

**Московский авиационный институт
(национальный исследовательский университет)**

**Факультет информационных технологий и прикладной
математики**

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «Операционные системы»

Создание динамических библиотек.

**Создание программ, которые используют функции динамических
библиотек.**

Студент: А. Р. Боташев
Преподаватель: Е. С. Миронов
Группа: М8О-201Б-21
Вариант: 9
Дата:
Оценка:
Подпись:

Москва, 2023

1 Постановка задачи

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами:

- Во время компиляции (на этапе «линковки»/linking)
- Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В лабораторной работе необходимо получить следующие части:

- Динамические библиотеки, реализующие контракты
- Тестовая программа (программа №1), которая использует 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 должен быть организован следующим образом:

- Команда «0»: переключить одну реализацию контракты на другую
- Команда «1 args»: вызов первой функции контрактов
- Команда «2 args»: вызов второй функции контрактов

Контракты:

- Подсчет количества простых чисел на промежутке [A; B]
- Отсортировать целочисленный массив

2 Сведения о программе

Программа написана на Си в Unix подобной операционной системе на базе ядра Linux.

Оба контракта реализованы в двух вариантах:

Нахождение простых чисел на промежутке наивным способом(перебором делителей) и решето Эратосфена

Сортировка массива пузырьком и quick-sort(сортировка Хоара)

Контракты описаны в файле signature.h, а реализация implem1.cpp и implem2.cpp

1. Создание объектных файлов
2. Компиляция библиотек с ключем `-shared`. Получаем динамические библиотеки с расширением `.so`
3. Линковка библиотеки к необходимой программе

Для динамической загрузки библиотек используется библиотека `dlfcn.h`

3 Общий метод и алгоритм решения

Программа принимает в себя команды:

- В случае команды 1, мы считываем `A` и `B` и находим все простые числа на отрезке `[A; B]`.
- В случае команды 2, мы считываем размер массива и сам массив и выводим его в отсортированном виде.
- В случае команды 0, мы закрываем старую библиотеку, открываем вторую и заменяем указатели на функции.

Для завершения программы нужно ввести комбинацию завершения ввода – `CTRL+D`.

4 Листинг программы

`signature.h`

```
1 | #ifndef OS_LABS_SIGNATURE_H
2 | #define OS_LABS_SIGNATURE_H
3 |
4 |
5 | #include <cstdint>
6 | #include <algorithm>
7 | #include <cmath>
8 | #include <vector>
9 |
10 | extern "C" {
11 |     int primeCount(int a, int b);
12 |     std::vector<int> sort(std::vector<int> array, int low, int high);
13 | }
14 |
15 |
16 | #endif //OS_LABS_SIGNATURE_H
```

implem1.cpp

```
1 #include "signature.h"
2
3 int primeCount(int a, int b) {
4     int res = 0;
5     for (int i = a; i <= b; i++){
6         if (i > 1) {
7             bool is_prime = true;
8             for (int j = 2; j * j <= i; j++) {
9                 if (i % j == 0) {
10                     is_prime = false;
11                     break;
12                 }
13             }
14             if (is_prime) {
15                 res++;
16             }
17         }
18     }
19     return res;
20 }
21
22 std::vector<int> sort(std::vector<int> array, int low, int high) {
23     for (int i = 0; i < high; i++) {
24         for (int j = 0; j < high - 1; j++) {
25             if (array[j] > array[j + 1]) {
26                 std::swap(array[j], array[j + 1]);
27             }
28         }
29     }
30     return array;
31 }
```

implem2.cpp

```
1 #include "signature.h"
2 #include <stack>
3
4 int primeCount(int a, int b) {
5     int prime[b + 1];
6     for (int i = 0; i <= b; i++) {
7         prime[i] = true;
8     }
9     prime[0] = prime[1] = false;
10    int res = 0;
11    for (int i = 2; i <= b; ++i)
12        if (prime[i]) {
13            if (i * 111 * i <= b)
14                for (int j = i * i; j <= b; j += i)
15                    prime[j] = false;
```

```

16
17         res += (i >= a);
18     }
19     return res;
20 }
21
22 std::vector<int> sort(std::vector<int>array, int low, int high) {
23     int i = low;
24     int j = high;
25     int pivot = array[(i + j) / 2];
26
27     while (i <= j)
28     {
29         while (array[i] < pivot)
30             i++;
31         while (array[j] > pivot)
32             j--;
33         if (i <= j)
34         {
35             std::swap(array[i], array[j]);
36             i++;
37             j--;
38         }
39     }
40     if (j > low)
41         array = sort(array, low, j);
42     if (i < high)
43         array = sort(array, i, high);
44
45     return array;
46 }

```

main_static.cpp

```

1  #include <iostream>
2  #include "signature.h"
3
4
5  int main() {
6      int command;
7      while (std::cin >> command) {
8          if (command == 1) {
9              int a, b;
10             std::cin >> a >> b;
11             if (a > b){
12                 std::cout << R("a" have to be less or equal then "b"\n");
13             }
14             else{
15                 std::cout << primeCount(a, b) << "\n";
16             }
17         }
18     }
19 }

```

```

17
18     } else if (command == 2) {
19         int n;
20         std::cin >> n;
21         std::vector<int> arr(n);
22         for (int i = 0; i < n; i++){
23             std::cin >> arr[i];
24         }
25         arr = sort(arr, 0, n);
26
27         for (int i = 0; i < n; i++){
28             std::cout << arr[i] << " ";
29         }
30         std::cout << "\n";
31     } else {
32         std::cout << "you have to enter 1 or 2" << std::endl;
33     }
34 }
35 }

```

main_{dynamic.cpp}

```

1  #include <array>
2  #include <cstdio>
3  #include <iostream>
4  #include <cstdlib>
5  #include <dlfcn.h>
6  #include <string>
7  #include <vector>
8
9
10 int main() {
11     const std::vector<std::string> LIB = {"/libd1_dynamic.so", "/libd2_dynamic.so"};
12     const std::vector<std::string> FUNC_NAME = {"primeCount", "sort"};
13     int curlib = 0;
14     int (*primeCount)(int a, int b);
15     std::vector<int> (*sort)(std::vector<int> array, int low, int high);
16     void* handle = dlopen(LIB[curlib].c_str(), RTLD_LAZY);
17     if (handle == nullptr) {
18         std::cout << "Fail dlopen\n";
19         return EXIT_FAILURE;
20     }
21     primeCount = ((int (*)(int, int)) dlsym(handle, FUNC_NAME[0].c_str()));
22     sort = (std::vector<int> (*)(std::vector<int>, int, int))dlsym(handle, FUNC_NAME
23         [1].c_str());
24     int command;
25     while (std::cin >> command) {
26         if (command == 1) {
27             int a, b;
28             std::cin >> a >> b;

```

```

28     if (a > b){
29         std::cout << R("a have to be less or equal then "b"\n");
30     }
31     else{
32         std::cout << primeCount(a, b) << "\n";
33     }
34 } else if (command == 2) {
35     int n;
36     std::cin >> n;
37     std::vector <int>arr(n);
38     for (int i = 0; i < n; i++){
39         std::cin >> arr[i];
40     }
41     sort(arr, 0, n);
42
43     for (int i = 0; i < n; i++){
44         std::cout << arr[i] << " ";
45     }
46     std::cout << "\n";
47 } else if (command == 0) {
48     dlclose(handle);
49     curlib ^= 1;
50     void* handle = dlopen(LIB[curlib].c_str(), RTLD_LAZY);
51     if (handle == nullptr) {
52         std::cout << "Fail dlopen\n";
53         return EXIT_FAILURE;
54     }
55     primeCount = ((int (*)(int, int)) dlsym(handle, FUNC_NAME[0].c_str()));
56     sort = (std::vector <int>(*)(std::vector <int>, int, int))dlsym(handle,
57         FUNC_NAME[1].c_str());
58 } else {
59     std::cout << "you have to enter 0, 1 or 2" << std::endl;
60 }
61 dlclose(handle);
62 }

```

5 Демонстрация работы программы

```
botashev@botashev-laptop:~/ClionProjects/os_labs/tests$ cat lab5_test.cpp
#include <fstream>
#include <gtest/gtest.h>
#include <string>
#include <dlfcn.h>
#include <signature.h>

TEST(Lab5Test, DynamicTest) {

    const std::vector <std::string>FUNC_NAME = {"primeCount", "sort"};

    std::vector <std::vector <int>>>input1 = {{1,10},{1,100},{107,107},{100,105}};

    std::vector <std::vector <int>>>input2 = {
        {3,2,1,4,5},
        {100,-1,0},
        {1,1,1,1,1},
        {1000000},
        {-1919,-66666,-789}
    };

    std::vector <int>>output1 = {4,25,1,2};

    std::vector <std::vector <int>>>output2= {
        {1,2,3,4,5},
        {-1,0,100},
        {1,1,1,1,1},
        {1000000},
        {-66666,-1919,-789}
    };

    int (*primeCountOne)(int a,int b);
    std::vector <int>(*sortOne)(std::vector <int>array,int low,int high);
    void* handleOne = dlopen(
        getenv("PATH_TO_libd1_dynamic.so"),RTLD_LAZY);
    ASSERT_NE(handleOne,nullptr);

    int (*primeCountTwo)(int a,int b);
    std::vector <int>(*sortTwo)(std::vector <int>array,int low,int high);
    void* handleTwo = dlopen(getenv("PATH_TO_libd2_dynamic.so"),RTLD_LAZY);
    ASSERT_NE(handleTwo,nullptr);

    primeCountOne = ((int (*)(int,int)) dlsym(handleOne,FUNC_NAME[0].c_str()));
    primeCountTwo = ((int (*)(int,int)) dlsym(handleTwo,FUNC_NAME[0].c_str()));
    sortOne = (std::vector <int>(*)(std::vector <int>,int,int))dlsym(handleOne,FUNC_NAME[1].c_str());
    sortTwo = (std::vector <int>(*)(std::vector <int>,int,int))dlsym(handleTwo,FUNC_NAME[1].c_str());

    for (size_t i = 0; i <input1.size(); i++) {
        auto primeCountOutOne = primeCountOne(input1[i][0],input1[i][1]);
        auto primeCountOutTwo = primeCountTwo(input1[i][0],input1[i][1]);
        EXPECT_EQ(primeCountOutOne,output1[i]);
        EXPECT_EQ(primeCountOutTwo,output1[i]);
    }
}
```



```

for (size_t i = 0; i <input2.size(); i++) {
    auto sortOutOne = sortOne(input2[i],0,(int)input2[i].size());
    auto sortOutTwo = sortTwo(input2[i],0,(int)input2[i].size() -1);

    EXPECT_EQ(sortOutOne,output2[i]);
    EXPECT_EQ(sortOutTwo,output2[i]);
}
}

TEST(Lab5Test,StaticOneTest) {

    std::vector <std::vector <int>>>input1 = {{1,10},{1,100},{107,107},{100,105}};

    std::vector <std::vector <int>>>input2 = {
        {3,2,1,4,5},
        {100,-1,0},
        {1,1,1,1,1},
        {1000000},
        {-1919,-66666,-789}
    };

    std::vector <int>output1 = {4,25,1,2};

    std::vector <std::vector <int>>>output2= {
        {1,2,3,4,5},
        {-1,0,100},
        {1,1,1,1,1},
        {1000000},
        {-66666,-1919,-789}
    };

    for (int i = 0; i <input1.size(); i++) {
        auto primeCountOut = primeCount(input1[i][0],input1[i][1]);
        EXPECT_EQ(primeCountOut,output1[i]);
    }

    for (size_t i = 0; i <input2.size(); i++) {
        auto sortOut = sort(input2[i],0,(int)input2[i].size());

        for (int j = 0; j <output2[i].size(); j++){
            EXPECT_EQ(sortOut[j],output2[i][j]);
        }
    }
}

botashev@botashev-laptop:~/ClionProjects/os_labs/tests$ ../../cmake-build-debug/tests/lab5_test
Running main() from /home/botashev/ClionProjects/os_labs/cmake-build-debug/_deps/googletest-src/googletest/src/gtest_main.cc
[=====] Running 2 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 2 tests from Lab5Test
[ RUN      ] Lab5Test.DynamicTest
[ OK       ] Lab5Test.DynamicTest (1 ms)
[ RUN      ] Lab5Test.StaticOneTest
[ OK       ] Lab5Test.StaticOneTest (0 ms)
[-----] 2 tests from Lab5Test (1 ms total)

[-----] Global test environment tear-down
[=====] 2 tests from 1 test suite ran. (1 ms total)
[ PASSED   ] 2 tests.

```

6 Вывод

Лабораторная работа была направлена на изучение динамических библиотек в Unix подобных операционных системах. Для изучения создания и работы с ними мною было написано 2 программ: одна подлючала динамичкие библиотеки на этапе компиляции, а вторая во время исполнения.

Динамические библиотеки содержат функционал отдельно от программы и прередают его непосредственно во время исполнения. Из плюсов такого подхода можно выделить, что во-первых, в таком случае размер результирующей программы меньше, во-вторых, одну и ту же библиотеку можно использовать в нескольких программах не встраивая в код, чем можно также добиться снижения общего занимаемого пространства на диске, и в-третьих, что после исправления ошибок в библиотеке не нужно перекомпилировать все программы, достаточно перекомпилировать саму библиотеку.

Однако у динамических библиотек есть и недостатки. Первый заключается в том, что вызов функции из динамической библиотеки происходит медленнее. Второй, что мы не можем подправить функционал библиотеки под конкретную программу не зацепив при этом других программ, работающих с этой библиотекой. И в-третьих, уже скомпилированная программа не будет работать на аналогичной системе без установленной динамической библиотеки.

Тем не менее плюсы динамичеких библиотек исчерпывают их минусы в большенстве задач, обратных случаях лучше обратиться к статическим библиотекам. В наше время с высокими мощностями вычислительных систем становится более важным сэкономить объем памяти, используемый программой, чем время обращения к функции. Поэтому динамические библиотеки используются в большенстве современных программ.