Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №4 по курсу

«Операционные системы»

Группа: М8О-216БВ-24

Студент: Гуськов А.В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 04.12.25

Москва, 2025

Постановка задачи

Вариант 8.

Требуется создать две динамические библиотеки, реализующие один и тот же интерфейс (контракт), но с различными алгоритмами обработки данных.

Контракты и реализации:

1. **Расчет интеграла функции *sin(x)*** на отрезке *[A, B]* с шагом e.
   - Реализация №1: Метод прямоугольников.
   - Реализация №2: Метод трапеций.
2. **Сортировка целочисленного массива.**
   - Реализация №1: Пузырьковая сортировка.
   - Реализация №2: Сортировка Хоара (*Quicksort*).

Необходимо разработать две программы:

1. **Программа №1:** Использует одну из библиотек, связывание происходит на этапе компиляции (*Link-time*). Программа жестко зависит от наличия библиотеки при запуске.
2. **Программа №2:** Загружает библиотеки во время выполнения (*Runtime*) по их относительным путям. Программа должна поддерживать переключение между реализациями (библиотеками) по команде пользователя без перезапуска.

Взаимодействие с пользователем осуществляется через консоль. Ввод команд и аргументов обрабатывается вручную, ввод-вывод осуществляется через системные вызовы.

Общий метод и алгоритм решения

Использованные системные вызовы:

- *void* dlopen(const char* filename, int flags); – открывает динамическую библиотеку и возвращает дескриптор (handle).*
- *void* dlsym(void* handle, const char* symbol); – возвращает адрес символа (функции) в памяти загруженной библиотеки.*
- *int dlclose(void* handle); – уменьшает счетчик ссылок на библиотеку и выгружает её, если счетчик равен 0.*
- *char* dlerror(void); – возвращает текстовое описание последней ошибки, возникшей в функциях dl*.*

В рамках лабораторной работы были созданы два файла исходного кода библиотек (*lib1.c*, *lib2.c*), которые компилируются с флагами *-fPIC* (позиционно-независимый код) и *-shared* для создания динамических библиотек *libd1.so* и *libd2.so*.

Библиотеки:

- ***libd1.so*** реализует функцию *sin_integral* методом прямоугольников и функцию *sort* методом пузырьковой сортировки.
- ***libd2.so*** реализует *sin_integral* методом трапеций и *sort* алгоритмом быстрой сортировки (*Quicksort*).

Программа №1 (*prog1*):

Реализует статическую компоновку. При компиляции указывается путь к библиотеке (*-L.*) и её имя (*-ld1*). Для корректного запуска используется флаг линковщика *-Wl,-rpath,.*, указывающий загрузчику искать библиотеку в текущей директории. Программа вызывает функции библиотеки напрямую через заголовочный файл *libs.h*. Ввод данных парсится с помощью *strtok* и *atoi/atof*, вывод осуществляется через буфер с использованием *snprintf* и системного вызова *write*.

Программа №2 (*prog2*):

Реализует динамическую загрузку. Программа не слинкована с библиотеками при компиляции.

1. При запуске или по команде «0» программа вызывает *dlopen*() для загрузки соответствующего *.so* файла.
2. С помощью *dlsym*() программа получает указатели на функции *sin_integral* и sort.
3. Вызовы функций происходят через полученные указатели.
4. При переключении библиотек старая библиотека выгружается через *dlclose*(), и загружается новая.
5. Обработка ввода-вывода аналогична первой программе, но добавлена обработка ошибок загрузки библиотек (через *dlerror*).

Таким образом, продемонстрирована разница между жесткой привязкой библиотек (требует перекомпиляции для смены реализации) и гибкой загрузкой плагинов во время выполнения.

Код программы

libs.h

```
#ifndef LIBS_H

#define LIBS_H


#include <stddef.h>


float sin_integral(float a, float b, float e);

int* sort(int* array, size_t size);




#endif // LIBS_H
```

lib1.c

```
#include "libs.h"

#include <math.h>

#include <stddef.h>
```

```
float sin_integral(float a, float b, float e)
{
  float sum = 0.0;

  for (float x = a; x < b; x += e)
  {
    sum += sinf(x) * e;
  }

  return sum;
}


int* sort(int* array, size_t size)
{
  if (size < 2) return array;

  for (size_t i = 0; i < size - 1; ++i)
  {
    for (size_t j = 0; j < size - i - 1; ++j)
    {
      if (array[j] > array[j + 1])
      {
        int temp = array[j];
        array[j] = array[j + 1];
        array[j + 1] = temp;
      }
    }
```

```
    }

    return array;
}
```

lib2.c

```
#include "libs.h"
#include <math.h>


float sin_integral(float a, float b, float e)
{
    float sum = 0.0;

    for (float x = a; x < b; x += e)
    {
        sum += 0.5 * (sinf(x) + sinf(x + e)) * e;
    }

    return sum;
}


int partition(int* array, int start, int end)
{
    int pivot = array[end];
    int i = start - 1;

    for (int j = start; j <= end - 1; ++j)
```

```
  {
    if (array[j] < pivot)
    {
      i++;
      int temp = array[i];
      array[i] = array[j];
      array[j] = temp;
    }
  }
  i++;
  int temp = array[i];
  array[i] = array[end];
  array[end] = temp;

  return i;
}


void quick_sort(int* array, int start, int end)
{
  if (end <= start) return;

  int pivot = partition(array, start, end);
  quick_sort(array, start, pivot - 1);
  quick_sort(array, pivot + 1, end);
}


int* sort(int* array, size_t size)
{
```

```
  if (size > 0) quick_sort(array, 0, (int)size - 1);
  return array;
}
```

prog1.c

```
#include <unistd.h>
#include <stdio.h>      // snprintf
#include <stdlib.h>
#include <string.h>

#include "libs.h"



#define BUFFER_SIZE 4096



void command_1()
{
  char* arg1 = strtok(NULL, " \t\n");
  char* arg2 = strtok(NULL, " \t\n");
  char* arg3 = strtok(NULL, " \t\n");

  int len = 0;
  char buffer[BUFFER_SIZE];

  if (arg1 && arg2 && arg3)
  {
     float res = sin_integral(atof(arg1), atof(arg2), atof(arg3));
     len = snprintf(buffer, BUFFER_SIZE, "Integral result: %.5f\n", res);
```

```
        write(STDOUT_FILENO, buffer, len);
    }
}

void command_2()
{
    char* size_str = strtok(NULL, " \t\n");
    if (!size_str) return;

    int size = atoi(size_str);
    int* array = (int*)malloc(size * sizeof(int));
    if (!array)
    {
        const char* message = "Unable to malloc memory for int* array\n";
        write(STDERR_FILENO, message, strlen(message));
        return;
    }

    for (int i = 0; i < size; ++i)
    {
        char* value = strtok(NULL, " \t\n");
        if (value) array[i] = atoi(value);
        else array[i] = 0;
    }

    sort(array, size);

    {
        const char* message = "Sorted array: ";
        write(STDOUT_FILENO, message, strlen(message));
```

```
    }

    char buffer[BUFFER_SIZE];
    for (int i = 0; i < size; ++i)
    {
        int len = snprintf(buffer, BUFFER_SIZE, "%d ", array[i]);
        write(STDOUT_FILENO, buffer, len);
    }

    write(STDOUT_FILENO, "\n", 1);
    free(array);
}



int main()
{
    {
        const char* message = "Program 1 (Static).\nCommands: 1 a b e | 2 Size Array... | Ctrl + D to
exit\n> ";
        write(STDOUT_FILENO, message, strlen(message));
    }

    int bytes_read = 0;
    char buffer[BUFFER_SIZE];

    while((bytes_read = read(STDIN_FILENO, buffer, BUFFER_SIZE - 1)) > 0)
    {
        buffer[bytes_read] = 0;

        char* token = strtok(buffer, " \t\n");
```

```
        if (!token) continue;

        int cmd = atoi(token);
        switch (cmd)
        {
            case 1:
            {
                command_1();
                break;
            }
            case 2:
            {
                command_2();
                break;
            }
        }

        write(STDOUT_FILENO, "> ", 2);
    }

    return 0;
}
```

prog2.c

```
#include <stddef.h>
#include <unistd.h>
#include <stdio.h>      // snprintf
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>
```

```
#define BUFFER_SIZE 4096


typedef float (*sin_integral_func)(float, float, float);
typedef int* (*sort_func)(int*, size_t);


enum ErrorCode
{
  OK = 0,
  ER_DLOPEN = -1,
};

enum CurrentLib
{
  FIRST = 0,
  SECOND = 1,
};


enum ErrorCode command_0(const char** LIB_NAMES, void** library, int* current_lib,
                         sin_integral_func* sin_integral, sort_func* sort)
{
  dlclose(*library);
  switch (*current_lib)
  {
     case FIRST: *current_lib = SECOND; break;
     case SECOND: *current_lib = FIRST; break;
```

```
    }

    char buffer[BUFFER_SIZE];

    *library = dlopen(LIB_NAMES[*current_lib], RTLD_LAZY);
    if (!(*library))
    {
        int len = snprintf(buffer, BUFFER_SIZE, "Error switching: %s\n", dlerror());
        write(STDERR_FILENO, buffer, len);
        return ER_DLOPEN;
    }


    *sin_integral = dlsym(*library, "sin_integral");
    if (!sin_integral)
    {
        const char msg[] = "warning: failed to find sin_integral function implementation\n";
        write(STDERR_FILENO, msg, sizeof(msg));
    }

    *sort = dlsym(*library, "sort");
    if (!sort)
    {
        const char msg[] = "warning: failed to find sort function implementation\n";
        write(STDERR_FILENO, msg, sizeof(msg));
    }

    {
        int len = snprintf(buffer, BUFFER_SIZE, "Switched to library: %s\n",
LIB_NAMES[*current_lib]);
```

```
        write(STDOUT_FILENO, buffer, len);
    }

    return OK;
}


void command_1(sin_integral_func sin_integral)
{
    char* arg1 = strtok(NULL, " \t\n");
    char* arg2 = strtok(NULL, " \t\n");
    char* arg3 = strtok(NULL, " \t\n");

    int len = 0;
    char buffer[BUFFER_SIZE];

    if (arg1 && arg2 && arg3)
    {
        float res = sin_integral(atof(arg1), atof(arg2), atof(arg3));
        len = snprintf(buffer, BUFFER_SIZE, "Integral result: %.5f\n", res);

        write(STDOUT_FILENO, buffer, len);
    }
}

void command_2(sort_func sort)
{
    char* size_str = strtok(NULL, " \t\n");
    if (!size_str) return;
```

```
int size = atoi(size_str);
int* array = (int*)malloc(size * sizeof(int));
if (!array)
{
    const char* message = "Unable to malloc memory for int* array\n";
    write(STDERR_FILENO, message, strlen(message));
    return;
}

for (int i = 0; i < size; ++i)
{
    char* value = strtok(NULL, " \t\n");
    if (value) array[i] = atoi(value);
    else array[i] = 0;
}

sort(array, size);

{
    const char* message = "Sorted array: ";
    write(STDOUT_FILENO, message, strlen(message));
}

char buffer[BUFFER_SIZE];
for (int i = 0; i < size; ++i)
{
    int len = snprintf(buffer, BUFFER_SIZE, "%d ", array[i]);
    write(STDOUT_FILENO, buffer, len);
}
```

```
    write(STDOUT_FILENO, "\n", 1);
    free(array);
}


int main()
{
    const char* LIB_NAMES[] = {"./libd1.so", "./libd2.so"};
    int current_lib = FIRST;

    sin_integral_func sin_integral = NULL;
    sort_func sort = NULL;

    char buffer[BUFFER_SIZE];

    void* library = dlopen(LIB_NAMES[current_lib], RTLD_LAZY);
    if (!library)
    {
        int len = snprintf(buffer, BUFFER_SIZE, "Error loading library: %s\n", dlerror());
        write(STDERR_FILENO, buffer, len);
        return ER_DLOPEN;
    }

    sin_integral = dlsym(library, "sin_integral");
    if (!sin_integral)
    {
        const char msg[] = "warning: failed to find sin_integral function implementation\n";
        write(STDERR_FILENO, msg, sizeof(msg));
    }
```

```
    sort = dlsym(library, "sort");
    if (!sort)
    {
        const char msg[] = "warning: failed to find sin_integral function implementation\n";
        write(STDERR_FILENO, msg, sizeof(msg));
    }

    {
        const char *msg = "Program 2 (Dynamic).\nCommands: 0 (Switch) | 1 A B E | 2 Size Arr...\n>
";
        write(STDOUT_FILENO, msg, strlen(msg));
    }

    int bytes_read;

    while ((bytes_read = read(STDIN_FILENO, buffer, BUFFER_SIZE - 1)) > 0)
    {
        buffer[bytes_read] = '\0';

        char *token = strtok(buffer, " \t\n");
        if (!token) continue;

        int cmd = atoi(token);
        switch (cmd)
        {
            case 0:
            {
                int result = command_0(LIB_NAMES, &library, &current_lib, &sin_integral, &sort);
                if (result != OK) return result;
                break;
```

```
            }
            case 1:
            {
                command_1(sin_integral);
                break;
            }
            case 2:
            {
                command_2(sort);
                break;
            }
        }

        write(STDOUT_FILENO, "> ", 2);
    }

    if (library) dlclose(library);
    return OK;
}
```

Протокол работы программы

```
❯ ./prog1
Program 1 (Static).
Commands: 1 a b e | 2 Size Array... | Ctrl + D to exit
> 1 0 3.14159 0.001
Integral result: 2.00000
> 1 -3.14159 3.14159 0.01
Integral result: 0.00001
> 1 0 1.5708 0.1
Integral result: 0.97836
> 2 5 50 40 30 20 10
Sorted array: 10 20 30 40 50
> 2 8 5 -10 5 0 100 -20 5 1
Sorted array: -20 -10 0 1 5 5 5 100
> 2 4 1 2 3 4
Sorted array: 1 2 3 4
> 2 1 42
Sorted array: 42
>
```

```
❯ ./prog2
Program 2 (Dynamic).
Commands: 0 (Switch) | 1 A B E | 2 Size Arr...
> 1 0 3.14159 0.1
Integral result: 1.99955
> 2 5 5 4 3 2 1
Sorted array: 1 2 3 4 5
> 0
Switched to library: ./libd2.so
> 1 0 3.14159 0.1
Integral result: 1.99663
> 2 6 100 -50 20 0 5 5
Sorted array: -50 0 5 5 20 100
>
```

Strace:

```
❯ strace -f ./prog1

execve("./prog1", ["./prog1"], 0x7ffec9346cc8 /* 61 vars */) = 0

brk(NULL)                               = 0x55f8ae3bd000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da75b4000

access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v4/libd1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libd1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libd1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT
(No such file or directory)
```

openat(AT_FDCWD, "./libd1.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=14952, ...}) = 0

getcwd("/home/avgu5kov/Projects/MAI-OS-LW/lab4/src", 128) = 43

mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da75af000

mmap(0x7f7da75b0000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f7da75b0000

mmap(0x7f7da75b1000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f7da75b1000

mmap(0x7f7da75b2000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f7da75b2000

close(3) = 0

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v4/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "./libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=162287, ...}) = 0

mmap(NULL, 162287, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7da7587000

close(3) = 0

openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000x\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896

fstat(3, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896

mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da7200000

mmap(0x7f7da7224000, 1515520, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7f7da7224000

mmap(0x7f7da7396000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x196000) = 0x7f7da7396000

mmap(0x7f7da7405000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x204000) = 0x7f7da7405000

```
mmap(0x7f7da740b000, 31792, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7da740b000

close(3)                        = 0

openat(AT_FDCWD, "/usr/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=1100400, ...}) = 0

mmap(NULL, 1102152, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f7da7479000

mmap(0x7f7da7488000, 569344, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f7da7488000

mmap(0x7f7da7513000, 466944, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9a000) = 0x7f7da7513000

mmap(0x7f7da7585000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10b000) = 0x7f7da7585000

close(3)                        = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f7da7476000

arch_prctl(ARCH_SET_FS, 0x7f7da7476740) = 0

set_tid_address(0x7f7da7476a10)         = 9090

set_robust_list(0x7f7da7476a20, 24)     = 0

rseq(0x7f7da7476680, 0x20, 0, 0x53053053) = 0

mprotect(0x7f7da7405000, 16384, PROT_READ) = 0

mprotect(0x7f7da7585000, 4096, PROT_READ) = 0

mprotect(0x7f7da75b2000, 4096, PROT_READ) = 0

mprotect(0x55f897ddd000, 4096, PROT_READ) = 0

mprotect(0x7f7da75f5000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

getrandom("\x75\x05\xe1\x24\xe0\x1c\xb6\xda", 8, GRND_NONBLOCK) = 8

munmap(0x7f7da7587000, 162287)          = 0

write(1, "Program 1 (Static).\nCommands: 1 "..., 77Program 1 (Static).

Commands: 1 a b e | 2 Size Array... | Ctrl + D to exit

> ) = 77

read(01 0 3.14159 0.001

, "1 0 3.14159 0.001\n", 4095)    = 18

write(1, "Integral result: 2.00000\n", 25Integral result: 2.00000
```

```
) = 25
write(1, "> ", 2> )                     = 2
read(01 -3.14159 3.14159 0.01
, "1 -3.14159 3.14159 0.01\n", 4095) = 24
write(1, "Integral result: 0.00001\n", 25Integral result: 0.00001
) = 25
write(1, "> ", 2> )                     = 2
read(02 5 50 40 30 20 10
, "2 5 50 40 30 20 10\n", 4095)   = 19
brk(NULL)                               = 0x55f8ae3bd000
brk(0x55f8ae3de000)                     = 0x55f8ae3de000
write(1, "Sorted array: ", 14Sorted array: )         = 14
write(1, "10 ", 310 )                   = 3
write(1, "20 ", 320 )                   = 3
write(1, "30 ", 330 )                   = 3
write(1, "40 ", 340 )                   = 3
write(1, "50 ", 350 )                   = 3
write(1, "\n", 1
)                       = 1
write(1, "> ", 2> )                     = 2
read(02 8 5 -10 5 0 100 -20 5 1
, "2 8 5 -10 5 0 100 -20 5 1\n", 4095) = 26
write(1, "Sorted array: ", 14Sorted array: )         = 14
write(1, "-20 ", 4-20 )                 = 4
write(1, "-10 ", 4-10 )                 = 4
write(1, "0 ", 20 )                     = 2
write(1, "1 ", 21 )                     = 2
write(1, "5 ", 25 )                     = 2
write(1, "5 ", 25 )                     = 2
write(1, "5 ", 25 )                     = 2
write(1, "100 ", 4100 )                 = 4
write(1, "\n", 1
```

```
)                       = 1
write(1, "> ", 2> )                    = 2
read(0, "", 4095)                       = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

```
❯ strace -f ./prog2
execve("./prog2", ["./prog2"], 0x7ffdf50ac748 /* 61 vars */) = 0
brk(NULL)                               = 0x5651ca9b3000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=162287, ...}) = 0
mmap(NULL, 162287, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f231d747000
close(3)                                = 0
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000x\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
fstat(3, {st_mode=S_IFREG|0755, st_size=2149728, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f231d745000
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 896, 64) = 896
mmap(NULL, 2174000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f231d400000
mmap(0x7f231d424000, 1515520, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x24000) = 0x7f231d424000
mmap(0x7f231d596000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x196000) = 0x7f231d596000
mmap(0x7f231d605000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x204000) = 0x7f231d605000
mmap(0x7f231d60b000, 31792, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f231d60b000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f231d742000
arch_prctl(ARCH_SET_FS, 0x7f231d742740) = 0
```

set_tid_address(0x7f231d742a10)   = 9241

set_robust_list(0x7f231d742a20, 24)   = 0

rseq(0x7f231d742680, 0x20, 0, 0x53053053) = 0

mprotect(0x7f231d605000, 16384, PROT_READ) = 0

mprotect(0x565190af8000, 4096, PROT_READ) = 0

mprotect(0x7f231d7ae000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

getrandom("\xda\x23\x5e\x3d\x2a\xb3\x44\x35", 8, GRND_NONBLOCK) = 8

munmap(0x7f231d747000, 162287)   = 0

brk(NULL)   = 0x5651ca9b3000

brk(0x5651ca9d4000)   = 0x5651ca9d4000

openat(AT_FDCWD, "./libd1.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=14952, ...}) = 0

getcwd("/home/avgu5kov/Projects/MAI-OS-LW/lab4/src", 128) = 43

**mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7f231d76a000**

**mmap(0x7f231d76b000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f231d76b000**

**mmap(0x7f231d76c000, 4096, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f231d76c000**

**mmap(0x7f231d76d000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f231d76d000**

close(3)   = 0

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=162287, ...}) = 0

mmap(NULL, 162287, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f231d71a000

close(3)   = 0

openat(AT_FDCWD, "/usr/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=1100400, ...}) = 0

mmap(NULL, 1102152, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f231d2f2000

mmap(0x7f231d301000, 569344, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f231d301000

mmap(0x7f231d38c000, 466944, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x7f231d38c000

mmap(0x7f231d3fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10b000) = 0x7f231d3fe000

close(3)   = 0

mprotect(0x7f231d3fe000, 4096, PROT_READ) = 0

mprotect(0x7f231d76d000, 4096, PROT_READ) = 0

munmap(0x7f231d71a000, 162287)   = 0

write(1, "Program 2 (Dynamic).\nCommands: 0"..., 70Program 2 (Dynamic).

Commands: 0 (Switch) | 1 A B E | 2 Size Arr...

>) = 70

read(01 0 3.14159 0.1

, "1 0 3.14159 0.1\n", 4095)   = 16

write(1, "Integral result: 1.99955\n", 25Integral result: 1.99955

) = 25

write(1, "> ", 2>)   = 2

read(02 5 5 4 3 2 1

, "2 5 5 4 3 2 1\n", 4095)   = 14

write(1, "Sorted array: ", 14Sorted array:)   = 14

write(1, "1 ", 21)   = 2

write(1, "2 ", 22)   = 2

write(1, "3 ", 23)   = 2

write(1, "4 ", 24)   = 2

write(1, "5 ", 25)   = 2

write(1, "\n", 1

)   = 1

write(1, "> ", 2>)   = 2

read(00

, "0\n", 4095)   = 2

munmap(0x7f231d76a000, 16408)   = 0

munmap(0x7f231d2f2000, 1102152)   = 0

openat(AT_FDCWD, "./libd2.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=15144, ...}) = 0

getcwd("/home/avgu5kov/Projects/MAI-OS-LW/lab4/src", 128) = 43

mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f231d76a000

mmap(0x7f231d76b000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f231d76b000

mmap(0x7f231d76c000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f231d76c000

mmap(0x7f231d76d000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f231d76d000

close(3) = 0

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=162287, ...}) = 0

mmap(NULL, 162287, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f231d71a000

close(3) = 0

openat(AT_FDCWD, "/usr/lib/libm.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0755, st_size=1100400, ...}) = 0

mmap(NULL, 1102152, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f231d2f2000

mmap(0x7f231d301000, 569344, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xf000) = 0x7f231d301000

mmap(0x7f231d38c000, 466944, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x7f231d38c000

mmap(0x7f231d3fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10b000) = 0x7f231d3fe000

close(3) = 0

mprotect(0x7f231d3fe000, 4096, PROT_READ) = 0

mprotect(0x7f231d76d000, 4096, PROT_READ) = 0

munmap(0x7f231d71a000, 162287) = 0

write(1, "Switched to library: ./libd2.so\n", 32Switched to library: ./libd2.so

) = 32

write(1, "> ", 2>) = 2

read(01 0 3.14159 0.1

, "1 0 3.14159 0.1\n", 4095) = 16

write(1, "Integral result: 1.99663\n", 25Integral result: 1.99663

```
) = 25
write(1, "> ", 2> )                     = 2
read(02 6 100 -50 20 0 5 5
, "2 6 100 -50 20 0 5 5\n", 4095) = 21
write(1, "Sorted array: ", 14Sorted array: )         = 14
write(1, "-50 ", 4-50 )                   = 4
write(1, "0 ", 20 )                      = 2
write(1, "5 ", 25 )                      = 2
write(1, "5 ", 25 )                      = 2
write(1, "20 ", 320 )                     = 3
write(1, "100 ", 4100 )                    = 4
write(1, "\n", 1
)                      = 1
write(1, "> ", 2> )                     = 2
read(0, "", 4095)                      = 0
munmap(0x7f231d76a000, 16424)          = 0
munmap(0x7f231d2f2000, 1102152)        = 0
exit_group(0)                          = ?
+++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были успешно созданы динамические библиотеки в *Linux* и реализованы программы для их использования двумя способами: через статическую компоновку и через динамическую подгрузку (*runtime*) с использованием *dlfcn.h.*