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: Шевлякова София Сергеевна

Группа: М8О–208Б–21

Вариант: 16

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Постановка задачи**

## Цель работы

Приобретение практических навыков в:

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

## Задание

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки двумя 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 следующее:

* Динамические библиотеки, реализующие контракты, которые заданы вариантом;
* Тестовая программа No1, которая использует одну из библиотек, используя знания, полученные на этапе компиляции;
* Тестовая программа No2, которая загружает библиотеки, используя 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 на другую (необходимо только для программы No2);
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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Описание | Сигнатура | Реализация 1 | Реализация 2 |
| 3 | Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные) | Int PrimeCount(int A, int B) | Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа. | Решето Эратосфена |
| 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. |

**Общие сведения о программе**

Программа компилируется из файлов main.c, main\_dyn.c, realisation1.c, realisation2.c. Также используется заголовочные файлы: stdio.h, realisation.h, dlfcn.h. В программе используются следующие системные вызовы:

1. **dlopen()** – загружает (открывает) динамическую библиотеку. Возвращает указатель на загруженную библиотеку, в случае ошибки возвращает NULL;
2. **dlsym()** – получение адреса функции или переменной из библиотеки. Возвращает адрес функции, в случае ошибки возвращает NULL;
3. **dlerror()** – возвращает понятную человеку строку, описывающую последнюю ошибку, которая произошла при вызове одной из функции dlopen, dlsym, dlclose. Возвращает NULL если не возникло ошибок с момента инициализации или с момента ее последнего вызова;
4. **dlclose()** – уменьшает на единицу счетчик ссылок на указатель динамической библиотеки. Возвращает 0 при удачном завершении и ненулевой результат при ошибке;

**Общий метод и алгоритм решения**

Описываем решения в библиотечных файлах, создаём общий заголовочный файл. Нам не потребуется два, так как в обеих реализациях одни и те же функции, соответственно, между двумя заголовочными файлами не было бы различия. Далее собираем всё в исполняемый файл.

**Основные файлы программы**

====================== main\_dyn.c ======================

#include <dlfcn.h>

#include <stdio.h>

#define check\_ok(VALUE, OKVAL, MSG) if (VALUE != OKVAL) { printf("%s", MSG); return 1; }

#define check\_wrong(VALUE, WRONG\_VAL, MSG) if (VALUE == WRONG\_VAL) { printf("%s", MSG); return 1; }

const char\* DYN\_LIB\_1 = "./libDyn1.so";

const char\* DYN\_LIB\_2 = "./libDyn2.so";

const char\* GCD\_NAME = "GCD";

const char\* PRIME\_COUNTER\_NAME = "PrimeCount";

int main(int argc, const char\*\* argv) {

int dynLibNum = 1;

void\* handle = dlopen(DYN\_LIB\_1, RTLD\_LAZY);

check\_wrong(handle, NULL, "Error opening dynamic library!\n")

int (\*GCD)(int, int);

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

GCD = dlsym(handle, GCD\_NAME);

PrimeCount = dlsym(handle, PRIME\_COUNTER\_NAME);

char\* error = dlerror();

check\_ok(error, NULL, error)

int q;

int x, y;

int A, B;

printf("Enter query: 0) change realisation 1) get GCD(x,y) 2) count primes between A, B\n");

while (scanf("%d", &q) > 0) {

switch (q) {

case 0:

check\_ok(dlclose(handle), 0, "Error closing dynamic library!\n")

if (dynLibNum) {

handle = dlopen(DYN\_LIB\_2, RTLD\_LAZY);

} else {

handle = dlopen(DYN\_LIB\_1, RTLD\_LAZY);

}

check\_wrong(handle, NULL, "Error opening dynamic library!\n")

GCD = dlsym(handle, GCD\_NAME);

error = dlerror();

check\_ok(error, NULL, error)

PrimeCount = dlsym(handle, PRIME\_COUNTER\_NAME);

error = dlerror();

check\_ok(error, NULL, error)

dynLibNum = dynLibNum ^ 1;

break;

case 1:

printf("enter integer x, y: ");

check\_ok(scanf("%d%d", &x, &y), 2, "Error reading integer!\n");

printf("GCD(%d, %d) = %d\n", x, y, GCD(x, y));

break;

case 2:

printf("enter integer A, B: ");

check\_ok(scanf("%d %d", &A, &B), 2, "Error reading floats!\n");

printf("There are %d primes between %d and %d\n", PrimeCount(A, B), A, B);

break;

default:

printf("End.\n");

check\_ok(dlclose(handle), 0, "Error closing dynamic library!\n")

return 0;

}

}

}

====================== main.c ======================

#include "realisation.h"

#include <stdio.h>

#define check\_ok(VALUE, OKVAL, MSG) if (VALUE != OKVAL) { printf("%s", MSG); return 1; }

int main(int argc, const char\*\* argv) {

int q;

int x, y;

int A, B;

printf("Enter query: 1) get GCD(x,y) 2) count primes between A, B\n");

while (scanf("%d", &q) > 0) {

switch (q) {

case 1:

printf("enter integer x, y: ");

check\_ok(scanf("%d%d", &x, &y), 2, "Error reading integer!\n");

printf("GCD(%d, %d) = %d\n", x, y, GCD(x, y));

break;

case 2:

printf("enter integer A, B: ");

check\_ok(scanf("%d %d", &A, &B), 2, "Error reading integers!\n");

printf("There are %d primes between %d and %d\n", PrimeCount(A, B), A, B);

break;

default:

printf("End.\n");

return 0;

}

}

}

====================realisation1.c =====================

#include "realisation.h"

void swap\_int(int\* x, int\* y) {

int tmp = \*x;

\*x = \*y;

\*y = tmp;

}

int GCD(int x, int y) {

while (y > 0) {

if (x >= y) {

x = x % y;

}

swap\_int(&x, &y);

}

return x;

}

int isPrime(int n) {

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

if (n % i == 0)

return 0;

}

return 1;

}

int PrimeCount(int A, int B) {

int res = 0;

if (A < 2) A = 2;

for (int i = A; i <= B; ++i) {

res += isPrime(i);

}

return res;

}

====================realisation2.c =====================

#include "realisation.h"

void swap\_int(int\* x, int\* y) {

int tmp = \*x;

\*x = \*y;

\*y = tmp;

}

int GCD(int x, int y) {

if (x > y) {

swap\_int(&x, &y);

}

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

if (x % i == 0 && y % i == 0) {

return i;

}

}

return 1;

}

int PrimeCount(int A, int B) {

if (A < 2) A = 2;

int res = 0;

char sieve [SQRT\_OF\_MAXINT];

for (int i = 0; i < SQRT\_OF\_MAXINT; ++i) sieve[i] = 1;

long long primes[SQRT\_OF\_MAXINT];

int t = 0;

for(long long i = 2; i < SQRT\_OF\_MAXINT; ++i) {

if (sieve[i] == 0) continue;

primes[t++] = i;

for(long long j = i \* i; j < SQRT\_OF\_MAXINT; j += i)

sieve[j] = 0;

}

int flag;

for (long long i = A; i <= B; ++i) {

flag = 1;

for (int j = 0; j < t && primes[j]\*primes[j] <= i; ++j) {

if (i % primes[j] == 0) {

flag = 0;

break;

}

}

res += flag;

}

return res;

}

====================realisation.h =====================

#ifndef LAB\_5\_REALISATION\_H

#define LAB\_5\_REALISATION\_H

const int SQRT\_OF\_MAXINT = 46341;

int GCD(int x, int y);

int PrimeCount(int A, int B);

#endif

======================== test1.txt ========================

1

167 85637

2

46578 465713

0

1

167 85637

2

46578 465713

5

======================== test2.txt ========================

1

100000 4635892

2

28372 35556

10

======================== test3.txt ========================

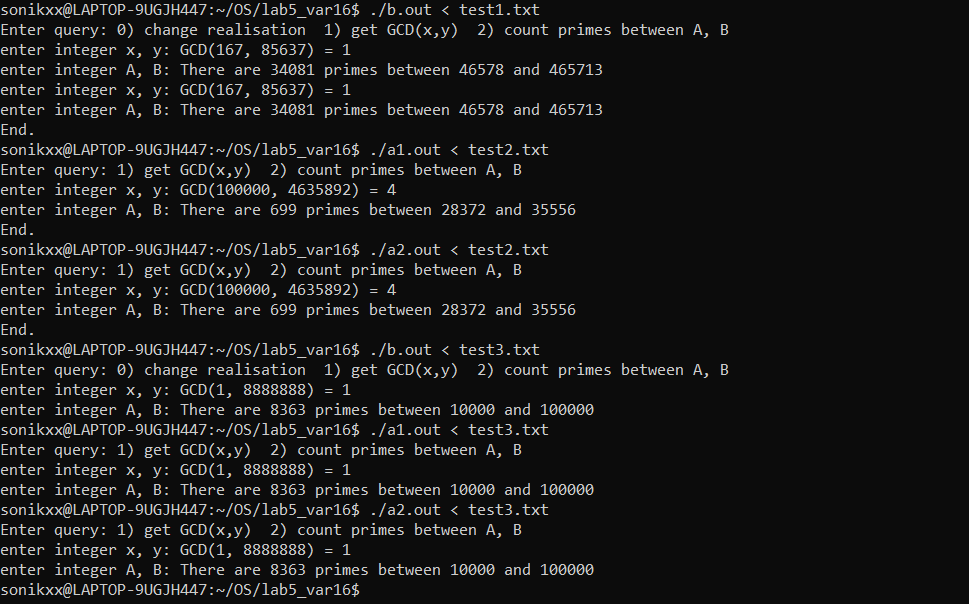
1

1 8888888

2

10000 100000

**Пример работы**

****

**Вывод**

Во время выполнения работы я изучила основы работы с динамическими библиотеками на операционных системах Linux, реализовала программу, которая использует созданные динамические библиотек. Выяснил некоторые различия в механизмах работы динамических и статических библиотек. Также я смогла сделать вывод что, использование библиотек добавляет модульность программе, что упрощает дальнейшую поддержку кода.