount = (int (*)(int, int))dlsym(library_handler_2, "PrimeCount");
                printf("primes count = %d\n", (*primecount)(a, b));
            }
        }
        else if (cmd == -1){
            break;
        }
        else{
            printf("Unknown command\n");
            menu();
        }
    }
    dlclose(library_handler_1);
    dlclose(library_handler_2);
    return 0;
}
```

Пример работы

```
Vanya:Src ivan$ ./program_2
================================================================
|| 0 - change realization                                      ||
|| 1 float float float - count integral sin(x)                 ||
|| 2 int int - count number of primes                          ||
|| -1 - Exit                                                    ||
================================================================

Enter command: 1 2 5 0.001
In 1-st realization
integral sin(x) = -0.696957
Enter command: 2 1 10
In 1-st realization
primes count = 4
Enter command: 0
Realization was changed from 1 to 2
Enter command: 1 2 5 0.001
In 2-nd realization
integral sin(x) = -0.697189
Enter command: 2 1 10
In 2-nd realization
primes count = 4
Enter command: -1
Vanya:Src ivan$
```

Вывод

На СИ можно удобно писать статические и динамические библиотеки, причем существует несколько механизмов работы с ними: используя знания полученные во время компиляции (этап линковки) или при помощи загрузки библиотек, их местоположения и контракта. Применяя библиотеки, мы можем писать более сложные вещи, которые используют простые функции, структуры, написанные ранее и сохраненные в этих самых библиотеках.