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: Старцев Иван Романович
Группа: М8О-201Б-21
Вариант: 20
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/IvanTvardovsky/OS-labs

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 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*).
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 20:

Контракт 1:

Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные)

Реализация 1:

Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.

Реализация 2:

Решето Эратосфена

Контракт 2:

Перевод числа x из десятичной системы счисления в другую

Реализация 1:

Другая система счисления двоичная

Реализация 2:

Другая система счисления троичная

Общие сведения о программе

Код состоит из 2 файлов библиотек и 2 файлов программ. Для решения данной лабораторной работы была использована библиотека dlfcn.h, она служит для реализации динамической загрузки.

Основные функции:

1. void *dlopen(const char *filename, int flag); - загружает динамическую библиотеку, имя которой указано в строке filename, и возвращает прямой указатель на начало динамической библиотеки. Если filename не является полным именем файла
2. void *dlsym(void *handle, char *symbol); - функция, возвращения адреса, по которому символ расположен в памяти
3. const char *dlerror(void); - функция описания ошибок в случае неудачи
4. int dlclose(void *handle); - выгружает динамическую библиотеку

Общий метод и алгоритм решения

В динамических библиотеках описаны функция, которые подсчитывают количество простых чисел в промежутке и перевод числа в другие системы счисления. С помощью программ main1.c и main2.c можно осуществить работу с этими функциями. Разница в работе двух программ заключается в том, что main1.c загружает библиотеку, используя знания полученные на этапе компиляции; mian2.c загружает библиотеки, используя только их местоположение и контракты;

Исходный код

main1.c

```
#include <stdlib.h>
#include <stdio.h>
#include "./include/library1.h"

int main(int argc, char* argv[]) {
int x, a, b;
long c;
for (;;) {
scanf("%d", &x);
if (x == 1) {
scanf("%d %d", &a, &b);
printf("Result: ");
int n = PrimeCount(a, b);
printf("%d\n", n);
} else if (x == 2) {
scanf("%ld", &c);
printf("Result: ");
char* res = Translation(c);
printf("%s\n", res);
} else {
return 0;
}
}
return 0;
}
```

main2.c

```
#include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
#include <math.h>
#include <stdbool.h>

const char LIBRARY1[] = "./library1.so";
const char LIBRARY2[] = "./library2.so";

int main(int argc, char* argv[]) {
void *library;
bool type = false;
int x, a, b;
long c;

library = dlopen(LIBRARY2, RTLD_LAZY);
if (!library) {
printf("Error dlopen(): %s\n", dlerror());
return 1;
}

int(*PrimeCount)(int x, int y);
char*(*Translation)(long x);
*(void**)(&PrimeCount) = dlsym(library, "PrimeCount");
```

```
*(void**)(&Translation) = dlsym(library, "Translation");

for (;;) {
scanf("%d", &x);
if (x == 0) {
dlclose(library);
if (type) {
library = dlopen(LIBRARY2, RTLD_LAZY);
type = false;
} else {
library = dlopen(LIBRARY1, RTLD_LAZY);
type = true;
}
if (!library) {
printf("Error dlopen(): %s\n", dlerror());
return 1;
}
*(void**)(&PrimeCount) = dlsym(library, "PrimeCount");
*(void**)(&Translation) = dlsym(library, "Translation");
} else if (x == 1 ) {
scanf("%d %d", &a, &b);
printf("Result: ");
int n = PrimeCount(a, b);
printf("%d\n", n);
} else if (x == 2) {
scanf("%ld", &c);
printf("Result: ");
char* res = Translation(c);
printf("%s\n", res);
} else {
dlclose(library);
return 0;
}
}
return 0;
}
```

functions1.c

```
#include "../include/functions.h"
#include "../include/library1.h"

int PrimeCount(int a, int b) {
int counter = 0;
for (int i = a; i <= b; ++i) {
short flag = 1;
for (int j = 2; j < i; ++j) {
if (i % j == 0) flag = 0;
}
if (flag == 1) ++counter;
}
return counter;
}

char* Translation(long x) {
char* res = (char*) malloc(64 * sizeof(char));
```

```
do {
*--res = x % 2 + '0';
x /= 2;
}
while (x != 0);
return res;
}
```

functions2.c

```
#include "../include/functions.h"

int PrimeCount(int a, int b) {
int counter = 0;
int mas[b + 1];
for (int i = 0; i < b + 1; ++i){
mas[i] = i;
}
for (int i = 2; i*i <= b; ++i){
if (mas[i]){
for (int j = i*i; j <= b; j += i){
mas[j] = 0;
}
}
}
for (int i = 0; i <= b; ++i){
if (mas[i] && i >= a && i != 1){
++counter;
}
}
return counter;
}

char* Translation(long x) {
char* res = (char*) malloc(64 * sizeof(char));
do {
*--res = x % 3 + '0';
x /= 3;
}
while (x != 0);
return res;
}
```

library1.h

```
#ifndef OS_LABS_LIBRARY1_H
#define OS_LABS_LIBRARY1_H

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int PrimeCount(int a, int b);
char* Translation(long x);
```

```
#endif //OS_LABS_LIBRARY1_H
```

Makefile

```
GCC=gcc

all: main1 main2

library1.o:
$(GCC) -fPIC -c ./src/functions1.c -lm -o library1.o

library2.o:
$(GCC) -fPIC -c ./src/functions2.c -lm -o library2.o

library1.so: library1.o
$(GCC) -shared -o library1.so library1.o -lm

library2.so: library2.o
$(GCC) -shared -o library2.so library2.o -lm

main1: library1.so
$(GCC) -o static main1.c -lm ./src/functions1.c

main2: main2.c library1.so library2.so
$(GCC) -o dynamic main2.c -lm -ldl

clean:
rm -rf *.o
rm -rf *.so
rm -rf static
rm -rf dynamic
```

Демонстрация работы программы

```
tvard@tvard-HVY-WXX9:~/os/OS-labs/lab5$ make
gcc -fPIC -c ./src/functions1.c -lm -o library1.o
gcc -shared -o library1.so library1.o -lm
gcc -o static main1.c -lm ./src/functions1.c
gcc -fPIC -c ./src/functions2.c -lm -o library2.o
gcc -shared -o library2.so library2.o -lm
gcc -o dynamic main2.c -lm -ldl
tvard@tvard-HVY-WXX9:~/os/OS-labs/lab5$ ./static
1 22 42
Result: 5
2 10
Result: 1010
^Z
[3]+  Stopped                 ./static
tvard@tvard-HVY-WXX9:~/os/OS-labs/lab5$ ./dynamic
1 22 42
Result: 5
2 10
Result: 101
0
1 22 42
```

```
Result: 5
2 10
Result: 1010
-1
tvard@tvard-HVY-WXX9:~/os/OS-labs/lab5$
```

Выводы

В ходе лабораторной работы я познакомился с созданием динамических библиотек в ОС Linux, а также с непосредственной работе с ними, то есть познакомился с возможностью загружать эти библиотеки в ходе выполнения программы.