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-214Б-24

Студент: Шведова Е. В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 07.12.25

Москва, 2025

Постановка задачи

Вариант 13.

Требуется создать динамические библиотеки, которые реализуют заданный вариантом функционал. Далее использовать данные библиотеки 2-мя способами:

1. Во время компиляции (на этапе линковки/linking)
2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

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

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа (программа №1), которая использует одну из библиотек, используя информацию, полученную на этапе компиляции;
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их относительные пути и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

- Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
- “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 ее выполнения;
- “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 ее выполнения.

Общий метод и алгоритм решения

В рамках лабораторной работы создана программа, демонстрирующая два подхода к использованию динамических библиотек: статическое связывание на этапе компиляции и динамическая загрузка во время выполнения. Программа состоит из двух независимых приложений, реализующих одинаковый функционал разными способами.

Статическая программа (static) связывается с библиотекой lib1 на этапе компиляции, используя классический подход линковки. Программа предоставляет интерфейс для вычисления производной функции cos(x) в заданной точке и расчета площади геометрической фигуры. Пользователь взаимодействует с программой через консольные команды, передавая необходимые параметры.

Динамическая программа (dynamic) использует механизм динамической загрузки библиотек через системные вызовы dlopen, dlsym и dlclose. Программа загружает одну из двух реализаций

библиотек (lib1 или lib2) по относительным путям и предоставляет возможность переключения между ними во время выполнения. Каждая библиотека содержит альтернативные реализации математических функций: первая использует правостороннюю разность для вычисления производной и формулу прямоугольника для площади, вторая - центральную разность и формулу прямоугольного треугольника.

Синхронизация доступа к функциям загружаемых библиотек осуществляется через указатели на функции, полученные с помощью dlsym. Пользовательский интерфейс обеих программ идентичен и поддерживает команды для вызова соответствующих функций и переключения между реализациями (в динамической версии).

По завершении работы динамическая программа корректно освобождает ресурсы с помощью dlclose, обеспечивая выгрузку библиотек из памяти. Оба подхода демонстрируют различные стратегии использования библиотечного кода: статическое связывание обеспечивает простоту и надежность, в то время как динамическая загрузка предоставляет гибкость и возможность обновления функционала без перекомпиляции основной программы.

Использованные системные вызовы:

- dlopen(const char *filename, int flags) – открывает динамическую библиотеку, указанную по пути filename, и возвращает указатель на дескриптор библиотеки.
- dlsym(void *handle, const char *symbol) – возвращает адрес функции с именем symbol из загруженной библиотеки, на которую указывает handle. Позволяет получить указатели на функции cos_derivative и area.
- dlclose(void *handle) – уменьшает счетчик ссылок на динамическую библиотеку. Если счетчик достигает нуля, библиотека выгружается из памяти. Используется для освобождения ресурсов при переключении библиотек и завершении программы.
- dlerror(void) – возвращает строку с описанием последней ошибки, возникшей при работе с функциями семейства dl*. Используется для диагностики ошибок загрузки библиотек и поиска символов.

Код программы

lib.h

```
#ifndef LIB_H
#define LIB_H


float cos_derivative(float a, float dx);
float area(float a, float b);


#endif
```

lib1.c

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


float cos_derivative(float a, float dx)
{
   return (cosf(a + dx) - cosf(a)) / dx;
}
```

```
float area(float a, float b)
{
   return a * b;
}
```

lib2.c

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

float cos_derivative(float a, float dx)
{
   return (cosf(a + dx) - cosf(a - dx)) / (2 * dx);
}

float area(float a, float b)
{
   return 0.5 * a * b;
}
```

static.c

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lib.h"

#define BUFFER_SIZE 1024

void process_derivative()
{
   char* a_str = strtok(NULL, " \t\n");
   char* dx_str = strtok(NULL, " \t\n");

   char output[BUFFER_SIZE];

   if (a_str && dx_str)
   {
      float a_val = atof(a_str);
      float dx_val = atof(dx_str);
      float result = cos_derivative(a_val, dx_val);

      int len = snprintf(output, BUFFER_SIZE, "Derivative of cos at %.2f: %.6f\n", a_val,
result);
      write(STDOUT_FILENO, output, len);
   }
   else
   {
```

```
        const char* error_msg = "Error: two arguments required - a and dx\n";
        write(STDOUT_FILENO, error_msg, strlen(error_msg));
    }
}

void process_area()
{
    char* a_str = strtok(NULL, " \t\n");
    char* b_str = strtok(NULL, " \t\n");

    char output[BUFFER_SIZE];

    if (a_str && b_str)
    {
        float a_val = atof(a_str);
        float b_val = atof(b_str);
        float result = area(a_val, b_val);

        int len = snprintf(output, BUFFER_SIZE, "Area with sides %.2f and %.2f: %.6f\n", a_val,
b_val, result);
        write(STDOUT_FILENO, output, len);
    }
    else
    {
        const char* error_msg = "Error: two arguments required - a and b\n";
        write(STDOUT_FILENO, error_msg, strlen(error_msg));
    }
}

int main()
{
    {
        const char* welcome_msg =
            "Static Linking Application\n"
            "Available commands:\n"
            "1 a dx - derivative of cos(x) at point a with step dx\n"
            "2 a b - area of figure with sides a and b\n"
            "exit - terminate program\n"
            "Enter command: ";
        write(STDOUT_FILENO, welcome_msg, strlen(welcome_msg));
    }

    int bytes_read = 0;
    char input_line[BUFFER_SIZE];

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

```
        char* command = strtok(input_line, " \t\n");
        if (!command) continue;

        if (strcmp(command, "1") == 0)
        {
            process_derivative();
        }
        else if (strcmp(command, "2") == 0)
        {
            process_area();
        }
        else if (strcmp(command, "exit") == 0)
        {
            const char* bye_msg = "Program terminated.\n";
            write(STDOUT_FILENO, bye_msg, strlen(bye_msg));
            break;
        }
        else
        {
            char error_msg[100];
            int len = snprintf(error_msg, 100, "Unknown command: '%s'. Use '1', '2' or 'exit'\n",
command);
            write(STDOUT_FILENO, error_msg, len);
        }

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

    return 0;
}
```

dynamic.c

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

#define BUFFER_SIZE 1024

typedef float (*cos_derivative_func)(float, float);
typedef float (*area_func)(float, float);

typedef enum
{
```

```
    SUCCESS = 0,
    LIBRARY_ERROR = -1,
} StatusCode;


typedef enum
{
    LIB_VERSION_1 = 0,
    LIB_VERSION_2 = 1,
} LibraryVersion;


StatusCode switch_library(const char** lib_paths, void** lib_handle, LibraryVersion*
current_version, cos_derivative_func* deriv_func, area_func* area_func_ptr)
{
    dlclose(*lib_handle);


    *current_version = (*current_version == LIB_VERSION_1) ? LIB_VERSION_2 :
LIB_VERSION_1;


    char message[BUFFER_SIZE];


    *lib_handle = dlopen(lib_paths[*current_version], RTLD_LAZY);
    if (!(*lib_handle))
    {
        int msg_len = snprintf(message, BUFFER_SIZE, "Library loading error: %s\n",
dlerror());
        write(STDERR_FILENO, message, msg_len);
        return LIBRARY_ERROR;
    }


    *deriv_func = dlsym(*lib_handle, "cos_derivative");
    *area_func_ptr = dlsym(*lib_handle, "area");


    if (!*deriv_func || !*area_func_ptr)
    {
        const char error_msg[] = "Error: failed to locate functions in library\n";
        write(STDERR_FILENO, error_msg, sizeof(error_msg));
        return LIBRARY_ERROR;
    }


    {
        int msg_len = snprintf(message, BUFFER_SIZE, "Loaded library: %s\n",
lib_paths[*current_version]);
        write(STDOUT_FILENO, message, msg_len);
    }


    return SUCCESS;
}
```

```
void execute_derivative(cos_derivative_func deriv_func)
{
    char* a_str = strtok(NULL, " \t\n");
    char* dx_str = strtok(NULL, " \t\n");

    char output[BUFFER_SIZE];


    if (a_str && dx_str)
    {
        float a_val = atof(a_str);
        float dx_val = atof(dx_str);
        float result = deriv_func(a_val, dx_val);

        int msg_len = snprintf(output, BUFFER_SIZE, "cos'(%.4f) = %.8f (dx=%.6f)\n", a_val,
result, dx_val);
        write(STDOUT_FILENO, output, msg_len);
    }
    else
    {
        const char* error_msg = "Error: a and dx values required\n";
        write(STDOUT_FILENO, error_msg, strlen(error_msg));
    }
}


void execute_area(area_func area_func_ptr)
{
    char* a_str = strtok(NULL, " \t\n");
    char* b_str = strtok(NULL, " \t\n");

    char output[BUFFER_SIZE];


    if (a_str && b_str)
    {
        float a_val = atof(a_str);
        float b_val = atof(b_str);
        float result = area_func_ptr(a_val, b_val);

        int msg_len = snprintf(output, BUFFER_SIZE, "Area with sides %.3f and %.3f = %.6f\n",
a_val, b_val, result);
        write(STDOUT_FILENO, output, msg_len);
    }
    else
    {
        const char* error_msg = "Error: two side values required\n";
        write(STDOUT_FILENO, error_msg, strlen(error_msg));
    }
}


int main()
```

```
{
    const char* library_files[] = {"./liblib1.so", "./liblib2.so"};
    LibraryVersion active_lib = LIB_VERSION_1;

    cos_derivative_func cos_derivative_func_ptr = NULL;
    area_func area_func_ptr = NULL;

    char message_buffer[BUFFER_SIZE];

    void* loaded_library = dlopen(library_files[active_lib], RTLD_LAZY);
    if (!loaded_library)
    {
        int msg_len = snprintf(message_buffer, BUFFER_SIZE, "Initialization error: %s\n",
dlerror());
        write(STDERR_FILENO, message_buffer, msg_len);
        return LIBRARY_ERROR;
    }

    cos_derivative_func_ptr = dlsym(loaded_library, "cos_derivative");
    area_func_ptr = dlsym(loaded_library, "area");

    if (!cos_derivative_func_ptr || !area_func_ptr)
    {
        const char error_msg[] = "Critical error: functions not found\n";
        write(STDERR_FILENO, error_msg, sizeof(error_msg));
        dlclose(loaded_library);
        return LIBRARY_ERROR;
    }

    {
        const char *welcome_text =
            "Dynamic Loading Application\n"
            "Available commands:\n"
            "0 - toggle between algorithm implementations\n"
            "1 a dx - derivative of cos(x) at point a with step dx\n"
            "2 a b - area with sides a and b\n"
            "exit - exit program\n"
            "Enter command: ";
        write(STDOUT_FILENO, welcome_text, strlen(welcome_text));
    }

    int read_bytes;

    while ((read_bytes = read(STDIN_FILENO, message_buffer, BUFFER_SIZE - 1)) > 0)
    {
        message_buffer[read_bytes] = '\0';
```

```
        char *user_command = strtok(message_buffer, " \t\n");
        if (!user_command) continue;


        if (strcmp(user_command, "0") == 0)
        {
            switch_library(library_files, &loaded_library, &active_lib, &cos_derivative_func_ptr,
&area_func_ptr);
        }
        else if (strcmp(user_command, "1") == 0)
        {
            execute_derivative(cos_derivative_func_ptr);
        }
        else if (strcmp(user_command, "2") == 0)
        {
            execute_area(area_func_ptr);
        }
        else if (strcmp(user_command, "exit") == 0)
        {
            const char* exit_msg = "Exiting program.\n";
            write(STDOUT_FILENO, exit_msg, strlen(exit_msg));
            break;
        }
        else
        {
            char error_msg[120];
            int len = snprintf(error_msg, 120, "Unknown command '%s'. Use: 0, 1, 2, or exit\n",
user_command);
            write(STDOUT_FILENO, error_msg, len);
        }


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


    if (loaded_library) dlclose(loaded_library);
    return SUCCESS;
}
```

CMakeLists.txt

```
  cmake_minimum_required(VERSION 3.10)


project(Lab2_Dynamic)


set(CMAKE_C_STANDARD 11)
set(CMAKE_C_STANDARD_REQUIRED ON)


add_compile_options(-Wall -Wextra -Wpedantic -g -fPIC)
```

```
add_library(lib1 SHARED lib1.c)
add_library(lib2 SHARED lib2.c)

target_link_libraries(lib1 m)
target_link_libraries(lib2 m)

add_executable(static static.c lib1.c)
target_link_libraries(static m)

add_executable(dynamic dynamic.c)
target_link_libraries(dynamic dl m)
```

Протокол работы программы

crane@Assus:~/op/Lab4/src/build$ ./static

Static Linking Application

Available commands:

1 a dx - derivative of cos(x) at point a with step dx

2 a b - area of figure with sides a and b

exit - terminate program

Enter command: 1 0 0.01

Derivative of cos at 0.00: -0.005001

> 1 1.5708 0.001

Derivative of cos at 1.57: -1.000047

> 1 3.14159 0.1

Derivative of cos at 3.14: 0.049956

> 2 5 3

Area with sides 5.00 and 3.00: 15.000000

> 2 2.5 4.2

Area with sides 2.50 and 4.20: 10.500000

> 2 0 10

Area with sides 0.00 and 10.00: 0.000000

> 3 1 2

Unknown command: '3'. Use '1', '2' or 'exit'

> exit

Program terminated.

```
crane@Assus:~/op/Lab4/src/build$ ./dynamic
Dynamic Loading Application
Available commands:
0 - toggle between algorithm implementations
1 a dx - derivative of cos(x) at point a with step dx
2 a b - area with sides a and b
exit - exit program
Enter command: 1 0 0.01
cos'(0.0000) = -0.00500083 (dx=0.010000)
> 2 5 3
Area with sides 5.000 and 3.000 = 15.000000
> 1 1.5708 0.001
cos'(1.5708) = -1.00004661 (dx=0.001000)
> 2 2.5 4.2
Area with sides 2.500 and 4.200 = 10.500000
> 0
Loaded library: ./liblib2.so
> 1 0 0.01
cos'(0.0000) = 0.00000000 (dx=0.010000)
> 2 5 3
Area with sides 5.000 and 3.000 = 7.500000
> 1 0.7854 0.01
cos'(0.7854) = -0.70709586 (dx=0.010000)
> 2 0 10
Area with sides 0.000 and 10.000 = 0.000000
> 4 1 2
Unknown command '4'. Use: 0, 1, 2, or exit
> 0
Loaded library: ./liblib1.so
> 1 0 0.01
cos'(0.0000) = -0.00500083 (dx=0.010000)
> exit
```

```
Exiting program.


crane@Assus:~/op/Lab4/src/build$ strace -f ./static
execve("./static", ["./static"], 0x7ffe8196e488 /* 79 vars */) = 0
brk(NULL)                               = 0x605c6cc1b000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7d2f992e6000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=75687, ...}) = 0
mmap(NULL, 75687, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7d2f992d3000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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|0644, st_size=952616, ...}) = 0
mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7d2f991ea000
mmap(0x7d2f991fa000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x10000) = 0x7d2f991fa000
mmap(0x7d2f99279000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8f000) = 0x7d2f99279000
mmap(0x7d2f992d1000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xe7000) = 0x7d2f992d1000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\0\220\243\2\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"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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"..., 784, 64) = 784
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7d2f98e00000
mmap(0x7d2f98e28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x28000) = 0x7d2f98e28000
mmap(0x7d2f98fb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1b0000) = 0x7d2f98fb0000
mmap(0x7d2f98fff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1fe000) = 0x7d2f98fff000
mmap(0x7d2f99005000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7d2f99005000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7d2f991e7000
arch_prctl(ARCH_SET_FS, 0x7d2f991e7740) = 0
set_tid_address(0x7d2f991e7a10)         = 6432
set_robust_list(0x7d2f991e7a20, 24)     = 0
rseq(0x7d2f991e8060, 0x20, 0, 0x53053053) = 0
mprotect(0x7d2f98fff000, 16384, PROT_READ) = 0
mprotect(0x7d2f992d1000, 4096, PROT_READ) = 0
mprotect(0x605c604c5000, 4096, PROT_READ) = 0
mprotect(0x7d2f99324000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
```

```
munmap(0x7d2f992d3000, 75687)           = 0
write(1, "Static Linking Application\nAvail"..., 183Static Linking Application
Available commands:
1 a dx - derivative of cos(x) at point a with step dx
2 a b - area of figure with sides a and b
exit - terminate program
Enter command: ) = 183
read(0,  1 1.5708 0.001
" 1 1.5708 0.001\n", 1023)      = 16
write(1, "Derivative of cos at 1.57: -1.00"..., 37Derivative of cos at 1.57: -1.000047
) = 37
write(1, "> ", 2> )                      = 2
read(0, 2 2.5 4.2
"2 2.5 4.2\n", 1023)            = 10
write(1, "Area with sides 2.50 and 4.20: 1"..., 41Area with sides 2.50 and 4.20: 10.500000
) = 41
write(1, "> ", 2> )                      = 2
read(0, exit
"exit\n", 1023)                 = 5
write(1, "Program terminated.\n", 20Program terminated.
)   = 20
exit_group(0)                           = ?
+++ exited with 0 +++


crane@Assus:~/op/Lab4/src/build$ strace -f ./dynamic
execve("./dynamic", ["./dynamic"], 0x7ffe0d908b18 /* 79 vars */) = 0
brk(NULL)                               = 0x61010847f000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7b45106f2000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=75687, ...}) = 0
mmap(NULL, 75687, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7b45106df000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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\0\220\243\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"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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"..., 784, 64) = 784
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7b4510400000
mmap(0x7b4510428000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x28000) = 0x7b4510428000
mmap(0x7b45105b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1b0000) = 0x7b45105b0000
mmap(0x7b45105ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1fe000) = 0x7b45105ff000
mmap(0x7b4510605000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7b4510605000
```

```
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x7b45106dc000
arch_prctl(ARCH_SET_FS, 0x7b45106dc740) = 0
set_tid_address(0x7b45106dca10)         = 7594
set_robust_list(0x7b45106dca20, 24)     = 0
rseq(0x7b45106dd060, 0x20, 0, 0x53053053) = 0
mprotect(0x7b45105ff000, 16384, PROT_READ) = 0
mprotect(0x6100cd422000, 4096, PROT_READ) = 0
mprotect(0x7b4510730000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
munmap(0x7b45106df000, 75687)           = 0
getrandom("\x86\x9b\x30\xaf\xf8\xd8\xc7\x14", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x61010847f000
brk(0x6101084a0000)                     = 0x6101084a0000
openat(AT_FDCWD, "./liblib1.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|0775, st_size=16936, ...}) = 0
getcwd("/home/crane/op/Lab4/src/build", 128) = 30
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7b45106ed000
mmap(0x7b45106ee000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1000) = 0x7b45106ee000
mmap(0x7b45106ef000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7b45106ef000
mmap(0x7b45106f0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7b45106f0000
close(3)                                = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=75687, ...}) = 0
mmap(NULL, 75687, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7b45106c9000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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|0644, st_size=952616, ...}) = 0
mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7b4510317000
mmap(0x7b4510327000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x10000) = 0x7b4510327000
mmap(0x7b45103a6000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8f000) = 0x7b45103a6000
mmap(0x7b45103fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xe7000) = 0x7b45103fe000
close(3)                                = 0
mprotect(0x7b45103fe000, 4096, PROT_READ) = 0
mprotect(0x7b45106f0000, 4096, PROT_READ) = 0
munmap(0x7b45106c9000, 75687)           = 0
write(1, "Dynamic Loading Application\nAvai"..., 214Dynamic Loading Application
Available commands:
0 - toggle between algorithm implementations
1 a dx - derivative of cos(x) at point a with step dx
```

```
2 a b - area with sides a and b
exit - exit program
Enter command: ) = 214
read(0, 1 1.5708 0.001
"1 1.5708 0.001\n", 1023)       = 15
write(1, "cos'(1.5708) = -1.00004661 (dx=0"..., 41cos'(1.5708) = -1.00004661 (dx=0.001000)
) = 41
write(1, "> ", 2> )                     = 2
read(0, 2 2.5 4.2
"2 2.5 4.2\n", 1023)           = 10
write(1, "Area with sides 2.500 and 4.200 "..., 44Area with sides 2.500 and 4.200 = 10.500000
) = 44
write(1, "> ", 2> )                     = 2
read(0, 0
"0\n", 1023)                   = 2
munmap(0x7b45106ed000, 16408)          = 0
munmap(0x7b4510317000, 950296)         = 0
openat(AT_FDCWD, "./liblib2.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\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|0775, st_size=17024, ...}) = 0
getcwd("/home/crane/op/Lab4/src/build", 128) = 30
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7b45106ed000
mmap(0x7b45106ee000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1000) = 0x7b45106ee000
mmap(0x7b45106ef000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7b45106ef000
mmap(0x7b45106f0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7b45106f0000
close(3)                               = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=75687, ...}) = 0
mmap(NULL, 75687, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7b45106c9000
close(3)                               = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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|0644, st_size=952616, ...}) = 0
mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7b4510317000
mmap(0x7b4510327000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x10000) = 0x7b4510327000
mmap(0x7b45103a6000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8f000) = 0x7b45103a6000
mmap(0x7b45103fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xe7000) = 0x7b45103fe000
close(3)                               = 0
mprotect(0x7b45103fe000, 4096, PROT_READ) = 0
mprotect(0x7b45106f0000, 4096, PROT_READ) = 0
munmap(0x7b45106c9000, 75687)          = 0
write(1, "Loaded library: ./liblib2.so\n", 29Loaded library: ./liblib2.so
```

```
) = 29
write(1, "> ", 2> )                     = 2
read(0, 1 0 0.01
"1 0 0.01\n", 1023)            = 9
write(1, "cos'(0.0000) = 0.00000000 (dx=0."..., 40cos'(0.0000) = 0.00000000 (dx=0.010000)
) = 40
write(1, "> ", 2> )                     = 2
read(0, 0
"0\n", 1023)                   = 2
munmap(0x7b45106ed000, 16408)          = 0
munmap(0x7b4510317000, 950296)         = 0
openat(AT_FDCWD, "./liblib1.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\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|0775, st_size=16936, ...}) = 0
getcwd("/home/crane/op/Lab4/src/build", 128) = 30
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7b45106ed000
mmap(0x7b45106ee000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1000) = 0x7b45106ee000
mmap(0x7b45106ef000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x2000) = 0x7b45106ef000
mmap(0x7b45106f0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|
MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7b45106f0000
close(3)                               = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=75687, ...}) = 0
mmap(NULL, 75687, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7b45106c9000
close(3)                               = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\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|0644, st_size=952616, ...}) = 0
mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7b4510317000
mmap(0x7b4510327000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x10000) = 0x7b4510327000
mmap(0x7b45103a6000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x8f000) = 0x7b45103a6000
mmap(0x7b45103fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0xe7000) = 0x7b45103fe000
close(3)                               = 0
mprotect(0x7b45103fe000, 4096, PROT_READ) = 0
mprotect(0x7b45106f0000, 4096, PROT_READ) = 0
munmap(0x7b45106c9000, 75687)          = 0
write(1, "Loaded library: ./liblib1.so\n", 29Loaded library: ./liblib1.so
) = 29
write(1, "> ", 2> )                     = 2
read(0, exit
"exit\n", 1023)                = 5
write(1, "Exiting program.\n", 17Exiting program.
)      = 17
```

```
munmap(0x7b45106ed000, 16408)           = 0
munmap(0x7b4510317000, 950296)          = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были успешно созданы типичные библиотеки, реализованы программы их использования двумя способами: через статическую компоновку и через типичную динамическую загрузку с использованием dlfcn.h.