

**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: П. Ф. Гришин
Преподаватель: Е. С. Миронов
Группа: М8О-201Б-21
Вариант: 4
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:

- Расчет интеграла функции $\sin(x)$ на отрезке $[A, B]$ с шагом e
- Расчет значения числа Π при заданной длине ряда (K)

2 Сведения о программе

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

Контракты описаны в файле func.h, а реализация lib1.c и lib2.c.

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 координаты левого и правого отрезка, интеграл функции которого хотим найти, а также шаг e , и находим интеграл либо методом прямоугольников, либо методом трапеций.
- В случае команды 2, мы считываем число - порядок ряда. Находим его либо с помощью ряда Лейбница, либо с помощью формулы Валлиса.
- 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

`functions.h`

```
1 | #pragma once
2 |
3 | float Square(float a, float b);
4 | char* Translation(long x);
```

`lib1.c`

```
1 | double Integrate(double a, double b, double epsilon)
2 | {
3 |     int steps = fabs(b - a) / epsilon;
4 |     double point = a;
5 |     double result = 0;
6 |
7 |     for (int i = 0; i < steps; ++i)
8 |     {
9 |         result += sin(point) * epsilon;
10 |        point += epsilon;
```

```

11     }
12
13     return result;
14 }
15
16 double Pi(int K) {
17     double pi = 0;
18     for(int i = 0; i <= K; i++) {
19         pi += 2.0/((4*i + 1)*(4*i + 3));
20     }
21     return pi * 4.0;
22 }

```

lib2.c

```

1 double Integrate(double a, double b, double epsilon)
2 {
3     int steps = fabs(b - a) / epsilon;
4     double point = a;
5     double result = 0;
6
7     for (int i = 0; i < steps; ++i)
8     {
9         result += sin(point + epsilon / 2) * epsilon;
10        point += epsilon;
11    }
12
13    return result;
14 }
15
16 double Pi(int K) {
17     double pi = 1;
18     for (int i = 1; i <= K; i++) {
19         pi *= (double)4*i*i/(4*i*i-1);
20     }
21     return (double)pi * 2;
22 }

```

main_dynamic.c

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <dlfcn.h>
4 #include <math.h>
5 #include <stdbool.h>
6
7 const char LIB1[] = "./libd1_dynamic.so";
8 const char LIB2[] = "./libd2_dynamic.so";
9
10 int main(int argc, char* argv[]) {
11     void *library;

```

```

12     bool flag = false;
13     int x, K;
14     double a, b, e;
15
16     library = dlopen(LIB2, RTLD_LAZY);
17     if (!library) {
18         printf("Error dlopen(): %s\n", dlerror());
19         return 1;
20     }
21
22     double(*Integrate)(double a, double b, double e);
23     double(*Pi)(int K);
24     *(void**)(&Integrate) = dlsym(library, "Integrate");
25     *(void**)(&Pi) = dlsym(library, "Pi");
26
27     for (;;) {
28         scanf("%d", &x);
29         if (x == 0) {
30             dlclose(library);
31             if (flag) {
32                 library = dlopen(LIB2, RTLD_LAZY);
33                 flag = false;
34             } else {
35                 library = dlopen(LIB1, RTLD_LAZY);
36                 flag = true;
37             }
38             if (!library) {
39                 printf("Error dlopen(): %s\n", dlerror());
40                 return 1;
41             }
42             *(void**)(&Integrate) = dlsym(library, "Integrate");
43             *(void**)(&Pi) = dlsym(library, "Pi");
44         } else if (x == 1) {
45             scanf("%lf %lf %lf", &a, &b, &e);
46             printf("Result: ");
47             double res = Integrate(a, b, e);
48             printf("%lf\n", res);
49         } else if (x == 2) {
50             scanf("%d", &K);
51             printf("Result: ");
52             double res = Pi(K);
53             printf("%lf\n", res);
54         } else {
55             dlclose(library);
56             return 0;
57         }
58     }
59     return 0;
60 }

```

main_static.c

```
1 | #include "include/func.h"
2 | #include <stdio.h>
3 |
4 | int main(int argc, char* argv[]) {
5 |     int x;
6 |     double a, b, e;
7 |     int K;
8 |
9 |     for (;;) {
10 |         scanf("%d", &x);
11 |         if (x == 1) {
12 |             scanf("%lf %lf %lf", &a, &b, &e);
13 |             double res = Integrate(a, b, e);
14 |             printf("%lf\n", res);
15 |             x = 0;
16 |         } else if (x == 2) {
17 |             scanf("%d", &K);
18 |             double res = Pi(K);
19 |             printf("%lf\n", res);
20 |             x = 0;
21 |         } else {
22 |             break;
23 |         }
24 |     }
25 |     return 0;
26 | }
```

CMakeList.txt

```
1 | add_library(d1_static STATIC src/lib1.c include/func.h)
2 | add_library(d2_static STATIC src/lib2.c include/func.h)
3 | add_library(d1_dynamic SHARED src/lib1.c include/func.h)
4 | add_library(d2_dynamic SHARED src/lib2.c include/func.h)
5 |
6 | # target_include_directories(d1_static PRIVATE include)
7 | # target_include_directories(d2_static PRIVATE include)
8 | # target_include_directories(d1_dynamic PRIVATE include)
9 | # target_include_directories(d2_dynamic PRIVATE include)
10 |
11 | add_executable(main_static_1 main_static.c)
12 | add_executable(main_static_2 main_static.c)
13 | add_executable(main_dynamic main_dynamic.c)
14 |
15 | target_include_directories(main_static_1 PRIVATE include)
16 | target_include_directories(main_static_2 PRIVATE include)
17 |
18 | target_link_libraries(main_dynamic ${CMAKE_DL_LIBS})
19 | target_link_libraries(main_static_1 PRIVATE d1_static)
20 | target_link_libraries(main_static_2 PRIVATE d2_static)
```

```
21 target_include_directories(main_dynamic PRIVATE include)
22
23 find_library(MATH_LIBRARY m)
24 if(MATH_LIBRARY)
25     target_link_libraries(d1_dynamic PUBLIC ${MATH_LIBRARY})
26     target_link_libraries(d2_dynamic PUBLIC ${MATH_LIBRARY})
27     target_link_libraries(main_static_1 PUBLIC ${MATH_LIBRARY})
28     target_link_libraries(main_static_2 PUBLIC ${MATH_LIBRARY})
29     target_link_libraries(main_dynamic PUBLIC ${MATH_LIBRARY})
30 endif()
```

5 Демонстрация работы программы

```
gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/lab5$ ./main_static_1
1
0
3.14
0.1
1.995390
2
10
3.096162
~C
gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/lab5$ ./main_static_2
1
0
3.14
0.1
1.999968
2
10
3.067704
~C
gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/lab5$ ./main_dynamic
1
0
3.14
0.1
Result: 1.999968
2
10
Result: 3.067704
0
1
0
3.14
0.1
Result: 1.995390
2
10
Result: 3.096162
~C
```



```
gpavel@gpavel-HP-Pavilion-Gaming-Laptop-17-cd1xxx:~/Desktop/OS/lab5$
```

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 подключала динамические библиотеки на этапе компиляции, а вторая во время исполнения.

Динамические библиотеки содержат функционал отдельно от программы и передают 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. Вторым, 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 динамических библиотек исчерпывают их минусы в большинстве 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 большинстве современных программ.