Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу

«Операционные системы»

ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ

Студент: Филиппов Владимир Михайлович
Группа: М8О–210Б–22
Вариант: 22
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023.

Постановка задачи

Цель работы

Целью является приобретение практических навыков в:

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

Задание

Требуется создать динамическую библиотеку, которая реализует определенный функционал. Далее использовать данную библиотеку 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов

В конечном итоге, программа должна состоять из следующих частей:

- Динамическая библиотека, реализующая заданных вариантом интерфейс;
- Тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции;
- Тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс.

Провести анализ между обоими типами использования библиотеки.

4	Подсчёт наибольшего общего делителя для двух натуральных чисел	Int GCF(int A, int B)	Алгоритм Евклида	Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.
5	Рассчет значения числа Пи при заданной длине ряда (K)	float Pi(int K)	Ряд Лейбница	Формула Валлиса

Общие сведения о программе

В программе используются следующие системные вызовы:

1. **dlopen** – загружает динамический общий объект (общую библиотеку) из файла, имя которого указано в строке filename (завершается null) и возвращает непрозрачный описатель на загруженный объект.

2. **dlsym** – функция возвращает адрес, по которому символ расположен в памяти(указывается одним из аргументов).
3. **dlclose** – уменьшает счётчик ссылок на динамически загружаемый общий объект, на который ссылается handle. Если счётчик ссылок достигает нуля, то объект выгружается. Все общие объекты, которые были автоматически загружены при вызове dlopen() для объекта, на который ссылается handle, рекурсивно закрываются таким же способом.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы dlsym, dlopen, dlclose.
2. Написать две реализации функций из задания.
3. Организовать простейший командный интерфейс в файлах static-main.cpp и dynamic-main.cpp
4. В файле static-main.cpp подключить библиотеку на этапе компиляции.
5. В файле dynamic-main.cpp загрузить библиотечные функции в runtime, с помощью dlsym, dlopen, dlclose.

Основные файлы программы

dynamic-main.cpp:

```
#include <dlfcn.h>
#include <stdlib.h>
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
void* libHandle = NULL;
float (*Pi)(int) = NULL;
int (*GCF)(int, int) = NULL;
using ptr_pi = float (*)(int);
using ptr_gcf = int (*)(int, int);
namespace Commands {
   const int Contract = 0;
   const int PI = 1;
   const int GCF = 2;
```

```
    const int EXIT = -1;
}
namespace Mode {
    const bool FIRST = 0;
    const bool SECOND = 1;
}
namespace LibsPath {
    const char * LIB1 = "../build/liblib1.so";
    const char * LIB2 = "../build/liblib2.so";
}
bool mode = Mode::SECOND;
void tokenize(std::vector<int> &args, const std::string &input) {
    std::stringstream ss(input);
    std::string s;
    while (std::getline(ss, s, ' ')) {
        args.push_back(std::stoi(s));
    }
}
void messageToUser() {
    std::cout << "~ Hello, world! ~" << std::endl;
    std::cout << "~ You in a dynamic-main.cpp ~" << std::endl;
    std::cout << "~ You can use 4 commands ~" << std::endl;
    std::cout << "~ Enter 0 to change contract ~" << std::endl;
    std::cout << "~ Enter 1 and the number - the length of the series (calculation
accuracy) to find the approximate value of Pi ~" << std::endl;
    std::cout << "~ Enter 2 and two numbers to find GCF of them ~" << std::endl;
    std::cout << "~ Enter -1 to stop programm ~" << std::endl;
}
void loadDynamicLib(const bool contract) {
    switch (contract) {
        case Mode::FIRST: {
                libHandle = dlopen(LibsPath::LIB1, RTLD_LAZY);
                break;
        }
        case Mode::SECOND:{
                libHandle = dlopen(LibsPath::LIB2, RTLD_LAZY);
                break;
        }
    }
    if (libHandle == NULL){
            perror("dlopen");
            exit(EXIT_FAILURE);
```

```
    }
}
void loadContract() {
    loadDynamicLib(mode);
    Pi = (ptr_pi)dlsym(libHandle, "Pi");
    GCF = (ptr_gcf)dlsym(libHandle, "GCF");
}
void changeContract(){
    dlclose(libHandle);
    switch (mode) {
            case Mode::FIRST:{
                    mode = Mode::SECOND;
                    break;
            }
            case Mode::SECOND:{
                mode = Mode::FIRST;
                break;
            }
    }
    loadContract();
}
int main() {
    loadContract();
    messageToUser();
    std::string input;
    std::vector<int> args;
    while (true) {
        std::getline(std::cin, input, '\n');
        tokenize(args, input);
        int cmd = args[0];
        if (args.size() > 3) {
            std::cout << "Wrong number of args" << std::endl;
            continue;
        }
        switch (cmd) {
            case Commands::PI: {
                std::cout << "Lenght of series = " << args[1] << " Pi = " << Pi(args[1])
<< std::endl;
                break;
            }
            case Commands::GCF: {
```

```
            std::cout << "Number 1 = " << args[1] << " Number 2 = " << args[2] <<
" GCF = " << GCF(args[1], args[2]) << std::endl;
            break;
        }
        case Commands::EXIT:
            return -1;
            break;
        case Commands::Contract:
            changeContract();
            std::cout << "Contract changed" << std::endl;
            break;
        default:
            std::cout << "Unknown command" << std::endl;
            break;
      }
      args.clear();
   }
   dlclose(libHandle);
}
```

static-main.cpp

```
extern "C" {
   #include "gcf-pi.h"
}
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
extern "C" {
   #include "gcf-pi.h"
}
namespace Commands {
   const int PI = 1;
   const int GCF = 2;
   const int EXIT = -1;
}
void tokenize(std::vector<int> &args, const std::string &input) {
   std::stringstream ss(input);
   std::string s;
   while (std::getline(ss, s, ' ')) {
      args.push_back(std::stoi(s));
   }
```

```
}
void messageToUser() {
    std::cout << "~ Hello, world! ~" << std::endl;
    std::cout << "~ You in a static-main.cpp ~" << std::endl;
    std::cout << "~ You can use 3 commands ~" << std::endl;
    std::cout << "~ Enter 1 and the number - the length of the series (calculation
accuracy) to find the approximate value of Pi ~" << std::endl;
    std::cout << "~ Enter 2 and two numbers to find GCF of them ~" << std::endl;
    std::cout << "~ Enter -1 to stop programm ~" << std::endl;
}
int main() {
    messageToUser();
    std::string input;
    std::vector<int> args;
    while (true) {
        std::getline(std::cin, input, '\n');
        tokenize(args, input);
        int cmd = args[0];
        if (args.size() > 3) {
            std::cout << "Wrong number of args" << std::endl;
            continue;
        }
        switch (cmd) {
            case Commands::PI:
                std::cout << "Lenght of series = " << args[1] << " Pi = " << Pi(args[1])
<< std::endl;
                break;
            case Commands::GCF:
                std::cout << "Number 1 = " << args[1] << " Number 2 = " << args[2] <<
" GCF = " << GCF(args[1], args[2]) << std::endl;
                break;
            case Commands::EXIT:
                return -1;
                break;
            default:
                std::cout << "Unknown command" << std::endl;
                break;
        }
        args.clear();
    }
}
```

gcf-pi.h

```
#ifndef _PI_CALC_
#define _PI_CALC_
#include <math.h>
#include <stdio.h>
float Pi(int k);
int GCF(int A, int B);
#endif
```

gcf-pi1.cpp:

```
#include "gcf-pi.h"
float Pi(int k) {
   float res = -1;
   for (int n = 0; n < k + 1; ++n) {
      res = res + (-pow(-1, n) / (2 * n - 1));
   }
   return 4 * res;
}
int GCF(int A, int B) {
   while (A != 0 && B != 0) {
      if (A > B) {
         A = A % B;
      } else {
         B = B % A;
      }
   }
   return A + B;
}
```

gcf-pi2.cpp:

```
#include "gcf-pi.h"
float Pi(int k) {
   float res = 1;
   for (int n = 1; n < k + 1; ++n) {
      res *= ((4 * pow(n, 2)) / (4 * pow(n, 2) - 1));
   }
   return 2 * res;
}
int GCF(int A, int B) {
   int del = 1;
   int max;
   while (del < A && del < B) {
      if (A % del == 0 && B % del == 0) {
```

```
            max = del;
        }
        ++del;
    }
    return max;
}
```

Пример работы

```
*[main][build]$ ./os_lab_4_static-main.exe
~ Hello, world! ~
~ You in a static-main.cpp ~
~ You can use 3 commands ~
~ Enter 1 and the number - the length of the series (calculation accuracy) to find the approximate value of Pi ~
~ Enter 2 and two numbers to find GCF of them ~
~ Enter -1 to stop programm ~
1 100
Lenght of series = 100 Pi = 3.13159
2 8 36
Number 1 = 8 Number 2 = 36 GCF = 4
-1
*[main][build]$ ./os_lab_4_dynamic-main.exe
~ Hello, world! ~
~ You in a dynamic-main.cpp ~
~ You can use 4 commands ~
~ Enter 0 to change contract ~
~ Enter 1 and the number - the length of the series (calculation accuracy) to find the approximate value of Pi ~
~ Enter 2 and two numbers to find GCF of them ~
~ Enter -1 to stop programm ~
1 100
Lenght of series = 100 Pi = 3.13379
2 100 1000
Number 1 = 100 Number 2 = 1000 GCF = 100
0
Contract changed
1 100
Lenght of series = 100 Pi = 3.13159
2 100 1000
Number 1 = 100 Number 2 = 1000 GCF = 100
-1
```

Вывод

В ходе выполнения Л.Р. я познакомился с созданием динамических библиотек в OS Linux, а также возможностью загружать эти библиотеки в ходе выполнения программы. Динамические библиотеки помогают уменьшить размер исполняемых файлов. Кроме того, я лучше разобрался как работают в том числе и статические библиотеки.

Думаю, что знание работы динамических библиотек полезно в промышленном коде, когда все стремятся уменьшить размер программ.