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: В. А. Амурский
Преподаватель: Е. С. Миронов
Группа: М8О-201Б-21
Вариант: 9
Дата:
Оценка:
Подпись:

Москва, 2023

1 Постановка задачи

Цель данной лабораторной работы заключается в создании динамических библиотек, реализующих определенный функционал, и использовании их в двух различных способах: на этапе компиляции и во время исполнения программы с помощью интерфейса ОС для работы с динамическими библиотеками.

Для выполнения лабораторной работы необходимо создать динамические библиотеки, реализующие контракты, а также две тестовые программы, одна из которых будет использовать одну из библиотек, используя знания полученные на этапе компиляции, а вторая будет загружать библиотеки, используя только их местоположение и контракты.

Анализ двух типов использования библиотек также является одним из заданий лабораторной работы.

2 Сведения о программе

Программа написана на языке 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 решетом Эратосфена; сортировка массива пузырьком и quick-sort (сортировка Хоара).

Контракты описаны в файле signature.h, а их реализация в файлах implem1.c и implem2.c. Для создания динамических библиотек необходимо выполнить три шага: создание объектных файлов, компиляция библиотек с ключем -shared, получение динамических библиотек с расширением .so и линковка библиотеки к необходимой программе. Для динамической загрузки библиотек используется библиотека dlfcn.h.

Программа принимает в себя команды для нахождения простых чисел на отрезке и сортировки массива. Команда 0 переключает одну реализацию контракта на другую, закрывая старую библиотеку, открывая вторую и заменяя указатели на функции. Для завершения программы необходимо ввести комбинацию завершения ввода - CTRL+D.

3 Листинг программы

signature.h

```
1 #ifndef OS-labs_SIGNATURE_H
2 #define OS-labs_SIGNATURE_H
3
4
5 #include <cstddef>
```

```
#include <algorithm>
#include <cmath>
#include <vector>

extern "C" {
    int primeCount(int a, int b);
    std::vector <int> sort(std::vector <int> array, int low, int high);
}


#endif //OS-labs_SIGNATURE_H
```

implem1.cpp

```
#include "signature.h"

int primeCount(int a, int b) {
    int res = 0;
    for (int i = a; i <= b; i++){
        if (i > 1) {
            bool is_prime = true;
            for (int j = 2; j * j <= i; j++) {
                if (i % j == 0) {
                    is_prime = false;
                    break;
                }
            }
            if (is_prime) {
                res++;
            }
        }
    }
    return res;
}

std::vector <int>sort(std::vector <int>array, int low, int high) {
    for (int i = 0; i < high; i++) {
        for (int j = 0; j < high - 1; j++) {
            if (array[j] > array[j + 1]) {
                std::swap(array[j], array[j + 1]);
            }
        }
    }
    return array;
}
```

implem2.cpp

```
#include "signature.h"
#include <stack>

```

```cpp
int primeCount(int a, int b) {
    int prime[b + 1];
    for (int i = 0; i <= b; i++) {
        prime[i] = true;
    }
    prime[0] = prime[1] = false;
    int res = 0;
    for (int i = 2; i <= b; ++i)
        if (prime[i]) {
            if (i * 1ll * i <= b)
                for (int j = i * i; j <= b; j += i)
                    prime[j] = false;

            res += (i >= a);
        }
    return res;
}

std::vector <int> sort(std::vector <int>array, int low, int high) {
    int i = low;
    int j = high;
    int pivot = array[(i + j) / 2];

    while (i <= j)
    {
        while (array[i] < pivot)
            i++;
        while (array[j] > pivot)
            j--;
        if (i <= j)
        {
            std::swap(array[i], array[j]);
            i++;
            j--;
        }
    }
    if (j > low)
        array = sort(array, low, j);
    if (i < high)
        array = sort(array, i, high);

    return array;
}
```

$\mathbf{main}_{s}tatic.cpp$

```cpp
#include <iostream>
#include "signature.h"


```

```cpp
int main() {
    int command;
    while (std::cin >> command) {
        if (command == 1) {
            int a, b;
            std::cin >> a >> b;
            if (a > b){
                std::cout << R"("a" have to be less or equal then "b"\n)";
            }
            else{
                std::cout << primeCount(a, b) << "\n";
            }

        } else if (command == 2) {
            int n;
            std::cin >> n;
            std::vector <int> arr(n);
            for (int i = 0; i < n; i++){
                std::cin >> arr[i];
            }
            arr = sort(arr, 0, n);

            for (int i = 0; i < n; i++){
                std::cout << arr[i] << " ";
            }
            std::cout << "\n";
        } else {
            std::cout << "you have to enter 1 or 2" << std::endl;
        }
    }
}
```

main$_{d}$*ynamic.cpp*

```cpp
#include <array>
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <dlfcn.h>
#include <string>
#include <vector>


int main() {
    const std::vector <std::string> LIB = {"./libd1_dynamic.so", "./libd2_dynamic.so"};
    const std::vector <std::string> FUNC_NAME = {"primeCount", "sort"};
    int curlib = 0;
    int (*primeCount)(int a, int b);
    std::vector <int> (*sort)(std::vector <int>array, int low, int high);
    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
            [1].c_str());
23      int command;
24      while (std::cin >> command) {
25          if (command == 1) {
26              int a, b;
27              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,
                    FUNC_NAME[1].c_str());
57          } else {
58              std::cout << "you have to enter 0, 1 or 2" << std::endl;
59          }
60      }
61      dlclose(handle);
62  }
```

4 Демонстрация работы программы

```
vazy1@vazy1-legion:~/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]);
}
}
}
vazy1@vazy1-legion:~/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.
```

5 Вывод

Я написал две программы для изучения динамических библиотек в Unix-подобных операционных системах. Одна программа подключает библиотеки на этапе компиляции, а другая - во время исполнения. Динамические библиотеки содержат функционал, который передается в программу во время ее выполнения. Их использование имеет ряд преимуществ, таких как уменьшение размера программы, возможность использования одной библиотеки в нескольких программах без встраивания в код, а также возможность обновления библиотеки без перекомпиляции всех программ, которые ее используют. Однако, есть и недостатки - вызов функции из динамической библиотеки может быть медленнее, изменение функционала библиотеки может повлиять на другие программы, которые ее используют, и программа не будет работать на другой системе, если там нет нужной библиотеки. В целом, преимущества динамических библиотек перевешивают недостатки, и они широко используются в современных программах. Однако в некоторых случаях статические библиотеки могут быть более подходящим выбором.