**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные системы»

**Лабораторная работа № 5**

Тема: Динамические библиотеки

Студент: Тимофеев А.В.

Группа: 80-207

Преподаватель: Миронов Е. С.

Дата:

Оценка:

**Постановка задачи**

Требуется создать динамические библиотеки, которые реализуют определенный 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 место положение 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»;

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.

Варианты:

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.

7 Подсчет площади плоской геометрической фигуры по двум сторонам Float Square(float A, float B) Фигура прямоугольник Фигура прямоугольный треугольник

**Алгоритм решения**

Запишем первую реализацию 2-х функций в файл functions1.c, а вторую реализацию в functions2.c. Общее описание вынесем в functions.h. Для «статического» использования библиотеки необходимо при компиляции указать ее имя и путь, по которому она находится. Также необходимо, чтобы сам исполняемый файл знал где искать эту библиотеку, так как она находится не в стандартном месте. Для динамического использования необходимо знать полные имена библиотек, поэтому поместим их в массив. При необходимости изменить версию функций, загрузим следующую библиотеку. Для выполнения функции необходимо загружать её из библиотеки в соответствующий указатель.

**Листинг программы**

**dynamic.c**

#include "functions.h"

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

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

int main() {

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

int lib = 0;

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

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

void \*handle;

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

check(handle, dlerror(), NULL)

int f;

int a, b;

float x, y;

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

if (f == 0) {

if (dlclose(handle) != 0) {

perror(dlerror());

exit(1);

}

lib = (lib + 1) % 2;

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

check(handle, dlerror(), NULL);

} else if (f == 1) {

if (scanf("%d %d", &a, &b) != 2) {

perror("invalid input");

exit(1);

}

GCD = (int (\*)(int, int))dlsym(handle, "mygcd");

//PrimeCount = (int (\*)(int, int))dlsym(handle, "PrimeCount");

check(GCD, dlerror(), NULL);

printf("GCD: %d\n", (\*GCD)(a, b));

} else if (f == 2) {

if (scanf("%f %f", &x, &y) != 2) {

perror("invalid input");

exit(1);

}

mysquare = (float (\*)(float, float))dlsym(handle, "mysquare");

check(mysquare, dlerror(), NULL);

printf("mysquare = %f\n", (\*mysquare)(x, y));

}

}

if (dlclose(handle) != 0) {

perror(dlerror());

exit(1);

}

}

**functions.h**

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

extern int mygcd(int, int);

extern float mysquare(float, float);

#endif

**functions1.с**

#include "functions.h"

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

int mygcd(int a, int b) {

while (b) {

a %= b;

int tmp = a;

a = b;

b = tmp;

}

return a;

}

float mysquare(float x, float y) {

return x\*y;

}

**functions2.с**

#include "functions.h"

#include <stdlib.h>

#include <stdio.h>

int mygcd(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;

}

float mysquare(float x, float y) {

return (x\*y)/2;

}

**static.c**

#include "functions.h"

#include <stdio.h>

#include <stdlib.h>

int main() {

int f;

int a, b;

float x, y;

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

if (f == 1) {

if (scanf("%d %d", &a, &b) != 2) {

perror("invalid input");

exit(1);

}

printf("GCD: %d\n", mygcd(a, b));

} else if (f == 2) {

if (scanf("%f %f", &x, &y) != 2) {

perror("invalid input");

exit(1);

}

printf("mysquare = %f\n", mysquare(x, y));

}

}

}

**Тесты и протокол исполнения:**

dude@dude:~/Рабочий стол/oc/OC/mylaba5$ ./static

1 100 20

GCD: 20

1 5 35

GCD: 5

2 4 8

mysquare = 32.000000

2 8 5

mysquare = 40.000000

dude@dude:~/Рабочий стол/oc/OC/mylaba5$ ./dynamic

1 100 20

GCD: 20

1 5 35

GCD: 5

2 4 8

mysquare = 32.000000

2 8 5

mysquare = 40.000000

0

1 100 20

GCD: 20

2 4 8

mysquare = 16.000000

2 3 3

mysquare = 4.500000

**Вывод**:

В ходе лабораторной работы я познакомился с созданием динамических библиотек в ОС Linux, а также с возможностью загружать эти библиотеки в ходе выполнения программы. Динамические библиотеки помогают уменьшить размер исполняемых файлов. Загрузка динамических библиотек во время выполнения также упрощает компиляцию. Однако также можно подключить библиотеку к программе на этапе линковки. Она все равно загрузится при выполнении, но теперь программа будет изначально знать что и где искать. Если библиотека находится не в стандартной для динамических библиотек директории, необходимо также сообщить линкеру, чтобы тот передал необходимый путь в исполняемый файл.

**Список литературы**

1. Таненбаум Э., Бос Х. Современные операционные системы. – 4-е изд. – СПб.: Издательский дом «Питер», 2018.
2. Разделяемые библиотеки (shared libraries) (lib share gcc ldd elf) – URL: https://www.opennet.ru/base/dev/shared\_lib\_intro.txt.html