Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «Операционные системы»

Динамические библиотеки

Студент: В. М. Ватулин
Преподаватель: А. А. Соколов
Группа: М8О-206Б-19
Дата: 17.04.2021
Оценка:
Подпись:

Москва, 2021

1 Постановка задачи

Цель работы:
Приобретение практических навыков в:

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

Задание (вариант 28):
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, загрузив библиотеки в память с помощью системных вызовов

В конечном итоге, в лабораторной работе необходимо получить следующие части:

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа (программа No1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
- Тестовая программа (программа No2), 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 No2). 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 28

1. Рассчет значения числа Пи при заданной длине ряда
   Реализация 1 Ряд Лейбница
   Реализация 2 Формула Валлиса

2. Подсчет площади плоской геометрической фигуры по двум сторонам
   Реализация 1 Фигура прямоугольник
   Реализация 2 Фигура прямоугольный треугольник

2 Общие сведения о программе

Лабораторная состоит из двух программ. В первом случае мы подключаем библиотеку на этапе линковки, как мы обычно подключаем math библиотеку в Си. Во втором случае нам необходимо подключить shared библиотеку в рантайме при помощи системных вызовов dlopen, dlsym, dlclose. Для того, чтобы скомпилировать shared библиотеки, нам необходимо использовать особые ключи компилятора.

3 Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Изучить принципы создания и использования динамических библиотек
2. Написать функции библиотеки
3. Написать первую программу
4. Написать вторую программу, используя системные вызовы
5. Собрать проект
6. Написать тесты

4 Исходный код

Функции библиотек

pi_1.c

```
1
2  #include "pi.h"
3
4  float pi(int n) {
5      if (n < 1) {
6          return 0;
7      }
8      float res = 1;
9      int sign = -1;
10     for (int i = 1; i < n; ++i) {
11         res += sign * ((float) 1 / (1 + i*2));
12         sign *= -1;
13     }
14     return res * 4;
15 }
```

pi_2.c

```
1
2  #include "pi.h"
3
4  float pi(int n) {
5      if (n < 1) {
6          return 0;
7      }
8      float res = 1;
9      for (int i = 1; i <= n; ++i) {
10         float a = 2 * ((i + 1)/2);
11         float b = 1 + 2 * (i/2);
12         res *= a / b;
13     }
14     return res * 2;
15 }
```

square_1.c

```
1
2  #include "square.h"
3
4  float square(float a, float b) {
5      return a * b;
6  }
```

square_2.c

```
1
2  #include "square.h"
3
4  float square(float a, float b) {
5      return a * b / 2;
6  }
```

Первая программа

main_link.c

```
1
2  #include <stdio.h>
3  #include <string.h>
4  #include <ctype.h>
5  #include <stdlib.h>
6
7  #include "pi.h"
8  #include "square.h"
9
10 #define MAX_INPUT_LEN 256
11
12 int main() {
13     char input[MAX_INPUT_LEN + 1];
14     while (fgets(input, MAX_INPUT_LEN + 1, stdin) != NULL) {
15         char dup[MAX_INPUT_LEN + 2];
16         strcpy(dup + 1, input);
17         dup[0] = '\0';
18         size_t len = strlen(input);
19         int argc = 0;
20         for (size_t i = 1; i < len; ++i) {
21             if (!isspace(dup[i]) && isspace(dup[i+1])) {
22                 ++argc;
23             }
24         }
25         char *argv[argc];
26         int cur = 0;
27         for (size_t i = 1; i < len + 1; ++i) {
28             if (isspace(dup[i])) {
29                 dup[i] = '\0';
30             }
31             else if (dup[i-1] == '\0') {
32                 argv[cur] = &dup[i];
33                 ++cur;
34             }
35         }
36         if (argc == 0) {
37             continue;
38         }
```

```
39          if (strcmp(argv[0], "1") == 0) {
40              if (argc != 2) {
41                  fprintf(stderr, "bad arguments\n");
42                  continue;
43              }
44              int n = atoi(argv[1]);
45              printf("pi = %f\n", pi(n));
46          }
47          else if (strcmp(argv[0], "2") == 0) {
48              if (argc != 3) {
49                  fprintf(stderr, "bad arguments\n");
50                  continue;
51              }
52              float a = atof(argv[1]);
53              float b = atof(argv[2]);
54              printf("square = %f\n", square(a, b));
55          }
56          else {
57              fprintf(stderr, "no such command\n");
58          }
59      }
60
61      return 0;
62  }
```

Вторая программа

main_runt.c

```c

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <dlfcn.h>

#define MAX_INPUT_LEN 256

int main() {
    int cur_impl = 1;
    void *pi_1_h = dlopen("./bin/libpi_1.so", RTLD_NOW);
    void *pi_2_h = dlopen("./bin/libpi_2.so", RTLD_NOW);
    void *square_1_h = dlopen("./bin/libsquare_1.so", RTLD_NOW);
    void *square_2_h = dlopen("./bin/libsquare_2.so", RTLD_NOW);
    if (pi_1_h == NULL || pi_2_h == NULL ||
        square_1_h == NULL || square_2_h == NULL)
    {
        perror("error");
        exit(1);
    }
    float (*pi)(int) = dlsym(pi_1_h, "pi");
    float (*square)(float, float) = dlsym(square_1_h, "square");
    char input[MAX_INPUT_LEN + 1];
    while (fgets(input, MAX_INPUT_LEN + 1, stdin) != NULL) {
        char dup[MAX_INPUT_LEN + 2];
        strcpy(dup + 1, input);
        dup[0] = '\0';
        size_t len = strlen(input);
        int argc = 0;
        for (size_t i = 1; i < len; ++i) {
            if (!isspace(dup[i]) && isspace(dup[i+1])) {
                ++argc;
            }
        }
        char *argv[argc];
        int cur = 0;
        for (size_t i = 1; i < len + 1; ++i) {
            if (isspace(dup[i])) {
                dup[i] = '\0';
            }
            else if (dup[i-1] == '\0') {
                argv[cur] = &dup[i];
                ++cur;
            }
        }
```

```
47          if (argc == 0) {
48              continue;
49          }
50          if (strcmp(argv[0], "0") == 0) {
51              if (argc != 1) {
52                  fprintf(stderr, "bad arguments\n");
53                  continue;
54              }
55              if (cur_impl == 1) {
56                  pi = dlsym(pi_2_h, "pi");
57                  square = dlsym(square_2_h, "square");
58                  cur_impl = 2;
59              }
60              else {
61                  pi = dlsym(pi_1_h, "pi");
62                  square = dlsym(square_1_h, "square");
63                  cur_impl = 1;
64              }
65              printf("implementation changed to %d\n", cur_impl);
66          }
67          else if (strcmp(argv[0], "1") == 0) {
68              if (argc != 2) {
69                  fprintf(stderr, "bad arguments\n");
70                  continue;
71              }
72              int n = atoi(argv[1]);
73              printf("pi = %f\n", pi(n));
74          }
75          else if (strcmp(argv[0], "2") == 0) {
76              if (argc != 3) {
77                  fprintf(stderr, "bad arguments\n");
78                  continue;
79              }
80              float a = atof(argv[1]);
81              float b = atof(argv[2]);
82              printf("square = %f\n", square(a, b));
83          }
84          else {
85              fprintf(stderr, "no such command\n");
86          }
87      }
88      dlclose(pi_1_h);
89      dlclose(pi_2_h);
90      dlclose(square_1_h);
91      dlclose(square_2_h);
92
93      return 0;
94  }
```

5 Пример работы

Продемонстрирую процесс сборки программ:

```
eri412@Eri-PC:~/Desktop/study/OS/OSlab5$ make
gcc -Wall -Wextra -c -I./include src/main_runt.c -o bin/main_runt.o
gcc -Wall -Wextra -c -fpic -I./include src/pi_1.c -o bin/pi_1.o
gcc -shared bin/pi_1.o -o bin/libpi_1.so
gcc -Wall -Wextra -c -fpic -I./include src/square_1.c -o bin/square_1.o
gcc -shared bin/square_1.o -o bin/libsquare_1.so
gcc -Wall -Wextra -c -fpic -I./include src/pi_2.c -o bin/pi_2.o
gcc -shared bin/pi_2.o -o bin/libpi_2.so
gcc -Wall -Wextra -c -fpic -I./include src/square_2.c -o bin/square_2.o
gcc -shared bin/square_2.o -o bin/libsquare_2.so
gcc bin/main_runt.o -o main_runt -ldl
gcc -Wall -Wextra -c -I./include src/main_link.c -o bin/main_link.o
gcc -L./bin -Wl,-rpath=./bin bin/main_link.o bin/libpi_1.so bin/libsquare_1.so
-o main_link -lpi_1 -lsquare_1
```

Работа первой программы:

```
eri412@Eri-PC:~/Desktop/study/OS/OSlab5$ ./main_link
0
no such command
1 1
pi = 4.000000
1 10
pi = 3.041840
1 30
pi = 3.108268
1 100
pi = 3.131593
2 2 3
square = 6.000000
2 1.5 3
square = 4.500000
```

Работа второй программы:

```
eri412@Eri-PC:~/Desktop/study/OS/OSlab5$ ./main_runt
1 1
pi = 4.000000
1 10
pi = 3.041840
1 30
pi = 3.108268
1 100
pi = 3.131593
2 2 3
square = 6.000000
2 1.5 3
square = 4.500000
0
implementation changed to 2
1 1
pi = 4.000000
1 10
pi = 3.002177
1 30
pi = 3.091339
1 100
pi = 3.126081
2 2 3
square = 3.000000
2 1.5 3
square = 2.250000
```

6 Вывод

В процессе работы над лабораторной я научился основам работы с динамическими библиотеками в Си. Это важный и мощный инструмент в разработке ПО, поскольку позволяет экономить память, а также подгружать библиотеки динамически. Так же динамические библиотеки важны, потому что при фиксе проблемы в динамической библиотеке, эта проблема исчезнет во всех программах, которые подгружают ее.