Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу

«Операционные системы»

Тема работы

Динамические библиотеки

Студент: Шаларь Игорь Павлович
Группа: М8О-208Б-20
Вариант: 27
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

Репозиторий

https://github.com/IgShalar/OS/tree/main/os_lab5

Постановка задачи

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.

Контракты и реализация функций:

1. Расчет значения числа Пи при заданной длине ряда (K) (float Pi(int K))

   Ряд Лейбница;Формула Валлиса

2. Расчет 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!))

Общие сведения о программе

Запуск:

_$ make

_$ ./main1

или

_$ ./main2

Написано для Unix.

lib1.cpp и lib2.cpp - содержат реализации контрактов.

1.cpp - использует знания, полученные на этапе компиляции

2.cpp - динамически загружает библиотеки, используя следующие вызовы:

dlopen() - загружает динамическую библиотеку.

dlsym() - загружает определенную функцию.

dlclose() - уменьшает на единицу счетчик ссылок на указатель динамической библиотеки, если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается.

Общий метод и алгоритм решения

Реализую функции, описанные в варианте.

Исходный код

Makefile:

```
all: main1 main2

main1: lib1.so lib2.so 1.cpp
	g++ 1.cpp -L. -ld1 -ld2 -o main1 -Wl,-rpath -Wl,.

main2: lib1.so libd2.so 2.cpp
	g++ 2.cpp -L. -ldl -o main2 -Wl,-rpath -Wl,.

lib1.so: d1.o
	g++ -shared d1.o -o libd1.so

lib2.so: d2.o
	g++ -shared d2.o -o libd2.so

d1.o: lib1.cpp
```

```
	g++ -fPIC -c lib1.cpp -o d1.o

d2.o: lib2.cpp
	g++ -fPIC -c lib2.cpp -o d2.o
clean:
	rm -r *.so *.o main1 main2
```

lib1.cpp:

```
#include<math.h>

extern "C" float Leibniz(int a);
extern "C" float Wallis(int a);

float Leibniz(int n){
    float res = 0;
    for (int i = 0; i < n; i++){
        res += pow(-1, i) / (2*i + 1);
    }
    return res * 4;
}

float Wallis(int n){
    float res = 1;
    for (int i = 1; i <= n; i++){
        res *= (4 * pow(i, 2)) / (4 * pow(i, 2) - 1);
    }
    return res * 2;
}
```

lib2.cpp:

```
#include<math.h>

extern "C" float formula(int x);
extern "C" float row (int x);

float formula(int x){
    float x1 = x;
```

```
    float res = pow(1 + 1 / x1, x1);
    return res;
}

float f(float x){
    float res = 1;
    for (int i = 1; i <= x; i++) res *= i;
    return res;
}

float row (int x){
    float res = 0;
    for (float i = 0; i <= x; i++) res += 1 / f(i);
    return res;
}
```

1.cpp:

```
#include<iostream>

extern "C" float Leibniz(int a);
extern "C" float Wallis(int a);
extern "C" float formula(int x);
extern "C" float row (int x);

using namespace std;

int main(){
    cout << endl << "USAGE:" << endl;
    cout << "calculate pi: 1 [x]" << endl;
    cout << "example: 1 10000" << endl;
    cout << "calculate e: 2 [x]" << endl;
    cout << "change pi realisation: 0 1" << endl;
    cout << "change e realisation: 0 2" << endl;
    cout << "exit: q" << endl;
    cout << "_____" << endl << endl;
    bool m1 = 0, m2 = 0;
```

```
    int a;
    char ch;
    while (1){
        cin >> ch;
        switch (ch){
            case '1':{
                cin >> a;
                if (m1) cout << Wallis(a) << endl;
                else cout << Leibniz(a) << endl;
                break;
            }
            case '2': {
                cin >> a;
                if (m2) cout << row(a) << endl;
                else cout << formula(a) << endl;
                break;
            }
            case '0': {
                cin >> a;
                if (a == 1) m1 = !(m1);
                else if (a == 2) m2 = !(m2);
                break;
            }
            case 'q': {
                return 0;
            }
        }
    }
    return 0;
}
```

2.cpp:

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

using namespace std;
```

```
int main(){
    void * lib_handle1 = dlopen("libd1.so", RTLD_LAZY);
    if (!lib_handle1){
        perror("dlopen lib1 error.");
        return -1;
    }
    void * lib_handle2 = dlopen("libd2.so", RTLD_LAZY);
    if (!lib_handle1){
        perror("dlopen lib1 error.");
        return -1;
    }
    float (* fn1)(int);
    float (* fn2)(int);
    fn1 = (float (*)(int)) dlsym(lib_handle1, "Leibniz");
    if (fn1 == NULL){
        perror("dlsym error.");
        return -1;
    }
    fn2 = (float (*)(int)) dlsym(lib_handle2, "formula");
    if (fn2 == NULL){
        perror("dlsym error.");
        return -1;
    }

    cout << endl << "USAGE:" << endl;
    cout << "calculate pi: 1 [x]" << endl;
    cout << "example: 1 10000" << endl;
    cout << "calculate e: 2 [x]" << endl;
    cout << "change pi realisation: 0 1" << endl;
    cout << "change e realisation: 0 2" << endl;
    cout << "exit: q" << endl;
    cout << "_____" <<
endl << endl;
    bool m1 = 0, m2 = 0;
    int a;
    char ch;
```

```
while (1){
    cin >> ch;
    switch (ch){
        case '1':{
            cin >> a;
            cout << fn1(a) << endl;
            break;
        }
        case '2': {
            cin >> a;
            cout << fn2(a) << endl;
            break;
        }
        case '0': {
            cin >> a;
            if (a == 1){
                if (m1){
                    fn1 = (float (*)(int)) dlsym(lib_handle1, "Leibniz");
                    if (fn1 == NULL){
                        perror("dlsym error.");
                        return -1;
                    }
                }
                else{
                    fn1 = (float (*)(int)) dlsym(lib_handle1, "Wallis");
                    if (fn1 == NULL){
                        perror("dlsym error.");
                        return -1;
                    }
                }
                m1 = !(m1);
            }
            else if (a == 2){
                if (m2){
                    fn2 = (float (*)(int)) dlsym(lib_handle2, "formula");
                    if (fn2 == NULL){
                        perror("dlsym error.");
```

```
                    return -1;
                }
            }
            else{
                fn2 = (float (*)(int)) dlsym(lib_handle2, "row");
                if (fn2 == NULL){
                    perror("dlsym error.");
                    return -1;
                }
            }
            m2 = !(m2);
        }
        break;
    }
    case 'q': {
        return 0;
    }
  }
 }
 dlclose (lib_handle1);
 dlclose (lib_handle2);
 return 0;
}
```

Демонстрация работы программы

1.cpp:

1)Ввод:

1 100

Вывод:
3.13159

2)Ввод:

0 1
1 100

Вывод:
3.13379

2.cpp:

1)Ввод:

2 100

Вывод:
2.70481

2)Ввод:
0 2
2 100

Вывод:
2.71828

Выводы

Познакомился с динамическими библиотеками. У них есть множество преимуществ: несколько запущенных приложений могут использовать одну и ту же динамическую библиотеку без необходимости иметь собственную копию для каждого из них, также динамическая библиотека может изменяться в ходе выполнения программы. Однако то, что библиотека может меняться во время выполнения делает программу менее устойчивой к ошибкам.