

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу
«Операционные системы»

Студент: Пирогов М.Д.
Группа: М8О-207Б-21
Вариант: 13
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

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/pirogovmark/OS-Labs>

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 использует 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 обеих 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.

Вариант 13:

Контракт 1:

Расчет производной функции $\cos(x)$ в точке A с приращением δx .

Float Derivative(float A, float deltaX)

Реализация 1:

$$f'(x) = (f(A + \delta x) - f(A)) / \delta x$$

Реализация 2:

$$f'(x) = (f(A + \delta x) - f(A - \delta x)) / (2 * \delta x)$$

Контракт 2:

Подсчет площади плоской геометрической фигуры по двум сторонам

Float Square(float A, float B)

Реализация 1:

Фигура прямоугольник

Реализация 2:

Фигура прямоугольный треугольник

Вариант 1:

Система сборки: Makefile.

Общие сведения о программе

Программа состоит из двух интерфейсов, объединенных в один. Каждая реализация контрактов представляет из себя отдельный файл: lib1.c и lib2.c. Для объявления необходимых функций также используется заголовочный файл lib.h. В проекте присутствует Makefile

Общий метод и алгоритм решения

Объявим необходимые функции внутри файла lib.h. Используем спецификатор хранения extern, который сообщает компилятору, что

находящиеся за ним типы и имена переменных объявляются где-то в другом месте.

Так как по заданию необходимо подключать библиотеки на этапе линковки, то подключать `lib.h` в реализации `lib1.c` и `lib2.c` не следует. В этих файлах просто напишем логику работы необходимых функций. Важно, чтобы они назывались также, как и те, что объявлены в `lib.h`.

Интерфейс 1:

Подключаем `lib.h` и пользуемся функциями так, как будто библиотека обычная. Различия наступают в сборке программы. Если бы мы собирали такой код в терминале, то прописали бы `gcc -c -fPIC lib1.c`. Опция `-fPIC` - требует от компилятора, при создании объектных файлов, порождать позиционно-независимый код. Формат позиционно-независимого кода позволяет подключать исполняемые модули к коду основной программы в момент ее загрузки. Далее `gcc -shared -o liblib1.so lib1.o -lm`. Опция `-shared` - указывает gcc, что в результате должен быть собран не исполняемый файл, а разделяемый объект — динамическая библиотека.

Интерфейс 2:

Воспользуемся системными вызовами из библиотеки `<dlfcn.h>`.

Функция `dlopen` открывает динамическую библиотеку (объект `.so`) по названию.

Функция `dlsym` - обработчик динамически загруженного объекта вызовом `dlopen`.

Функция `dlclose`, соответственно, закрывает динамическую библиотеку.

Собираем с помощью `gcc -L. -Wall -o main.out main2.c -llib2 -llib1`. Флаг `-L.` Означает, что поиск файлов библиотек будет начинаться с текущей директории.

Исходный код

lib.h

```
#ifndef __LIB_H__
```

```
#define __LIB_H__
```

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

```
extern float Square(float A, float B);
```

```
#endif
```

lib1.c

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

```
#include <math.h>
```

```
float Derivative(float A, float deltaX) {  
    printf("Calculation of derivative function  $f(x) = \cos(x)$ \n");  
    printf("at the point %f with approximation %f\n", A, deltaX);  
    printf("by formula  $f'(x) = (f(A + \text{deltaX}) - f(A)) / \text{deltaX}$ \n");  
    float dfdx = (cos(A + deltaX) - cos(A)) / deltaX;  
    return dfdx;  
}
```

```
float Square(float A, float B) {  
    printf("Calculation square of rectangle\n");  
    printf("with length: %f, width: %f\n", A, B);  
    return A * B;  
}
```

lib2.c

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

```
#include <math.h>
```

```

float Derivative(float A, float deltaX) {
    printf("Calculation of derivative function  $f(x) = \cos(x)$ \n");
    printf("at the point %f with approximation %f\n", A, deltaX);
    printf("by formula  $f'(x) = (f(A + \text{deltaX}) - f(A - \text{deltaX})) / (2 * \text{deltaX})$ \n");
    float dfdx = (cos(A + deltaX) - cos(A - deltaX)) / (2 * deltaX);
    return dfdx;
}

```

```

float Square(float A, float B) {
    printf("Calculation square of right triangle\n");
    printf("with length: %f, width: %f\n", A, B);
    return (A * B) / ((float) 2);
}

```

main.c

```

#include <stdio.h>

#include <dlfcn.h>

#include "lib.h"

```

```

void printMenu(){
    printf("\nEnter command:\n");
    printf("\n0 - to change methods of calculation\n");
}

```

```
    printf("\n1 - to compute the derived function  $f(x) = \cos(x)$  with arguments point  
and delta\n");
```

```
    printf("\n2 - to calculate the area of flat figure with arguments length and  
width\n");
```

```
    printf("\n3 - to end program\n");  
}
```

```
const char* lib1 = "./liblib1.so";
```

```
const char* lib2 = "./liblib2.so";
```

```
int main(int argc, char const *argv[]) {  
    float A = 0, deltaX = 0, B = 0;  
    int command = 0, link = 0, flag = 1;  
    void *currentLib = dlopen(lib1, RTLD_LAZY);  
    printMenu();  
    printf("\nCurrent method: %d\n\n", link + 1);  
    float (*Derivative)(float A, float deltaX);  
    float (*Square)(float A, float B);  
    Derivative = dlsym(currentLib, "Derivative");  
    Square = dlsym(currentLib, "Square");  
  
    while (flag) {  
        scanf("%d", &command);  
        switch (command) {
```


case 0:

```
dlclose(currentLib);  
if (link == 0) {  
    currentLib = dlopen(lib2, RTLD_LAZY);  
} else {  
    currentLib = dlopen(lib1, RTLD_LAZY);  
}  
link = !link;  
Derivative = dlsym(currentLib, "Derivative");  
Square = dlsym(currentLib, "Square");  
break;
```

case 1:

```
scanf("%f%f", &A, &deltaX);  
printf("\n-----\n");  
printf("Answer: %f\n", Derivative(A, deltaX));  
printf("\n-----\n");  
break;
```

case 2:

```
scanf("%f%f", &A, &B);  
printf("-----\n");  
printf("Answer: %f\n", Square(A, B));  
printf("-----\n");  
break;
```

```

    case 3:
        flag = 0;
        break;

    default:
        printf("Wrong command\n");
        break;
}

if (flag == 1) {
    printMenu();
    printf("\nCurrent method: %d\n\n", link + 1);
} else {
    printf("Program completed!\n");
}
}

return 0;
}

```

Makefile

compile:

```

gcc -c -Wall -Werror -fpic lib1.c
gcc -c -Wall -Werror -fpic lib2.c
gcc -shared -o liblib1.so lib1.o -lm

```

```
gcc -shared -o liblib2.so lib2.o -lm
```

```
gcc -Wall -o main.out main.c
```

clean:

```
rm *.o
```

```
rm *.so
```

```
rm *.out
```

Демонстрация работы программы

```
[(base) markp@MacBook-Air-Mark a5 % make ]
gcc -c -Wall -Werror -fpic lib1.c
gcc -c -Wall -Werror -fpic lib2.c
gcc -shared -o liblib1.so lib1.o -lm
gcc -shared -o liblib2.so lib2.o -lm
gcc -Wall -o main.out main.c
[(base) markp@MacBook-Air-Mark a5 % ./main.out ]
```

Enter command:

0 - to change methods of calculation

1 - to compute the derived function $f(x) = \cos(x)$ with arguments point and delta

2 - to calculate the area of flat figure with arguments length and width

3 - to end program

Current method: 1

1

4 0.1

Calculation of derivative function $f(x) = \cos(x)$
at the point 4.000000 with approximation 0.100000
by formula $f'(x) = (f(A + \text{deltaX}) - f(A))/\text{deltaX}$
Answer: 0.788196

Enter command:

0 - to change methods of calculation

1 - to compute the derived function $f(x) = \cos(x)$ with arguments point and delta

2 - to calculate the area of flat figure with arguments length and width

3 - to end program

Current method: 1

2

4.5 4.1

Calculation square of rectangle
with length: 4.500000, width: 4.100000
Answer: 18.449999

Enter command:

0 - to change methods of calculation

1 - to compute the derived function $f(x) = \cos(x)$ with arguments point and delta

2 - to calculate the area of flat figure with arguments length and width

3 - to end program

Current method: 1

0

Enter command:

0 - to change methods of calculation

1 - to compute the derived function $f(x) = \cos(x)$ with arguments point and delta

2 - to calculate the area of flat figure with arguments length and width

3 - to end program

Current method: 2

1

4 0.1

Calculation of derivative function $f(x) = \cos(x)$
at the point 4.000000 with approximation 0.100000
by formula $f'(x) = (f(A + \text{deltaX}) - f(A - \text{deltaX})) / (2 * \text{deltaX})$
Answer: 0.755541

Enter command:

0 – to change methods of calculation

1 – to compute the derived function $f(x) = \cos(x)$ with arguments point and delta

2 – to calculate the area of flat figure with arguments length and width

3 – to end program

Current method: 2

2

4.5 9

Calculation square of right triangle
with length: 4.500000, width: 9.000000
Answer: 20.250000

Выводы

В ходе лабораторной работы я познакомился с созданием динамических библиотек в операционных системах unix, а также с возможностью загружать эти библиотеки в ходе выполнения программы.