МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

ЛАБОРАТОРНАЯ РАБОТА №5

по курсу операционные системы I семестр, 2021/22 уч. год

Студент *Сикорский Александр Александрович, группа М8О-208Б-20*
Преподаватель *Миронов Евгений Сергеевич*
Вариант *4*
Оценка
Дата
Подпись

Содержание

Репозиторий

`https://github.com/sikorskii/os/tree/master/lab5`

Постановка задачи

Задание: Вариант 4:

Функция 1: 1, функция 2: 5. Цель работы:

Целью является 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 её выполнения.

Общие сведения о программе

Исходный код лежит в 4 файлах, еще один - makefile:

1. main.cpp - файл, в котором происходит создание дочерних процессов, работа с пользователем и открытие семафоров.

2. child1.cpp- код дочерних процессов. Принимают строки от родителя, разворачивают их и пишут в указанный файл.

3. mem.h - Некоторые константы и значения, нужные для работы с mmap.

Программа собирается с помощью Makefile. Нужно получитьдва исполняемых файла и два файла с динамическими библиотеками, как указано в задании.

Общий метод и алгоритм решения

Имеем два исполняемых файла. В первом случае библиотека подключается к программе на этапе компиляции. Тут мы работаем с ней практически как с любой другой библиотекой. Во втором случае нам необходимо пользоваться средствами языка для открытия динамических библиотек и получения функций из них. В каждой библиотеке содержатся определения двух функций, кроме этого есть общий заголовочный файл. Сами функции - простые и сводятся к арифметике. В программе с динамическим использованием библиотеки нужно открывать файл динамической библиотеки, получать из него функции и переключаться между разными библиотекаками с разными реализациями по команде пользователя. Используем переменную, которая позволяет понять, с какой библиотекой мы сейчас работаем. Перед закрытием программы нужно закрыть библиотеку.

Исходный код

main.c

```
//
// Created by aldes on 28.11.2021.
//

#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <stdlib.h>
#include <dlfcn.h>

double (*integrateSin)(double leftBound, double rightBound, double step) = NULL;
double (*calculatePI)(int seriesLength) = NULL;
void *lib = NULL;
bool firstLib = true;
```

```
char* path1 = "libFirst.so";
char* path2 = "libSecond.so";

void openLib(char* pathToLib) {
    if (lib != NULL) {
        dlclose(lib);
    }

    lib = dlopen(pathToLib, RTLD_LAZY);
    if (lib == NULL) {
        perror("cant open lib\n");
        exit(1);
    }
    integrateSin = dlsym(lib, "integrateSin");
    calculatePI = dlsym(lib, "calculatePI");
    if (integrateSin == NULL || calculatePI == NULL) {
        perror("unable to find method\n");
        exit(1);
    }
}

void reloadLib() {
    firstLib ? openLib(path2) : openLib(path1);
    firstLib = !firstLib;
    printf("lib changed\n");
}

double parseAndPerform(char* query) {
    int typeOfQuery;
    char* type = strtok(query, " ");
    sscanf(type, "%d", &typeOfQuery);

    if (typeOfQuery == 1) {
        char* firstArg = strtok(NULL, " ");
        if (firstArg == NULL)
            return 0;
        double first;
        sscanf(firstArg, "%lf", &first);
        char* secondArg = strtok(NULL, " ");
        if (secondArg == NULL)
            return 0;
        double second;
```

```
        sscanf(secondArg, "%lf", &second);
        char* thirdArg = strtok(NULL, " ");
        if (thirdArg == NULL)
            return 0;
        double third;
        sscanf(thirdArg, "%lf", &third);
        while (thirdArg != NULL) {
            thirdArg = strtok(NULL, " \n\0");
        }
        printf("first arg %lf second %lf third %lf\n", first, second, third);
        return integrateSin(first, second, third);
    }

    if (typeOfQuery == 2) {
        char* firstArg = strtok(NULL, " ");
        if (firstArg == NULL)
            return 0;
        //printf("first arg is %s\n", firstArg);
        int first;
        sscanf(firstArg, "%d", &first);
        while (firstArg != NULL) {
            firstArg = strtok(NULL, " \n\0");
        }
        //printf("first arg %d\n", first);
        return calculatePI(first);
    }

    if (typeOfQuery == 0) {
        reloadLib();
        return 0;
    }
    else {
        printf("invalid query, 0 returned\n");
        return 0;
    }
}


int main() {
    char queryBuf[100];
    openLib(path1);
    printf("enter query:\n");
```

```
    while (fgets(queryBuf, 100, stdin) != NULL) {
        printf("calculated %lf\n", parseAndPerform(queryBuf));
        printf("enter query:\n");
    }
    firstLib ? dlclose(path1) : dlclose(path2);
}
```

main2.c

```
//
// Created by aldes on 28.11.2021.
//

#include <stdio.h>
#include <string.h>
#include "lib1.h"

double parseAndPerform(char* query) {
    int typeOfQuery;
    char* type = strtok(query, " ");
    sscanf(type, "%d", &typeOfQuery);

    if (typeOfQuery == 1) {
        char* firstArg = strtok(NULL, " ");
        if (firstArg == NULL)
            return 0;
        double first;
        sscanf(firstArg, "%lf", &first);
        char* secondArg = strtok(NULL, " ");
        if (secondArg == NULL)
            return 0;
        double second;
        sscanf(secondArg, "%lf", &second);
        char* thirdArg = strtok(NULL, " ");
        if (thirdArg == NULL)
            return 0;
        double third;
        sscanf(thirdArg, "%lf", &third);
        while (thirdArg != NULL) {
            thirdArg = strtok(NULL, " \n\0");
        }
        printf("first arg %lf second %lf third %lf\n", first, second, third);
        return integrateSin(first, second, third);
```

```
    }

    if (typeOfQuery == 2) {
        char* firstArg = strtok(NULL, " ");
        if (firstArg == NULL)
            return 0;
        //printf("first arg is %s\n", firstArg);
        int first;
        sscanf(firstArg, "%d", &first);
        while (firstArg != NULL) {
            firstArg = strtok(NULL, " \n\0");
        }
        //printf("first arg %d\n", first);
        printf("stat function begin\n");
        return calculatePI(first);
    }
    else {
        printf("invalid query, 0 returned\n");
        return 0;
    }

}

int main() {
    char queryBuf[100];
    printf("enter query:\n");
    while (fgets(queryBuf, 100, stdin) != NULL) {
        printf("calculated %lf\n", parseAndPerform(queryBuf));
        printf("enter query:\n");
    }
}
```

lib1.h

```
//
// Created by aldes on 28.11.2021.
//

#ifndef LAB5NEW_LIB1_H
#define LAB5NEW_LIB1_H

extern double integrateSin(double leftBound, double rightBound, double step);
```

```
extern double calculatePI(int seriesLength);

#endif //LAB5NEW_LIB1_H
```

lib1.c

```
//
// Created by aldes on 28.11.2021.
//

#include "lib1.h"
#include <math.h>

double integrateSin(double leftBound, double rightBound, double step) {
    double sinNumeric = 0;

    double left = leftBound;
    double right = left + step;
    double point = (left + right) / 2;
    while (right <= rightBound) {
        sinNumeric += (sin(point) * step);
        left = right;
        right += step;
        point = (left + right) / 2;
    }

    return sinNumeric;
}

double calculatePI(int seriesLength) {
    double PI = 0;

    for (int i = 0; i < seriesLength - 1; i++) {
        PI += pow(-1, i) / (2 * i + 1);
    }

    return 4 * PI;
}
```

lib2.c

```
//
// Created by aldes on 28.11.2021.
```

```c
//

#include "lib1.h"
#include <math.h>

double integrateSin(double leftBound, double rightBound, double step) {
    double sinNumeric = 0;

    double left = leftBound;
    double right = left + step;
    while (right <= rightBound) {
        sinNumeric += (sin(left) + sin(right)) / 2 * step;
        left = right;
        right += step;
    }

    return sinNumeric;
}

double calculatePI(int seriesLength) {
    double PI = 1;

    for (double i = 1; i < seriesLength; ) {
        PI *= (2 * i / (2 * i - 1) * 2 * i / (2 * i + 1));
        i += 1;
    }

    return 2 * PI;
}
```

Makefile

```
CC = gcc

all:
	$(CC) -fPIC -c lib1.c -o lib1.o -lm
	$(CC) -fPIC -c lib2.c -o lib2.o -lm
	$(CC) -shared -o libFirst.so lib1.o -lm
	$(CC) -shared -o libSecond.so lib2.o -lm
	sudo cp libFirst.so /usr/lib
	sudo cp libSecond.so /usr/lib
	$(CC) main2.c -lFirst -lm -o static.out -fsanitize=address
	$(CC) main.c -ldl -lm -o dynamic.out -fsanitize=address
```

```
clean:
	rm -f *.o *.so *.out
```

Демонстрация работы программы

Ниже приведен пример работы программы.

test.txt

```
aldes@aldes:~/dev/os/lab5$ ./static.out
enter query:
1 0 1 0.001
first arg 0.000000 second 1.000000 third 0.001000
calculated 0.458857
enter query:
0
invalid query, 0 returned
calculated 0.000000
enter query:
2
calculated 0.000000
enter query:
2 1000
stat function begin
calculated 3.142594
enter query:
```

```
aldes@aldes:~/dev/os/lab5$ ./dynamic.out
enter query:
0
lib changed
calculated 0.000000
enter query:
0
lib changed
calculated 0.000000
enter query:
1 0 1 0.0001
```

```
first arg 0.000000 second 1.000000 third 0.000100
calculated 0.459698
enter query:
0
lib changed
calculated 0.000000
enter query:
1 0 1 0.0000001
first arg 0.000000 second 1.000000 third 0.000000
calculated 0.459698
enter query:
2 1000000
stat function begin
calculated 3.141592
enter query:
0
lib changed
calculated 0.000000
enter query:
2 1000
stat function begin
calculated 3.142594
```

Выводы

В ходе работы я познакомился с тем, как создавать и использовать shared библиотеки. Пришлось написать две почти одинаковых программы для работы с библиотеками. Можно заметить, что сам код программ несложный и даже сказать, что лабораторная заточена на написание makefile и использование пары функций для открытия библиотеки.