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: Фролов М.А.

Группа: М8О–206Б–20

Вариант: 9

Преподаватель: Соколов Андрей Алексеевич

Оценка: 5

Дата: 04.12.21

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

* Создание динамических библиотек
* Создание программ, которые используют функции динамических библиотек

**Задание**

Требуется создать динамические библиотеки, которые реализуют определённый 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 No1*), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (*программа No2*), 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 No2*). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;

1. «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;
2. «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 9.

2. Рассчет производной функции cos(x) в точке A с приращением deltaX Float Derivative(float A, float DeltaX) двумя способами: и .

7. Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные) Int PrimeCount(int A, int B): наивный алгоритм и решето Эратосфена.

В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамическую библиотеку с указанным именем. В случае неуспеха возвращает NULL.
2. **dlclose –** уменьшает на единицу счетчик ссылок на указатель динамической библиотеки handle. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается.  В случае успеха возвращает 0, иначе ненулевой результат.
3. **dlsym** – использует указатель на динамическую библиотеку, возвращаемую dlopen, и оканчивающееся нулем символьное имя, а затем возвращает адрес, указывающий на нужный символ. В случае неуспеха dlsym возвращает NULL.
4. **dlerror** – возвращает сообщение об ошибке, если ошибки не произошло, то возвращает NULL.

**Основные файлы программы**

**Makefile:**

done: 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/michael/Desktop/os/os\_laba\_5" -Wl,-R. -l1 -o prog1 –Wall

**functions.h**

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

float Derivative(float a, float deltax);

int PrimeCount(int a, int b);

#endif

**lib1.c**

#include <math.h>

#include <stdbool.h>

float Derivative(float a, float deltax){

return (cos(a + deltax) - cos(a)) / deltax;

}

int PrimeCount(int a, int b){

int cnt = 0;

bool fnd\_divider = false;

if(a == 1) ++a;

for(int i = a; i <= b; ++i){

for(int j = i - 1; j > 1; --j){

if(i % j == 0){

fnd\_divider = true;

break;

}

}

if(!fnd\_divider){

++cnt;

}

fnd\_divider = false;

}

return cnt;

}

**lib2.c**

#include <math.h>

float Derivative(float a, float deltax){

return (cos(a + deltax) - cos(a - deltax)) / (2 \* deltax);

}

int PrimeCount(int a, int b){

int cnt = 0;

int sieve[b + 1];

for(int i = 0; i < b + 1; ++i){

sieve[i] = 0;

}

sieve[0] = 1;

sieve[1] = 1;

for(int i = 2; i <= b; ++i){

if(sieve[i] != 0){

continue;

}

for(int j = 2 \* i; j <= b; j += i){

sieve[j] = 1;

}

++cnt;

}

return cnt;

}

**main1.c**

#include <stdio.h>

#include "functions.h"

int main(){

printf("1 + float A + float deltaX - derivative of cos in dot A with deltaX as delta\n");

printf("2 + int A + int B - number of prime numbers in [a,b]\n");

int command;

while(scanf("%d", &command) != EOF){

switch(command){

case 1:{

float a, deltax;

if(scanf("%f%f", &a, &deltax) != 2){

printf("Wrong arguements!\n");

continue;

}

printf("%f\n", Derivative(a, deltax));

break;

}

case 2:{

int a, b;

if(scanf("%d%d", &a, &b) != 2){

printf("Wrong arguements!\n");

continue;

}

printf("%d\n", PrimeCount(a, b));

break;

}

default:{

printf("Wrong command!\n");

}

}

}

}

**main2.c**

#include <stdio.h>

#include <dlfcn.h>

int main(){

int version = 0;

int command;

float (\*Derivative)(float, float) = NULL;

int (\*PrimeCount)(int, int) = NULL;

void \*lib1\_handler = dlopen("./lib1.so",RTLD\_LAZY);

if (!lib1\_handler){

fprintf(stderr,"dlopen() error: %s\n", dlerror());

return -1;

}

void \*lib2\_handler = dlopen("./lib2.so",RTLD\_LAZY);

if (!lib2\_handler){

fprintf(stderr,"dlopen() error: %s\n", dlerror());

return -1;

}

Derivative = dlsym(lib1\_handler,"Derivative");

PrimeCount = dlsym(lib1\_handler,"PrimeCount");

printf("0 - to switch realizations\n");

printf("1 + float A + float deltaX - derivative of cos in dot A with deltaX as delta\n");

printf("2 + int A + int B - number of prime numbers in [a,b]\n");

while(scanf("%d", &command) != EOF){

switch(command){

case 0:{

version ^= 1;

if(!version){

Derivative = dlsym(lib1\_handler,"Derivative");

PrimeCount = dlsym(lib1\_handler,"PrimeCount");

} else{

Derivative = dlsym(lib2\_handler,"Derivative");

PrimeCount = dlsym(lib2\_handler,"PrimeCount");

}

printf("Switched to realization %d\n", version + 1);

break;

}

case 1:{

float a, deltax;

if(scanf("%f%f", &a, &deltax) != 2){

printf("Wrong arguements!\n");

continue;

}

printf("%f\n", Derivative(a, deltax));

break;

}

case 2:{

int a, b;

if(scanf("%d%d", &a, &b) != 2){

printf("Wrong arguements!\n");

continue;

}

printf("%d\n", PrimeCount(a, b));

break;

}

default:{

printf("Wrong command!\n");

}

}

}

dlclose(lib1\_handler);

dlclose(lib2\_handler);

}

**Пример работы**

michael@michael-VirtualBox:~/Desktop/os/os\_laba\_5$ make

make: \*\*\* No targets. Stop.

michael@michael-VirtualBox:~/Desktop/os/os\_laba\_5$ ./prog1

1 + float A + float deltaX - derivative of cos in dot A with deltaX as delta

2 + int A + int B - number of prime numbers in [a,b]

1 2.4 0.04

-0.660536

2 1 7

4

^C

michael@michael-VirtualBox:~/Desktop/os/os\_laba\_5$ ./prog2

0 - to switch realizations

1 + float A + float deltaX - derivative of cos in dot A with deltaX as delta

2 + int A + int B - number of prime numbers in [a,b]

1 2.4 0.04

-0.660536

2 1 7

4

0

Switched to realization 2

1 2.4 0.04

-0.675282

2 1 7

4

^C

**Вывод**

В ходе выполнения данной лабораторной работы я познакомился с тем, как создавать и использовать динамические библиотеки. Стоить отметить, что в простых программах с минимальной функциональностью использование статических библиотек может быть предпочтительней, однако в больших программах, использующих несколько библиотек, использование динамических библиотек позволяет снизить потребление оперативной и дисковой памяти, поскольку динамическую библиотеку достаточно один раз выгрузить в память, что ею могли пользоваться все нуждающиеся программы.