Московский авиационный институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

Лабораторная работа по предмету "Операционные системы" №4

Студент: Пирязев М.А.

Преподаватель: Миронов Е.С.

Группа: М8О-207Б-22

Дата: 23.11.2023

Оценка:

Подпись:

Москва 2023 г.

Оглавление

Цель работы

Целью является приобретение практических навыков в:

- Создание динамических библиотек
- Создание программ, которые используют функции динамических библиотек

Постановка задачи

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»;

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 30.

Рассчет значения числа Пи при заданной длине ряда (K) используя ряд Лейбница и формулу Валлиса.

Отсортировать целочисленный массив при помощи сортировки пузырьком и сортировки Хоара.

Общие сведения о программе

В программе есть 3 варианта запуска: с динамически подгружаемыми библиотеками, либо же подключаемыми изначально на этапе компиляции программы. Чтобы увидеть разницу между ними нужно сначала протестировать программу запуская ее статические варианты (их исполняемые файлы static_first и static_second). Как понятно из названия эти программы включают в себя разные библиотеки. Для того чтобы увидеть в действии динамическую работу библиотек, нужно запустить файл dynamic. В нем же библиотеки загружаются во время работы программы при переключении между ними с помощью команды 0.

Общий алгоритм решения

Программа динамически загружает две библиотеки, переключается между ними в зависимости от значения switcher_condition, и вызывает функции из этих библиотек в ответ на пользовательский ввод. Пользователь может вводить различные значения, чтобы переключаться между библиотеками, сортировать массив и вычислять значение числа π с заданной точностью. Программа начинает с динамической загрузки двух библиотек с использованием функции dlopen, вызывает функции из соответствующей библиотеки с помощью указателей на функции, полученных с помощью dlsym. Пользователь может вводить различные значения, где -1 завершает программу, 0 переключает между библиотеками, 1 вычисляет значение числа π с заданной точностью, а 2 сортирует массив и выводит его содержимое.

Реализация

static.c

```c
#include "lib.h"
#include "constants.h"

int main() {
    int received_digit;

    int array[constant_size];
    for (int iteration = 0; iteration < constant_size; iteration++){
        array[iteration] = constant_size - iteration;
    }

    while(scanf("%d", &received_digit) != EOF) {
        if (received_digit == -1){
                return 0;
        }
        else if (received_digit == 1){
                int number;
                printf("Enter your accuracy value: ");
                scanf("%d", &number);
                printf("%f\n", (*Pi)(number));
        }
        else if (received_digit == 2){
                printf("Sorted array: ");
                Sort(array, constant_size);
                for (int iteration = 0; iteration < constant_size; iteration++){
                    printf("%d, ", array[iteration]);
                }
```

```
                printf("\n");
        }
    }
}
```

---

first_lib.c

```c
#include "lib.h"

float Pi(int accuracy) {
    float pi = 0.0;
    int sign = 1;
    for (int i = 0; i < accuracy; i++) {
        pi += (4.0 / (2 * i + 1)) * sign;
        sign *= -1;
    }
    return pi;
}

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]) {
                int temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }
    }
    return array;
}
```

---

second_lib.c

```c
#include "lib.h"

float Pi(int accuracy)
{
    float pi = 2.0;
    for (int i = 1; i <= accuracy; i++) {
        pi *= (4.0 * i * i) / ((4.0 * i * i) - 1);
    }
    return pi;
}

void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
```

```
    *b = t;
}
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
int* Sort(int* array, int size) {
    quickSort(array, 0, size - 1);
    return array;
}
```

---

dynamic.c

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <dlfcn.h>
#include "constants.h"


const char* FIRST_PATH_FOR_LIBRARY = "build/libfirst_lib.so";
const char* SECOND_PATH_FOR_LIBRARY = "build/libsecond_lib.so";
void* descriptor = NULL;
int switcher_condition = 2;

float (*Pi)(int accuracy) = NULL;

int*(*Sort)(int * array, int size) = NULL;

void switch_library() {
    if(switcher_condition == 1) {
```

```
        if(descriptor != NULL)
            dlclose(descriptor);

        descriptor = dlopen(SECOND_PATH_FOR_LIBRARY, RTLD_LAZY);

        Pi = dlsym(descriptor, "Pi");
        Sort = dlsym(descriptor, "Sort");

        switcher_condition = 2;

    } else {

        if(descriptor != NULL)
            dlclose(descriptor);

        descriptor = dlopen(FIRST_PATH_FOR_LIBRARY, RTLD_LAZY);

        Pi = dlsym(descriptor, "Pi");
        Sort = dlsym(descriptor, "Sort");

        switcher_condition = 1;

    }
    printf("Switcher condition changed\n");
}

int main() {
    switch_library();
    int received_digit;

    int array[constant_size];
    for (int iteration = 0; iteration < constant_size; iteration++){
        array[iteration] = constant_size - iteration;
    }

    while(scanf("%d", &received_digit) != EOF) {
        if (received_digit == -1){
                dlclose(descriptor);
                return 0;
        }
        else if (received_digit == 0){
                switch_library();
        }
        else if (received_digit == 1){
                int number;
                printf("Enter your accuracy value: ");
                scanf("%d", &number);
                printf("%f\n", (*Pi)(number));
        }
        else if (received_digit == 2){
```

```
                printf("Sorted array: ");
                Sort(array, constant_size);
                for (int iteration = 0; iteration < constant_size; iteration++){
                    printf("%d, ", array[iteration]);
                }
                printf("\n");
        }
    }
}
```

Пример работы

Test 1

Input	Output
1 2 2	Switcher condition changed Enter your accuracy value: 2.666667 Sorted array: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100,

Вывод

Данная лабораторная работа связана с динамической загрузкой библиотек в языке Си с использованием библиотеки dlfcn.h. Код демонстрирует переключение между двумя библиотеками и вызов функций из этих библиотек. Данная лабораторная познакомила меня с динамически загружаемыми библиотеками и предоставила точное понимание какие выгоды открываются перед пользователем который может динамически изменять функционал программы, не перезапуская при этом каждый раз исполняемые файлы. Также произошло знакомство с несколькими функциями которые нужны при работе с динамическими библиотеками, а именно dlopen, dlsym и dlclose.