Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

Студент: Уваров А.П.

Группа: М8О–206Б–19

Вариант: 13

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**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. 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»;

Я не реализовал такую функцию.

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 13:

Функция 1: Вычисление функций:

float res1 = (cos(A + deltaX) - cos(A)) / deltaX

float res2 = (cos(A + deltaX) - cos(A - deltaX)) / (2 \* deltaX)

Функция 2: Подсчет площади плоской геометрической фигуры по двум сторонам. float Square(float A, float B). Фигура прямоугольник. Фигура прямоугольный треугольник.

**Общие сведения о программе**

Программа компилируется при помощи Makefile в 2 исполняемых файла dynLib.c, statLib.c и 2 библиотеки libstat.a, libdynn.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. **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.
3. **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 два heder-a 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 mainStat.c.
      2. при помощи загрузки библиотек при помощи dlopen в программе mainDyn.c.

**Основные файлы программы**

**makefile:**

**CC = gcc**

**all: run**

**run: libstat.a libdynn.so mainStat.o mainDyn.o**

**$(CC) -o stat-prog mainStat.o -L. -lstat -lm**

**$(CC) -o dyn-prog mainDyn.o -L. -ldynn -ldl**

**libstat.a: statLib.o**

**ar rc libstat.a statLib.o**

**statLib.o: statLib.c**

**$(CC) -c statLib.c**

**mainStat.o: mainStat.c**

**$(CC) -c mainStat.c**

**libdynn.so: dynLib.o**

**$(CC) -shared -o libdynn.so dynLib.o**

**dynLib.o: dynLib.c**

**$(CC) -c -fPIC dynLib.c -lm**

**mainDyn.o: mainDyn.c**

**$(CC) -c mainDyn.c -lm**

**clean:**

**rm \*.o \*.a libdynn.so stat-prog dyn-prog**

**mainStat.c:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <math.h>**

**#include "stLib.h"**

**void WriteCommand(){**

**printf("Write command: \n");**

**printf(" 1 <A> <deltaX> - first function with two arguments\n");**

**printf(" 2 <A> <deltaX> - second function with two arguments\n");**

**printf(" <e> - close programm\n");**

**}**

**int main(){**

**printf("[Stat] Program started!\n");**

**WriteCommand();**

**char \*x = malloc(sizeof(char));**

**float A, deltaX;**

**while(scanf("%s",x)){**

**if(!strcmp(x, "1")){**

**printf("Enter A and deltaX: ");**

**scanf("%f",&A); scanf("%f",&deltaX);**

**printf("[1] Result: %f\n",function1(A, deltaX));**

**}**

**else if(!strcmp(x, "2")){**

**printf("Enter A and deltaX: ");**

**scanf("%f",&A); scanf("%f",&deltaX);**

**printf("[2] Result: %f\n",function2(A, deltaX));**

**}**

**else if(!strcmp(x, "e")){**

**break;**

**}**

**else{**

**WriteCommand();**

**}**

**}**

**return 0;**

**}**

**mainDyn.c:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <dlfcn.h>**

**#include <math.h>**

**#include "dnLib.h"**

**void WriteCommand(){**

**printf("Write command: \n");**

**printf(" 1 <A> <B> - first function with two arguments\n");**

**printf(" 2 <A> <B> - second function with two arguments\n");**

**printf(" <exit, e> - close programm\n");**

**}**

**int main(){**

**void\* handle = dlopen("libdynn.so", RTLD\_LAZY);**

**if(!handle){**

**printf("%s\n",dlerror());**

**return 1;**

**}**

**float (\*Square1) (float A, float B) = dlsym(handle, "Square1");**

**float (\*Square2) (float A, float B) = dlsym(handle, "Square2");**

**printf("[Stat] Program started!\n");**

**WriteCommand();**

**char \*x = malloc(sizeof(char));**

**float A,B;**

**while(scanf("%s",x)){**

**if(!strcmp(x, "1")){**

**printf("Enter A and B: ");**

**scanf("%f",&A); scanf("%f",&B);**

**printf("[1] Result: %f\n", (\*Square1)(A, B));**

**}**

**else if(!strcmp(x, "2")){**

**printf("Enter A and B: ");**

**scanf("%f",&A); scanf("%f",&B);**

**printf("[2] Result: %f\n", (\*Square2)(A, B));**

**}**

**else if(!strcmp(x, "e")){**

**break;**

**}**

**else{**

**printf("%s is unknown command!\n", x);**

**WriteCommand();**

**}**

**}**

**dlclose(handle);**

**return 0;**

**}**

**dynLib.c:**

**#include "dnLib.h"**

**float Square1(float A, float B){**

**return A\*B;**

**}**

**float Square2(float A, float B){**

**return A\*B/2;**

**}**

**statLib.c:**

**#include "stLib.h"**

**extern float function1(double A, double deltaX){**

**return (cos(A + deltaX) - cos(A)) / deltaX;**

**}**

**extern float function2(double A, double deltaX){**

**return (cos(A + deltaX) - cos(A - deltaX)) / (2 \* deltaX);**

**}**

**Пример работы**

**andrey@andrey-VivoBook:~ /Рабочий стол/OC/OChelp/5$ ./dyn-prog**

**[Stat] Program started!**

**Write command:**

**1 < A > < B > - first function with two arguments**

**2 < A > < B > - second function with two arguments**

**<exit, e> - close programm**

**1 2 2**

**Enter A and B: [1] Result: 4.000000**

**2 2 2**

**Enter A and B: [2] Result: 2.000000**

**E**

**andrey@andrey-VivoBook:~ /Рабочий стол/OC/OChelp/5$ ./stat-prog**

**[Stat] Program started!**

**Write command:**

**1 < A > - first function with two arguments**

**2 < A > - second function with two arguments**

**- close programm**

**1 0.1 0.1**

**Enter A and deltaX: [1] Result: -0.149376**

**2 0.1 0.1**

**Enter A and deltaX: [2] Result: -0.099667**

**e**

**Вывод**

В СИ можно использовать динамические библиотеки, в основном применяются, когда программист не хочет делиться со своим исходным кодом. Однако, каждая библиотека имеет свои недостатки. В этой лабораторной работе я научился их создавать и использовать на практике.