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: Суханов Е.А.
Группа: М8О–206Б–19
Вариант:11
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

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. 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

Лабораторная работа состоит из 4-ех частей:

- app1. Первая программа, которая использует первую реализацию данных функций (библиотека libimp1). Она подключает данную библиотеку на этапе компиляции;
- app2. Вторая программа, которая использует обе реализации, и может их менять “на лету”. Для этого используется функционал ОС dl. Данная библиотека имеет 4 вызова:
    1. dlopen - открывает динамическую библиотеку по указанному пути. возвращает указатель на void. Который потом используется для работы с данной библиотекой;
    2. dlsym - возвращает указатель на функцию из библиотеки. Функция указывается с помощью ее строкового идентификатора;
    3. dlerror - возвращает указатель на строку с описанием ошибки, если та произошла. Если ошибки не было, то возвращается NULL;
    4. dlclose - закрывает открытую библиотеку.
- lib_first_* . Поиск производной функции cos(x) в точке A с приращением deltaX;
- lib_second_*. Вычисление числа Pi с помощью формул, в которых K членов.

Общий метод и алгоритм решения

Ничего сложного в данных компонентах нет. Довольно стандартная обработка пользовательского ввода с помощью цикла и switch в функции main. Вполне стандартные реализации алгоритмов.

Собирается данная программа с помощью Makefile.

Основные файлы программы

app1.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include "libs/first/first.h"
#include "libs/second/second.h"

int main()
{
    char mod;

    while(scanf("%c", &mod) == 1)
    {
        // Derivative
        if(mod == '1')
        {
            float A, deltaX;
            if(scanf("%f%f", &A, &deltaX) != 2)
            {
                fprintf(stderr, "Ошибка ввода\n");
                continue;
            }
            printf("Derivative: %f\n", Derivative(A, deltaX));
        }
        //Pi
        else if(mod == '2')
        {
            int K;
            if(scanf("%d", &K) != 1)
            {
                fprintf(stderr, "Ошибка ввода\n");
                continue;
            }
            if(K < 1)
```

```
            {
                fprintf(stderr, "Введено некорректное значение\n");
                continue;
            }

            printf("Pi: %f\n", Pi(K));
        }
    }

    return 0;
}
```

app2.c:

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


void* load_function(void* lib, const char* name)
{
    char* error;
    void* fun;
    fun = dlsym(lib, name);
    if((error = dlerror()) != NULL)
    {
        fprintf(stderr, "Произошла ошибка загрузки функции: %s\n", error);
        exit(1);
    }

    return fun;
}

int main()
{
    char mod;

    void* libs[2];
    float (*Derivative)(float A, float deltaX);
```

```
float (*Pi)(int K);

if((libs[0] = dlopen("libs/libimp1.so", RTLD_LAZY)) == NULL)
{
    fprintf(stderr, "Не удалось загрузить библиотеку libs/libimp1.so\n");
    exit(1);
}
if((libs[1] = dlopen("libs/libimp2.so", RTLD_LAZY)) == NULL)
{
    fprintf(stderr, "Не удалось загрузить библиотеку libs/libimp2.so\n");
    exit(1);
}

Derivative = load_function(libs[0], "Derivative");
Pi = load_function(libs[0], "Pi");
printf("Используется первая реализация\n");

while(scanf("%c", &mod) == 1)
{
    // Change imp
    if(mod == '0')
    {
        int number;
        if(scanf("%d", &number) != 1)
        {
            fprintf(stderr, "Ошибка ввода\n");
            continue;
        }
        if(number < 0 || number > 1)
        {
            fprintf(stderr, "Недопустимое значение\n");
            continue;
        }
        Derivative = load_function(libs[number], "Derivative");
        Pi = load_function(libs[number], "Pi");
    }
```

```
        // Derivative
        if(mod == '1')
        {
            float A, deltaX;
            if(scanf("%f%f", &A, &deltaX) != 2)
            {
                fprintf(stderr, "Ошибка ввода\n");
                continue;
            }
            printf("Derivative: %f\n", Derivative(A, deltaX));
        }
        //Pi
        else if(mod == '2')
        {
            int K;
            if(scanf("%d", &K) != 1)
            {
                fprintf(stderr, "Ошибка ввода\n");
                continue;
            }
            if(K < 1)
            {
                fprintf(stderr, "Введено некорректное значение\n");
                continue;
            }

            printf("Pi: %f\n", Pi(K));
        }
    }

    dlclose(libs[0]);
    dlclose(libs[1]);
    return 0;
}
```

first_imp_1.c:

```c
#include "first.h"
#include <math.h>

float Derivative(float A, float deltaX)
{
    return (cos(A + deltaX) - cos(A)) / deltaX;
}
```

second_imp.c:

```c
// Формула Валлиса
#include <stdlib.h>
#include <stdbool.h>
#include "second.h"

float Pi(int K)
{
    float sum = 1;
    for(int i = 1; i < K; ++i)
    {
        float a = 4 * i * i;
        sum *= a / (a - 1);
    }

    return 2 * sum;
}
```

Пример работы

```
reterer@serv:~/OS/os_exercise_05/src$ ./app2
Используется первая реализация
2 10
Pi: 3.041840
0 1
2 10
Pi: 3.060036
```

Вывод

Динамические библиотеки можно подключить двумя способами:

1. На этапе компиляции указывается путь к библиотеке и заголовочный файл;
2. “На лету” с помощью ОС можно подключить динамическую библиотеку и вызвать ее функции, зная их названия.

Динамические библиотеки могут использоваться несколькими приложениями одновременно, поэтому они позволяют экономить память как дисковую, так и оперативную. Возможность подключения динамических библиотек “на лету” позволяет создавать расширяемые системы. Например поддержку плагинов или модификаций.