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: Фаттяхетдинов Сильвестр Динарович
Группа: М8О-208Б-20
Вариант: 30
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/silverfatt

Постановка задачи

Задача: реализовать 2 динамические библиотеки и 2 программы для работы с ними. Первая программа будет загружать библиотеку (одну) на этапе компиляции при помощи ключа -lmylib, а вторая программа будет подключать две динамические библиотеки при помощи dl-функций в самом коде.

Общие сведения о программе

Для выполнения данной лабораторной работы я предварительно создал 5 файлов: первые два – realization1.cpp и realization2.cpp являются исходным кодом для наших динамических библиотек. Файлы main1.cpp и main2.cpp являются двумя программами, которые нужно было реализовать по заданию. Main1.cpp является программой, к которой библиотека подгружается на этапе компиляции, а main2.cpp является программой, к которой библиотека подключается непосредственно во время работы программы. Файл realization.h служит для объявления функций.

Помимо этого, для удобства компиляции всех программ я создал MakeFile со следующим набором команд:

```
g++ -fPIC -c realization1.cpp -o dynamic1.o
g++ -fPIC -c realization2.cpp -o dynamic2.o
```

При помощи этих команд cpp-реализации превращаются в объектные файлы. Это, так называемый, “промежуточный этап” создания динамических библиотек.

```
g++ -shared -o libdynamic1.so dynamic1.o
g++ -shared -o libdynamic2.so dynamic2.o
```

При помощи флага -shared мы создаем наши нужные по заданию динамические библиотеки.

g++ main1.cpp -L. -Wl, -rpath,. -ldynamic1 -o main1
Этой строчкой мы делаем исполняемый файл из нашей программы compilation.cpp, при этом компилируем мы только с одной библиотекой (то есть компиляция может проходить либо с ключом -ld1, либо с ключом -ld2).
g++ main2.cpp -ldl -Wl, -rpath,. -o main2
Этой строчкой мы делаем исполняемый файл из нашей программы launch.cpp, только теперь с флагом -ldl. Далее в нашей программе main2 будут доступны 2 динамические библиотеки, действия над которыми будут обрабатываться при помощи следующих функций:
void* dlopen(...) - вгружает нашу библиотеку;
void* dlsym(...) - присваивает указателю на функцию ее адрес в библиотеке
int dlclose(...) - освобождает указатель на библиотеку

Общий метод и алгоритм решения

В самом начале выполнения лабораторной работы я реализовал две библиотеки: realization1.cpp и realization2.cpp. В библиотеке realization1.cpp реализованы вычисление значения числа Пи при заданной длине ряда (K) (ряд Лейбница) и реализована пузырьковая сортировка. В библиотеке realization2.cpp реализованы те же функции, но другими методами – формула Валлиса, сортировка Хоара. Далее в файле main1.cpp я реализовал обычное считывание команды при помощи проверки равенства функции scanf на -1 (вводится EOF - Ctrl+D на Ubuntu) и конструкции switch-case. Если вводится команда, отличная от 1 или 2, вылезает сообщение о том, что ввод был осуществлен неправильно. Если вводится 1, то считается число пи. Если вводится 2, то вводится и сортируется массив.
Что же касается main2.cpp, то там суть почти та же. В начале создаю необходимые указатели, позже загружаю какую-либо дин. библиотеку в зависимости от ввода пользователя. При помощи известного нам считывания до EOF я считываю команду. Если это не 0, не 1 и не 2, то прошу ввести правильную команду. Если эта команда 0, то программа меняет библиотеки.

Если команда 1, то считается значение числа Пи. Если 2, то сортируется массив. В конце освобождается указатель на библиотеку в целях избежания утечек памяти, программа завершается.

Исходный код

realizations.h

```
extern "C" float Pi(int K);
//int * Sort(int * array, int size);
extern "C" int * Sort(int * array, int right, int left = 0);
```

realization1.cpp

```
#include "realizations.h"

 float Pi(int K){
    float pi = 0;
    float del = 1;
    for (int i = 1; i<=K; i++){
        if(i % 2 == 1) pi = pi + 4/del;
        else pi = pi-4/del;
        del = del+2;
    }
    return pi;
}

 int * Sort(int * array, int size, int left){
    ++size;
    int tmp;
        for (int i = 0; i < size; i++) {
        for (int j = 0; j < size - 1; j++) {
            if (array[j] > array[j + 1]) {
                tmp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = tmp;
            }
        }
    }
}
```

realization2.cpp

```cpp
#include "realizations.h"

 float Pi(int K){
    float del = 2;
    float div = 1;
    int count = 1;
    float pi1 = 1;
    for (int i = 1; i<=K; i++){
        pi1 = pi1*del/div;
        if(count % 2 == 1) div = div + 2;
        else del = del+2;
        ++count;
    }
    float pi = 2*pi1;
    return pi;
}


int* Sort(int* s_arr, int last, int first){
  int i = first, j = last, x = s_arr[(first + last) / 2];
    do {
        while (s_arr[i] < x) i++;
        while (s_arr[j] > x) j--;

        if(i <= j) {
            if (s_arr[i] > s_arr[j]){
              int t;
              t = s_arr[i];
              s_arr[i] = s_arr[j];
              s_arr[j] = t;
            }
            i++;
            j--;
        }
    } while (i <= j);

    if (i < last)
        Sort(s_arr, last, i);
    if (first < j)
        Sort(s_arr, j, first);
}
```

main1.cpp

```cpp
#include <iostream>
//#include "realizations.h"

extern "C" float Pi(int K);
extern "C" int * Sort(int * array, int left, int right = 0);
```

```
int main(){
    int command;
    std::cout << "Insert a command\n 1 - engage Pi function\n 2 - engage Sort
function\n";
    while(scanf("%d", &command) != EOF){
        //std::cin >> command;
        switch(command){
            case 1:{ //pi function
                std::cout << "Insert accuracy K\n";
                int K;
                std::cin >> K;
                float pi = Pi(K);
                std::cout << "Result is: " << pi << "\n";
                break;
            }
            case 2:{ //sort function
                std::cout << "Insert array length\n";
                int N;
                std::cin >> N;
                int array[N];
                std::cout << "Insert array elements\n";
                for(int i = 0; i<N; ++i){
                    std::cin >> array[i];
                };
                int size = sizeof(array)/sizeof(array[0]);
                Sort(array, size-1, 0);
                std::cout << "Result is:\n";
                for(int i = 0; i<N; ++i){
                    std::cout << array[i] << " ";
                }
                std::cout << "\n";
                break;
            }
                default:{
                std::cout << "--Wrong command!--\n";
                std::cout << "Insert a command\n 0 - change library\n 1 - engage
Pi function\n 2 - engage Sort function\n";
            }
        }
    }
    std::cout << "Shutting down...\n";

}
```

main2.cpp

```
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
```

```
#include <dlfcn.h>

int main(){
    int command = 1;
    int libmode = 1;
    void *handle;
    handle = dlopen("libdynamic1.so",RTLD_LAZY);
    if (!handle){
        fprintf(stderr, "dlopen() error: %s\n", dlerror());
        exit(1);
    }
    float (*Pi)(int);
    int* (*Sort)(int*, int, int);
    Pi = (float(*)(int))dlsym(handle, "Pi");
    Sort = (int*(*)(int*,int,int))dlsym(handle, "Sort");
    //std::cout << handle << "\n";
    std::cout << "Now using lib1\n";
    std::cout << "Insert a command\n 0 - change library\n 1 - engage Pi
function\n 2 - engage Sort function\n";
    while(scanf("%d", &command) != EOF){
        //std::cin >> command;
        switch(command){
            case 0:{ //change lib
                dlclose(handle);
                if(libmode == 1){
                    handle = dlopen("libdynamic2.so", RTLD_LAZY);
                if (!handle){
                fprintf(stderr, "dlopen() error: %s\n", dlerror());
                exit(1);
                }
                    libmode = 2;
                    std::cout << "Now using lib2\n";
                } else if(libmode == 2){
                if (!handle)
                {
                fprintf(stderr, "dlopen() error: %s\n", dlerror());
                exit(1);
                }
                    handle = dlopen("libdynamic1.so", RTLD_LAZY);
                    libmode = 1;
                    std::cout << "Now using lib1\n";
                }
                Pi = (float(*)(int))dlsym(handle, "Pi");
                Sort = (int*(*)(int*,int,int))dlsym(handle, "Sort");
                break;
            }
            case 1:{ //pi function
                std::cout << "Insert accuracy K\n";
                int K;
                std::cin >> K;
```

```
                float pi = Pi(K);
                std::cout << "Result is: " << pi << "\n";
                break;
            }
            case 2:{ //sort function
                std::cout << "Insert array length\n";
                int N;
                std::cin >> N;
                int array[N];
                std::cout << "Insert array elements\n";
                for(int i = 0; i<N; ++i){
                    std::cin >> array[i];
                };
                int size = sizeof(array)/sizeof(array[0]);
                Sort(array, size-1, 0);
                std::cout << "Result is:\n";
                for(int i = 0; i<N; ++i){
                    std::cout << array[i] << " ";
                }
                std::cout << "\n";
                break;
            }
            default:{
                std::cout << "--Wrong command!--\n";
                std::cout << "Insert a command\n 0 - change library\n 1 - engage
Pi function\n 2 - engage Sort function\n";
            }
        }
    }
    dlclose(handle);
    std::cout << "Library closed.\n";

}
```

Демонстрация работы программы

```
silverfatt@DESKTOP-AGNE5GI:~/lab5$ ./main1
Insert a command
 1 - engage Pi function
 2 - engage Sort function
1
Insert accuracy K
100
Result is: 3.13159
2
Insert array length
3
Insert array elements
1 2 3
Result is:
1 2 3
```

```
silverfatt@DESKTOP-AGNE5GI:~/lab5$ ./main2
Now using lib1
Insert a command
 0 - change library
 1 - engage Pi function
 2 - engage Sort function
1
Insert accuracy K
100
Result is: 3.13159
0
Now using lib2
1
Insert accuracy K
100
Result is: 3.12608
2
Insert array length
4
Insert array elements
3 2 1 4
Result is:
1 2 3 4
4
--Wrong command!--
Insert a command
 0 - change library
 1 - engage Pi function
 2 - engage Sort function
1
Insert accuracy K
1000
Result is: 3.14002
```

Выводы

Данная лабораторная работа научила меня пользоваться dl-функциями, благодаря реализации исполняемых файлов по заданию, я закрепил навык работы с динамическими библиотеками и полностью осознал их отличие от статических библиотек.