**Московский авиационный институт**
**(национальный исследовательский университет)**

**Институт: «Информационные технологии и прикладная математика»**
**Кафедра: 806 «Вычислительная математика и программирование»**

# Лабораторная работа №5
# по курсу «Операционные системы»

# ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ

Студент: Лагуткина Мария Сергеевна
Группа: М8О-206Б-19
Преподаватель: Соколов А. А.
Дата:
Оценка:

**Москва, 2020**

# 1 Постановка задачи

**Цель работы**
Целью является приобретение практических навыков в:
• Создание динамических библиотек
• Создание программ, которые используют функции динамических библиотек
**Задание**
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 36.**

1. Перевод числа x из десятичной системы счисления в другую
   **Реализация 1** Другая система счисления двоичная
   **Реализация 2** Другая система счисления двоичная

2. Отсортировать целочисленный массив
   **Реализация 1** Пузырьковая сортировка
   **Реализация 2** Сортировка Хоара

# 2 Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Понять этапы и флаги компиляции динамических библиотек.
2. Изучить принципы работы функций dlopen, dlclose, dlsym и dlerror.
3. Подключить библиотеки, необходимые для работы с вышеперечисленными функциями и их аргументами.
4. Написать исходные файлы библиотек.
5. Написать файлы программ
6. Изучить принципы компиляции библиотек и программ, использующих их
7. Проверить работоспособность
8. Написать Makefile и собрать проект

Библиотеки являются динамическими, с позиционно-независимым кодом. Первая программа загружает библиотеки исходя из информации, полученной на этапе линковки, а вторая с помощью функции dlopen.

# 3 Основные файлы программы

**translation.h**

```
1
2 #ifndef TRANSLATION_H
3 #define TRANSLATION_H
4
5 extern char* Translation(long x);
6 extern void Sort(int* arr, const int size);
7
8 #endif
```

**translation.c**

```c
#include "translation.h"
#include <stddef.h>
#include <stdlib.h>

char* Translation(long x) {
    int size = 20;
    char tmp;
    char* res = (char*)calloc(size, sizeof(char));
    int i = 0;
    if (x == 0) {
        res[0] = '0';
        return res;
    }
    while (x != 0) {
        if (i > size) {
        char* new_res = (char*)calloc(i, sizeof(char));
        res = new_res;
        }
        if (x % 2 == 0) {
            res[i] = '0';
            x /= 2;
        }
        else {
            res[i] = '1';
            x /= 2;
        }
        i++;
    }
    for (int k = 0; k < i / 2; k++) {
        tmp = res[k];
        res[k] = res[i - k - 1];
        res[i - k - 1] = tmp;
    }
    return res;
}

void Swap(int* x, int* y) {
  int tmp = *x;
  *x = *y;
  *y = tmp;
}

void Sort(int* arr, const int size) {
  for (size_t i = 0; i < size; ++i) {
    for (size_t j = 1; j < size; ++j) {
      if (arr[j - 1] > arr[j]) {
        Swap(&arr[j - 1], &arr[j]);
      }
    }
```

```
  }
}
```

### static_main.c

```
#include "translation.h"
#include <stdio.h>
#include <stdlib.h>

int main() {
    int cmd = 0;
  printf("Usage:\n");
  printf("1 number - translation to 2 (or 3) base\n");
  printf("2 size a1 a2 ... - sort \n");

    while (scanf("%d", &cmd) != EOF) {
        if (cmd == 1) {
      long x;
            if((scanf("%ld", &x)== EOF)||(x<0)){
        printf("Error input\n");
                return 1;
            }
            char* string = Translation(x);
      printf("from 10 base to 2(or 3): ");
            printf("%s\n", string);
            free(string);
        }
    if (cmd == 2) {
            int size;
      if((scanf("%d", &size)<1)||(size<0)){
        printf("Error input\n");
        return 1;
      }
      int* a = malloc(sizeof(int) * size);
      for (size_t i = 0; i < size; ++i) {
        if(scanf("%d", &a[i])<1){
          printf("Error input\n");
          return 1;
        }
      }
      Sort(a, size);
      printf("sorted: ");
      for (size_t i = 0; i < size; ++i) {
        printf("%d ", a[i]);
      }
      printf("\n");
      free(a);
        }
    }
}
```

### dynamic_main.c

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <dlfcn.h>
5  #include <stddef.h>
6
7  const char* libName1 = "./libsort1.so";
8  const char* libName2 = "./libsort2.so";
9
10 char* (*Translation)(long x);
11 void (*Sort)(int*, const int);
12 char *err;
13
14 void *libHandle = NULL;
15
16 typedef enum {
17     FIRST,
18     SECOND,
19 } realization;
20
21 realization rel = FIRST;
22
23 void load_realization(){
24   const char *name;
25     if(rel == FIRST){
26         name = libName1;
27     } else{
28         name = libName2;
29     }
30     libHandle = dlopen(name, RTLD_LAZY);
31     if(!libHandle){
32         fprintf(stderr, "%s\n", dlerror());
33         exit(EXIT_FAILURE);
34     }
35
36   *(void **) (&Sort) = dlsym(libHandle, "Sort");
37     *(void **) (&Translation) = dlsym(libHandle, "Translation");
38
39     if((err = dlerror())) {
40         fprintf(stderr, "%s\n", err);
41         exit(EXIT_FAILURE);
42     }
43 }
44
45 void change_realization(){
46   dlclose(libHandle);
47     if(rel == FIRST){
48         rel = SECOND;
```

```
49      } else {
50          rel = FIRST;
51      }
52
53      load_realization();
54  }
55
56  int main(){
57      rel = FIRST;
58      load_realization();
59
60      int cmd = 0;
61    printf("Usage:\n");
62    printf("0 - change realization\n");
63    printf("1 number - translation to 2 (or 3) base\n");
64    printf("2 size a1 a2 ... - sort \n");
65      while (scanf("%d", &cmd) != EOF){
66
67          if(cmd == 0){
68              change_realization();
69              printf("Ok\n");
70              if(rel == FIRST){
71                  printf("now 1 realization is running\n");
72              } else{
73                  printf("now 2 realization is running\n");
74              }
75              continue;
76          }
77          if(cmd == 1) {
78              long x;
79              if((scanf("%ld", &x)== EOF)||(x<0)){
80          printf("Error input\n");
81                  return 1;
82              }
83              printf("Translate to ");
84              if(rel == FIRST) {
85                  printf("2 base: ");
86              }else{
87                  printf("3 base: ");
88              }
89              char* string = Translation(x);
90              printf("%s\n", string);
91              free(string);
92          }
93          else if(cmd == 2){
94        int size;
95        if((scanf("%d", &size) <1)||(size<0)){
96          printf("Incorrent input\n");
97          return 1;
```

```
98        }
99        int* arr = malloc(sizeof(int) * size);
100       for(int i = 0; i<size; i++){
101
102         if(scanf("%d", &arr[i])<1){
103           printf("Incorrent input");
104           return 1;
105         }
106       }
107       Sort(arr, size);
108       printf("Sorted: ");
109       for (int i = 0; i < size; ++i) {
110         printf("%d ", arr[i]);
111
112       }
113       printf("\n");
114       free(arr);
115         }
116     else{
117       printf("incorrect command\n");
118     }
119     }
120     dlclose(libHandle);
121 }
```

# 4 Пример работы

```
maria@DESKTOP-6CRUDOR:~/pro/os5_tmp$ ./solution_s1
Usage:
1 number -translation to 2 (or 3) base
2 size a1 a2 ... -sort
1 2341
from 10 base to 2(or 3): 100100100101
2
4 2 3 1 -7
sorted: -7 1 2 3
maria@DESKTOP-6CRUDOR:~/pro/os5_tmp$ ./solution
Usage:
0 -change realization
1 number -translation to 2 (or 3) base
2 size a1 a2 ... -sort
1 23
Translate to 2 base: 10111
2 5 1 -3 0 5 1
```

```
Sorted: -3 0 1 1 5
0
Ok
now 2 realization is running
1 23
Translate to 3 base: 212
2 5 1 -3 0 5 1
Sorted: -3 0 1 1 5
```

# 5 Вывод

В этой лабораторной я узнала про существование файлов dll, а также, что файлы представляет собой динамические библиотеки, об отличиях статических и динамических библиотек, сделала свои собственные динамические библиотеки и попробовал разные способы их загрузки.