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: Рылов Александр Дмитриевич
Группа: М8О-207Б-21
Вариант: 16
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/Brokiloene/os

Постановка задачи

Цель работы

Изучить создание и использование 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.

Задание варианта

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 GCD(int A, int B)	Алгоритм Евклида	Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.

Сборочная система - Make

Общие сведения о программе

Программа компилируется из файлов main_static.c, main_dynamic.c, lib5_v1.c, lib5_v2.c. Также используется заголовочные файлы: stdlib.h, stdio.h, lib5.h, dlfcn.h. В программе используются следующие системные вызовы:

1. dlopen() – загружает общий динамический объект и возвращает «handle» на него.
2. dlsym() – указывает адресс в объекте, откуда загружать символ.
3. dlerror() – возвращает строку ошибки, связанную с работой динамического объекта.
4. dlclose() – уменьшает на единицу счетчик ссылок на указатель динамической библиотеки «handle».

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.

Исходный код

main_static.c | Makefile | main_dynamic.c | dyn_start.sh

```c
// var16 foo1-3(PrimeCount) foo2-4(GCD) 1-make

#include <stdio.h>

#include "lib5.h"



int main(int argc, char const *argv[])
{
    int mode, a, b;
    while(scanf("%d", &mode) > 0) {
        switch(mode) {
            case 1:
                check_ans(scanf("%d %d", &a, &b), 2, "scanf error\n");
                printf("PrimeCount = %d\n", PrimeCount(a, b));
                break;
            case 2:
                check_ans(scanf("%d %d", &a, &b), 2, "scanf error\n");
                printf("GCD = %d\n", GCD(a, b));
                break;
            default:
                return 0;
        }

    }
    return 0;
}

```

main_static.c | Makefile | main_dynamic.c | dyn_start.sh | lib5.h

```c
const char* DLL_1 = "lib5_v1.so";
const char* DLL_2 = "lib5_v2.so";

int main(int argc, char const *argv[])
{

    void *handle = dlopen(DLL_1, RTLD_LAZY);
    check_not_null(handle, "dll open error\n");

    int (*PrimeCount)(int, int);
    int (*GCD)(int, int);
    PrimeCount = dlsym(handle, "PrimeCount");
    GCD = dlsym(handle, "GCD");

    char *err = dlerror();
    check_null(err, err);

    int mode, a, b;
    int cur_lib = 1;

    while(scanf("%d", &mode) > 0) {
        switch(mode) {
            case 0:
                check_ans(dlclose(handle), 0, "error with closing dll\n");
                if (cur_lib == 1) {
                    handle = dlopen(DLL_2, RTLD_LAZY);
                    cur_lib = 0;
                    printf("switch lib from %s to %s\n", DLL_1, DLL_2);
                } else {
                    handle = dlopen(DLL_1, RTLD_LAZY);
                    cur_lib = 1;
                    printf("switch lib from %s to %s\n", DLL_2, DLL_1);
                }
                check_not_null(handle, "dll open error\n");
                PrimeCount = dlsym(handle, "PrimeCount");
                GCD = dlsym(handle, "GCD");
                char *err = dlerror();
                check_null(err, err);
                break;
            case 1:
                check_ans(scanf("%d %d", &a, &b), 2, "scanf error\n");
                printf("PrimeCount = %d\n", PrimeCount(a, b));
                break;
            case 2:
                check_ans(scanf("%d %d", &a, &b), 2, "scanf error\n");
                printf("GCD = %d\n", GCD(a, b));
                break;
            default:
                check_ans(dlclose(handle), 0, "error with closing dll\n");
                return 0;
```

main_static.c | Makefile | main_dynamic.c | dyn_start.sh | lib5.h

```
CC     = gcc
CFLAGS = -Wall -g
BINS   = main_static_v1 main_static_v2 lib5_v1.so lib5_v2.so main_dynamic

all: $(BINS)

%.o: %.c
    $(CC) $(CFLAGS) -c $<
%.a: %.o
    ar rcs $@ $<
%.so: %.o
    $(CC) $(CFLAGS) -fPIC -shared -o $@ $^ -lc

main_static_v1: main_static.c lib5_v1.a
    $(CC) $(CFLAGS) $^ -o main_static_v1

main_static_v2: main_static.c lib5_v2.a
    $(CC) $(CFLAGS) $^ -o main_static_v2

main_dynamic: main_dynamic.c
    $(CC) $(CFLAGS) -o $@ $^ -L. -l5_v1 -l5_v2


clean:
    rm *.a *.so main_static_v1 main_static_v2 main_dynamic
```

lib5_v1.c | main_static.c | Makefile | main_dynamic.c | dyn_start.sh

```c
// PrimeCount - Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.
// GCD - Алгоритм Евклида

#include "lib5.h"

void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int PrimeCount(int a, int b)
{
    printf("(lib5_v1)\n");
    int res = 0;
    for (int i = a; i <= b; ++i) {
        for (int j = 2; j < i; ++j) {
            if (i % j == 0 && i != j) {
                res -= 1;
                break;
            }
        }
        res += 1;
    }
    return res;
}

int GCD(int a, int b)
{
    printf("(lib5_v1)\n");
    while(b > 0) {
        if (a >= b) a = a % b;
        swap(&a, &b);
    }
    return a;
}
```

lib5_v1.c | main_static.c | Makefile | lib5.h | main_dynamic.c

```c
#ifndef _LAB_5_
#define _LAB_5_

#include <stdio.h> // perror
#include <stdlib.h> // EXIT_FAILURE & exit

#define check_ans(foo, ok, msg) do { \
if (foo != ok) { perror(msg); exit(EXIT_FAILURE); } \
} while(0);

#define check_null(foo, msg) do { \
if (foo != NULL) { perror(msg); exit(EXIT_FAILURE); } \
} while(0);

#define check_not_null(foo, msg) do { \
if (foo == NULL) { perror(msg); exit(EXIT_FAILURE); } \
} while(0);

int PrimeCount(int A, int B);
int GCD(int A, int B);

#endif
```

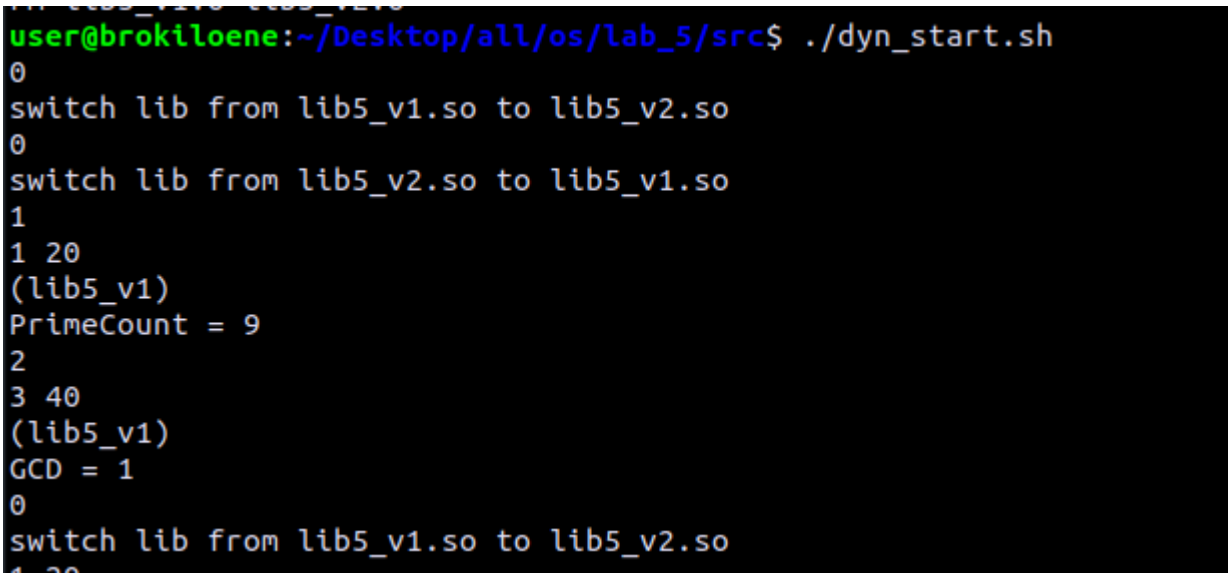
lib5_v1.c | lib5_v2.c | main_static.c | Makefile | lib5.h

```c
// PrimeCount - Решето Эратосфена
// GCD - Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.
#include <stdio.h> // printf
#include <stdlib.h> // calloc & free & exit & EXIT_FAILURE

#include "lib5.h"

void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

int PrimeCount(int a, int b)
{
    printf("(lib5_v2)\n");
    int *erat = (int *) calloc(b + 1, sizeof(int));
    if (erat == NULL) {
        printf("calloc error\n");
        exit(EXIT_FAILURE);
    }
    for (int i = 2; i * i <= b; ++i) {
        for (int j = i * i; j <= b; j += i) {
            erat[j] = 1;
        }
    }
    int res = 0;
    for (int i = a; i <= b; ++i) {
        if (erat[i] == 0) res += 1;
    }
    free(erat);
    return res;
}

int GCD(int a, int b)
{
    printf("(lib5_v2)\n");
    if (a > b) swap(&a, &b);
    for (int i = a; i > 0; --i) {
        if (a % i == 0 && b % i == 0) {
            return i;
        }
    }
    return 1;
}
```

dyn_start.sh | lib5.h

```
#!/bin/sh
LD_LIBRARY_PATH=$(pwd):$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
./main_dynamic
```

Демонстрация работы программы



Выводы

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