Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Тема работы**

**Динамические библиотеки**

Студент: Щеглаков Илья Викторович

Группа: М8О-206Б-20

Вариант: 26

Преподаватель:

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Постановка задачи**

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»;

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.

**Вариант 26:** Подсчёт наибольшего общего делителя для двух натуральных чисел алгоритмом Евклида и наивным алгоритмом (пытаться разделить числа на все меньшие числа), отсортировать целочисленный массив пузырьковой сортировкой и сортировкой Хоара.

**Общие сведения о программе:** Программа состоит из четырёх файлов: re1.cpp и re2.cpp — содержат в себе реализацию функций двумя различными способами, prog1.cpp — использует библиотеки на этапе компиляции, prog2.cpp — загружает библиотеки, используя их местоположение и контракты.

**Общий метод и алгоритм решения**

В библиотеке re1.cpp реализованы алгоритм Евклида и пузырьковая сортировка, в библиотеке re2.cpp реализованы наивный алгоритм нахождения НОД и сортировка Хоара. Создание динамических библиотек осуществляется с помощью команды g++ -fPIC -c re1.cpp -o d1.o && g++ -shared d1.o -o libd1.so для re1.cpp и g++ -fPIC -c re2.cpp -o d2.o && g++ -shared d2.o -o libd2.so для re2.cpp. В prog1.cpp выбор подключаемой библиотеки осуществляется на этапе компиляции программы, g++ prog1.cpp -L. -ld1 -o main1 -Wl,-rpath -Wl,. подключает re1.cpp, g++ prog1.cpp -L. -ld2 -o main2 -Wl,-rpath -Wl,. подключает re2.cpp. В prog2.cpp возможен выбор необходимой библиотеки непосредственно при выполнении программы, скомпилированной при помощи g++ prog2.cpp -ldl -o main .

**Исходный код**

re1.cpp

extern "C" int GCF(int A, int B); //g++ -fPIC -c re1.cpp -o d1.o

extern "C" int \* Sort(int \* array); //g++ -shared d1.o -o libd1.so

int GCF(int A, int B)

{

if (B == 0)

{

return A;

}

else

{

return GCF(B, A % B);

}

}

int\* Sort(int\* array)

{

for (int i = 1; i < array[0] + 1; ++i)

{

for (int j = 1; j < array[0]; ++j)

{

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

{

int a = array[j];

array[j] = array[j + 1];

array[j + 1] = a;

}

}

}

return array;

}

re2.cpp

extern "C" int GCF(int A, int B); //g++ -fPIC -c re2.cpp -o d2.o

extern "C" int \* Sort(int \* array); //g++ -shared d2.o -o libd2.so

using namespace std;

int min(int a, int b)

{

return a < b ? a : b;

}

int GCF(int A, int B)

{

int m = 1;

for (int i = 1; i <= min(A, B); ++i)

{

if ((A % i == 0)&&(B % i == 0)&&(i > m))

{

m = i;

}

}

return m;

}

void \_sort(int\* a, int first, int last)

{

int i = first, j = last;

int tmp, x = a[(first + last) / 2];

do

{

while (a[i] < x)

{

i++;

}

while (a[j] > x)

{

j--;

}

if (i <= j)

{

if (i < j)

{

tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

i++;

j--;

}

} while (i <= j);

if (i < last)

{

\_sort(a, i, last);

}

if (first < j)

{

\_sort(a, first, j);

}

}

int\* Sort(int\* array)

{

\_sort(array, 1, array[0]);

return array;

}

prog1.cpp

#include <stdio.h> //g++ prog1.cpp -L. -ld1 -o main1 -Wl,-rpath -Wl,.

#include <vector> //g++ prog1.cpp -L. -ld2 -o main2 -Wl,-rpath -Wl,.

using namespace std;

extern "C" int GCF(int A, int B);

extern "C" int\* Sort(int\* array);

int main()

{

int command;

char c1 = '1', c2 = '1';

printf("1 for GCF, 2 for Sort:\n");

scanf("%d", &command);

while (1)

{

if (command == 1)

{

int a, b;

scanf("%d%c%d%c", &a, &c1, &b, &c2);

if ((a < 1)||(b < 1))

{

printf("Input error\n");

}

else

{

printf("%d\n", GCF(a, b));

}

}

if (command == 2)

{

int a;

char c = '1';

vector<int> v;

while (c != '\n')

{

scanf("%d%c", &a, &c);

v.push\_back(a);

}

c = '1';

int\* arr = new int[v.size() + 1];

arr[0] = v.size();

for (int i = 0; i < v.size(); ++i)

{

arr[i + 1] = v[i];

}

Sort(arr);

for (int i = 1; i < arr[0] + 1; ++i)

{

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

}

printf("\n");

delete [] arr;

}

if (command == 3)

{

break;

}

scanf("%d", &command);

}

return 0;

}

**prog2.cpp**

#include <stdio.h> //g++ prog2.cpp -L. -ldl -o main -Wl,-rpath -Wl,.

#include <stdlib.h>

#include <dlfcn.h>

#include <vector>

using namespace std;

int main()

{

void\* h = NULL;

int (\*GCF)(int A, int B);

int\* (\*Sort)(int\* array);

int lib;

printf("0 for change libs, 1 for re1.cpp, 2 for re2.cpp, 3 for exit:\n");

scanf("%d", &lib);

while ((lib != 1)&&(lib != 2))

{

printf("Input error, try again:\n");

scanf("%d", &lib);

}

if (lib == 1)

{

h = dlopen("libd1.so", RTLD\_LAZY);

}

if (lib == 2)

{

h = dlopen("libd2.so", RTLD\_LAZY);

}

GCF = (int(\*)(int, int))dlsym(h, "GCF");

Sort = (int\*(\*)(int\*))dlsym(h, "Sort");

unsigned command;

printf("1 for GCF, 2 for Sort:\n");

char c1 = '1', c2 = '1';

scanf("%d", &command);

while (1)

{

if (command == 0)

{

if (lib == 1)

{

dlclose(h);

h = dlopen("libd2.so", RTLD\_LAZY);

GCF = (int(\*)(int, int))dlsym(h, "GCF");

Sort = (int\*(\*)(int\*))dlsym(h, "Sort");

lib = 2;

printf("re1 changed on re2\n");

scanf("%d", &command);

continue;

}

else

{

dlclose(h);

h = dlopen("libd1.so", RTLD\_LAZY);

GCF = (int(\*)(int, int))dlsym(h, "GCF");

Sort = (int\*(\*)(int\*))dlsym(h, "Sort");

lib = 1;

printf("re2 changed on re1\n");

scanf("%d", &command);

continue;

}

}

if (command == 1)

{

int a, b;

scanf("%d%c%d%c", &a, &c1, &b, &c2);

if ((a < 1)||(b < 1))

{

printf("Input error\n");

}

else

{

printf("%d\n", GCF(a, b));

}

}

if (command == 2)

{

int a;

char c = '1';

vector<int> v;

while (c != '\n')

{

scanf("%d%c", &a, &c);

v.push\_back(a);

}

c = '1';

int\* arr = new int[v.size() + 1];

arr[0] = v.size();

for (int i = 0; i < v.size(); ++i)

{

arr[i + 1] = v[i];

}

Sort(arr);

for (int i = 1; i < arr[0] + 1; ++i)

{

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

}

printf("\n");

delete [] arr;

}

if (command == 3)

{

break;

}

scanf("%d", &command);

}

dlclose(h);

return 0;

}

**Выводы**

Проделав лабораторную работу, я приобрёл практические навыки, необходимые для работы с динамическими библиотеками.