Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа 4 по курсу

«Операционные системы»

III Семестр

Создание программ, которые используют функции динамических библиотек

Студент: Шляхтуров А.В
Группа: М8О-201Б-22
Преподаватель: Миронов Е. С.
Вариант 30
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023

Цель работы.

Целью является приобретение практических навыков в создании динамических библиотек и создании программ, которые используют функции динамических библиотек

Задание

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами: во время компиляции (на этапе «линковки»/linking) или 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.

5	Рассчет значения числа Пи при заданной длине ряда (K)	float Pi(int K)	Ряд Лейбница	Формула Валлиса
9	Отсортировать целочисленный массив	Int * Sort(int * array)	Пузырьковая сортировка	Сортировка Хоара

Решение

Создадим две динамические библиотеки, которые будут иметь функции одинаковой сигнатуры, но реализованные по-разному. Создадим программу, которая будет использовать эти функции. Сделаем Makefile для удобной компиляции соответствующих программ.

Код

Lib1.cpp

```
#include "../include/lib.hpp"

extern "C"
{


    int *Sort(int *array, int size)
    {
        for (int i = 0; i < size - 1; i++)
        {
            for (int j = 0; j < size - i - 1; j++)
            {
                if (array[j] > array[j + 1])
                {
                    std::swap(array[j], array[j + 1]);
                }
            }
        }
        return array;
    }

    float Pi(int K)
    {
        float sum = 0.0;
        for (int i = 0; i < K; i++)
        {
            int sign = (i % 2 == 0) ? 1 : -1;
            float term = static_cast<float>(sign) / (2 * i + 1);
            sum += term;
        }
        return 4 * sum;
    }
}
```

Lib2.cpp

```
#include "../include/lib.hpp"


extern "C"
{

    float Pi(int K)
    {
        float prod = 1;
```

```
        float elem;
        for (int i = 1; i < K; i++)
        {
            elem = (4.0 * i * i) / (4.0 * i * i - 1);
            prod *= elem;
            // std:: cout << prod << " " << elem << " " << i << std::endl;
        }
        return 2 * prod;
    }

    int partition(int *array, int low, int high)
    {
        int pivot = array[high];
        int i = low - 1;
        for (int j = low; j <= high - 1; j++)
        {

            if (array[j] <= pivot)
            {
                i++;
                std::swap(array[i], array[j]);
            }
        }

        std::swap(array[i + 1], array[high]);
        return i + 1;
    }

    void quickSort(int *array, int low, int high)
    {
        if (low < high)
        {
            int pi = partition(array, low, high);

            quickSort(array, low, pi - 1);
            quickSort(array, pi + 1, high);
        }
    }

    // Функция сортировки Хоара
    int *Sort(int *array, int size)
    {
        quickSort(array, 0, size - 1);
        return array;
    }
}
```

Main_dynamic.cpp

```
#include <iostream>
#include <dlfcn.h>

const char *libs[2] = {"/home/alexander/labsos/lab4/build/libMyLib1.so", "/home/al-
exander/labsos/lab4/build/libMyLib2.so"};

int main()
{

    int choiseLib;
    std::cout << "Enter interested lib : ";
    std::cin >> choiseLib;
    if (choiseLib != 1 && choiseLib != 2)
    {
        std::cerr << "incorrect lib number" << std::endl;
        exit(-1);
    }

    --choiseLib;
    void *lib_header = dlopen(libs[choiseLib], RTLD_LAZY);
    if (lib_header == NULL)
    {
        std::cerr << "Cannot load library" << std::endl;
    }

    float (*Pi)(int a) = (float (*)(int))dlsym(lib_header, "Pi");
    int *(*Sort)(int *a, int b) = (int *(*)(int *, int))dlsym(lib_header, "Sort");

    if (Pi == NULL || Sort == NULL)
    {
        std::cerr << "Cannot load functions" << std::endl;
        exit(-1);
    }

    int choise;
    while (std::cout << "Введите 1 для выполнения сортировки или 2 для подсчета
числа пи" << std::endl && std::cin >> choise)
    {
        switch (choise)
        {
        case (1):
        {
            int len;
            int *array;
            std::cout << "Введите размер массива: ";
            std::cin >> len;
            std::cout << std::endl;
```

```
            array = new int[len];
            for (int i = 0; i < len; i++)
            {
                std::cout << "Введите " << i << " элемент: ";
                std::cin >> array[i];
            }

            int *sortedArr1 = Sort(array, len);

            std::cout << "\nОтсортированный массив: ";
            for (int i = 0; i < len; i++)
            {
                std::cout << sortedArr1[i] << " ";
            }
            std::cout << std::endl;

            break;
        }
        case (2):
        {
            int K;
            std::cout << "Введите длину ряда K для вычисления числа π: ";
            std::cin >> K;
            if (K <= 0)
            {
                std::cerr << "K должно быть положительным числом." << std::endl;
                break;
            }
            float result1 = Pi(K);
            std::cout << "Число π при использовании ряда длиной K = " << K << ": "
<< result1 << std::endl;

            break;
        }

        case (0):
        {
            if (dlclose(lib_header) != 0)
            {
                std::cerr << "Cannot close library" << std::endl;
                exit(-1);
            }

            choiseLib = 1 - choiseLib;

            lib_header = dlopen(libs[choiseLib], RTLD_LAZY);
            if (lib_header == NULL)
            {
                std::cerr << "Cannot load library" << std::endl;
            }
```

```
            float (*Pi)(int a) = (float (*)(int))dlsym(lib_header, "Pi");
            int *(*Sort)(int *a, int b) = (int *(*)(int *, int))dlsym(lib_header,
"Sort");

            if (Pi == NULL || Sort == NULL)
            {
                std::cerr << "Cannot load functions" << std::endl;
                exit(-1);
            }
            break;
        }
        default:
            std::cout << "unknowned command" << std::endl;
        }
    }

    if (dlclose(lib_header) != 0)
    {
        std::cerr << "Cannot close library" << std::endl;
        exit(-1);
    }

    return 0;
}
```

Main_static.cpp

```
#include <iostream>

#include "../include/lib.hpp"

extern "C"
{

    int *Sort(int *, int);
    float Pi(int);
}

int main()
{

    size_t choise;
    while (std::cout << "Введите 1 для выполнения сортировки или 2 для подсчета
числа пи" << std::endl && std::cin >> choise)
    {
        switch (choise)
        {
        case (1):{
            int len;
            int *array;
            std::cout << "Введите размер массива: ";
            std::cin >> len;
```

```
            std::cout << std::endl;
            array = new int[len];
            for (int i = 0; i < len; i++)
            {
                std::cout << "Введите " << i << " элемент: ";
                std::cin >> array[i];
            }

            int *sortedArr1 = Sort(array, len);

            std::cout << "\nОтсортированный массив: ";
            for (int i = 0; i < len; i++)
            {
                std::cout << sortedArr1[i] << " ";
            }
            std::cout << std::endl;

            break;
        }
        case (2): {
            int K;
            std::cout << "Введите длину ряда K для вычисления числа π: ";
            std::cin >> K;
            if (K <= 0)
            {
                std::cerr << "K должно быть положительным числом." << std::endl;
                break;
            }
            float result1 = Pi(K);
            std::cout << "Число π при использовании ряда длиной K = " << K << ": "
<< result1 << std::endl;


            break;
        }

        default:
            std::cout << "unknowned command" << std::endl;
        }
    }

    return 0;
}
```

Пример работы

```
Enter interested lib : 1
Введите 1 для выполнения сортировки или 2 для подсчета числа пи
2
Введите длину ряда K для вычисления числа π: 100
Число π при использовании ряда длиной K = 100: 3.13159
```

```
Enter interested lib : 1
Введите 1 для выполнения сортировки или 2 для подсчета числа пи
1
Введите размер массива: 15

Введите 0 элемент: 5
Введите 1 элемент: 7
Введите 2 элемент: 6
Введите 3 элемент: 5
Введите 4 элемент: 8
Введите 5 элемент: 7
Введите 6 элемент: 6
Введите 7 элемент: 5
Введите 8 элемент: 4
Введите 9 элемент: 9
Введите 10 элемент: 8
Введите 11 элемент: 7
Введите 12 элемент: 1
Введите 13 элемент: 56
Введите 14 элемент: 44

Отсортированный массив: 1 4 5 5 5 6 6 7 7 7 8 8 9 44 56
```

Выводы

В ходе лабораторной работы я реализовал две библиотеки, каждая из которых имеет возможность подключаться к программам во время линковки или во время runtime, т.е непосредственно во время исполнения кода.

Знания, полученные при выполнении работы, помогут мне более структурированно подходить к написанию кода, создавать библиотеки, которыми можно делиться с другими программистами, чтобы общими усилиями развивать какой-либо проект