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: Соколов Арсений Игоревич

Группа: М8О-207Б-21

Вариант: 30

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/valerasaray/os

**Постановка задачи**

**Цель работы**

приобретение практических навыков в создании динамических библиотек, создании 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:

1. Динамические библиотеки
2. реализующие контракты, которые заданы вариантом
3. Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
4. 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:

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»;

«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;

«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 30**

Расчет значения числа Пи при заданной длине ряда (K) (методы Лагранжа, Валлиса)

Сортировка целочисленного массива (пузырьковая, быстрая)

**Общие сведения о программе**

Программа состоит из двух программ main1.cpp и main2.cpp), а каждая реализация контрактов представляет из себя отдельный файл: lib1.c и lib2.c. Для объявления необходимых функций также используется заголовочный файл lib.h.

**Общий метод и алгоритм решения**

В зависимости от запуска первой или второй программы запускать соответственную реализацию контрактов на основе динамических библиотек (вычисление числа пи методами Лагранжа или Валлиса, быстрая сортировка массива или пузырьком)

**Исходный код**

**config.h**

**#ifndef CONFIG\_H\_IN**

**#define CONFIG\_H\_IN**

**#define PROJECT\_NAME "main"**

**#define COMP\_ID "GNU"**

**#define COMP\_VER "12.2.1"**

**#define TIME\_NOW "2023-03-20T14:32:33"**

**#endif // CONFIG\_H\_IN**

**config.h.in**

**#ifndef CONFIG\_H\_IN**

**#define CONFIG\_H\_IN**

**#define PROJECT\_NAME "@PROJECT\_NAME@"**

**#define COMP\_ID "@COMP\_ID@"**

**#define COMP\_VER "@COMP\_VER@"**

**#define TIME\_NOW "@TIME\_NOW@"**

**#endif // CONFIG\_H\_IN**

**lib1.cpp**

**#include <iostream>**

**#include <stdlib.h>**

**using namespace std;**

**// Функция перестановки элементов**

**void Swap(int \* a, int \* b)**

**{**

**int temp = \*a;**

**\*a = \*b;**

**\*b = temp;**

**}**

**// Пузырькова сортировка**

**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]) {**

**Swap(&array[j], &array[j+1]);**

**}**

**}**

**}**

**return array;**

**}**

**// ряд Валлиса**

**extern "C" float Pi(int K)**

**{**

**float pi = 0.0;**

**int sign = 1;**

**for (int i = 1; i <= K; i += 2) {**

**pi += (float)sign \* 4.0 / (float)i;**

**sign = -sign;**

**}**

**return pi;**

**}**

**lib2.cpp**

**#include <iostream>**

**using namespace std;**

**// сортировка Хоара (быстрая)**

**void quickSort(int \*array, int low, int high) {**

**// верхняя и нижняя границы**

**int i = low; // индекс первого элемента**

**int j = high; // индекс последнего элемента**

**// средний элемент подмассива, заданный нижней и верхней границами**

**int pivot = array[(i + j) / 2]; // опорный элемент**

**int temp;**

**while (i <= j) {**

**while (array[i] < pivot) i++;**

**while (array[j] > pivot) j--;**

**if (i <= j) {**

**temp = array[i];**

**array[i] = array[j];**

**array[j] = temp;**

**++i;**

**--j;**

**}**

**}**

**if (j > low) quickSort(array, low, j);**

**if (i < high) quickSort(array, i, high);**

**}**

**extern "C" int \* Sort(int \* array, int size)**

**{**

**quickSort(array, 0, size - 1);**

**return array;**

**}**

**// Ряд Лейбница**

**extern "C" float Pi(int k)**

**{**

**float res = 1;**

**// итеративное вычисление суммы ряда**

**for (int i = 1; i <= k; i++) {**

**float num = (float)(2\*i);**

**res \*= (num / (num - 1)) \* (num / (num + 1));**

**}**

**return res \* 2;**

**}**

**main1.cpp**

**#include<iostream>**

**#include<stdio.h>**

**#include<string>**

**#include"lib.h"**

**#include"config.h"**

**using namespace std;**

**int main()**

**{**

**cout<<"Реализация номер 2"<<endl;**

**cout<<"Версия компилятора: "<<COMP\_VER<<endl;**

**cout<<"ID компилятора: "<<COMP\_ID<<endl;**

**cout<<"Дата компиляции: "<<TIME\_NOW<<endl;**

**int command = 0;**

**cout<<"Введите номер команды\n0 для смены режима\n1 для сортировки массива\n2 для числа пи:\n";**

**while(cin>>command) {**

**if (command == 1) {**

**int n;**

**cout<<"2 Введите число элементов массива: ";**

**cin>>n;**

**cout<<"2 Введите элементы массива: ";**

**int \*array = (int \*) malloc(n \* sizeof(int));**

**for (int i = 0; i < n; i++) {**

**cin>>array[i];**

**}**

**array = Sort(array, n);**

**cout << "Отсортированный массив:\n";**

**for (int i = 0; i < n; i++) {**

**cout<<array[i]<<" ";**

**}**

**cout<<"\n";**

**continue;**

**}**

**if (command == 2) {**

**int k;**

**float answ;**

**cout<<"2 Введите число k: ";**

**cin>>k;**

**answ = Pi(k);**

**cout<<"2 Результат трансляции: " << answ << endl;**

**continue;**

**}**

**cout<<"2 Неверный номер команды"<<endl;**

**}**

**}**

**main2.cpp**

**#include<iostream>**

**#include<stdio.h>**

**#include<dlfcn.h>**

**#include"lib.h"**

**#include<string>**

**#include"config.h"**

**using namespace std;**

**int main()**

**{**

**// информация о номере реализации**

**cout<<"Реализация номер 1"<<endl;**

**// информации о версии компилятора, его ID и дате компиляции,**

**// которые определены в файле "config.h"**

**cout<<"Версия компилятора: "<<COMP\_VER<<endl;**

**cout<<"ID компилятора: "<<COMP\_ID<<endl;**

**cout<<"Дата компиляции: "<<TIME\_NOW<<endl;**

**// пути к двум динамическим библиотекам "liblib1.so" и "liblib2.so"**

**string lib1 = "./liblib1.so";**

**string lib2 = "./liblib2.so";**

**int command; // номер команды**

**// Функция возвращает указатель на загруженную библиотеку,**

**// который сохраняется в переменной "curlib".**

**void\* curlib = dlopen(lib1.c\_str(), RTLD\_LAZY); // загружаем первую библиотеку**

**// переменные-указатели на функции: "Sort" и "Pi"**

**int \*(\*Sort)(int \*, int); // принимает указатель на массив, возвращает указатель на отсортированный**

**float (\*Pi)(int k); // принимает число k, возвращает значение числа π при заданной длине ряда k**

**Sort = (int\*(\*)(int \*, int)) dlsym(curlib, "Sort");**

**Pi = (float(\*)(int))dlsym(curlib, "Pi");**

**int lib\_id = 1; // номер текущей реализации**

**cout<<"Введите номер команды\n0 для смены режима\n1 для сортировки массива\n2 для числа пи:\n";**

**while(cin>>command) {**

**// смена библиотеки**

**if (command == 0) {**

**dlclose(curlib); // закрытие текущей библиотеки**

**// открытие другой библиотеки**

**if (lib\_id == 1) {**

**curlib = dlopen(lib2.c\_str(), RTLD\_LAZY);**

**lib\_id = 2;**

**} else {**

**curlib = dlopen(lib1.c\_str(), RTLD\_LAZY);**

**lib\_id= 1;**

**}**

**Sort = (int\*(\*)(int \*, int)) dlsym(curlib, "Sort");**

**Pi = (float(\*)(int))dlsym(curlib, "Pi");**

**continue;**

**}**

**if (command == 1) {**

**int n; // число элементов массива**

**cout<<"Введите число элементов массива: ";**

**cin>>n;**

**cout<<"Введите элементы массива: ";**

**int \*array = (int \*) malloc(n \* sizeof(int)); // создание массива размера n**

**// циклический ввод элементов массива**

**for (int i = 0; i < n; i++) {**

**cin>>array[i];**

**}**

**array = Sort(array, n); // сортировка массива**

**cout << "Отсортированный массив:\n";**

**// циклический вывод элементов массива**

**for (int i = 0; i < n; i++) {**

**cout<<array[i]<<" ";**

**}**

**cout<<"\n";**

**continue;**

**}**

**if (command == 2) {**

**int k; // число k**

**float answer; // ответ**

**cout<<"Введите число k: ";**

**cin>>k;**

**answer = Pi(k);**

**cout<<"Результат трансляции: " << answer << endl;**

**continue;**

**}**

**cout<<"Неверный номер команды"<<endl;**

**}**

**}**

**Демонстрация работы программы**

**[arseny@arseny build]$** ./main

Реализация номер 1

Версия компилятора: 12.2.1

ID компилятора: GNU

Дата компиляции: 2023-03-20T14:32:33

Введите номер команды

0 для смены режима

1 для сортировки массива

2 для числа пи:

1

Введите число элементов массива: 5

Введите элементы массива: 34 0 234 -34 1

Отсортированный массив:

-34 0 1 34 234

2

Введите число k: 125

Результат трансляции: 3.15746

2

Введите число k: 3

Результат трансляции: 2.66667

0 2

Введите число k: 125

Результат трансляции: 3.13535

0 1

Введите число элементов массива: 4

Введите элементы массива: 23 4342 -2 0

Отсортированный массив:

-2 0 23 4342

**Выводы**

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