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: Гаптулхаков Руслан Рамилевич

Группа: М80 – 208Б-19

Вариант: 20

Преподаватель: Миронов Евгений Сергеевич

Дата: 20.04.2021

Оценка:

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

1. **Постановка задачи**

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

* во время компиляции (на этапе линковки)
* во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В лабораторной работе необходимо получить следующие части:

* Динамические библиотеки, реализующие контракты
* 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 использования двух типов библиотек.

Пользовательский ввод должен быть организован следующим образом:

* Команда «0»: переключить одну реализацию контрактов на другую
* Команда «1 args»: вызов первой функции контрактов
* Команда «2 args»: вызов второй функции контрактов

Контракты:

* Поиск колличества простых чисел на отрезке – наивный алгоритм и решето Эратосфена.
* Перевод из десятичной СС в иную – двоичная и троичная.

1. **Общие сведения о программе**

Программа написана на языке Си. Операционная система – Ubuntu.

Контракты описаны в заголовочном файле functions.h. Реализации контрактов описаны в файлах functions\_01.c и functions\_02.c.

Сборка программы производится с использованием makefile. Для подключения библиотеки на этапе компиляции выполняются следующие действия:

1. Получение объектного файла основной программы.
2. Компиляция файла библиотеки с ключом -shared. Получаем .so файл.
3. Линковка библиотеки и объектного файла основной программы.
4. Указание пути к библиотеке в переменной LD\_LIBRARY\_PATH.

export LD\_LIBRARY\_PATH=/home/rusya/lab5\_OS:$LD\_LIBRARY\_PATH

Для динамической загрузки библиотек используются средства библиотеки dlfcn.h.

1. **Общий метод и алгоритм решения**

Программа принимает на вход неограниченное количество команд следующего вида:

* 0: переключение библиотеки. Выполняется при помощи функций dlopen, dlclose, dlsym, dlerror.
* 1 A B: Для любого x из [A,B] ищем делмтели числа x из отрезка [2,x] и алгоритм Эратосфена.
* 2 x: перевод путём деления на основание.

При вводе неверной команды программа выводит соответствующее сообщения. Для завершения программы можно использовать комбинацию Ctrl + D.

1. **Основные файлы программы**

***functions.h***

#ifndef \_\_FUNCTIONS\_\_H

#define \_\_FUNCTIONS\_\_H

int PrimeCount(int A, int B);

char\* Translation(long x);

#endif

***functions\_01.c***

#include <math.h>

#include <stdlib.h>

#include "functions.h"

int PrimeCount(int A, int B){

    int n = B + 1;

    int flag, count = 0;

    for(int i = A; i < n; ++i){

        flag = 1;

        for(int j = i - 1; j >= 2; --j){

            if(i % j == 0){

                flag = 0;

            }

        }

    count+=flag;

    }

    return count;

}

char\* Translation(long x){

    char\* result = NULL;

    int size = floor(log2(x)) + 2;

    result = malloc(sizeof(char) \* size);

    result[size - 1] = '\0';

    while(x > 0){

        --size;

        result[size - 1] = (x % 2) + '0';

        x /= 2;

    }

    return result;

}

***functions\_02.c***

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include "functions.h"

int PrimeCount(int A, int B){

    int n = B + 1;

    int prime[n];

    for(int i = 0; i < n; ++i){

        prime[i] = 1;

    }

    for (int i = 2; i\*i <= B; ++i)

        if (prime[i])

            for(int j=i\*i; j <= B; j+=i)

                prime[j] = 0;

    int count = 0;

    for(int i = A; i < B; ++i){

        count += prime[i]? 1:0;

    }

    return count;

}

double my\_log(double num, double base){

    return log2(num)/log2(base);

}

char\* Translation(long x){

    char \*result = NULL;

    int size = floor(my\_log(x,3)) + 2;

    result = malloc(sizeof(char) \* size);

    result[size - 1] = '\0';

    while(x > 0){

        --size;

        result[size - 1] = (x % 3) + '0';

        x /= 3;

    }

    return result;

}

***Main\_static.c***

#include <stdio.h>

#include "functions.h"

int main(){

    int sw = -1;

    while(scanf("%d",&sw) > 0){

        if(sw == 1){

            int A, B;

            scanf("%d",&A);

            scanf("%d",&B);

            printf("%d\n",PrimeCount(A,B));

        } else if(sw == 2) {

            long x = 0;

            scanf("%ld",&x);

            printf("%s\n",Translation(x));

        }

    }

    return 0;

}

***main\_dynamic.c***

#include <dlfcn.h>

#include <stdio.h>

int main(){

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

    char\* (\*Translation)(long) = NULL;

    int sw = -1;

    char\* libs[] = {"lib1.so","lib2.so"};

    int lib = 0;

    void\* handle = NULL;

    handle = dlopen(libs[lib],RTLD\_LAZY);

    if (!handle) {

        printf("%s\n", dlerror());

        return 1;

    }

    printf("%p \n", handle);

    PrimeCount = dlsym(handle,"PrimeCount");

    Translation = dlsym(handle,"Translation");

    printf("%p %p \n", PrimeCount, Translation);

    while(scanf("%d",&sw) > 0){

        if(sw == 1){

            int A, B;

            scanf("%d %d",&A, &B);

            printf("%d\n",(PrimeCount)(A,B));

        } else if(sw == 2) {

            long x = 0;

            scanf("%ld",&x);

            printf("%s\n",Translation(x));

        } else if(sw == 0) {

            dlclose(handle);

            lib = (lib + 1) % 2;

            handle = dlopen(libs[lib],RTLD\_LAZY);

            PrimeCount = dlsym(handle,"PrimeCount");

            Translation = dlsym(handle,"Translation");

        }

    }

    dlclose(handle);

}

***makefile***

CC = gcc

CFLAGS = -g3 -c -Wall

all: prog1 prog2

prog1: lib1.so

    $(CC) main\_static.c -L. -l1  -Wall -o main\_static

prog2: lib1.so lib2.so

    $(CC) main\_dynamic.c  -o  main\_dynamic -ldl

lib1.so: lib1.o

    $(CC) lib1.o  -shared -o lib1.so -lm

lib2.so: lib2.o

    $(CC) lib2.o  -shared -o lib2.so -lm

lib1.o:

    $(CC) $(CFLAGS) functions\_01.c -lm -Werror -fpic  -o lib1.o

lib2.o:

    $(CC) $(CFLAGS) functions\_02.c -lm -Werror -fpic -o lib2.o

clean:

    rm -rf \*.so \*.o main\_dynanmic main\_static

1. **Демонстрация работы программы**

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/Lab5\_OS$** make clean

rm -rf \*.so \*.o main\_dynanmic main\_static

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/Lab5\_OS**$ make

gcc -g3 -c -Wall functions\_01.c -lm -Werror -fpic -o lib1.o

gcc lib1.o -shared -o lib1.so -lm

gcc main\_static.c -L. -l1 -Wall -o main\_static

gcc -g3 -c -Wall functions\_02.c -lm -Werror -fpic -o lib2.o

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

gcc main\_dynamic.c -o main\_dynamic -ldl

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/Lab5\_OS$** ./main\_static

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/Lab5\_OS$** export LD\_LIBRARY\_PATH=./:$LD\_LIBRARY\_PATH

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/Lab5\_OS$** ./main\_static

1

1 10

5

1

20 30

2

2

10

1010

2

11

1011

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/Lab5\_OS$** ./main\_dynamic

0x7fffc3ba12c0

0x7f0a9961c159 0x7f0a9961c1c6

1

1

10

5

1

20 30

2

2

10

1010

2

11

1011

0

1

1 10

5

20 30

1

20 30

2

2

10

101

2

11

102

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5\_2020$** ltrace -o ltrace.log ./main\_static  
**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5\_2020$** cat ltrace.log

\_\_isoc99\_scanf(0x7f8eccc00a44, 0x7fffc71b4ec8, 0x7fffc71b4fd8, 0x7f8eccc009c0) = 1  
\_\_isoc99\_scanf(0x7f8eccc00a44, 0x7fffc71b4ecc, 0x7f8ecc3dd8d0, 16) = 1  
\_\_isoc99\_scanf(0x7f8eccc00a44, 0x7fffc71b4ed0, 0x7f8ecc3dd8d0, 16) = 1  
PrimeCount(1, 10, 10, 16) = 5  
printf("%d\n", 5) = 2  
\_\_isoc99\_scanf(0x7f8eccc00a44, 0x7fffc71b4ec8, 0, 0) = 1  
\_\_isoc99\_scanf(0x7f8eccc00a4b, 0x7fffc71b4ed0, 0x7f8ecc3dd8d0, 16) = 1  
Translation(15, 1, 0x7f8ecc3dd8d0, 16) = 0x7fffbe7f3280  
puts("1111") = 5  
\_\_isoc99\_scanf(0x7f8eccc00a44, 0x7fffc71b4ec8, 0x7f8ecc3dd8c0, 0x7fffbe7f1010) = 0xffffffff  
+++ exited (status 0) +++

**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5\_2020$** ltrace -o ltrace.log ./main\_dynamic  
**rusya@DESKTOP-93JGKCU:/mnt/c/Users/rusya/Desktop/lab5\_2020$** cat ltrace.log

dlopen("lib1.so", 1) = 0x7fffd6725290  
printf("%p \n", 0x7fffd6725290) = 16  
dlsym(0x7fffd6725290, "PrimeCount") = 0x7f70af1e06ea  
dlsym(0x7fffd6725290, "Translation") = 0x7f70af1e0753  
printf("%p %p \n", 0x7f70af1e06ea, 0x7f70af1e0753) = 31  
\_\_isoc99\_scanf(0x7f70afe00bc6, 0x7fffdec275d4, 0, 0) = 1  
\_\_isoc99\_scanf(0x7f70afe00bb8, 0x7fffdec275d8, 0x7fffdec275e0, 16) = 2  
printf("%d\n", 5) = 2  
\_\_isoc99\_scanf(0x7f70afe00bc6, 0x7fffdec275d4, 0, 0) = 1  
\_\_isoc99\_scanf(0x7f70afe00bc2, 0x7fffdec275e0, 0x7f70af7dd8d0, 16) = 1  
puts("1111") = 5  
\_\_isoc99\_scanf(0x7f70afe00bc6, 0x7fffdec275d4, 0x7f70af7dd8c0, 0x7fffd6725010) = 1  
dlclose(0x7fffd6725290) = 0  
dlopen("lib2.so", 1) = 0x7fffd6725290  
dlsym(0x7fffd6725290, "PrimeCount") = 0x7f70af1e07aa  
dlsym(0x7fffd6725290, "Translation") = 0x7f70af1e094d  
\_\_isoc99\_scanf(0x7f70afe00bc6, 0x7fffdec275d4, 1, 0) = 1  
\_\_isoc99\_scanf(0x7f70afe00bb8, 0x7fffdec275d8, 0x7fffdec275e0, 16) = 2  
printf("%d\n", 5) = 2  
\_\_isoc99\_scanf(0x7f70afe00bc6, 0x7fffdec275d4, 0, 0) = 1  
\_\_isoc99\_scanf(0x7f70afe00bc2, 0x7fffdec275e0, 0x7f70af7dd8d0, 16) = 1  
puts("120") = 4  
\_\_isoc99\_scanf(0x7f70afe00bc6, 0x7fffdec275d4, 0x7f70af7dd8c0, 0x7fffd6725010) = 0xffffffff  
dlclose(0x7fffd6725290) = 0  
+++ exited (status 0) +++

1. **Выводы**

Данная лабораторная работа была направлена на изучение динамических библиотек в операционной системе Linux. Для демонстрации различий двух библиотек были реализованы две программы. Одна подключала динамическую библиотеку на стадии линковки, а вторая во время исполнения.

Динамические библиотеки содержат функции, которые будут загружены в оперативную память только тогда, когда они действительно понадобятся. Использование динамических библиотек существенно экономит память и ускоряет процесс сборки программы.

Операционная система Linux позволяет совершенствовать библиотеки «на ходу», когда программа уже запущена и использует какие-то библиотеки. Динамические библиотеки легче обновлять: достаточно исправить только код самой библиотеки.

Однако у динамических библиотек есть и свои недостатки. Например, это достаточно запутанная сборка программ. Также стоит отметить, что вызов функции из динамической библиотеки происходит немного медленнее.

Но преимущества использования динамических библиотек перекрывают все их недостатки. Сейчас компьютеры обладают достаточной мощностью для быстрого выполнения системных вызовов. Важнее сэкономить объём памяти, используемой программой. Поэтому динамические библиотеки используются в большинстве современных программ.