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-214БВ-24

Студент: Зайцев Т.И.

Преподаватель: Бахарев В.Д.

Оценка:

_____

Дата: 12.12.25

Москва, 2025

Постановка задачи

Вариант 35.

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 используют одну из библиотек, используя информацию полученные 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.

7. Подсчет площади плоской геометрической фигуры по двум сторонам:

Сигнатура функции: float area(float a, float b);
● Реализация №1: Фигура прямоугольник
● Реализация №2: Фигура прямоугольный треугольник

9. Отсортировать целочисленный массив:
Сигнатура функции: int *sort(int *array, size_t n);
● Реализация №1: Пузырьковая сортировка :^)
● Реализация №2: Сортировка Хоара (за использование qsort из libc или std::sort и его варианты из libstdc++ будет бан; реализуйте его самостоятельно)

Общий метод и алгоритм решения

Использованные системные вызовы и функции:

• dlopen() — загрузка динамической библиотеки (shared object) в адресное пространство процесса
• dlsym() — получение адреса функции из загруженной библиотеки по её имени
• dlclose() — выгрузка библиотеки из памяти и освобождение ресурсов
• read() — чтение данных из стандартного потока ввода (аналог fgets())
• write() — запись данных в стандартный поток вывода (аналог printf())

Вспомогательные функции стандартной библиотеки C:
• snprintf() — форматированный вывод в буфер
• strtof(), strtol() — преобразование строк в числа
• strtok(), strchr() — обработка строковых данных

В программе реализованы 2 типа используемых библиотек – статические (которые программа видит на этапе линковки) и динамические, которые программа загружает прямо во время выполнения.

Первая программа (статическая линковка)

Библиотечные функции area() и sort() из файлов area1.c и sort1.c статически линкуются в исполняемый файл. На вход программа принимает команды:

• 0 – выводится информация об используемых библиотеках

• 1 a b – вычисляется площадь фигуры с заданными сторонами a и b

• 2 n a1 a2 ... an – сортируется массив из n чисел

Программа проверяет корректность введённых данных (размер массива 1-100, наличие всех аргументов) и вызывает соответствующие функции для расчетов. Результаты выводятся на экран. Какими реализациями функций будут производиться вычисления, зависит от того, какие файлы были слинкованы при компиляции.

Вторая программа (динамическая загрузка) использует механизм динамической загрузки библиотек во время выполнения.

Программа не содержит код библиотечных функций на этапе компиляции. Вместо этого используются две пары динамических библиотек:

• area1.so (прямоугольник) и area2.so (треугольник)

• sort1.so (пузырьковая сортировка) и sort2.so (быстрая сортировка)

При запуске программа автоматически загружает первую версию библиотек. Основной цикл обрабатывает команды аналогично первой программе, но с ключевым отличием: при вводе 0 происходит переключение между реализациями функций. Программа выгружает текущие библиотеки через dlclose() и загружает альтернативные через dlopen(), затем получает указатели на функции через dlsym(). Это позволяет менять реализацию алгоритмов без перекомпиляции программы.

Протокол работы программы

```
tim@tim-potato-pc:~/OS_labs/lab_04/src$ gcc -shared -fPIC -o area1.so area1.c
tim@tim-potato-pc:~/OS_labs/lab_04/src$ gcc -shared -fPIC -o area2.so area2.c
tim@tim-potato-pc:~/OS_labs/lab_04/src$ gcc -shared -fPIC -o sort1.so sort1.c
tim@tim-potato-pc:~/OS_labs/lab_04/src$ gcc -shared -fPIC -o sort2.so sort2.c
tim@tim-potato-pc:~/OS_labs/lab_04/src$ gcc program1.c area1.c sort1.c -o
program1_rect_bubble
tim@tim-potato-pc:~/OS_labs/lab_04/src$ gcc program1.c area2.c sort2.c -o
program1_triangle_quick
tim@tim-potato-pc:~/OS_labs/lab_04/src$ gcc program2.c -o program2 -ldl
tim@tim-potato-pc:~/OS_labs/lab_04/src$ ./program1_triangle_quick
Program 1: Static linking
Commands:
1 a b - area
2 n a1 a2 ... an - sort
exit - exit

> 1 4 5
Area(4.000000, 5.000000) = 10.000000
> 1
Error: need two numbers
> 2 3 7 1 5
Array: 7 1 5
Sorted: 1 5 7
> exit
Exiting
tim@tim-potato-pc:~/OS_labs/lab_04/src$ ./program2
Program 2: Dynamic loading
Commands:
0 - switch version (1 or 2)
1 a b - area
2 n a1 ... an - sort
exit - exit

Loaded version 1
> 1 2 10
Area(2.000000, 10.000000) = 20.000000 (v1: rectangle)
> 0
Loaded version 2
> 1 2 10
Area(2.000000, 10.000000) = 10.000000 (v2: triangle)
```

```
> 2
Error: array size missing
> 2 4 5 7 2 3
Array: 5 7 2 3
Sorted: 2 3 5 7
(v2: quick)
> exit
Exiting
```

Код программы

area1.c

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

area2.c

```c
float area(float a, float b) {
   return (a * b) / 2.0f;
}
```

sort1.c

```c
#include <stdlib.h>
#include <string.h>

int* sort(int* array, size_t n) {
   if (n == 0) return NULL;
   int* result = (int*)malloc(n * sizeof(int));
   if (!result) return NULL;
   memcpy(result, array, n * sizeof(int));

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

sort2.c

```c
#include <stdlib.h>
#include <string.h>
```

```c
static void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

static int hoare_partition(int* arr, int low, int high) {
    int pivot = arr[(low + high) / 2];
    int i = low - 1;
    int j = high + 1;

    while (1) {
        i++;
        while (i <= high && arr[i] < pivot) {
            i++;
        }

        j--;
        while (j >= low && arr[j] > pivot) {
            j--;
        }

        if (i >= j) return j;

        swap(&arr[i], &arr[j]);
    }
}

static void quick_sort_hoare(int* arr, int low, int high) {
    if (low < high) {
        int p = hoare_partition(arr, low, high);
        quick_sort_hoare(arr, low, p);
        quick_sort_hoare(arr, p + 1, high);
    }
}

int* sort(int* array, size_t n) {
    if (n == 0) return NULL;
    int* result = (int*)malloc(n * sizeof(int));
    if (!result) return NULL;
    memcpy(result, array, n * sizeof(int));
    quick_sort_hoare(result, 0, n - 1);
    return result;
}
```

contract.h

```c
#ifndef CONTRACT_H
#define CONTRACT_H

#include <stddef.h>

float area(float a, float b);
int* sort(int* array, size_t n);

#endif
```

program1.c

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

#include "contract.h"

static void write_str(const char *str) {
   write(STDOUT_FILENO, str, strlen(str));
}

static void write_float(float x) {
   char buf[64];
   int len = snprintf(buf, sizeof(buf), "%.6f", x);
   write(STDOUT_FILENO, buf, len);
}

static void write_int(int x) {
   char buf[32];
   int len = snprintf(buf, sizeof(buf), "%d", x);
   write(STDOUT_FILENO, buf, len);
}

static void write_array(int *arr, size_t n) {
   for (size_t i = 0; i < n; i++) {
       write_int(arr[i]);
       write_str(" ");
   }
   write_str("\n");
}

static int read_line(char *buf, size_t size) {
   ssize_t bytes = read(STDIN_FILENO, buf, size - 1);
```

```
    if (bytes <= 0) {
        return 0;
    }
    buf[bytes] = '\0';

    char *newline = strchr(buf, '\n');
    if (newline) {
        *newline = '\0';
    }
    return 1;
}

static int parse_two_floats(const char *str, float *a, float *b) {
    char *end;
    *a = strtof(str, &end);
    if (end == str) return 0;

    while (*end == ' ') end++;
    *b = strtof(end, &end);
    if (end == str) return 0;

    return 1;
}

static int parse_array(const char *str, int *arr, int n) {
    char *ptr = (char *)str;
    for (int i = 0; i < n; i++) {
        while (*ptr == ' ') ptr++;
        if (*ptr == '\0') return 0;

        arr[i] = strtol(ptr, &ptr, 10);
    }
    return 1;
}

int main(void) {
    char input[256];

    write_str("Program 1: Static linking\n");
    write_str("Commands:\n");
    write_str(" 1 a b - area \n");
    write_str(" 2 n a1 a2 ... an - sort\n");
    write_str(" exit - exit\n\n");

    while (1) {
        write_str("> ");
```

```
    if (!read_line(input, sizeof(input))) {
        break;
    }

    if (strcmp(input, "exit") == 0) {
        write_str("Exiting\n");
        break;
    }

    if (strlen(input) == 0) {
        continue;
    }

    else if (input[0] == '1') {
        float a, b;
        if (parse_two_floats(input + 2, &a, &b)) {
            float res = area(a, b);
            write_str("Area(");
            write_float(a);
            write_str(", ");
            write_float(b);
            write_str(") = ");
            write_float(res);
            write_str("\n");
        } else {
            write_str("Error: need two numbers\n");
        }
    }
    else if (input[0] == '2') {
        char *space = strchr(input, ' ');
        if (!space) {
            write_str("Error: array size missing\n");
            continue;
        }

        int n = strtol(space + 1, &space, 10);
        if (n <= 0 || n > 100) {
            write_str("Error: size 1..100\n");
            continue;
        }

        int *arr = malloc(n * sizeof(int));
        if (!arr) {
            write_str("Memory error\n");
            continue;
        }
```

```
            if (parse_array(space, arr, n)) {
                write_str("Array: ");
                write_array(arr, n);
                int *sorted = sort(arr, n);
                if (sorted) {
                    write_str("Sorted: ");
                    write_array(sorted, n);
                    free(sorted);
                } else {
                    write_str("Sort failed\n");
                }
            } else {
                write_str("Error: not enough elements\n");
            }
            free(arr);
        }
        else {
            write_str("Unknown command\n");
        }
    }

    return 0;
}
```

program2.c

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

#include "contract.h"

static void write_str(const char *str) {
    write(STDOUT_FILENO, str, strlen(str));
}

static void write_float(float x) {
    char buf[64];
    int len = snprintf(buf, sizeof(buf), "%.6f", x);
    write(STDOUT_FILENO, buf, len);
}

static void write_int(int x) {
    char buf[32];
```

```
    int len = snprintf(buf, sizeof(buf), "%d", x);
    write(STDOUT_FILENO, buf, len);
}

static void write_array(int *arr, size_t n) {
    for (size_t i = 0; i < n; i++) {
        write_int(arr[i]);
        write_str(" ");
    }
    write_str("\n");
}

static int read_line(char *buf, size_t size) {
    ssize_t bytes = read(STDIN_FILENO, buf, size - 1);
    if (bytes <= 0) {
        return 0;
    }
    buf[bytes] = '\0';

    char *newline = strchr(buf, '\n');
    if (newline) {
        *newline = '\0';
    }
    return 1;
}

static int parse_two_floats(const char *str, float *a, float *b) {
    char *end;
    *a = strtof(str, &end);
    if (end == str) return 0;

    while (*end == ' ') end++;
    *b = strtof(end, &end);
    if (end == str) return 0;

    return 1;
}

static int parse_array(const char *str, int *arr, int n) {
    char *ptr = (char *)str;
    for (int i = 0; i < n; i++) {
        while (*ptr == ' ') ptr++;
        if (*ptr == '\0') return 0;

        arr[i] = strtol(ptr, &ptr, 10);
    }
```

```
    return 1;
}

int main(void) {
    char input[256];

    write_str("Program 1: Static linking\n");
    write_str("Commands:\n");
    write_str(" 1 a b - area \n");
    write_str(" 2 n a1 a2 ... an - sort\n");
    write_str(" exit - exit\n\n");

    while (1) {
        write_str("> ");
        if (!read_line(input, sizeof(input))) {
            break;
        }

        if (strcmp(input, "exit") == 0) {
            write_str("Exiting\n");
            break;
        }

        if (strlen(input) == 0) {
            continue;
        }

        else if (input[0] == '1') {
            float a, b;
            if (parse_two_floats(input + 2, &a, &b)) {
                float res = area(a, b);
                write_str("Area(");
                write_float(a);
                write_str(", ");
                write_float(b);
                write_str(") = ");
                write_float(res);
                write_str("\n");
            } else {
                write_str("Error: need two numbers\n");
            }
        }
        else if (input[0] == '2') {
            char *space = strchr(input, ' ');
            if (!space) {
                write_str("Error: array size missing\n");
```

```
            continue;
        }

        int n = strtol(space + 1, &space, 10);
        if (n <= 0 || n > 100) {
            write_str("Error: size 1..100\n");
            continue;
        }

        int *arr = malloc(n * sizeof(int));
        if (!arr) {
            write_str("Memory error\n");
            continue;
        }

        if (parse_array(space, arr, n)) {
            write_str("Array: ");
            write_array(arr, n);
            int *sorted = sort(arr, n);
            if (sorted) {
                write_str("Sorted: ");
                write_array(sorted, n);
                free(sorted);
            } else {
                write_str("Sort failed\n");
            }
        } else {
            write_str("Error: not enough elements\n");
        }
        free(arr);
    }
    else {
        write_str("Unknown command\n");
    }
  }

  return 0;
}
```

Strace

```
tim@tim-potato-pc:~/OS_labs/lab_04/src$ strace ./program2
execve("./program2", ["./program2"], 0x7ffd24bd5200 /* 84 vars */) = 0
brk(NULL)                               = 0x5f0b8966e000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7bac5d9b7000
```

```
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=69395, ...}) = 0
mmap(NULL, 69395, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7bac5d9a6000
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) = 0x7bac5d600000
mmap(0x7bac5d628000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7bac5d628000
mmap(0x7bac5d7b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7bac5d7b0000
mmap(0x7bac5d7ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7bac5d7ff000
mmap(0x7bac5d805000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7bac5d805000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7bac5d9a3000
arch_prctl(ARCH_SET_FS, 0x7bac5d9a3740) = 0
set_tid_address(0x7bac5d9a3a10)         = 24403
set_robust_list(0x7bac5d9a3a20, 24)     = 0
rseq(0x7bac5d9a4060, 0x20, 0, 0x53053053) = 0
mprotect(0x7bac5d7ff000, 16384, PROT_READ) = 0
mprotect(0x5f0b849f4000, 4096, PROT_READ) = 0
mprotect(0x7bac5d9f5000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7bac5d9a6000, 69395)           = 0
write(1, "Program 2: Dynamic loading\n", 27Program 2: Dynamic loading
) = 27
write(1, "Commands:\n", 10Commands:
)             = 10
write(1, " 0 - switch version (1 or 2)\n", 29 0 - switch version (1 or 2)
) = 29
write(1, " 1 a b - area\n", 14 1 a b - area
)         = 14
write(1, " 2 n a1 ... an - sort\n", 22 2 n a1 ... an - sort
) = 22
write(1, " exit - exit\n\n", 14 exit - exit

)        = 14
```

```
getrandom("\x9d\xd7\xa6\x1c\x86\x1d\xb4\x86", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5f0b8966e000
brk(0x5f0b8968f000)                     = 0x5f0b8968f000
openat(AT_FDCWD, "./area1.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\0"..., 832) =
832
fstat(3, {st_mode=S_IFREG|0775, st_size=15112, ...}) = 0
getcwd("/home/tim/OS_labs/lab_04/src", 128) = 29
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7bac5d9b2000
mmap(0x7bac5d9b3000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1000) = 0x7bac5d9b3000
mmap(0x7bac5d9b4000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
= 0x7bac5d9b4000
mmap(0x7bac5d9b5000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7bac5d9b5000
close(3)                                = 0
mprotect(0x7bac5d9b5000, 4096, PROT_READ) = 0
openat(AT_FDCWD, "./sort1.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\0"..., 832) =
832
fstat(3, {st_mode=S_IFREG|0775, st_size=15528, ...}) = 0
getcwd("/home/tim/OS_labs/lab_04/src", 128) = 29
mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7bac5d9ad000
mmap(0x7bac5d9ae000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1000) = 0x7bac5d9ae000
mmap(0x7bac5d9af000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
= 0x7bac5d9af000
mmap(0x7bac5d9b0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7bac5d9b0000
close(3)                                = 0
mprotect(0x7bac5d9b0000, 4096, PROT_READ) = 0
write(1, "Loaded version 1\n", 17Loaded version 1
)      = 17
write(1, "> ", 2> )                      = 2
read(0, 1 4 5
"1 4 5\n", 255)                 = 6
write(1, "Area(", 5Area()                   = 5
write(1, "4.000000", 84.000000)                 = 8
write(1, ", ", 2, )                       = 2
write(1, "5.000000", 85.000000)                 = 8
write(1, ") = ", 4) = )                     = 4
write(1, "20.000000", 920.000000)                = 9
write(1, " (v", 3 (v)                      = 3
write(1, "1", 11)                        = 1
write(1, ": rectangle)\n", 13: rectangle)
)          = 13
write(1, "> ", 2> )                       = 2
```

```
read(0, 2 5 4 5 9 2 3
"2 5 4 5 9 2 3\n", 255)         = 14
write(1, "Array: ", 7Array: )                  = 7
write(1, "4", 14)                        = 1
write(1, " ", 1 )                        = 1
write(1, "5", 15)                        = 1
write(1, " ", 1 )                        = 1
write(1, "9", 19)                        = 1
write(1, " ", 1 )                        = 1
write(1, "2", 12)                        = 1
write(1, " ", 1 )                        = 1
write(1, "3", 13)                        = 1
write(1, " ", 1 )                        = 1
write(1, "\n", 1
)                       = 1
write(1, "Sorted: ", 8Sorted: )                 = 8
write(1, "2", 12)                        = 1
write(1, " ", 1 )                        = 1
write(1, "3", 13)                        = 1
write(1, " ", 1 )                        = 1
write(1, "4", 14)                        = 1
write(1, " ", 1 )                        = 1
write(1, "5", 15)                        = 1
write(1, " ", 1 )                        = 1
write(1, "9", 19)                        = 1
write(1, " ", 1 )                        = 1
write(1, "\n", 1
)                       = 1
write(1, " (v", 3 (v)                      = 3
write(1, "1", 11)                        = 1
write(1, ": bubble)\n", 10: bubble)
)             = 10
write(1, "> ", 2> )                       = 2
read(0, 0
"0\n", 255)                     = 2
munmap(0x7bac5d9b2000, 16400)           = 0
munmap(0x7bac5d9ad000, 16416)           = 0
openat(AT_FDCWD, "./area2.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\0"..., 832) =
832
fstat(3, {st_mode=S_IFREG|0775, st_size=15184, ...}) = 0
getcwd("/home/tim/OS_labs/lab_04/src", 128) = 29
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7bac5d9b2000
mmap(0x7bac5d9b3000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1000) = 0x7bac5d9b3000
mmap(0x7bac5d9b4000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
= 0x7bac5d9b4000
```

```
mmap(0x7bac5d9b5000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7bac5d9b5000
close(3)                                = 0
mprotect(0x7bac5d9b5000, 4096, PROT_READ) = 0
openat(AT_FDCWD, "./sort2.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\0"..., 832) =
832
fstat(3, {st_mode=S_IFREG|0775, st_size=15616, ...}) = 0
getcwd("/home/tim/OS_labs/lab_04/src", 128) = 29
mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7bac5d9ad000
mmap(0x7bac5d9ae000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1000) = 0x7bac5d9ae000
mmap(0x7bac5d9af000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000)
= 0x7bac5d9af000
mmap(0x7bac5d9b0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7bac5d9b0000
close(3)                                = 0
mprotect(0x7bac5d9b0000, 4096, PROT_READ) = 0
write(1, "Loaded version 2\n", 17Loaded version 2
)      = 17
write(1, "> ", 2> )                      = 2
read(0, 2 3 5 9 1
"2 3 5 9 1\n", 255)             = 10
write(1, "Array: ", 7Array: )                  = 7
write(1, "5", 15)                       = 1
write(1, " ", 1 )                       = 1
write(1, "9", 19)                       = 1
write(1, " ", 1 )                       = 1
write(1, "1", 11)                       = 1
write(1, " ", 1 )                       = 1
write(1, "\n", 1
)                       = 1
write(1, "Sorted: ", 8Sorted: )                 = 8
write(1, "1", 11)                       = 1
write(1, " ", 1 )                       = 1
write(1, "5", 15)                       = 1
write(1, " ", 1 )                       = 1
write(1, "9", 19)                       = 1
write(1, " ", 1 )                       = 1
write(1, "\n", 1
)                       = 1
write(1, " (v", 3 (v)                     = 3
write(1, "2", 12)                       = 1
write(1, ": quick)\n", 9: quick)
)               = 9
write(1, "> ", 2> )                      = 2
read(0, exit
```

```
"exit\n", 255)                  = 5
write(1, "Exiting\n", 8Exiting
)               = 8
munmap(0x7bac5d9b2000, 16400)           = 0
munmap(0x7bac5d9ad000, 16416)           = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были исследованы и применены на практике два принципиально различных метода работы с библиотеками: статическое связывание на стадии компиляции и динамическая загрузка в процессе выполнения программы.

Работа наглядно показала ключевые различия между этими подходами применительно к операциям вычисления площади геометрических фигур и сортировки числовых массивов. Каждый из методов обладает специфическими преимуществами: статическая линковка даёт выигрыш в быстродействии благодаря непосредственным вызовам функций, тогда как динамическая загрузка открывает возможность мгновенного переключения между альтернативными реализациями алгоритмов без пересборки всей программы.