Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу
«Операционные системы»

Динамические библиотеки

Студент: Недосеков Иван Дмитриевич
Группа: М8О-206Б-19
Вариант: 3
Преподаватель: Соколов Андрей Алексеевич
Дата: _____
Оценка: _____
Подпись: _____

Москва, 2020

Постановка Задачи

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, загрузив библиотеки в память с помощью системных вызовов

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.

Вариант: 3. Функция: Рассчет интеграла функции $\sin(x)$ на отрезке $[A, B]$ с шагом e. Сигнатура: $float\ SinIntegral(float\ A, float\ B, float\ e)$ Реализации :

1. Подсчет интеграла методом прямоугольников;
2. Подсчет интеграла методом трапеций.

Функция: Подсчёт наибольшего общего делителя для двух натуральных чисел. Сигнатура: $int\ GCF(int\ A, int\ B)$ Реализации :

1. Алгоритм Евклида;
2. Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.

Общие сведения о программе

Программы, использующие динамические библиотеки компилируется из файла program1.c, program2.c; библиотеки из gcdEvc.c, gcdSimple.c, SinIntegralRectangle.c, SinIntegralTrapezoid.c, lib1.c, lib2.c. Также используется заголовочные файлы: *math.h, stdlib.h, stdio.h, dlfcn.h*. В программе используются следующие системные вызовы:

1. **dlopen** — загружает динамическую библиотеку, имя которой указано в строке.
2. **dlclose** — уменьшает на единицу счетчик ссылок на указатель динамической библиотеки. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается.
3. **dlsym** — использует указатель на динамическую библиотеку, возвращаемую dlopen, и оканчивающееся нулем символьное имя, а затем возвращает адрес, указывающий, откуда загружается этот символ. Если символ не найден, то возвращаемым значением dlsym является NULL.
4. **dlerror** — возвращает NULL, если не возникло ошибок с момента инициализации или его последнего вызова.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

- Изучить принципы работы dlopen, dlclose, dlsym, dlerror.
- Изучить принципы работы динамической загрузки и динамической компановки.
- Написать библиотеки.
- Проверить программу на тестах.

Основные файлы программы

gcdEvc.c

```
int GCF (int A, int B) {
    while (B) {
        A %= B;
        int temp = A;
        A = B;
        B = temp;

    }
    return A;
}
```

gcdSimple.c

```c
#define max(a, b) ((a) > (b) ? (a) : (b))
#define min(a, b) ((a) < (b) ? (a) : (b))

int GCF (int A, int B) {
    for( int i = min ( A, B ); i > 1; i-- ){
        if ( A % i == 0 && B % i == 0 ){
            return i;
        }
    }
    return 1;
}
```

SinIntegralRectangle.c

```c
#include <math.h>

float SinIntegral( float A, float B, float e){
    double integral = 0;

    for( float dot = A; dot < B; dot+=e ){
        integral += sin( dot ) * e;
    }
    return integral;

}
```

SinIntegralTrapezoid.c

```c
#include <math.h>

float SinIntegral( float A, float B, float e){
    double integral = 0;
    for( float dot = A + e; dot < B; dot+=e ){
        integral += (sin( dot ) + sin( dot - e )) / 2 * e;
    }
    return integral;

}
```

lib1.c

```c
#include <stdio.h>
void who(){
   printf("I am lib 1\n");
}
```

lib2.c

```c
#include <stdio.h>
void who(){
   printf("I am lib 2\n");
}
```

program1.h

```c
#pragma once



float SinIntegral( float A, float B, float e);
int GCF (int A, int B);
```

program1.c

```c
#include <stdio.h>
#include "program1.h"
#include <stdlib.h>


int main(){

    char comand;
```

```
    while( scanf( "%c", &comand ) > 0 ){
        if ( comand == '1' ){
            float A,B,e;
            scanf( "%f%f%f", &A, &B, &e );
            printf( "Integral = %f\n", SinIntegral( A, B, e) );
        } else if ( comand == '2' ){
            int A,B;
            scanf( "%d%d", &A, &B );
            printf( "GCD = %d\n", GCF( A, B ) );
        }else if ( comand == EOF  ){
            exit(0);
        }else if ( comand == 'q' ){
            exit(0);
        }else if ( comand == '\n' || comand == '\t' || comand == ' ' ){
            continue;
        }else{
            printf("Wrong format\n");
        }
    }
}
```

program2.h

```
#pragma once



const char LibFisrstReal[] = "./libs/libshared1.so";
const char LibSecondReal[] = "./libs/libshared2.so";
```

program2.c

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include "program2.h"

#define FIRST 1
#define SECOND 2
```

```
int curRealisation = SECOND;
void* libFD;
float (*SinIntegral) ( float, float, float );
int (*GCF) ( int, int );
void (*who)();
char* err = NULL;

void swapRealisation(){
    if ( dlclose( libFD ) != 0 ){
        perror( "Cant close dl" );
        exit(1);
    }
    char* nextName;
    if ( curRealisation == SECOND ){
        curRealisation = FIRST;
        nextName = LibFisrstReal;
    }else{
        curRealisation = SECOND;
        nextName = LibSecondReal;
    }

    libFD = dlopen (nextName, RTLD_LAZY);
    if (!libFD) {
        perror("err cant swap lib" );
        exit(1);
    }
    SinIntegral = dlsym(libFD, "SinIntegral");
    if ((err = dlerror()) != NULL)  {
        fprintf (stderr, "%s\n", err);
        dlclose( libFD );
        exit(1);
    }
    GCF = dlsym(libFD, "GCF");
    if ((err = dlerror()) != NULL)  {
        fprintf (stderr, "%s\n", err);
        dlclose( libFD );
        exit(1);
    }
    who = dlsym(libFD, "who");
    if ((err = dlerror()) != NULL){
        fprintf (stderr, "%s\n", err);
        dlclose(libFD);
        exit(1);
    }
}
```

```
int main(){

    libFD = dlopen (LibFisrstReal, RTLD_LAZY);
    if (!libFD) {
        perror("err cant open standart second lib" );
        exit(1);
    }
    SinIntegral = dlsym(libFD, "SinIntegral");
    if ((err = dlerror()) != NULL)  {
        fprintf (stderr, "%s\n", err);
        dlclose( libFD );
        exit(1);
    }
    GCF = dlsym(libFD, "GCF");
    if ((err = dlerror()) != NULL)  {
        fprintf (stderr, "%s\n", err);
        dlclose( libFD );
        exit(1);
    }
    who = dlsym(libFD, "who");
    if ((err = dlerror()) != NULL){
       fprintf(stderr,"%s\n", err);
       dlclose( libFD);
       exit(1);
    }


    char comand;
    while( scanf( "%c", &comand ) > 0 ){
        if ( comand == '1' ){
            float A,B,e;
            scanf( "%f%f%f", &A, &B, &e );
            printf( "Integral = %f\n", SinIntegral( A, B, e) );
        } else if ( comand == '2' ){
            int A,B;
            scanf( "%d%d", &A, &B );
            printf( "GCD = %d\n", GCF( A, B ) );
        } else if ( comand == '0' ){
            swapRealisation();
   } else if ( comand == '!'){
      who();
        }else if ( comand == EOF  ){
            dlclose( libFD );
            exit(0);
```

```
        }else if ( comand == 'q' ){
            dlclose( libFD );
            exit(0);
        }else if ( comand == '\n' || comand == '\t' || comand == ' ' ){
            continue;
        }else{
            printf("Wrong format\n");
        }
    }
}
```

Пример работы

```
mx$ cat ../tests/01.t
1 0 3.14 0.1
2 1 5
2 121 33
mx$ ./program1 < ../tests/01.t
Integral = 1.999548
GCD = 1
GCD = 11
mx$ cat ../tests/02.t
1 0 3.14 0.1
2 1 5
2 121 33
0
1 0 3.14 0.1
2 1 5
2 121 33
0
1 0 3.14 0.1
2 1 5
2 121 33
mx$ ./program2 < ../tests/02.t
Integral = 1.999548
GCD = 1
GCD = 11
Integral = 1.999548
GCD = 1
GCD = 11
Integral = 1.997469
GCD = 1
```

```
GCD = 11
```

GitHub

Ссылка на репозиторий GitHub с проектом:
https://github.com/GrozniyToaster/os_lab_05

Вывод

Использование динамических библиотек помогает сократить размер исполняемого файла и минимизировать копии кода библиотек в исполняемых файлах. При использовании динамической компоновки код не усложняется и не меняется относительно статических, а загрузке же нужно обрабатывать результаты что усложняет читаемость кода, но тогда мы можем менять напрмер используемую реализацию не затрачиваю доп помять на не используемые библиотеки.