Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики Кафедра вычислительной математики и программирования

Лабораторная работа по предмету "операционные системы" №5

Студент: Мокеева С.А.
Преподаватель: Соколов А.А.
Группа: М8О-206Б-20
Дата: 12.04.2022
Оценка:
Подпись:

Москва 2022г.

Вариант 12.

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.

2	Рассчет производной функции	Float	f'(x) = (f(A + deltaX)	f'(x) = (f(A + deltaX)

	cos(x) в точке A с приращением deltaX	Derivative(float A, float deltaX)	– f(A))/deltaX	– f(A-deltaX))/(2*deltaX)

6	Рассчет значения числа e(основание натурального логарифма)	Float E(int x)	(1 + 1/x) ^ x	Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!))

Реализация

Файл lib1.c

```
#include <math.h>

//реализация 1

float E(int x) { //рассчёт значения числа e
    return pow(1 + 1. / x, x);
}

float Derivative(float A, float deltaX) { //функция рассчёта производной
функции
                                          //cos(x) в точке A с приращением
deltaX
    float ans;
    ans = (cos(A + deltaX) - cos(A)) / deltaX;
    return ans;
}
```

Файл lib2.c

```
#include <math.h>

//реализация 2

float E(int x) { //рассчёт значения числа e
    float answer = 0;
    long long s = 1;
    for (int i = 0; i <= x; ++i) {
        if (i!=0) {
            s*=i;
        }
        answer += 1. / s;
    }
    return answer;
}

float Derivative(float A, float deltaX) { //функция рассчёта производной
функции
                                          //cos(x) в точке A с приращением
deltaX
    float ans;
    ans = (cos(A + deltaX) - cos(A - deltaX)) / (2 * deltaX);
    return ans;
}
```

Файл prog1.c

```
#include<stdio.h>

float E(int x);
```

```
float Derivative(float A, float deltaX);

int main()
{
    int operation;
    while(scanf("%d", &operation) > 0) {//считываем операции, чтобы выйти
ввести -1
        if (operation == 1) { //если операция 1
            float A, deltaX;
            scanf("%f%f", &A, &deltaX); //считываем A, deltaX
            printf("%s(%f, %f) = %f\n", "Derivative", A, deltaX,
Derivative(A, deltaX));
            //печатаем значение производной
        } else if (operation == 2) { //если операция 2
            int x;
            scanf("%d", &x); //считываем x
            printf("%s(%d) = %f\n", "E", x, E(x)); //печатаем значение e
        } else {
            printf("ERROR\n"); //иначе ошибка
        }
    }
}
```

Файл prog2.c

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

int main()
{
    void* cur_lib = NULL; //пустой указатель
    float (*Derivative)(float A, float deltaX); // указатель на функцию
производной
    float (*E)(int x); // указатель на функцию e

    int key;
    int lib_num;
    printf("Library 1 or 2?\n"); //Выберите библиотеку
    scanf("%d",&lib_num);

    // Открываем нужную библиотеку:
    if (lib_num == 1) { //если первая библиотека
        //dlopen(const char *filename, int flag), dlopen загружает
динамическую библиотеку и возвращает
        //прямой указатель на начало динамической библиотеки
        //RTLD_LAZY подразумевает разрешение неопределённых символов в виде
кода из динамической библиотеке
        cur_lib = dlopen("./libd1.so",RTLD_LAZY);  //загружаем и открываем
библиотеку 1
    } else if (lib_num == 2) { //если вторая библиотека
        cur_lib = dlopen("./libd2.so",RTLD_LAZY); //загружаем и открываем
библиотеку 2
    } else {
        return 1;
    }

    // Находим адреса функций:
    //void *dlsym(void *handle, char *symbol), dlsym использует указатель
на динамическую библиотеку, возвращаемую dlopen,
    //и оканчивающееся нулем символьное имя, а затем возвращает адрес,
указывающий, откуда загружается этот символ
    Derivative = dlsym(cur_lib, "Derivative");
    E = dlsym(cur_lib, "E");
```

```
    while (scanf("%d",&key) > 0) {
        if (key == 1) { //если значение 1
            float A,deltaX;
            scanf("%f%f",&A,&deltaX); //считываем A, deltaX
            printf("%s(%f, %f) = %f\n","Derivative", A,
deltaX,(*Derivative)(A, deltaX)); //выводим значение производной
        } else if (key == 2) { //если значение 2
            int x;
            scanf("%d",&x); //считываем x
            printf("%s(%d) = %f\n","E",x,(*E)(x)); //выводим значение e
        } else if (key == 0) { //если значение 0
            //dlclose(void* handle) уменьшает на единицу счетчик ссылок на
указатель динамической библиотеки handle
            dlclose(cur_lib);  //закрываем библиотеку

            if (lib_num == 1) { //если значение библиотеки 1
                lib_num = 2; //меняем значение на 2
                cur_lib = dlopen("./libd2.so",RTLD_LAZY); //загружаем и
открываем библиотеку 2
            } else {
                lib_num = 1; //иначе меняем значение на 1
                cur_lib = dlopen("./libd1.so",RTLD_LAZY); //загружаем и
открываем библиотеку 1
            }
            //находим начало адреса функций Derivative и E другой
библиотеки
            Derivative = dlsym(cur_lib,"Derivative");
            E = dlsym(cur_lib, "E");
        }
    }

    dlclose(cur_lib); //закрываем библиотеку
}
```

Файл run.sh

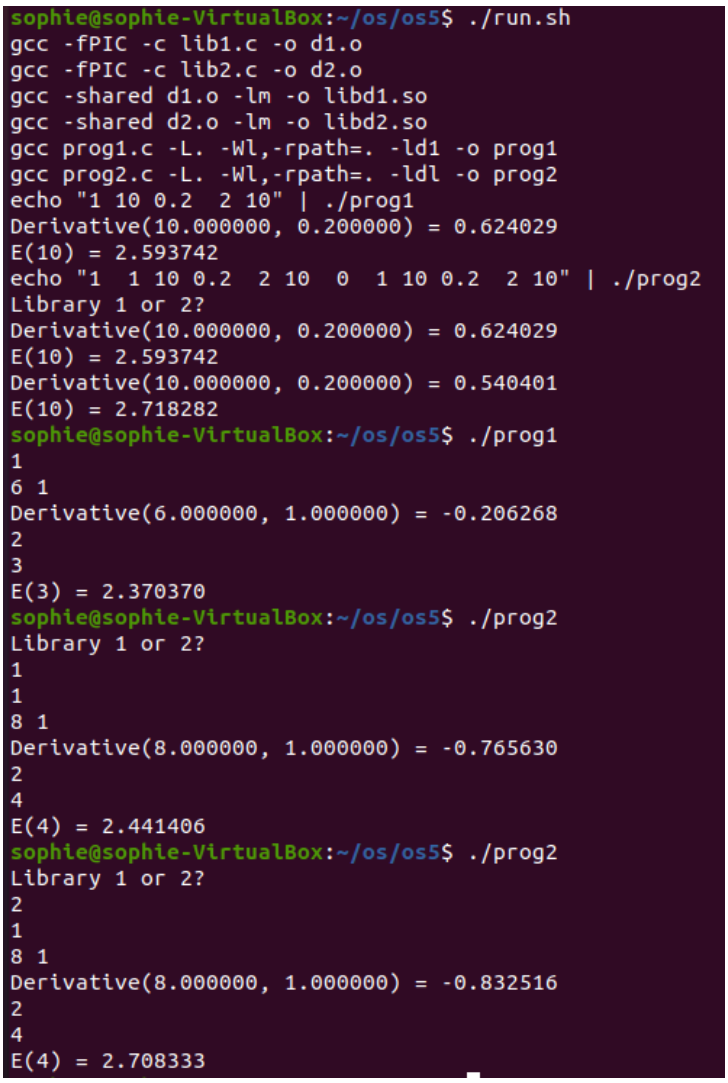
```
#!/bin/bash

exec >& >(tee run.log)

make
```

Файл make

```
.PHONY: all clean

all: build test

# lib1.c -> d1.o -> libd1.so -(runtime)-> prog2
build:
      gcc -fPIC -c lib1.c -o d1.o
      gcc -fPIC -c lib2.c -o d2.o

      @# `-lm` -- libmath
      gcc -shared d1.o -lm -o libd1.so
      gcc -shared d2.o -lm -o libd2.so

      @# `-rpath=.` -- сообщаем линковщику (`ld`), что при запуске
программы
      @# ей нужно искать динамические библиотеки в текущем каталоге.
      @#
```

```
	@# `-L.` -- сообщаем линковщику, что и ему самому нужно искать
библиотеки
	@# в текущем каталоге (в частности, чтобы проверить корректность их
	@# использования).
	gcc prog1.c -L. -Wl,-rpath=. -ld1 -o prog1
	gcc prog2.c -L. -Wl,-rpath=. -ldl -o prog2

test:
	echo "1 10 0.2  2 10" | ./prog1
	echo "1  1 10 0.2  2 10  0  1 10 0.2  2 10" | ./prog2

clean:
	rm d1.o d2.o libd1.so libd2.so prog1 prog2
```

Пример работы



Вывод

В данной лабораторной работе я научилась работать с динамическими библиотеками, изучила ряд функций для этого, а также использовала их для математических вычислений по различным формулам.