Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

**ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ**

Студент: Комбаров Владислав Александрович

Группа: М8О–212Б–22

Вариант: 25

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

## Цель работы

Целью является приобретение практических навыков в:

* Создании динамических библиотек
* Создании программ, которые используют функции динамических библиотек

## Задание

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 местоположение и контракты. Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

* 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 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 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.

**Функция 1:**

Описание: Подсчёт наибольшего общего делителя для двух натуральных чисел

Сигнатура: Int GCF(int A, int B)

Реализация 1: Алгоритм Евклида

Реализация 2: Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.

**Функция 2:**

Описание: Перевод числа x из десятичной системы счисления в другую

Сигнатура: Char\* translation(long x)

Реализация 1: Другая система счисления двоичная

Реализация 2: Другая система счисления троичная

**Общие сведения о программе**

*functions.h* – Заголовочный файл для функций;

*first\_real.c* – Содержит реализацию 1 данных функций;

*second\_real.c* – Содержит реализацию 2 данных функций;

*main1.c* - Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;

*main2.c* - Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты;

Программа компилируется стандартно в случае с main1.c и путём создания динамических библиотек с помощью принципа позиционно-независимого кода и линковки их с библиотекой доступа к функциям загрузки библиотек в Linux, а также указания директории с лабораторной работой как места для поиска библиотек в случае с main2.c.

**Общий метод и алгоритм решения**.

Для реализации, поставленной задачи необходимо:

1. Реализовать заданные функции двумя способами, в соответствии с заданием;
2. Написать тестовую программу №1, используя данные библиотеки во время компиляции;
3. Написать тестовую программу №2, используя данные библиотеки во время исполнения программы;
4. Скомпилировать и запустить программу, проверить правильность её выполнения.

**Основные файлы программы**

**functions.h**

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

int GCF(int A, int B);

char\* translation(long x);

#endif

**first\_real.c**

#include <stdio.h>

#include <stdlib.h>

#include "functions.h"

int GCF(int A, int B) {

while (B != 0) {

int temp = B;

B = A % B;

A = temp;

}

return A;

}

char\* translation(long x)

{

char\* binary = (char\*)malloc(64 \* sizeof(char)); // выделяем память под строку для двоичного представления

int index = 0; // индекс для заполнения строки

// Если входное число равно нулю, возвращаем строку "0"

if (x == 0) {

binary[0] = '0';

binary[1] = '\0';

return binary;

}

// Проверяем, является ли число отрицательным

int isNegative = 0;

if (x < 0) {

isNegative = 1;

x = (-1)\*x; // берем модуль числа

}

// Переводим число в двоичную систему счисления

while (x > 0) {

binary[index++] = (x % 2) + '0'; // получаем остаток от деления и добавляем его в строку, преобразуя в символ

x = x / 2; // делим на 2 для получения следующей цифры

}

if (isNegative) {

binary[index++] = '-'; // если число было отрицательным, добавляем знак минуса в конец строки

}

binary[index] = '\0'; // добавляем завершающий символ

// Переворачиваем строку, т.к. мы добавляли цифры в обратном порядке

int i, j;

char temp;

for (i = 0, j = index - 1; i < j; i++, j--) {

temp = binary[i];

binary[i] = binary[j];

binary[j] = temp;

}

return binary;

}

**main\_1.c**

#include <stdio.h>

#include <stdlib.h>

#include "first\_real.c"

int main(){

printf("1 arg1 arg2 - GCF function\n");

printf("2 arg - Translation function\n");

printf("3 - Exit\n");

int c;

int x,y;

long a;

printf("Enter the key number: ");

scanf("%d", &c);

while(c != 3){

switch (c){

case 1:

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

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

printf("\nEnter the key number: ");

scanf("%d",&c);

break;

case 2:

scanf("%ld" ,&a);

char\* binary = translation(a);

printf("Translation(%ld) = %s\n",a,binary);

free(binary);

printf("\nEnter the key number: ");

scanf("%d",&c);

break;

case 3:

break;

default:

printf("\nIncorrect Input\n");

exit(-1);

break;

}

}

}

**second\_real.c**

#include <stdio.h>

#include <stdlib.h>

#include "functions.h"

int GCF(int A, int B) {

int min = A < B ? A : B; // Находим минимальное из двух чисел

int gcf = 1; // Переменная для хранения наибольшего общего делителя

for (int i = 1; i <= min; i++) {

if (A % i == 0 && B % i == 0) {

gcf = i; // Если i делит оба числа, обновляем значение наибольшего общего делителя

}

}

return gcf; // Возвращаем наибольший общий делитель

}

char\* translation(long x) {

if (x == 0) {

char\* result = (char\*)malloc(2 \* sizeof(char)); // выделение памяти под "0\0"

result[0] = '0';

result[1] = '\0';

return result;

}

int num = labs(x); // берем модуль числа

int count = 0;

long temp = num;

while (temp > 0) {

temp = temp / 3;

count++;

}

char\* result = (char\*)malloc((count + 1) \* sizeof(char)); // выделение памяти под троичное представление числа

int index = count - 1;

while (num > 0) {

result[index] = (num % 3) + '0';

num = num / 3;

index--;

}

result[count] = '\0';

if (x < 0) {

char\* finalResult = (char\*)malloc((count + 2) \* sizeof(char));

finalResult[0] = '-'; // добавляем знак минуса в начало строки

for (int i = 0; i <= count; i++) {

finalResult[i + 1] = result[i]; // добавляем троичное представление числа после знака

}

free(result); // освобождаем память, выделенную для временного троичного представления

return finalResult;

}

return result;

}

**main2.c**

#include <stdio.h>

#include <dlfcn.h>

#include <string.h>

#include <stdlib.h>

int main(){

printf("0 - Change realization\n");

printf("1 arg1 arg2 - GCF function\n");

printf("2 arg - Translation function\n");

printf("3 - Exit\n");

int flag\_lib = 1; // номер текущей библиотеки

void\* library\_handler; // указатель на начало динамической библиотеки

library\_handler = dlopen("libFirst.so",RTLD\_LAZY);

if (!library\_handler){

// если ошибка, то вывести ее на экран

fprintf(stderr,"dlopen() error: %s\n", dlerror());

exit(1); // в случае ошибки закончить работу программу

};

int c;

int x,y;

long a;

printf("Enter the key number: ");

scanf("%d",&c);

while(c != 3){

switch (c){

case 0:

dlclose(library\_handler); // закрывается текущая библиотека

if(flag\_lib == 1){

library\_handler = dlopen("libSecond.so",RTLD\_LAZY);

flag\_lib = 2;

}

else{

library\_handler = dlopen("libFirst.so",RTLD\_LAZY);

flag\_lib = 1;

}

printf("\nRealization changed\n");

printf("Enter the key number: ");

scanf("%d",&c);

break;

case 1:

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

int (\*gcffunc)(int, int); // указатель на функцию GCF

char name1[] = "GCF";

gcffunc = dlsym(library\_handler,name1);

if (!gcffunc){

fprintf(stderr,"dlopen() error: %s\n", dlerror());

exit(1);

};

if(flag\_lib == 1) printf("\nGCF realization №1\n");

else printf("\nGCF realization №2\n");

printf("GCD(%d, %d) = %d\n",x,y,(\*gcffunc)(x, y));;

printf("\nEnter the key number: ");

scanf("%d",&c);

break;

case 2:

scanf("%ld",&a);

char\* (\*tranfunc)(long); // указатель на функцию translation

char name2[] = "translation";

if(flag\_lib == 1) printf("\nTranslation realization №1\n");

else printf("\nTranslation realization №2\n");

tranfunc = dlsym(library\_handler,name2);

char\* res = (\*tranfunc)(a);

printf("Translation(%ld) = %s\n",a,res);

free(res);

printf("\nEnter the key number: ");

scanf("%d",&c);

break;

case 3:

break;

default:

printf("\nIncorrect Input\n");

exit(-1);

break;

}

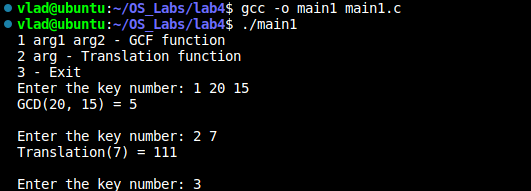
}

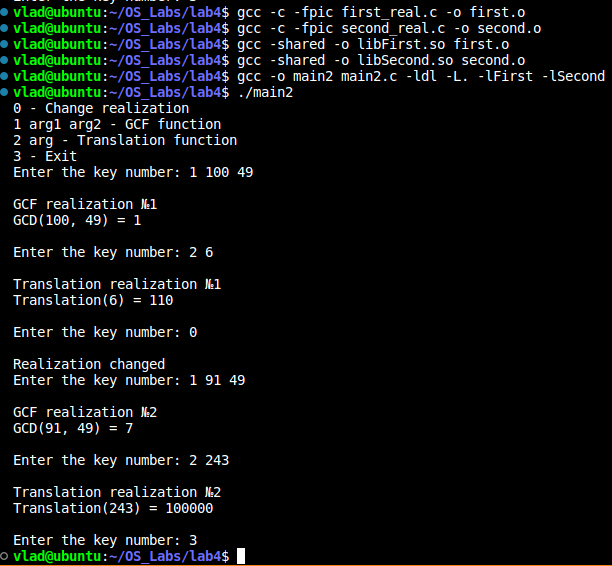
dlclose(library\_handler);

return 0;

}

**Примеры работы**

****

****

**Примечание: запуск программы main2 оказался возможен только после обозначения директории с лабораторной как одним из для поиска динамических библиотек. Это достигается с помощью команды export LD\_LIBRARY\_PATH=/home/vlad/OS Labs/lab4:$LD\_LIBRARY\_PATH ,введённой в командную строку.**

**Вывод**

В результате выполнения лабораторной работы мной были приобретены практические навыки в создании динамических библиотек и программ, которые используют функции этих библиотек разными способами: используя знания, полученные на этапе компиляции, и подключая библиотеки к работе непосредственно во время исполнения программы.

Также в процессе выполнения работы были отточены навыки написания одинаковых функций двумя способами, работы с динамической памятью, работы с типом long, встречающимся в практике гораздо реже, чем int, char и double.

Я узнал о способе компиляции программы с применением динамических библиотек с помощью принципа позиционно-независимого кода, создания динамических библиотек и их линковки с “библиотекой подгружения других библиотек” и намеренного поиска библиотек именно в папке с лабораторной работой.

Работа требовала хорошей концентрации – так, например, при использовании функции перевода числа в систему счисления в строке tranfunc =dlsym(library\_handler, name2), переменная name2 содержит название функции “translation”; когда название здесь писалось с большой буквы, программа выдавала ошибку сегментирования. Примечательным для меня этоо момент стал тем, что выдалась именно ошибка сегментирования, а не сообщение о том, что не удалось найти в библиотеке требуемую функцию.