

**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: С. А. Арапов  
Преподаватель: Е. С. Миронов  
Группа: М8О-208Б-19  
Дата: Оценка:  
Подпись:

**Москва, 2021**

## Лабораторная работа №5

**Тема:** Целью является приобретение практических навыков в:

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

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

- Во время компиляции (на этапе «линковки»/linking)
- 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 двух типов использования библиотек.

Пользовательский ввод организован следующим образом:

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

**Вариант №15:**

- Функция 1: расчет производной функции  $\cos(x)$  в точке  $A$  с приращением  $\delta X$ .
  - Реализация 1:  $f'(a) = (f(a + \delta X) - f(a)) / \delta X$ .
  - Реализация 2:  $f'(a) = (f(a + \delta X) - f(a - \delta X)) / 2 * \delta X$ .
- Функция 2: отсортировать целочисленный массив.
  - Реализация 1: Пузырьковая сортировка.
  - Реализация 2: Сортировка Хоара.

# 1 Описание программы

Динамические библиотеки компилируются из файлов `liblab_first.c`, `liblab_second.c`. Основная программа компилируется из файлов `prog1.c`, `prog2.c`. Внутри исходных файлов используется заголовочные файлы: `stdio.h`, `stdlib.h`, `math.h`, `dlfcn.h`, `liblab.h`.

# 2 Алгоритм решения

Для реализации поставленной задачи необходимо: Изучить принципы работы `dlsym`, `dlopen`, `dlclose`. Написать заголовочный файл с объявлениями экспортируемых символов из наших библиотек `liblab.h`. Написать две реализации библиотеки: `liblab_first.c` и `liblab_second.c`. Организовать простейший командный интерфейс в файлах `prog1.c` и `prog2.c`. В файле `prog1.c` подключить первую реализацию библиотеки на этапе компиляции и статического связывания. В файле `prog2.c` загрузить библиотечные функции в процессе выполнения, с помощью библиотеки `dl`. Организовать возможность переключаться между реализациями библиотеки.

# 3 Исходный код

`prog1.c`

```
1 | #include "../inc/liblab.h"
2 |
3 | #include <stdio.h>
4 | #include <stdlib.h>
5 |
6 | void printArray(int* a, unsigned int s) {
7 |     for (unsigned int i = 0; i < s; ++i) {
8 |         printf("%d ", a[i]);
9 |     }
10 | }
11 |
12 | void initArray(int* a, unsigned int s) {
13 |     for (unsigned int i = 0; i < s; ++i) {
14 |         a[i] = rand() % 100;
15 |     }
16 | }
17 |
18 | int main() {
19 |
20 |     printf("prog1.c: Shared library - Dynamic Linking\n");
21 |     printName();
22 |
23 |     int cmd = 0;
```

```

24
25     int array[20];
26     const unsigned int SIZE = 20;
27     initArray(array, SIZE);
28
29     float A = 0.0f;
30     float deltaX = 0.0f;
31
32     while (scanf("%d", &cmd) != EOF) {
33         switch (cmd) {
34             case 0:
35                 printf("Not supported for Dynamic linking\n");
36                 break;
37             case 1:
38                 scanf("%f %f", &A, &deltaX);
39                 printf("Calling: float Derivative(float A, float deltaX);\n\tresult: %f\n",
40                     Derivative(A, deltaX));
41                 break;
42             case 2:
43                 printf("Array: ");
44                 printArray(array, SIZE);
45                 printf("\n");
46                 Sort(array, SIZE);
47                 printf("Calling: int* Sort(int* array, unsigned int size);\n\tresult: ");
48                 printArray(array, SIZE);
49                 printf("\n");
50                 initArray(array, SIZE);
51                 break;
52         }
53     }
54     return 0;
55 }

```

prog2.c

```

1  #include "../inc/liblab.h"
2
3  #include <dlfcn.h>
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  void printArray(int* a, unsigned int s) {
9      for (unsigned int i = 0; i < s; ++i) {
10         printf("%d ", a[i]);
11     }
12 }
13
14 void initArray(int* a, unsigned int s) {
15     for (unsigned int i = 0; i < s; ++i) {

```

```

16     a[i] = rand() % 100;
17 }
18 }
19
20 const char libname_first[] = "./lib/liblab_first.so";
21 const char libname_second[] = "./lib/liblab_second.so";
22
23 char* error = NULL;
24
25 void get_function(void* dl_hdl, const char* name, void** func_ptr) {
26     *func_ptr = dlsym(dl_hdl, name);
27     error = dlerror();
28     if (error != NULL) {
29         printf("dlsym() %s\n", error);
30         exit(13);
31     }
32 }
33
34 void get_handle(const char* libname, void** dl_hdl) {
35     *dl_hdl = dlopen(libname, RTLD_LAZY | RTLD_LOCAL);
36     if (*dl_hdl == NULL) {
37         printf("dlopen() error %s\n", dlerror());
38         exit(12);
39     }
40 }
41
42 #define SIZE 20
43
44 int main() {
45
46     printf("prog1.c: Shared library - Dynamic Loading\n");
47
48     float (*Derivative_ptr)(float, float) = NULL;
49     int* (*Sort_ptr)(int*, unsigned int) = NULL;
50     void* (*printName_ptr)() = NULL;
51
52     void* dl_handles[2];
53     get_handle(libname_first, &dl_handles[0]);
54     printf("god handle 1: %p\n", dl_handles[0]);
55     get_handle(libname_second, &dl_handles[1]);
56     printf("god handle 2: %p\n", dl_handles[1]);
57
58     int cmd = 0;
59     int curr = 0;
60
61     get_function(dl_handles[curr], "Derivative", (void**)&Derivative_ptr);
62     get_function(dl_handles[curr], "Sort", (void**)&Sort_ptr);
63     get_function(dl_handles[curr], "printName", (void**)&printName_ptr);
64

```

```

65     printName_ptr();
66
67     int array[SIZE];
68     initArray(array, SIZE);
69
70     float A = 0.0f;
71     float deltaX = 0.0f;
72
73     while (scanf("%d", &cmd) != EOF) {
74         switch (cmd) {
75             case 0:
76                 printf("Supported for Dynamic loading. Switching...\n");
77
78                 curr ^= 1;
79
80                 get_function(dl_handles[curr], "Derivative", (void**>(&Derivative_ptr));
81                 get_function(dl_handles[curr], "Sort", (void**>(&Sort_ptr));
82                 get_function(dl_handles[curr], "printName", (void**>(&printName_ptr));
83
84                 printName_ptr();
85
86                 break;
87             case 1:
88                 scanf("%f %f", &A, &deltaX);
89                 printf("Calling: float Derivative(float A, float deltaX);\n\tresult: %f\n",
90                     Derivative_ptr(A, deltaX));
91                 break;
92             case 2:
93                 printf("Array: ");
94                 printArray(array, SIZE);
95                 printf("\n");
96                 Sort_ptr(array, SIZE);
97                 printf("Calling: int* Sort(int* array, unsigned int size);\n\tresult: ");
98                 printArray(array, SIZE);
99                 printf("\n");
100                initArray(array, SIZE);
101                break;
102        }
103    }
104
105    if (dlclose(dl_handles[0])
106        || dlclose(dl_handles[1])) {
107        exit(15);
108    }
109    return 0;
110 }

```

liblab.h

```

1 || extern void printName();

```

```

2 |
3 | extern float Derivative(float A, float deltaX);
4 |
5 | extern int* Sort(int* array, unsigned int size);
6 |
liblab_first.c
7 | #include <stdio.h>
8 | #include <math.h>
9 |
10 | void printName() {
11 |     printf("liblab_first.c\n");
12 | }
13 | //      cos(x)      A      deltaX
14 | extern float Derivative(float A, float deltaX) {
15 |     if (deltaX == 0.0) {
16 |         printf("NAN\n");
17 |         return NAN;
18 |     }
19 |     return (cos(A + deltaX) - cos(A)) / deltaX;
20 | }
21 |
22 | //
23 | extern int* Sort(int* array, unsigned int size) {
24 |     for (int i = 0; i < size; ++i) {
25 |         for (int j = 0; j < size - i - 1; ++j) {
26 |             if (array[j + 1] < array[j]) {
27 |                 int tmp = array[j + 1];
28 |                 array[j + 1] = array[j];
29 |                 array[j] = tmp;
30 |             }
31 |         }
32 |     }
33 |     return array;
34 | }
35 |
liblab_second.c
36 | #include <stdio.h>
37 | #include <math.h>
38 |
39 | void printName() {
40 |     printf("liblab_second.c\n");
41 | }
42 | //f'(x) = (f(A + deltaX) - f(A - deltaX)) / (2*deltaX)
43 | //      cos(x)      A      deltaX
44 | extern float Derivative(float A, float deltaX) {
45 |     if (deltaX == 0.0) {
46 |         printf("NAN\n");

```

```

47     return NAN;
48 }
49 return (cos(A + deltaX) - cos(A - deltaX)) / (2 * deltaX);
50 }
51
52 unsigned int partition(int* a, unsigned int l, unsigned int r) {
53     int p = (r + l) / 2;
54     unsigned int i = l;
55     unsigned int j = r;
56     while (i < j) {
57         while (a[i] <= a[p] && i < r) {
58             i++;
59         }
60         while (a[j] > a[p]) {
61             j--;
62         }
63         if (i >= j)
64             break;
65         int tmp = a[i];
66         a[i] = a[j];
67         a[j] = tmp;
68     }
69     int tmp = a[p];
70     a[p] = a[j];
71     a[j] = tmp;
72     return j;
73 }
74
75 int* quicksort(int* a, unsigned int l, unsigned int r) {
76     if (l < r) {
77         unsigned int p = partition(a, l, r);
78         quicksort(a, l, p - 1);
79         quicksort(a, p + 1, r);
80     }
81     return a;
82 }
83
84 //
85 extern int* Sort(int* array, unsigned int size) {
86     return quicksort(array, 0, size - 1);
87 }

```

makefile

```

1 CC=gcc
2 CFLAGS=-pedantic -Wall -Werror -g3
3 LFLAGS=-L ./lib -llab_first
4 LDFLAGS=-Wl,-rpath,./lib
5
6 all: liblab_first liblab_second prog1 prog2

```



```

7 |
8 | liblab_first:
9 |     $(CC) $(CFLAGS) -shared -fPIC ./lib/src/liblab_first.c -o ./lib/liblab_first.so -lm
10 |
11 | liblab_second:
12 |     $(CC) $(CFLAGS) -shared -fPIC ./lib/src/liblab_second.c -o ./lib/liblab_second.so -
    lm
13 |
14 | prog1:
15 |     $(CC) $(CFLAGS) ./src/prog1.c -o program_1 $(LFLAGS) $(LDFLAGS)
16 |
17 | prog2:
18 |     $(CC) $(CFLAGS) ./src/prog2.c -o program_2 -ldl $(LDFLAGS)
19 |
20 | clean:
21 |     rm ./obj/*.o ./bin/*.out

```

## 4 Запуск программы и демонстрация работы

```

ogrocket@LAPTOP-ADULJM7A:/mnt/d/OS/OS/os_lab5$ make
gcc -pedantic -Wall -Werror -g3 -shared -fPIC
./lib/src/liblab_first.c -o ./lib/liblab_first.so -lm
gcc -pedantic -Wall -Werror -g3 -shared -fPIC ./lib/src/liblab_second.c -o
./lib/liblab_second.so -lm
gcc -pedantic -Wall -Werror -g3 ./src/prog1.c -o program_1 -L ./lib -llab_first
-Wl,-rpath,./lib
gcc -pedantic -Wall -Werror -g3 ./src/prog2.c -o program_2 -ldl -Wl,-rpath,./lib
ogrocket@LAPTOP-ADULJM7A:/mnt/d/OS/OS/os_lab5$ ./program_1
prog1.c: Shared library -Dynamic Linking
liblab_first.c
0
Not supported for Dynamic linking
2
Array: 83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36
Calling: int* Sort(int* array,unsigned int size);
result: 15 21 26 26 27 35 36 40 49 59 62 63 72 77 83 86 86 90 92 93
2
Array: 11 68 67 29 82 30 62 23 67 35 29 2 22 58 69 67 93 56 11 42
Calling: int* Sort(int* array,unsigned int size);
result: 2 11 11 22 23 29 29 30 35 42 56 58 62 67 67 67 68 69 82 93
2
Array: 29 73 21 19 84 37 98 24 15 70 13 26 91 80 56 73 62 70 96 81

```

```

Calling: int* Sort(int* array,unsigned int size);
result: 13 15 19 21 24 26 29 37 56 62 70 70 73 73 80 81 84 91 96 98
2
Array: 5 25 84 27 36 5 46 29 13 57 24 95 82 45 14 67 34 64 43 50
Calling: int* Sort(int* array,unsigned int size);
result: 5 5 13 14 24 25 27 29 34 36 43 45 46 50 57 64 67 82 84 95
1 0.3 0.01
Calling: float Derivative(float A,float deltaX);
result: -0.300292
1 -0.3 0.01
Calling: float Derivative(float A,float deltaX);
result: 0.290738
1 2 -0.04
Calling: float Derivative(float A,float deltaX);
result: -0.917376
1 222 111
Calling: float Derivative(float A,float deltaX);
result: 0.013467

ogrocket@LAPTOP-ADULJM7A:/mnt/d/OS/OS/os_lab5$ ./program_2 <test2
prog1.c: Shared library -Dynamic Loading
god handle 1: 0x7fffd97ae2d0
god handle 2: 0x7fffd97af030
liblab_first.c
Supported for Dynamic loading. Switching...
liblab_second.c
Calling: float Derivative(float A,float deltaX);
result: -0.999982
Calling: float Derivative(float A,float deltaX);
result: 0.278745
Calling: float Derivative(float A,float deltaX);
result: -0.001589
Array: 83 86 77 15 93 35 86 92 49 21 62 27 90 59 63 26 40 26 72 36
Calling: int* Sort(int* array,unsigned int size);
result: 15 21 77 83 35 49 86 86 92 93 26 26 27 36 40 59 62 63 72 90
Supported for Dynamic loading. Switching...
liblab_first.c
Calling: float Derivative(float A,float deltaX);
result: -0.367600
Calling: float Derivative(float A,float deltaX);
result: -0.956449

```

```
Calling: float Derivative(float A,float deltaX);  
result: -0.071473  
Array: 11 68 67 29 82 30 62 23 67 35 29 2 22 58 69 67 93 56 11 42  
Calling: int* Sort(int* array,unsigned int size);  
result: 2 11 11 22 23 29 29 30 35 42 56 58 62 67 67 67 68 69 82 93  
Array: 29 73 21 19 84 37 98 24 15 70 13 26 91 80 56 73 62 70 96 81  
Calling: int* Sort(int* array,unsigned int size);  
result: 13 15 19 21 24 26 29 37 56 62 70 70 73 73 80 81 84 91 96 98
```

## 5 Выводы

Данная лабораторная работа была направлена на то, чтобы изучить динамические библиотеки в Unix подобных ОС. Я написал две программы. Первая подключает динамические библиотеки на этапе компиляции. Вторая подключает динамические библиотеки во время исполнения.

Динамические библиотеки имеют как свои плюсы так и минусы. Объем итоговой программы может быть меньше, также одну и ту же библиотеку можно использовать в нескольких программах, не встраивая в код. Однако вызов функции может происходить несколько медленнее.