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: Ивенкова Л.В

Группа: М80 – 208Б-19

Вариант: 24

Преподаватель: Миронов Е. С.

Дата: \_\_\_\_\_\_\_\_\_\_\_

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:

* Подсчёт наибольшего общего делителя для двух натуральных чисел (основание натурального логарифма) – при помощи Алгоритма Евклида и при помощи наивного алгоритма (попытаться разделить числа на все числа, что меньше A и B).
* Подсчет площади плоской геометрической фигуры по двум сторонам – площадь прямоугольника и прямоугольного треугольника.

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 functions1.c и functions2.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.

Для динамической загрузки библиотек используются средства библиотеки dlfcn.h.

Для указания пути к библиотеке в Makefile используем флаг -Wl,-rpath=./

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. Все системные ошибки обрабатываются.
* 1.1: расчет НОД с помощью Алгоритма Евклида: сравниваем числа, большему числу присваиваем значение разности входных чисел, до тех пор, пока они не станут равными. Во втором случае применяем наивный алгоритм – делим числа на все что меньше меньшего из них и сравниваем остатки.
* 2: Формулу прямоугольника вычисляем как a\*b, прямоугольного треугольника – как a\*b/2.

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

1. **Основные файлы программы**

***functions.h.***

extern float Square(float a, float b);

extern int GCF(int x, int y);

***functions1.c***

#include "functions.h"

float Square(float a, float b){

return a\*b;

}

int GCF(int x, int y){

while (x != y){

if (x>=y)

x = x - y;

else

y = y - x;

}

return x;

}

***functions2.c***

#include "functions.h"

float Square(float a, float b){

return a\*b/2;

}

void Swap(int\* x, int\* y) {

int tmp = \*x;

\*x = \*y;

\*y = tmp;

}

int GCF(int x, int y){

if (x>y)

Swap(&x,&y);

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

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

return i;

}

return 1;

}

***static.c***

# #include "functions.h"

#include <stdio.h>

#include <stdlib.h>

int main(){

int c, x, y;

float a, b;

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

if (c == 1){

scanf("%d %d", &x, &y);

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

}

else if (c == 2){

scanf("%f %f",&a, &b);

printf("Square = %f\n", Square(a, b));

}

}

}

***dynamic.c***

//#include "functions.h"

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

#include <math.h>

#define check(VALUE,BADVALUE,MSG) if (VALUE == BADVALUE){perror(MSG); exit(1); }

int main(){

char \*libnames[] = {"./libfunctions1.so", "./libfunctions2.so"};

int lib = 0;

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

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

void \*handle;

handle = dlopen(libnames[lib], RTLD\_NOW);

check(handle,NULL,dlerror());

GCF = dlsym(handle,"GCF");

check(GCF,NULL,dlerror());

Square = dlsym(handle,"Square");

check(Square,NULL,dlerror());

int f,x,y;

float a, b;

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

if (f == 0){

if (dlclose(handle) != 0)

perror(dlerror());

lib = (lib+1)%2;

handle = dlopen(libnames[lib],RTLD\_NOW);

check(handle,NULL, dlerror());

GCF = dlsym(handle,"GCF");

check(GCF,NULL,dlerror());

Square = dlsym(handle,"Square");

check(Square,NULL,dlerror());

}

else if (f == 1){

scanf("%d %d", &x, &y);

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

}

else if (f == 2){

scanf("%f %f", &a, &b);

printf("Square = %f\n", Square(a, b));

}

}

if (dlclose(handle) != 0)

perror(dlerror());

}

***Makefile***

all: static dynamic libfunctions

static: libfunctions

gcc static.c -o static -L./ -lfunctions1 -Wl,-rpath=./

dynamic: libfunctions

gcc dynamic.c -o dynamic -ldl

libfunctions: functions

gcc -shared -o libfunctions1.so functions1.o

gcc -shared -o libfunctions2.so functions2.o

functions: functions.h functions1.c functions2.c

gcc -fPIC -c functions1.c

gcc -fPIC -c functions2.c

clean:

rm -f \*.so \*.o static dynamic

1. **Демонстрация работы программы**

**parsifal@DESKTOP-3G70RV4:~/OS/Lab5\_1**$ make

gcc -fPIC -c functions1.c

gcc -fPIC -c functions2.c

gcc -shared -o libfunctions1.so functions1.o

gcc -shared -o libfunctions2.so functions2.o

gcc static.c -o static -L./ -lfunctions1 -g -O2 -std=c11 -Wl,-rpath=./

gcc dynamic.c -o dynamic -ldl

parsifal@DESKTOP-3G70RV4:~/OS/Lab5\_1$ ./static

1

35 125

GCF(35 125) = 5

2

3 4

Square = 12.000000

**parsifal@DESKTOP-3G70RV4:~/OS/Lab5\_1**$ ./dynamic

1

45 15

GCF(45,15) = 15

2

3 8

Square = 24.000000

0

2

3 8

Square = 12.000000

1

45 15

GCF(45,15) = 15

**parsifal@DESKTOP-3G70RV4:~/OS/Lab5**$ltrace -o ltrace.log ./static

1 x --> count 'e' number

2 size\_of\_array array --> sort array

1 10

Counting 'e' number using second remarkable limit

2.593742

2 3 2 3 1

Sorting an array usuing bubble sort

1 2 3

**parsifal@DESKTOP-3G70RV4:~/OS/Lab5**$cat ltrace.log

\_\_isoc99\_scanf(0x7f7857000a47, 0x7ffffcf6b894, 0x7ffffcf6b9c8, 0x7f78570009c0) = 1

\_\_isoc99\_scanf(0x7f7857000a44, 0x7ffffcf6b898, 0x7ffffcf6b89c, 16) = 2

GCF(123, 9, 0x7f78569dd8d0, 24) = 3

\_\_printf\_chk(1, 0x7f7857000a4a, 123, 9) = 15

\_\_isoc99\_scanf(0x7f7857000a47, 0x7ffffcf6b894, 0x7f78569dd8c0, 0) = 1

\_\_isoc99\_scanf(0x7f7857000a5b, 0x7ffffcf6b8a0, 0x7ffffcf6b8a4, 16) = 2

Square(0, 2, 0x7f78569dd8d0, 0x7ffffcf6b351) = 2

\_\_printf\_chk(1, 0x7f7857000a61, 0x7f78569dd8d0, 0x7ffffcf6b351) = 19

\_\_isoc99\_scanf(0x7f7857000a47, 0x7ffffcf6b894, 0x7f78569dd8c0, 0) = 0xffffffff

+++ exited (status 0) +++

**parsifal@DESKTOP-3G70RV4:~/OS/Lab5**$ltrace -o ltrace.log ./dynamic

dlopen("./libfunctions1.so", 2) = 0x7fffc3083280

dlsym(0x7fffc3083280, "GCF") = 0x7f42a55e05b4

dlsym(0x7fffc3083280, "Square") = 0x7f42a55e059a

\_\_isoc99\_scanf(0x7f42a6200cdf, 0x7fffcae19150, 1, 0) = 1

\_\_isoc99\_scanf(0x7f42a6200cb5, 0x7fffcae19154, 0x7fffcae19158, 16) = 2

printf("GCF(%d,%d) = %d\n", 123, 9, 3) = 15

\_\_isoc99\_scanf(0x7f42a6200cdf, 0x7fffcae19150, 0, 0) = 1

\_\_isoc99\_scanf(0x7f42a6200ccc, 0x7fffcae1915c, 0x7fffcae19160, 16) = 2

printf("Square = %f\n", 12.000000) = 19

\_\_isoc99\_scanf(0x7f42a6200cdf, 0x7fffcae19150, 0, 0) = 1

dlclose(0x7fffc3083280) = 0

dlopen("./libfunctions2.so", 2) = 0x7fffc3083280

dlsym(0x7fffc3083280, "GCF") = 0x7f42a55e063d

dlsym(0x7fffc3083280, "Square") = 0x7f42a55e05ea

\_\_isoc99\_scanf(0x7f42a6200cdf, 0x7fffcae19150, 1, 0) = 1

\_\_isoc99\_scanf(0x7f42a6200ccc, 0x7fffcae1915c, 0x7fffcae19160, 16) = 2

printf("Square = %f\n", 6.000000) = 18

\_\_isoc99\_scanf(0x7f42a6200cdf, 0x7fffcae19150, 0, 0) = 1

\_\_isoc99\_scanf(0x7f42a6200cb5, 0x7fffcae19154, 0x7fffcae19158, 16) = 2

printf("GCF(%d,%d) = %d\n", 175, 15, 5) = 16

\_\_isoc99\_scanf(0x7f42a6200cdf, 0x7fffcae19150, 0, 0) = 0xffffffff

dlclose(0x7fffc3083280) = 0

+++ exited (status 0) +++

Примечание: красным цветом выделены системные вызовы, связанные с динамическими библиотеками.

1. **Выводы**

Данная лабораторная работа была направлена на изучение динамических библиотек в операционной системе Linux. Чтобы разобраться с применением таких библиотек, мной были написаны две программы: одна подключала динамическую библиотеку на стадии линковки, а вторая – непосредственно 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.