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: Сальманов Э.Р.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 05.12.25

Москва, 2025

Постановка задачи

Вариант 17.

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.

Контракты:

1. Подсчёт количества простых чисел на отрезке [a, b] (a, b – натуральные):
   Сигнатура функции: `int prime_count(int a, int b);`
   ● Реализация №1: Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.
   ● Реализация №2: Решето Эратосфена
2. Расчет значения числа π при заданной длине ряда (k):
   Сигнатура функции: `float pi(int k);`
   ● Реализация №1: Ряд Лейбница
   ● Реализация №2: Формула Валлиса

Общий метод и алгоритм решения

Использованные системные вызовы:

- *ssize_t read(int fd, void *buf, size_t count);* - читает данные из файлового дескриптора и записывает их в буфер.
- *ssize_t write(int fd, const void *buf, size_t count);* - записывает данные из буфера в файловый дескриптор.
- *void *dlopen(const char *path, int flags);* - поиск и загрузка динамической библиотеки в память.
- *int dlclose(void *handle);* - выгрузка из памяти динамической библиотеки и освобождение ресурсов, связанных с ней.
- *void *dlsym(void *handle, const char *symbol);* - поиск и возврат запрошенного символа в загруженной динамической библиотеке.

Создано две программы с демонстрацией различных подходов к линковке динамических библиотек. В первой программе происходит обычный вызов функций-контрактов, как если бы они были статически слинкованы с основной программой. Во второй программе происходит динамическая загрузка контрактов: производится загрузка динамической библиотеки с помощью *dlopen* с нужной сейчас реализацией функций-контрактов, происходит получение адресов функций посредством вызова *dlsym.* Завершение происходит при вводе команды *exit*. При завершении выполнения второй программы (нормальном и аварийном) происходит освобождение ресурсов библиотеки с помощью *dlclose.*

Код программы

lab4_1.c

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

#include <lab4/commands.h>

#define MAX_BUFFER_SIZE 1024


int main(void) {
    char command[MAX_BUFFER_SIZE + 1] = {0};
    while (read(STDIN_FILENO, command, MAX_BUFFER_SIZE) > 0) {
        if (strncmp(command, EXIT_COMMAND, strlen(EXIT_COMMAND)) == 0) {
            break;
        }

        char *type = strtok(command, DELIMITERS);

        if (!type) {
            memset(command, 0, MAX_BUFFER_SIZE);

            continue;
        }

        char message[MAX_BUFFER_SIZE + 1] = {0};
        if (strcmp(type, "1") == 0) {
```

```
            ProcessCommand1(prime_count, message);
        } else if (strcmp(type, "2") == 0) {
            ProcessCommand2(pi, message);
        } else {
            sprintf(message, "Invalid command!\n");
        }

        write(STDOUT_FILENO, message, strlen(message));

        memset(command, 0, MAX_BUFFER_SIZE);
    }

    return 0;
}
```

lab4_2.c

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

#include <lab4/commands.h>

#define MAX_BUFFER_SIZE 1024


static PrimeCountFn prime_count_fn;

static PiFn pi_fn;

static const char *IMPL_1_NAME = "./libmathematics_impl1.so";
static const char *IMPL_2_NAME = "./libmathematics_impl2.so";


int set_mathematics_impl(int impl_number, void **library) {
    void *tmp_library = NULL;

    if (impl_number == 1) {
        tmp_library = dlopen(IMPL_1_NAME, RTLD_LOCAL | RTLD_NOW);
    } else if (impl_number == 2) {
        tmp_library = dlopen(IMPL_2_NAME, RTLD_LOCAL | RTLD_NOW);
    } else {
        char message[] = "Invalid impl number!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return 1;
    }

    if (!tmp_library) {
```

```
        char message[] = "Can`t change impl!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return 1;
    }

    prime_count_fn = (PrimeCountFn) dlsym(tmp_library, "prime_count");

    if (!prime_count_fn) {
        dlclose(tmp_library);

        char message[] = "Can`t find function `prime_count`!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return 1;
    }

    pi_fn = (PiFn) dlsym(tmp_library, "pi");

    if (!pi_fn) {
        dlclose(tmp_library);

        char message[] = "Can`t find function `pi`!\n";
        write(STDOUT_FILENO, message, sizeof(message));

        return 1;
    }

    if (*library) {
        dlclose(*library);
    }

    *library = tmp_library;

    return 0;
}

int main(void) {
    int impl_number = 1;

    void *library = NULL;

    if (set_mathematics_impl(impl_number, &library)) {
        return 1;
    }

    char command[MAX_BUFFER_SIZE + 1] = {0};
    while (read(STDIN_FILENO, command, MAX_BUFFER_SIZE) > 0) {
        if (strncmp(command, EXIT_COMMAND, strlen(EXIT_COMMAND)) == 0) {
```

```
            break;
        }

        char *type = strtok(command, DELIMITERS);

        if (!type) {
            memset(command, 0, MAX_BUFFER_SIZE);

            continue;
        }

        char message[MAX_BUFFER_SIZE + 1] = {0};
        if (strcmp(type, "0") == 0) {
            impl_number = impl_number == 1 ? 2 : 1;

            if (set_mathematics_impl(impl_number, &library)) {
                dlclose(library);

                return 1;
            }

            sprintf(message, "Impl successfully changed!\n");
        } else if (strcmp(type, "1") == 0) {
            ProcessCommand1(prime_count_fn, message);
        } else if (strcmp(type, "2") == 0) {
            ProcessCommand2(pi_fn, message);
        } else {
            sprintf(message, "Invalid command!\n");
        }

        write(STDOUT_FILENO, message, strlen(message));

        memset(command, 0, MAX_BUFFER_SIZE);
    }

    dlclose(library);

    return 0;
}
```

commands.h

```
#ifndef MAI_OS_2025_COMMANDS_H
#define MAI_OS_2025_COMMANDS_H

#include <lab4/mathematics.h>

#define EXIT_COMMAND "exit\n"
#define DELIMITERS   " \t\n"
```

```
int ProcessCommand1(PrimeCountFn prime_count_fn,
                    char *message);

int ProcessCommand2(PiFn pi_fn,
                    char *message);

#endif //MAI_OS_2025_COMMANDS_H
```

commands.c

```
#include <float.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <lab4/commands.h>


int ParseInt(char *string,
             int *number) {
    if (!string) {
        return 1;
    }

    char *end;

    long tmp = strtol(string, &end, 10);

    if (tmp < INT_MIN || tmp > INT_MAX) {
        return 1;
    }

    if (*end != '\0') {
        return 1;
    }

    *number = (int) tmp;

    return 0;
}

int ProcessCommand1(PrimeCountFn prime_count_fn,
                    char *message) {
    int a = 0, b = 0;

    if (ParseInt(strtok(NULL, DELIMITERS), &a)) {
        sprintf(message, "Can`t parse number!\n");
```

```
        return 0;
    }

    if (ParseInt(strtok(NULL, DELIMITERS), &b)) {
        sprintf(message, "Can`t parse number!\n");

        return 0;
    }

    if (strtok(NULL, DELIMITERS) != NULL) {
        sprintf(message, "Unknown arguments!\n");

        return 0;
    }

    int prime_result = prime_count_fn(a, b);

    if (prime_result == -1) {
        sprintf(message, "Invalid number range!\n");
    } else {
        sprintf(message, "Primes count in [a, b]: %d\n", prime_result);
    }

    return 0;
}

int ProcessCommand2(PiFn pi_fn,
                    char *message) {
    int k = 0;

    if (ParseInt(strtok(NULL, DELIMITERS), &k)) {
        sprintf(message, "Can`t parse number!\n");

        return 0;
    }

    if (strtok(NULL, DELIMITERS) != NULL) {
        sprintf(message, "Unknown arguments!\n");

        return 0;
    }

    float pi_result = pi_fn(k);

    if (pi_result + 1.f < FLT_EPSILON) {
        sprintf(message, "Invalid number of series!\n");
    } else {
        sprintf(message, "Pi by series: %f\n", pi_result);
    }
```

```
    return 0;
}
```

mathematics.h

```
#ifndef MAI_OS_2025_MATHEMATICS_H
#define MAI_OS_2025_MATHEMATICS_H

typedef int (*PrimeCountFn)(int a,
                            int b);

typedef float (*PiFn)(int k);

int prime_count(int a,
                int b);

float pi(int k);

#endif //MAI_OS_2025_MATHEMATICS_H
```

mathematics_impl1.c

```
#include <stdbool.h>

#include <lab4/mathematics.h>


int prime_count(int a,
                int b) {
    if (a < 1 || b < 1
     || a >= b) {
        return -1;
    }

    int primes_count = 0;

    for (int i = a; i <= b; ++i) {
        bool prime = true;

        for (int j = 2; j < i; ++j) {
            if (i % j == 0) {
                prime = false;

                break;
            }
        }

        if (prime) {
            ++primes_count;
        }
```

```
    }

    return primes_count;
}

float pi(int k) {
    if (k <= 0) {
        return -1.f;
    }

    double result = 0.f;

    for (int n = 0, sign = 1; n < k; ++n, sign = -sign) {
        result += (double) sign / (double) (2 * n + 1);
    }

    return 4.f * (float) result;
}
```

mathematics_impl2.c

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

#include <lab4/mathematics.h>


int prime_count(int a,
                int b) {
    if (a < 1 || b < 1
     || a >= b) {
        return -1;
     }

    bool *primes = (bool *) malloc((b + 1) * sizeof(bool));

    if (!primes) {
        return 0;
    }

    memset(primes, true, (b + 1) * sizeof(bool));

    for (int i = 2; i <= b + 1; i++) {
        if (!primes[i]) {
            continue;
        }

        for (int j = 2 * i; j <= b + 1; j += i) {
            primes[j] = false;
```

```
        }
    }

    int primes_count = 0;
    for (int i = a; i <= b; ++i) {
        if (primes[i]) {
            ++primes_count;
        }
    }

    free(primes);

    return primes_count;
}

float pi(int k) {
    if (k < 1) {
        return -1.f;
    }

    double result = 4.0 / 3.0;

    for (int n = 2; n <= k; ++n) {
        double n2 = 2.0 * (double) n;
        double up = n2 * n2;
        double down = (n2 - 1.0) * (n2 + 1.0);

        result *= up / down;
    }

    return 2.f * (float) result;
}
```

Протокол работы программы

Тестирование:

```
$ ./lab4_1
1 2 10
1 4 18
2 1000
2 100000
exit
$ ./lab4_2
1 2 10
1 4 18
2 1000
2 100000
0
```

```
1 2 10
1 4 18
2 1000
2 100000
0
exit
```

Strace:

lab4_1

```
46980 execve("./lab4_1", ["./lab4_1"], 0x7ffeae76a8d8 /* 29 vars */) = 0
46980 brk(NULL)                           = 0x5baa0fd3d000
46980 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x701246377000
46980 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
46980 openat(AT_FDCWD,
"/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4/glibc-hwcaps/x86-64-v3/libmathematics_im
pl1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
46980 newfstatat(AT_FDCWD,
"/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4/glibc-hwcaps/x86-64-v3/",
0x7ffe0df01930, 0) = -1 ENOENT (No such file or directory)
46980 openat(AT_FDCWD,
"/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4/glibc-hwcaps/x86-64-v2/libmathematics_im
pl1.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
46980 newfstatat(AT_FDCWD,
"/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4/glibc-hwcaps/x86-64-v2/",
0x7ffe0df01930, 0) = -1 ENOENT (No such file or directory)
46980 openat(AT_FDCWD,
"/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4/libmathematics_impl1.so",
O_RDONLY|O_CLOEXEC) = 3
46980 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
46980 fstat(3, {st_mode=S_IFREG|0755, st_size=16848, ...}) = 0
46980 mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x701246372000
46980 mmap(0x701246373000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x701246373000
46980 mmap(0x701246374000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x701246374000
46980 mmap(0x701246375000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x701246375000
46980 close(3)                            = 0
46980 openat(AT_FDCWD, "/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4/libc.so.6",
O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory)
46980 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
46980 fstat(3, {st_mode=S_IFREG|0644, st_size=23471, ...}) = 0
46980 mmap(NULL, 23471, PROT_READ, MAP_PRIVATE, 3, 0) = 0x70124636c000
46980 close(3)                            = 0
46980 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
46980 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
```

```
     46980 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
     46980 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
     46980 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
     46980 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x701246000000
     46980 mmap(0x701246028000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x701246028000
     46980 mmap(0x7012461b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7012461b0000
     46980 mmap(0x7012461ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7012461ff000
     46980 mmap(0x701246205000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x701246205000
     46980 close(3)                          = 0
     46980 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x701246369000
     46980 arch_prctl(ARCH_SET_FS, 0x701246369740) = 0
     46980 set_tid_address(0x701246369a10)   = 46980
     46980 set_robust_list(0x701246369a20, 24) = 0
     46980 rseq(0x70124636a060, 0x20, 0, 0x53053053) = 0
     46980 mprotect(0x7012461ff000, 16384, PROT_READ) = 0
     46980 mprotect(0x701246375000, 4096, PROT_READ) = 0
     46980 mprotect(0x5ba9e269a000, 4096, PROT_READ) = 0
     46980 mprotect(0x7012463af000, 8192, PROT_READ) = 0
     46980 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
     46980 munmap(0x70124636c000, 23471)     = 0
     46980 read(0, "1 2 10\n", 1024)         = 7
     46980 write(1, "Primes count in [a, b]: 4\n", 26) = 26
     46980 read(0, "1 4 18\n", 1024)         = 7
     46980 write(1, "Primes count in [a, b]: 5\n", 26) = 26
     46980 read(0, "2 1000\n", 1024)         = 7
     46980 write(1, "Pi by series: 3.140593\n", 23) = 23
     46980 read(0, "2 100000\n", 1024)       = 9
     46980 write(1, "Pi by series: 3.141583\n", 23) = 23
     46980 read(0, "exit\n", 1024)           = 5
     46980 exit_group(0)                     = ?
     46980 +++ exited with 0 +++
```

lab4_2

```
     49533 execve("./lab4_2", ["./lab4_2"], 0x7fff2e83fdd8 /* 29 vars */) = 0
     49533 brk(NULL)                         = 0x5b74c3837000
     49533 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x747f69b89000
     49533 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
     49533 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
     49533 fstat(3, {st_mode=S_IFREG|0644, st_size=23471, ...}) = 0
```

```
     49533 mmap(NULL, 23471, PROT_READ, MAP_PRIVATE, 3, 0) = 0x747f69b83000
     49533 close(3)                          = 0
     49533 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
     49533 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
     49533 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
     49533 fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
     49533 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
     49533 mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x747f69800000
     49533 mmap(0x747f69828000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x747f69828000
     49533 mmap(0x747f699b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x747f699b0000
     49533 mmap(0x747f699ff000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x747f699ff000
     49533 mmap(0x747f69a05000, 52624, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x747f69a05000
     49533 close(3)                          = 0
     49533 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x747f69b80000
     49533 arch_prctl(ARCH_SET_FS, 0x747f69b80740) = 0
     49533 set_tid_address(0x747f69b80a10)   = 49533
     49533 set_robust_list(0x747f69b80a20, 24) = 0
     49533 rseq(0x747f69b81060, 0x20, 0, 0x53053053) = 0
     49533 mprotect(0x747f699ff000, 16384, PROT_READ) = 0
     49533 mprotect(0x5b749247b000, 4096, PROT_READ) = 0
     49533 mprotect(0x747f69bc1000, 8192, PROT_READ) = 0
     49533 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
= 0
     49533 munmap(0x747f69b83000, 23471)     = 0
     49533 getrandom("\x2f\xc2\xd3\x7a\x6e\xa5\x00\xc9", 8, GRND_NONBLOCK) = 8
     49533 brk(NULL)                         = 0x5b74c3837000
     49533 brk(0x5b74c3858000)               = 0x5b74c3858000
     49533 openat(AT_FDCWD, "./libmathematics_impl1.so", O_RDONLY|O_CLOEXEC) = 3
     49533 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
     49533 fstat(3, {st_mode=S_IFREG|0755, st_size=16848, ...}) = 0
     49533 getcwd("/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4", 128) = 51
     49533 mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x747f69b84000
     49533 mmap(0x747f69b85000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x747f69b85000
     49533 mmap(0x747f69b86000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x747f69b86000
     49533 mmap(0x747f69b87000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x747f69b87000
     49533 close(3)                          = 0
     49533 mprotect(0x747f69b87000, 4096, PROT_READ) = 0
```

```
     49533 read(0, "1 2 10\n", 1024)         = 7
     49533 write(1, "Primes count in [a, b]: 4\n", 26) = 26
     49533 read(0, "1 4 18\n", 1024)         = 7
     49533 write(1, "Primes count in [a, b]: 5\n", 26) = 26
     49533 read(0, "2 1000\n", 1024)         = 7
     49533 write(1, "Pi by series: 3.140593\n", 23) = 23
     49533 read(0, "2 100000\n", 1024)       = 9
     49533 write(1, "Pi by series: 3.141583\n", 23) = 23
     49533 read(0, "0\n", 1024)              = 2
     49533 openat(AT_FDCWD, "./libmathematics_impl2.so", O_RDONLY|O_CLOEXEC) = 3
     49533 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
     49533 fstat(3, {st_mode=S_IFREG|0755, st_size=17928, ...}) = 0
     49533 getcwd("/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4", 128) = 51
     49533 mmap(NULL, 16424, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x747f69b7b000
     49533 mmap(0x747f69b7c000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x747f69b7c000
     49533 mmap(0x747f69b7d000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x747f69b7d000
     49533 mmap(0x747f69b7e000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x747f69b7e000
     49533 close(3)                          = 0
     49533 mprotect(0x747f69b7e000, 4096, PROT_READ) = 0
     49533 munmap(0x747f69b84000, 16400)     = 0
     49533 write(1, "Impl successfully changed!\n", 27) = 27
     49533 read(0, "1 2 10\n", 1024)         = 7
     49533 write(1, "Primes count in [a, b]: 4\n", 26) = 26
     49533 read(0, "1 4 18\n", 1024)         = 7
     49533 write(1, "Primes count in [a, b]: 5\n", 26) = 26
     49533 read(0, "2 1000\n", 1024)         = 7
     49533 write(1, "Pi by series: 3.140808\n", 23) = 23
     49533 read(0, "2 100000\n", 1024)       = 9
     49533 write(1, "Pi by series: 3.141585\n", 23) = 23
     49533 read(0, "0\n", 1024)              = 2
     49533 openat(AT_FDCWD, "./libmathematics_impl1.so", O_RDONLY|O_CLOEXEC) = 3
     49533 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
     49533 fstat(3, {st_mode=S_IFREG|0755, st_size=16848, ...}) = 0
     49533 getcwd("/home/eldar/MAI_OS_2025/cmake-build-debug-wsl/lab4", 128) = 51
     49533 mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x747f69b84000
     49533 mmap(0x747f69b85000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x747f69b85000
     49533 mmap(0x747f69b86000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x2000) = 0x747f69b86000
     49533 mmap(0x747f69b87000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x747f69b87000
     49533 close(3)                          = 0
     49533 mprotect(0x747f69b87000, 4096, PROT_READ) = 0
     49533 munmap(0x747f69b7b000, 16424)     = 0
```

```
49533 write(1, "Impl successfully changed!\n", 27) = 27
49533 read(0, "exit\n", 1024)           = 5
49533 munmap(0x747f69b84000, 16400)     = 0
49533 exit_group(0)                     = ?
49533 +++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы были успешно приобретены практические навыки в создании и использовании библиотек статической и динамической линковки в ОС и обработке ошибок вызова системных вызовов ОС. Также был освоен способ создания этих библиотек с использованием современных средств сборки.

Серьёзных проблем при выполнении лабораторной работы не возникло.