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: Федоров А.Д.

Группа: М8О–206Б–20

Вариант: 26

Преподаватель: Соколов Андрей Алексеевич

Оценка: 3

Дата: 11.04.2022

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 No1*), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (*программа No2*), 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 No2*). 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 9.

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.

9. Отсортировать целочисленный массив Int \* Sort(int \* array) Пузырьковая сортировка Сортировка Хоара

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:**

down: main\_1 main\_2

main\_1: lib1.o main\_1.c

gcc main\_1.c lib1.o -o main\_1

main\_2: main\_2.o lib1.o lib1.so lib2.o lib2.so

gcc main\_2.o -Wl,-rpath,. -ldl -o main\_2

lib1.so: lib1.o

gcc -shared -o lib1.so lib1.o

main\_2.o: main\_2.c

gcc -c main\_2.c

lib1.o: lib1.c

gcc -fPIC -c lib1.c

lib2.so: lib2.o

gcc -shared -o lib2.so lib2.o

lib2.o: lib2.c

gcc -fPIC -c lib2.c

**functions.h**

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

float Derivative(float a, float deltax);

int PrimeCount(int a, int b);

#endif

**lib1.c**

#include "my\_lib.h"

#include <stdlib.h>

#include <stdio.h>

int gcf(int a, int b){

if (a % b == 0){

return b;

}

return gcf(b, a % b);

}

void swap(int\* a, int\* b) {

int t = \*a;

\*a = \*b;

\*b = t;

}

void sort(int \*array, int size) {

printf("Sorting integers using the first implementation.\n");

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

for (int j = 0; j < size - 1; ++j) {

if (array[j] > array[j + 1]) {

swap(array + j, array + j + 1);

}

}

}

}

**lib2.c**

#include "my\_lib.h"

#include <stdlib.h>

#include <stdio.h>

int gcf(int a, int b){

if (a < b){

int tmp = b;

b = a;

a = tmp;

}

for (int i = b; i > 0; --i){

if ((a % i == 0) && (b % i == 0)){

return i;

}

}

return -1;

}

int compare(const void\* a, const void\* b) {

return (\*(int\*)a - \*(int\*)b);

}

void sort(int\* array, int size) {

printf("Sorting integers using the second implementation.\n");

qsort(array, size, sizeof(int), compare);

}

**main1.c**

#include <stdio.h>

#include <stdlib.h>

#include "my\_lib.h"

void do\_gcf() {

int a, b;

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

fprintf(stderr, "Function gcf error input.\n");

exit(1);

}

printf("Greatest common divisor is %d\n", gcf(a, b));

}

void read\_array(int\* array, size\_t size) {

for (size\_t i = 0; i < size; ++i) {

if (scanf("%d", array + i) != 1) {

fprintf(stderr, "Function sort array input error\n");

exit(2);

}

}

}

void print\_array(int\* array, size\_t size) {

printf("Sorted sequence: ");

for (size\_t i = 0; i < size; ++i) {

printf("%d ", array[i]);

}

printf("\n");

}

void do\_sort() {

int size;

if (scanf("%d", &size) != 1) {

fprintf(stderr, "Function sort size input error\n");

exit(3);

}

int\* array = malloc(size \* sizeof(int));

read\_array(array, size);

sort(array, size);

print\_array(array, size);

free(array);

}

int main() {

int cmd;

printf("gcf - 1 (int)arg1 (int)arg2\n");

printf("sort - 2 (int)size of array (int)arg1 (int)arg2 ... \n");

printf("Enter number the command and args: ");

while (scanf("%d", &cmd) == 1) {

if (cmd == 1) {

do\_gcf();

}

else if (cmd == 2) {

do\_sort();

}

else {

fprintf(stderr, "Undefined command.\n");

}

printf("Enter number the command and args: ");

}

return 0;

}

**main2.c**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <dlfcn.h>

void\* lib1;

void\* lib2;

int (\*gcf)(int, int);

void (\*sort)(int\*, int);

void do\_gcf() {

int a, b;

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

fprintf(stderr, "Function prime\_count error input.\n");

exit(1);

}

printf("The count of prime numbers is %d\n", gcf(a, b));

}

void read\_array(int\* array, size\_t size) {

for (size\_t i = 0; i < size; ++i) {

if (scanf("%d", array + i) != 1) {

fprintf(stderr, "Function sort array input error\n");

exit(2);

}

}

}

void print\_array(int\* array, size\_t size) {

printf("Sorted sequence: ");

for (size\_t i = 0; i < size; ++i) {

printf("%d ", array[i]);

}

printf("\n");

}

void do\_sort() {

int size;

if (scanf("%d", &size) != 1) {

fprintf(stderr, "Function sort size input error\n");

exit(3);

}

int\* array = malloc(size \* sizeof(int));

read\_array(array, size);

sort(array, size);

print\_array(array, size);

free(array);

}

int main() {

lib1 = dlopen("lib1.so", RTLD\_LAZY);

if (!lib1) {

fprintf(stderr, "Cannot open library lib1.so\n");

return -2;

}

lib2 = dlopen("lib2.so", RTLD\_LAZY);

if (!lib2) {

fprintf(stderr, "Cannot open library lib2.so\n");

return -2;

}

gcf = dlsym(lib1, "gcf");

sort = dlsym(lib1, "sort");

int cmd;

bool first\_realise = false;

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

printf("gcf - 1 (int)arg1 (int)arg2\n");

printf("sort - 2 (int)arg1 (int)arg2 ... \n");

printf("Enter number the command and args: ");

while (scanf("%d", &cmd) == 1) {

if (cmd == 0) {

if (first\_realise) {

gcf = dlsym(lib1, "gcf");

sort = dlsym(lib1, "sort");

}

else {

gcf = dlsym(lib2, "gcf");

sort = dlsym(lib2, "sort");

}

first\_realise = !first\_realise;

if (gcf == NULL || sort == NULL) {

fprintf(stderr, "Cannot find functions\n");

exit(2);

}

}

else if (cmd == 1) {

do\_gcf();

}

else if (cmd == 2) {

do\_sort();

}

else {

fprintf(stderr, "Undefined command\n");

dlclose(lib1);

dlclose(lib2);

return -1;

}

printf("Enter number the command and args: ");

}

dlclose(lib1);

dlclose(lib2);

return 0;

}

**Пример работы**

mediocrity@LAPTOP-EEF7UMGS:~$ ./main\_2

Change lib - 0

gcf - 1 (int)arg1 (int)arg2

sort - 2 (int)arg1 (int)arg2 ...

Enter number the command and args: 1 2 5

The count of prime numbers is 1

Enter number the command and args: 2 3 3 2 1

Sorting integers using the first implementation.

Sorted sequence: 1 2 3

Enter number the command and args: 0

mediocrity@LAPTOP-EEF7UMGS:~$ ./main\_1

gcf - 1 (int)arg1 (int)arg2

sort - 2 (int)size of array (int)arg1 (int)arg2 ...

Enter number the command and args: 1 5 6

Greatest common divisor is 1

Enter number the command and args: 2 3 3 2 1

Sorting integers using the first implementation.

Sorted sequence: 1 2 3

**Вывод**

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