**Московский авиационный институт (национальный исследовательский университет)**

Институт №8 «Информационные технологии и прикладная математика» Кафедра 806 «Вычислительная математика и программирование» Дисциплина «Операционные системы»

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Динамические библиотеки.**

## Студент: Кондратьев Егор Алексеевич

## Группа: М80-206Б-19

Преподаватель: Миронов Е.С.

Дата:

## Оценка:

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 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. Подсчёт наибольшего общего делителя для двух натуральных чисел

Реализация 1 Алгоритм Евклида

Реализация 2 Наивный алгоритм. Пытаться разделить числа на все числа,

что меньше A и B

2. Отсортировать целочисленный массив

Реализация 1 Пузырьковая сортировк

Реализация 2 Сортировка Хоара

**Общие сведения о программе**

Программа компилируется при помощи Makefile в 2 исполняемых файла main1, main2 и 2 библиотеки liblab1.so, liblab2.so. В первом случае мы используем библиотеку, которая использует знания полученные во время компиляции (на этапе линковки). Во втором случае программа загружает библиотеки и взаимодействует с ними при помощи следующих системных вызовов:

1. **dlopen** – загружает динамическую библиотеку, имя которой указано первым аргументом, и возвращает прямой указатель на начало динамической библиотеки. Второй аргумент отвечает за разрешение неопределенных символов, возвращает 0 при успешном завершении и значение != 0 в случае ошибки.
2. **exit** – завершение работы программы с кодом, указанным в качестве аргумента.
3. **dlsym** – использует указатель на динамическую библиотеку – первый аргумент, возвращаемую dlopen, и оканчивающееся нулем символьное имя – второй аргумент, а затем возвращает адрес, указывающий, откуда загружается этот символ. Если символ не найден, то возвращаемым значением dlsym является NULL.
4. **dlclose –** уменьшает на единицу счетчик ссылок на указатель динамической библиотеки, передаваемый в качестве аргумента. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается.

**Общий метод и алгоритм решения**.

Создаем по два исполняемых файла и header для них с реализациями и объявлениями для каждой из двух функций, собираем из них библиотеку и используем 2-мя способами:

* + - 1. на этапе компиляции (стадия линковки) при помощи #include в программе main1.
      2. при помощи загрузки библиотек при помощи dlopen в программе main2.

**Основные файлы программы**

**Makefile:**

CC = gcc

CFLAGS = -std=c99

LDFLAGS = -L.

CLIBFLAGS = -c -fPIC

LDLIBFLAGS = -shared

EXECUTABLE1 = task1

EXECUTABLE2 = task2

SRC1 = main1.c

SRC2 = main2.c

OBJ1 = $(SRC1:.c=.o)

OBJ2 = $(SRC2:.c=.o)

SRCLIB1 = ./libs/lib1.c

ND\_SRCLIB1 = $(notdir $(SRCLIB1))

OBJLIB1 = $(ND\_SRCLIB1:.c=.o)

SOLIB1 = $(ND\_SRCLIB1:.c=.so)

SRCLIB2 = ./libs/lib2.c

ND\_SRCLIB2 = $(notdir $(SRCLIB2))

OBJLIB2 = $(ND\_SRCLIB2:.c=.o)

SOLIB2 = $(ND\_SRCLIB2:.c=.so)

all: $(SOLIB1) $(SOLIB2) $(EXECUTABLE1) $(EXECUTABLE2)

$(EXECUTABLE1): $(OBJ1)

    $(CC) $(LDFLAGS) $< $(SOLIB1) -o $@ -lm

$(EXECUTABLE2): $(OBJ2)

    $(CC)  $< -o $@ -ldl

$(SOLIB1): $(OBJLIB1)

    $(CC) $(LDLIBFLAGS) $< -o $@ -lm

$(OBJLIB1): $(SRCLIB1)

    $(CC) $(CLIBFLAGS) $<

$(SOLIB2): $(OBJLIB2)

    $(CC) $(LDLIBFLAGS) $< -o $@ -lm

$(OBJLIB2): $(SRCLIB2)

    $(CC) $(CLIBFLAGS) $<

clean:

    rm \*.o $(SOLIB1) $(SOLIB2) $(EXECUTABLE1) $(EXECUTABLE2)

**main1.c:**

#include "./libs/lib.h"

#include <stdio.h>

int main() {

    int cmd = 0;

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

        if(cmd == 1){

            float a, deltaX;

            scanf("%f%f", &a, &deltaX);

            printf("Calculating derivative!\n");

            float res = Derivative(a, deltaX);

            printf("Derivative is %f\n", res);

        }

        else if (cmd == 2){

            int x;

            scanf("%d", &x);

            printf("Calculating E!\n");

            float e = E(x);

            printf("E is %f\n", e);

        }

        else{

            printf("Wrong command!\n");

            return -1;

        }

    }

}

**main2.c:**

#include <stdio.h>

#include <stdlib.h>

#include <dlfcn.h>

typedef enum {

FIRST,

SECOND,

} CONTEXT;

CONTEXT r = FIRST;

const char\* libName1 = "lib1.so";

const char\* libName2 = "lib2.so";

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

float (\*E)(int x) = NULL;

char \*err;

void \*libHandle = NULL;

void loadLibs(CONTEXT context){

const char \*name;

if(context == FIRST){

name = libName1;

} else{

name = libName2;

}

libHandle = dlopen(name, RTLD\_LAZY);

if(!libHandle){

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

exit(EXIT\_FAILURE);

}

}

void unloadLibs(){

dlclose(libHandle);

}

void loadContext(){

loadLibs(r);

Derivative = (float (\*)(float, float))dlsym(libHandle, "Derivative");

E = (float (\*)(int))dlsym(libHandle, "E");

if((err = dlerror())) {

fprintf(stderr, "%s\n", err);

exit(EXIT\_FAILURE);

}

}

void changeContext(){

unloadLibs();

if(r == FIRST){

r = SECOND;

} else {

r = FIRST;

}

loadContext();

}

int main(){

r = FIRST;

loadContext();

int cmd = 0;

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

if(cmd == 0){

changeContext();

printf("Ok. Contract was changed\n");

if(r == FIRST){

printf("Now context is first\n");

} else{

printf("Now context is second\n");

}

continue;

}

else if(cmd == 1){

float a, deltaX;

scanf("%f%f", &a, &deltaX);

printf("Calculating derivative!\n");

float res = Derivative(a, deltaX);

printf("Derivative is %f\n", res);

}

else if (cmd == 2){

int x;

scanf("%d", &x);

printf("Calculating E!\n");

float e = E(x);

printf("E is %f\n", e);

}

else{

printf("Wrong command!\n");

return -1;

}

}

unloadLibs();

}

**lib1.c:**

#include <math.h>

#include "lib.h"

float Derivative(float A, float deltaX){

float ans = (cosf(A+deltaX) - cosf(A))/deltaX;

return ans;

}

float E(int x){

float res = powf((1+1/(float)x), (float)x);

return res;

}

**lib2.c:**

#include <math.h>

#include "lib.h"

float Derivative(float A, float deltaX){

float ans = (cosf(A+deltaX) - cosf(A-deltaX))/(2\*deltaX);

return ans;

}

int factorial(int x) {

if (x == 0) {

return 1;

}

else if (x == 1) {

return 1;

}

return x \* factorial(x - 1);

}

float E(int x){

float sum = 0;

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

sum += 1. / factorial(i);

}

return sum;

}

**lib.h:**

#ifndef INC\_5\_LAB\_H

#define INC\_5\_LAB\_H

extern float Derivative(float A, float deltaX);

extern float E(int x);

#endif

**Пример работы**

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_5/src$ cat ../tests/test1

1 42 0.3

1 10 0.4

2 5

2 15

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_5/src$ ./task1 < ../tests/test1

Calculating derivative!

Derivative is 0.962382

Calculating derivative!

Derivative is 0.695217

Calculating E!

E is 2.488321

Calculating E!

E is 2.632881

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_5/src$ cat ../tests/test2

1 42 0.3

1 10 0.4

2 5

2 15

0

1 42 0.3

1 10 0.4

2 5

2 1

root@Du:/mnt/c/Users/egork/Desktop/os/lab\_5/src$ ./task2 < ../tests/test2

Calculating derivative!

Derivative is 0.962382

Calculating derivative!

Derivative is 0.695217

Calculating E!

E is 2.488321

Calculating E!

E is 2.632881

Ok. Contract was changed

Now context is second

Calculating derivative!

Derivative is 0.902833

Calculating derivative!

Derivative is 0.529629

Calculating E!

E is 2.708333

Calculating E!

E is 2.718282

**Вывод**

В ходе выполнения данной лабораторной работы я познакомился с тем, как создавать и использовать динамические библиотеки.

Существуют два способа использования динамических библиотек: динамическая компоновка и динамическая загрузка. Динамическая компоновка происходит в момент компиляции, загрузку ресурсов выполняет операционная система и ее динамических компоновщик. Динамическая загрузка происходит в *runtime* и предоставляет более широкие возможности: можно выбрать какие библиотеки подгружать, какие функции из них взять.