Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу
«Операционные системы»

Студент: Абдыкалыков Нурсултан Абдыкалыкович
Группа: М8О-206Б-23
Вариант: 1
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

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 заданный вариантом 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 информацию полученные на этапе компиляции;
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути 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.

Контракты и реализации функций:

1. Рассчет интеграла функции sin(x) на отрезке [A, B] с шагом e
2. Рассчет производной функции cos(x) в точке A с приращением deltaX

Общий метод и алгоритм решения

1. Реализация математических функций:

   Созданы две версии математические функции с разной реализацией в каждой библиотеке

   - SinIntegral(float A, float B, float e)
   - Derivative(float A, float deltaX)

2. Модульная архитектура

   Функции реализованы в виде двух динамических библиотек (libmath1.so и libmath2.so), что позволяет:

   - Легко переключаться между разными методами вычислений
   - Изолировать реализации друг от друга
   - Загружать нужную версию во время выполнения

3. Два способа использования библиотек:
   - Статическая линковка: функции жестко линкуются на этапе компиляции (программа **static_program**)
   - Динамическая загрузка: библиотеки загружаются во время выполнения с возможностью переключения (программа **dynamic_program**)

4. Обработка ошибок.

Исходный код

Mathlib1.c:

```
#include <math.h>

float SinIntegral(float A, float B, float e) {
    float integral = 0.0;
    for(float x = A; x < B; x += e) {
        integral += sin(x) * e;
    }
    return integral;
}

float Derivative(float A, float deltaX) {
    return (cos(A + deltaX) - cos(A)) / deltaX;
}
```

Mathlib2.c:

```
#include <math.h>

float SinIntegral(float A, float B, float e) {
    float integral = 0.0;
    for(float x = A; x < B; x += e) {
        integral += (sin(x) + sin(x + e)) * e / 2;
    }
    return integral;
}

float Derivative(float A, float deltaX) {
    return (cos(A + deltaX) - cos(A - deltaX)) / (2 * deltaX);
}
```

Main_dynamic.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include <string.h>
#include <ctype.h>
#include <stdbool.h>

typedef float (*func_float3)(float, float, float);
typedef float (*func_float2)(float, float);

bool is_number(const char* str) {
    if (*str == '-' || *str == '+') str++;
    bool has_dot = false;
    while (*str) {
        if (*str == '.') {
            if (has_dot) return false;
            has_dot = true;
```

```
        }
        else if (!isdigit(*str)) {
            return false;
        }
        str++;
    }
    return true;
}

bool validate_integral_params(float A, float B, float e) {
    if (e <= 0.0f) {
        printf("Ошибка: шаг 'e' должен быть > 0\n");
        return false;
    }
    if (A >= B) {
        printf("Ошибка: A должно быть меньше B\n");
        return false;
    }
    return true;
}

bool validate_derivative_params(float deltaX) {
    if (deltaX == 0.0f) {
        printf("Ошибка: deltaX не может быть нулём\n");
        return false;
    }
    return true;
}

int main() {
    int current_lib = 1;
    char lib_path[256];
    void *handle = NULL;
    func_float3 SinIntegral = NULL;
    func_float2 Derivative = NULL;

    while(1) {
        snprintf(lib_path, sizeof(lib_path), "./lib/libmath%d.so",
current_lib);
        handle = dlopen(lib_path, RTLD_LAZY);

        if(!handle) {
            fprintf(stderr, "Ошибка загрузки %s: %s\n", lib_path,
dlerror());
            return 1;
        }

        SinIntegral = (func_float3)dlsym(handle, "SinIntegral");
        Derivative = (func_float2)dlsym(handle, "Derivative");
```

```
        if(!SinIntegral || !Derivative) {
            fprintf(stderr, "Ошибка загрузки функций: %s\n",
dlerror());
            dlclose(handle);
            return 1;
        }

        printf("Используется libmath%d.so\n", current_lib);
        printf("Команды:\n1 A B e - интеграл sin(x)\n2 A dx -
производная cos(x)\n0 - переключить метод\n> ");

        char cmd[256];
        while(fgets(cmd, sizeof(cmd), stdin)) {
            cmd[strcspn(cmd, "\n")] = '\0';

            if(cmd[0] == '0') {
                dlclose(handle);
                current_lib = (current_lib == 1) ? 2 : 1;
                printf("Переключено на libmath%d.so\n", current_lib);
                break;
            }
            else if(cmd[0] == '1') {
                char* endptr;
                float params[3];
                char* token = strtok(cmd + 1, " ");
                int i = 0;
                bool valid = true;

                while (token && i < 3) {
                    if (!is_number(token)) {
                        printf("Ошибка: '%s' не является числом\n",
token);
                        valid = false;
                        break;
                    }
                    params[i++] = strtof(token, &endptr);
                    token = strtok(NULL, " ");
                }

                if (valid && i == 3) {
                    if (validate_integral_params(params[0],
params[1], params[2])) {
                        printf("Результат интеграла: %f\n",
SinIntegral(params[0], params[1], params[2]));
                    }
                } else if (i != 3) {
                    printf("Ошибка! Требуется 3 числа: 1 A B e\n");
                }
```

```
            }
            else if(cmd[0] == '2') {
                char* endptr;
                float params[2];
                char* token = strtok(cmd + 1, " ");
                int i = 0;
                bool valid = true;

                while (token && i < 2) {
                    if (!is_number(token)) {
                        printf("Ошибка: '%s' не является числом\n",
token);
                        valid = false;
                        break;
                    }
                    params[i++] = strtof(token, &endptr);
                    token = strtok(NULL, " ");
                }

                if (valid && i == 2) {
                    if (validate_derivative_params(params[1])) {
                        printf("Результат производной: %f\n",
Derivative(params[0], params[1]));
                    }
                } else if (i != 2) {
                    printf("Ошибка! Требуется 2 числа: 2 A dx\n");
                }
            }
            else {
                printf("Неизвестная команда! Доступные: 1/2/0\n");
            }
            printf("> ");
        }
    }

    return 0;
}
```

Main_static.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <stdbool.h>
#include <string.h>

extern float SinIntegral(float A, float B, float e);
```

```
extern float Derivative(float A, float deltaX);

bool is_number(const char* str) {
    if (*str == '-' || *str == '+') str++;
    bool has_dot = false;
    while (*str) {
        if (*str == '.') {
            if (has_dot) return false;
            has_dot = true;
        }
        else if (!isdigit(*str)) {
            return false;
        }
        str++;
    }
    return true;
}

int main() {
    char input[100];

    printf("Калькулятор:\n1 A B e - интеграл sin(x)\n2 A dx -
производная cos(x)\n0 - выход\n");

    while(1) {
        printf("> ");
        if (!fgets(input, sizeof(input), stdin)) {
            printf("Ошибка чтения ввода\n");
            continue;
        }

        input[strcspn(input, "\n")] = '\0';

        if(input[0] == '1') {
            char* endptr;
            float params[3];
            char* token = strtok(input + 1, " ");
            int i = 0;
            bool valid = true;

            while (token && i < 3) {
                if (!is_number(token)) {
                    printf("Ошибка: '%s' не является числом\n",
token);
                    valid = false;
                    break;
                }
                params[i++] = strtof(token, &endptr);
                token = strtok(NULL, " ");
```

```
            }

            if (valid && i == 3) {
                if (params[2] <= 0.0f) {
                    printf("Ошибка: шаг 'e' должен быть > 0\n");
                }
                else if (params[0] >= params[1]) {
                    printf("Ошибка: A должно быть меньше B\n");
                }
                else {
                    printf("Результат: %f\n", SinIntegral(params[0],
params[1], params[2]));
                }
            }
            else if (i != 3) {
                printf("Ошибка! Нужно 3 числа: 1 A B e\n");
            }
        }
        else if(input[0] == '2') {
            char* endptr;
            float params[2];
            char* token = strtok(input + 1, " ");
            int i = 0;
            bool valid = true;

            while (token && i < 2) {
                if (!is_number(token)) {
                    printf("Ошибка: '%s' не является числом\n",
token);
                    valid = false;
                    break;
                }
                params[i++] = strtof(token, &endptr);
                token = strtok(NULL, " ");
            }

            if (valid && i == 2) {
                if (params[1] == 0.0f) {
                    printf("Ошибка: deltaX не может быть нулём\n");
                }
                else {
                    printf("Результат: %f\n", Derivative(params[0],
params[1]));
                }
            }
            else if (i != 2) {
                printf("Ошибка! Нужно 2 числа: 2 A dx\n");
            }
        }
```

```
        else if(input[0] == '0') {
            printf("Выход...\n");
            break;
        }
        else {
            printf("Неизвестная команда! Доступно: 1/2/0\n");
        }
    }

    return 0;
}
```

Демонстрация работы программы:

>make

>bin/static_program или bin/dynamic_program

>Далее будет диалоговое окно в терминале

Для автоматизации запуска использовался Makefile, в нём следующий код:

```
CC = gcc
CFLAGS = -shared -fPIC
SRC_DIR = src
LIB_DIR = lib
BIN_DIR = bin

all: libs programs

# Сборка библиотек
libs:
    mkdir -p $(LIB_DIR)
    $(CC) $(CFLAGS) $(SRC_DIR)/mathlib1.c -o $(LIB_DIR)/libmath1.so -
lm
    $(CC) $(CFLAGS) $(SRC_DIR)/mathlib2.c -o $(LIB_DIR)/libmath2.so -
lm

# Сборка программ
```

```
programs:
    mkdir -p $(BIN_DIR)
    $(CC) $(SRC_DIR)/main_static.c -L$(LIB_DIR) -lmath1 -o
$(BIN_DIR)/static_program -Wl,-rpath=$(LIB_DIR)
    $(CC) $(SRC_DIR)/main_dynamic.c -ldl -o
$(BIN_DIR)/dynamic_program

# Очистка
clean:
    rm -rf $(LIB_DIR) $(BIN_DIR)
```

Тестирование:

```
bin/static_program
Калькулятор:
1 A B e - интеграл sin(x)
2 A dx - производная cos(x)
0 - выход
> 1 1 4 0.001
Результат: 1.194035
> 2 5 0.01
Результат: 0.957512
> 0
Выход...
```

```
bin/dynamic_program
Используется libmath1.so
Команды:
1 A B e - интеграл sin(x)
2 A dx - производная cos(x)
0 - переключить метод
> 1 1 4 0.001
Результат интеграла: 1.194035
```

```
> 2 5 0.01
Результат производной: 0.957512
> 0
Переключено на libmath2.so
Используется libmath2.so
Команды:
1 A B e - интеграл sin(x)
2 A dx - производная cos(x)
0 - переключить метод
> 1 1 4 0.001
Результат интеграла: 1.193238
> 2 5 0.01
Результат производной: 0.958930
> ^C
```

Выводы

В ходе лабораторной работы я освоил создание и применение динамических библиотек в C, реализовав два способа их подключения - статическую линковку и динамическую загрузку во время выполнения. В результате я получил ценный опыт модульной разработки, научился создавать переключаемые алгоритмы и автоматизировать сборку проектов с помощью CMake.