Московский авиационный институт

(Национальный исследовательский университет)

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные Системы»

Лабораторная работа № 5

Тема: Создание динамических библиотек и создание программ, которые используют функции динамических библиотек

Студент: Семин Александр Витальевич

Группа: М8О-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата:

Оценка:

Москва, 2021

1. **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.
- 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 10

Функции:

2	Рассчет производной функции cos(x) в точке A с приращением deltaX	Float Derivative(float A, float deltaX)	f'(x) = (f(A + deltaX) – f(A))/deltaX	f'(x) = (f(A + deltaX) – f(A-deltaX))/(2*deltaX)

4	Подсчёт наибольшего общего делителя для двух натуральных чисел	Int GCF(int A, int B)	Алгоритм Евклида	Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.
5	Рассчет значения числа Пи	float Pi(int K)	Ряд Лейбница	Формула Валлиса

2. **Набор тестов**

Первый тестовый набор:

```
progger@asus:~/Desktop/OS_labs/l5$ ./prog1
1. Рассчет производной функции cos(x) в точке A с приращением deltaX
USAGE: float Derivative(float A, float deltaX)

2. Подсчёт наибольшего общего делителя для двух натуральных чисел A B
USAGE: int GCF(int A, int B)

1 0.123 0.01
-0.127650
2 14 49
7
```

Второй тестовый набор:

```
progger@asus:~/Desktop/OS_labs/l5$ ./prog2
1. Рассчет производной функции cos(x) в точке A с приращением deltaX
USAGE: float Derivative(float A, float deltaX)

2. Подсчёт наибольшего общего делителя для двух натуральных чисел A B
USAGE: int GCF(int A, int B)

1 0.123 0.01
-0.127650
0
Switched to version 2
1 0.123 0.01
-0.122688
2 49 14
7
```

3. **Листинг программы**

functions.h

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H


float Derivative(float A, float deltaX);
int GCF(int A, int B);


#endif
```

lib.c

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

//Рассчет производной функции cos(x) в точке A с приращением deltaX
float Derivative(float A, float deltaX) {
	float res = (cos(A+deltaX) - cos(A)) / deltaX;
	return res;
}

//Подсчёт наибольшего общего делителя для двух натуральных чисел A B
//Алгоритм Евклида
int GCF(int A, int B) {
	while (A != B) {
		if (A > B)
			A -= B;
		else B -= A;
	}
	return A;
}
```

prog1.c

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

void menu() {
	printf("1. Рассчет производной функции cos(x) в точке A с приращением
deltaX\n");
	printf("USAGE: float Derivative(float A, float deltaX)\n\n");
	printf("2. Подсчёт наибольшего общего делителя для двух натуральных
чисел A B\n");
	printf("USAGE: int GCF(int A, int B)\n\n");
}


int main() {
	int cmd;
	menu();
	while(scanf("%d", &cmd) != EOF) {
		if (cmd == 1) {
```

```
                float x, y;
                if(scanf("%f %f", &x, &y) != 2) {
                        printf("Invalid arguments!\n");
                        continue;
                }
                printf("%f\n", Derivative(x, y));
        } else if (cmd == 2) {
                int x, y;
                if(scanf("%d%d", &x, &y) != 2) {
                        printf("Invalid arguments!\n");
                        continue;
                }
                printf("%d\n", GCF(x, y));
        } else {
                printf("Invalid command!\n");
                menu();
        }
    }
}
```

lib2.c

```
#include <math.h>


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

//Подсчёт наибольшего общего делителя для двух натуральных чисел A B
//Наивный алгоритм.
int GCF(int A, int B) {
    int prev_nod = 0;
    int nod = 1;
    while (nod < A && nod < B) {
        if (A % nod == 0 && B % nod == 0) {
            prev_nod = nod;
        }
        nod++;
    }
    return prev_nod;
}
```

prog2.c

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

void menu() {
    printf("1. Рассчет производной функции cos(x) в точке A с приращением
```

```
deltaX\n");
      printf("USAGE: float Derivative(float A, float deltaX)\n\n");
      printf("2. Подсчёт наибольшего общего делителя для двух натуральных
чисел A B\n");
      printf("USAGE: int GCF(int A, int B)\n\n");
}



int main() {
      float (*Derivative)(float, float);
      int (*GCF)(int, int);

      void* l1_handler = dlopen("./lib1.so", RTLD_LAZY);
      void* l2_handler = dlopen("./lib2.so", RTLD_LAZY);
      if (!l1_handler || !l2_handler) {
          fprintf(stderr,"DLOPEN error: %s\n", dlerror());
          return -1;
      }
      Derivative = dlsym(l1_handler,"Derivative");
      GCF = dlsym(l1_handler,"GCF");

      int ver = 0;
      int cmd;
      menu();
      while (scanf("%d", &cmd) != EOF) {
            if (cmd == 0) {
                  ver ^= 1;
                  if (ver == 0) {
                        Derivative = dlsym(l1_handler,"Derivative");
                        GCF = dlsym(l1_handler,"GCF");
                  } else {
                        Derivative = dlsym(l2_handler,"Derivative");
                        GCF = dlsym(l2_handler,"GCF");
                  }
                  printf("Switched to version %d\n", ver + 1);
            } else if (cmd == 1) {
                  float x, y;
                  if (scanf("%f %f", &x, &y) != 2) {
                        printf("Invalid arguments!\n");
                        continue;
                  }
                  printf("%f\n", Derivative(x, y));
            } else if (cmd == 2) {
                  int x, y;
                  if (scanf("%d %d", &x, &y) != 2) {
                        printf("Invalid arguments!\n");
                        continue;
                  }
                  printf("%d\n", GCF(x, y));
            } else {
                  printf("Invalid command!\n");
                  menu();
```

```
            }

      }
      dlclose(l1_handler);
      dlclose(l2_handler);
}
```

Makefile:

```
      all: prog1 prog2

lib1.so: lib1.c
        gcc -shared lib1.c -o lib1.so -lm -Wall

lib2.so: lib2.c
        gcc -shared lib2.c -o lib2.so -lm -Wall

prog2: lib1.so lib2.so prog2.c
        gcc prog2.c -ldl -o prog2 -Wall

prog1: lib1.so prog1.c
        gcc prog1.c -L "/home/progger/Desktop/OS_labs/l5" -Wl,-R. -l1 -o
prog1 -Wall
```

4. **Выводы**

В ходе выполнения данной работы я разобрался с созданием и использованием динамических библиотек.

Такие библиотеки используются во всех крупных проектах, чтобы для внесения изменений была возможность перекомпилировать только одну библиотеку, а не весь проект целиком. Также одно из их достоинств заключается в том, что достаточно один раз выгрузить динамическую библиотеку в память и ей смогут пользоваться другие программы.