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: Гречников А.В.

Группа: М8О–206Б–20

Вариант: 24

Преподаватель: Соколов Андрей Алексеевич

Дата: 18.12.2021

Оценка: 4

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 №1*), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (*программа №2*), 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 №2*). 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 24.

4. Подсчёт наибольшего общего делителя для двух натуральных чисел Int GCF(int A, int B) – при помощи Алгоритма Евклида и при помощи наивного алгоритма (попытаться разделить числа на все числа, что меньше A и B).

7. Подсчёт площади плоской геометрической фигуры по двум сторонам Float Square(float A, float B) – площадь прямоугольника и прямоугольного треугольника.

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 -Wall

lib2.so: lib2.c

gcc -shared lib2.c -o lib2.so -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/artem" -Wl,-R. -l1 -o prog1 –Wall

**functions.h**

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

float Square(float a, float b);

int GCF(int x, int y);

#endif

**lib1.c**

float Square(float a, float b){

return a \* b;

}

int GCF(int x, int y){

while(x != y){

if(x >= y) x -= y;

else y -= x;

}

return x;

}

**lib2.c**

float Square(float a, float b){

return a \* b / 2;

}

int GCF(int x, int y){

if(x > y){ int tmp = x; x = y; y = tmp; }

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

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

return i;

}

return 1;

}

**prog1.c**

#include <stdio.h>

#include "functions.h"

int main(){

int c, x, y;

float a, b;

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

if(c == 1){

if(scanf("%d %d", &x, &y) == 2)

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

}

else if(c == 2){

if(scanf("%f %f", &a, &b) == 2)

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

}

else

printf("Incorrect comand!\n");

}

}

**prog2.c**

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

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

int main(){

char \*libnames[] = {"./lib1.so", "./lib2.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){

if (scanf("%d %d", &x, &y) == 2)

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

}

else if(f == 2){

if(scanf("%f %f", &a, &b) == 2)

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

}

else

printf("Incorrect comand!\n");

}

if (dlclose(handle) != 0)

perror(dlerror());

}

**Пример работы**

artem@ideapad:~$ ./prog1

1

35 125

GCF(35, 125) = 5

2

3 4

Square = 12.000000

artem@ideapad:~$ ./prog2

1

45 15

GCF(45, 15) = 15

2

3 8

Square = 24.000000

0

1

45 15

GCF(45, 15) = 15

2

3 8

Square = 12.000000

**Вывод**

В ходе выполнения данной лабораторной работы я познакомился с тем, как создавать и использовать динамические библиотеки. Стоит 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, чтобы ею могли пользоваться все нуждающиеся программы.