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: Дылдин С.В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 21.12.25

Москва, 2025

Постановка задачи

Вариант 33.

Расчет значения числа e (основание натурального логарифма):
Сигнатура функции: `float e(int x);`

- Реализация №1: $(1 + 1 / x)^x$
- Реализация №2: Сумма ряда по n от 0 до x, где элементы ряда равны: $(1 / (n!))$

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:

- int openat(int dirfd, const char *pathname, int flags, …); открывает файл библиотеки .so, возвращает файловый дескриптор
- void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset); отображает в виртуальную память (dlopen)
- int munmap(void *addr, size_t lenght); (dlclose) отключает отображение
- int mprotect… меняет права доступа к области памяти

Описание:
Пример работы статической и динамической библиотек .

- Реализованы 2 динамических библиотеки с функциями:
  1. расчет значения числа e
  2. сортировка массива
- в first_main.c библиотека подгружается на этапе компиляции и далее с функциями из этой библиотеки возможна работа через пользовательский ввод
- в second_main.c программа не знает о библиотеке на моменте компиляции, она подгружает их уже при выполнении, в моменте использования dlopen в runtime. Предоставляет более гибкое взаимодействие с библиотеками, т.к во время выполнения программы возможна замена одной библиотеки другой.

Код программы

first.h

```h
#ifndef IMPLEMENTATION1_H
#define IMPLEMENTATION1_H

#include <stddef.h>

float e(int x);

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

#endif
```

first.c

```c
#include "first.h"
#include <math.h>
#include <stdlib.h>

float e(int x) {
    if (x == 0) {
        return 1.0f;
    }
    return powf(1.0f + 1.0f / x, x);
}

int* sort(int* array, size_t n) {
    if (array == NULL || n == 0) {
        return array;
    }
    for (size_t i = 0; i < n - 1; i++) {
        for (size_t j = 0; j < n - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }

    return array;
}
```

second.h

```h
#ifndef IMPLEMENTATION2_H
```

```h
#define IMPLEMENTATION2_H

#include <stddef.h>

float e(int x);

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

#endif
```

second.c

```c
#include "second.h"
#include <stdlib.h>

static unsigned long long factorial(int n) {
    if (n <= 1){
        return 1;
    }
    unsigned long long result = 1;
    for (int i = 2; i <= n; i++){
        result *= i;
    }
    return result;
}

float e(int x) {
    if (x < 0){
        x = 0;
    }

    float sum = 0.0f;
    for (int n = 0; n <= x; n++){
        sum += 1.0f / factorial(n);
    }
    return sum;
}

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

    while (1) {
        do {
            i++;
        } while (array[i] < pivot);
        do {
            j--;
```

```
        } while (array[j] > pivot);
        if (i >= j) {
            return j;
        }
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }
}

//сортировка Хоара
static void quicksort(int* array, int low, int high) {
    if (low < high) {
        int pi = partition_hoare(array, low, high);
        quicksort(array, low, pi);
        quicksort(array, pi + 1, high);
    }
}


int* sort(int* array, size_t n) {
    if (array == NULL || n <= 1) {
        return array;
    }

    quicksort(array, 0, n - 1);
    return array;
}
```

first_main.c

```
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include "l1/first.h"

#define BUFSIZE 1024

int readLine(char *buffer, const ssize_t size) {
    ssize_t bytes_read = read(STDIN_FILENO, buffer, size);
    if (bytes_read < 0) {
        return 1;
    }
    if (bytes_read == 0 || buffer[0] == '\n') {
        return 2;
    }
    if (bytes_read == size && buffer[size - 1] != '\n') {
```

```
        return 3;
    }
    if (buffer[bytes_read - 1] == '\n') {
        buffer[bytes_read - 1] = '\0';
    } else {
        buffer[bytes_read] = '\0';
    }
    return 0;
}

int main() {
    char user_input[BUFSIZE];
    int running = 1;

    do {

        int err = readLine(user_input, BUFSIZE);
        switch (err) {
            case 1: {
                const char msg[] = "error: failed to read stdin\n";
                write(STDERR_FILENO, msg, sizeof(msg) - 1);
                exit(EXIT_FAILURE);
            }
            case 2: {
                running = 0;
            } break;
            case 3: {
                const char msg[] = "error: buffer overflow\n";
                write(STDERR_FILENO, msg, sizeof(msg) - 1);
                exit(EXIT_FAILURE);
            }
        }

        if (!running) {
            break;
        }

        if (strcmp(user_input, "exit") == 0) {
            break;
        }

        char *arg = strtok(user_input, " \t\n");
        if (!arg) {
            const char msg[] = "Ошибка: неверная команда\n";
            write(STDERR_FILENO, msg, sizeof(msg) - 1);
            continue;
        }

        int command = atoi(arg);
```

```
        switch (command) {
            case 1: {
                char *x_str = strtok(NULL, " \t\n");
                if (!x_str) {
                    const char msg[] = "Ошибка: необходим параметр x\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    break;
                }

                int x = atoi(x_str);
                float result = e(x);

                char result_msg[BUFSIZE];
                int msg_size = snprintf(result_msg, BUFSIZE, "e(%d) = %.6f\n", x,
result);
                write(STDOUT_FILENO, result_msg, msg_size);
            } break;

            case 2: {
                char *n_str = strtok(NULL, " \t\n");
                if (!n_str) {
                    const char msg[] = "Ошибка: необходимо указать размер массива n >
0\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    break;
                }

                int n = atoi(n_str);
                if (n <= 0) {
                    const char msg[] = "Ошибка: необходимо указать размер массива n >
0\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    break;
                }

                int* array = (int*)malloc(n * sizeof(int));
                if (!array) {
                    const char msg[] = "Ошибка выделения памяти\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    break;
                }

                int count = 0;
                int success = 1;

                for (int i = 0; i < n; i++) {
                    char* token = strtok(NULL, " \t\n");
                    if (!token) {
                        const char msg[] = "Ошибка: недостаточно элементов массива\n";
                        write(STDERR_FILENO, msg, sizeof(msg) - 1);
```

```
                    free(array);
                    success = 0;
                    break;
                }
                array[i] = atoi(token);
                count++;
            }

            if (success) {
                if (count != n) {
                    char msg[BUFSIZE];
                    int msg_size = snprintf(msg, BUFSIZE, "Ошибка: получено %d
элементов вместо %d\n", count, n);
                    write(STDERR_FILENO, msg, msg_size);
                    free(array);
                    break;
                }

                char output[BUFSIZE];
                int offset = 0;

                offset += snprintf(output + offset, BUFSIZE - offset, "Исходный
массив: ");
                for (int i = 0; i < n; i++) {
                    offset += snprintf(output + offset, BUFSIZE - offset, "%d ",
array[i]);
                }
                offset += snprintf(output + offset, BUFSIZE - offset, "\n");
                write(STDOUT_FILENO, output, offset);

                sort(array, n);

                offset = 0;
                offset += snprintf(output + offset, BUFSIZE - offset,
"Отсортированный массив: ");
                for (int i = 0; i < n; i++) {
                    offset += snprintf(output + offset, BUFSIZE - offset, "%d ",
array[i]);
                }
                offset += snprintf(output + offset, BUFSIZE - offset, "\n");
                write(STDOUT_FILENO, output, offset);

                free(array);
            }
        } break;

        default: {
            char msg[BUFSIZE];
            int msg_size = snprintf(msg, BUFSIZE, "Ошибка: неизвестная команда %d\n",
command);
            write(STDERR_FILENO, msg, msg_size);
        } break;
    }
```

```
    } while (running);

    return 0;
}
```

second_main.c

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

#define BUFSIZE 1024

typedef float (*e_func_t)(int);
typedef int* (*sort_func_t)(int*, size_t);

typedef struct {
    void* handle;
    e_func_t e_func;
    sort_func_t sort_func;
    int lib_number;
} Library;

Library current_lib = {NULL, NULL, NULL, 1};

int readLine(char *buffer, const ssize_t size) {
    ssize_t bytes_read = read(STDIN_FILENO, buffer, size);
    if (bytes_read < 0) {
        return 1;
    }
    if (bytes_read == 0 || buffer[0] == '\n') {
        return 2;
    }
    if (bytes_read == size && buffer[size - 1] != '\n') {
        return 3;
    }
    if (buffer[bytes_read - 1] == '\n') {
        buffer[bytes_read - 1] = '\0';
    } else {
        buffer[bytes_read] = '\0';
    }
    return 0;
}
```

```
int load_library(int lib_num) {
    if (current_lib.handle) {
        dlclose(current_lib.handle);
    }

    const char* lib_path;
    if (lib_num == 1) {
        lib_path = "./l1/libfirst.so";
    } else {
        lib_path = "./l2/libsecond.so";
    }

    current_lib.handle = dlopen(lib_path, RTLD_LAZY);
    if (!current_lib.handle) {
        char msg[BUFSIZE];
        int msg_size = snprintf(msg, BUFSIZE, "Ошибка загрузки библиотеки: %s\n",
dlerror());
        write(STDERR_FILENO, msg, msg_size);
        return 0;
    }

    current_lib.e_func = (e_func_t)dlsym(current_lib.handle, "e");
    if (!current_lib.e_func) {
        char msg[BUFSIZE];
        int msg_size = snprintf(msg, BUFSIZE, "Ошибка загрузки функции e: %s\n",
dlerror());
        write(STDERR_FILENO, msg, msg_size);
        dlclose(current_lib.handle);
        return 0;
    }

    current_lib.sort_func = (sort_func_t)dlsym(current_lib.handle, "sort");
    if (!current_lib.sort_func) {
        char msg[BUFSIZE];
        int msg_size = snprintf(msg, BUFSIZE, "Ошибка загрузки функции sort: %s\n",
dlerror());
        write(STDERR_FILENO, msg, msg_size);
        dlclose(current_lib.handle);
        return 0;
    }

    current_lib.lib_number = lib_num;

    char output[BUFSIZE];
    int offset = 0;
    offset += snprintf(output + offset, BUFSIZE - offset, "Загружена библиотека %d\n",
lib_num);
    if (lib_num == 1) {
        offset += snprintf(output + offset, BUFSIZE - offset, "Функция e: (1 +
1/x)^x\n");
        offset += snprintf(output + offset, BUFSIZE - offset, "Сортировка:
Пузырьковая\n");
    } else {
```

```
        offset += snprintf(output + offset, BUFSIZE - offset, "Функция e: Сумма ряда 1/n!\n");
        offset += snprintf(output + offset, BUFSIZE - offset, "Сортировка: Быстрая сортировка Хоара\n");
    }
    write(STDOUT_FILENO, output, offset);

    return 1;
}

int main() {
    char user_input[BUFSIZE];
    int running = 1;

    if (!load_library(1)) {
        const char msg[] = "Не удалось загрузить начальную библиотеку\n";
        write(STDERR_FILENO, msg, sizeof(msg) - 1);
        return 1;
    }

    do {
        int err = readLine(user_input, BUFSIZE);
        switch (err) {
            case 1: {
                const char msg[] = "error: failed to read stdin\n";
                write(STDERR_FILENO, msg, sizeof(msg) - 1);
                exit(EXIT_FAILURE);
            }
            case 2: {
                running = 0;
            } break;
            case 3: {
                const char msg[] = "error: buffer overflow\n";
                write(STDERR_FILENO, msg, sizeof(msg) - 1);
                exit(EXIT_FAILURE);
            }
        }

        if (!running) {
            break;
        }

        if (strcmp(user_input, "exit") == 0) {
            break;
        }

        char *arg = strtok(user_input, " \t\n");
        if (!arg) {
            const char msg[] = "Ошибка: неверная команда\n";
            write(STDERR_FILENO, msg, sizeof(msg) - 1);
            continue;
```

```
        }

        int command = atoi(arg);

        switch (command) {
            case 0: {
                int new_lib = (current_lib.lib_number == 1) ? 2 : 1;
                if (!load_library(new_lib)) {
                    char msg[BUFSIZE];
                    int msg_size = snprintf(msg, BUFSIZE, "Не удалось загрузить
библиотеку %d\n", new_lib);
                    write(STDERR_FILENO, msg, msg_size);
                }
            } break;

            case 1: {
                char *x_str = strtok(NULL, " \t\n");
                if (!x_str) {
                    const char msg[] = "Ошибка: необходим параметр x\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    break;
                }

                int x = atoi(x_str);
                float result = current_lib.e_func(x);

                char result_msg[BUFSIZE];
                int msg_size = snprintf(result_msg, BUFSIZE, "e(%d) = %.6f (библиотека
%d)\n", x, result, current_lib.lib_number);
                write(STDOUT_FILENO, result_msg, msg_size);
            } break;

            case 2: {
                char *n_str = strtok(NULL, " \t\n");
                if (!n_str) {
                    const char msg[] = "Ошибка: необходимо указать размер массива n >
0\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    break;
                }

                int n = atoi(n_str);
                if (n <= 0) {
                    const char msg[] = "Ошибка: необходимо указать размер массива n >
0\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    break;
                }

                int* array = (int*)malloc(n * sizeof(int));
                if (!array) {
                    const char msg[] = "Ошибка выделения памяти\n";
```

```
                write(STDERR_FILENO, msg, sizeof(msg) - 1);
                break;
            }

            int count = 0;
            int success = 1;

            for (int i = 0; i < n; i++) {
                char* token = strtok(NULL, " \t\n");
                if (!token) {
                    const char msg[] = "Ошибка: недостаточно элементов массива\n";
                    write(STDERR_FILENO, msg, sizeof(msg) - 1);
                    free(array);
                    success = 0;
                    break;
                }
                array[i] = atoi(token);
                count++;
            }

            if (success) {
                if (count != n) {
                    char msg[BUFSIZE];
                    int msg_size = snprintf(msg, BUFSIZE, "Ошибка: получено %d
элементов вместо %d\n", count, n);
                    write(STDERR_FILENO, msg, msg_size);
                    free(array);
                    break;
                }

                char output[BUFSIZE];
                int offset = 0;

                offset += snprintf(output + offset, BUFSIZE - offset, "Исходный
массив: ");
                for (int i = 0; i < n; i++) {
                    offset += snprintf(output + offset, BUFSIZE - offset, "%d ",
array[i]);
                }
                offset += snprintf(output + offset, BUFSIZE - offset, "\n");
                write(STDOUT_FILENO, output, offset);

                current_lib.sort_func(array, n);

                offset = 0;
                offset += snprintf(output + offset, BUFSIZE - offset,
"Отсортированный массив: ");
                for (int i = 0; i < n; i++) {
                    offset += snprintf(output + offset, BUFSIZE - offset, "%d ",
array[i]);
                }
                offset += snprintf(output + offset, BUFSIZE - offset, " (библиотека
%d)\n", current_lib.lib_number);
```

```
                write(STDOUT_FILENO, output, offset);

                free(array);
            }
        } break;

        default: {
            char msg[BUFSIZE];
            int msg_size = snprintf(msg, BUFSIZE, "Ошибка: неизвестная команда %d\n",
command);
            write(STDERR_FILENO, msg, msg_size);
        } break;
    }
} while (running);

if (current_lib.handle) {
    dlclose(current_lib.handle);
}

return 0;
}
```

Протокол работы программы

Тестирование:

```
/mnt/d/MAI/3semestr/OSI/laba4 → ./prog1
1 10
e(10) = 2.593743
1 0
e(0) = 1.000000
1 1
e(1) = 2.000000
2 10 1 2 3 6 5 4 9 8 7 0
Исходный массив: 1 2 3 6 5 4 9 8 7 0
Отсортированный массив: 0 1 2 3 4 5 6 7 8 9
exit


/mnt/d/MAI/3semestr/OSI/laba4 → strace ./prog1
execve("./prog1", ["./prog1"], 0x7ffed304d950 /* 30 vars */) = 0
brk(NULL)                               = 0x5b8cecd9c000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7159bcafc000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "./l1/glibc-hwcaps/x86-64-v3/libfirst.so", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./l1/glibc-hwcaps/x86-64-v2/libfirst.so", O_RDONLY|O_CLOEXEC) = -1
```

```
ENOENT (No such file or directory)
openat(AT_FDCWD, "./l1/libfirst.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|0777, st_size=15552, ...}) = 0
getcwd("/mnt/d/MAI/3semestr/OSI/laba4", 128) = 30
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7159bcaf7000
mmap(0x7159bcaf8000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7159bcaf8000
mmap(0x7159bcaf9000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7159bcaf9000
mmap(0x7159bcafa000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7159bcafa000
close(3)                                = 0

openat(AT_FDCWD, "./l1/glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./l1/glibc-hwcaps/x86-64-v2/libc.so.6", O_RDONLY|O_CLOEXEC) = -1
ENOENT (No such file or directory)
openat(AT_FDCWD, "./l1/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=34311, ...}) = 0
mmap(NULL, 34311, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7159bcaee000
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) = 0x7159bc800000
mmap(0x7159bc828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7159bc828000
mmap(0x7159bc9b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7159bc9b0000
mmap(0x7159bc9ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7159bc9ff000
mmap(0x7159bca05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7159bca05000
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) = 0x7159bc717000
mmap(0x7159bc727000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7159bc727000
mmap(0x7159bc7a6000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x7159bc7a6000
mmap(0x7159bc7fe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x7159bc7fe000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7159bcaeb000
arch_prctl(ARCH_SET_FS, 0x7159bcaeb740) = 0
```

```
set_tid_address(0x7159bcaeba10)         = 210575
set_robust_list(0x7159bcaeba20, 24)     = 0
rseq(0x7159bcaec060, 0x20, 0, 0x53053053) = 0
mprotect(0x7159bc9ff000, 16384, PROT_READ) = 0
mprotect(0x7159bc7fe000, 4096, PROT_READ) = 0
mprotect(0x7159bcafa000, 4096, PROT_READ) = 0
mprotect(0x5b8ceb6cc000, 4096, PROT_READ) = 0
mprotect(0x7159bcb34000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7159bcaee000, 34311)           = 0
read(0, 1 1
"1 1\n", 1024)                  = 4
write(1, "e(1) = 2.000000\n", 16e(1) = 2.000000
)       = 16
read(0, 1 0
"1 0\n", 1024)                  = 4
write(1, "e(0) = 1.000000\n", 16e(0) = 1.000000
)       = 16
read(0, 2 10 1 2 3 6 5 4 9 8 7 0
"2 10 1 2 3 6 5 4 9 8 7 0\n", 1024) = 25
getrandom("\x80\xd6\x80\xd1\xa2\x3c\x3f\x44", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x5b8cecd9c000
brk(0x5b8cecdbd000)                     = 0x5b8cecdbd000
write(1, "\320\230\321\201\321\205\320\276\320\264\320\275\321\213\320\271
\320\274\320\260\321\201\321\201\320\270\320\262: 1"..., 52Исходный массив: 1 2 3 6 5 4 9 8
7 0
) = 52
write(1,
"\320\236\321\202\321\201\320\276\321\200\321\202\320\270\321\200\320\276\320\262\320\260\32
0\275\320\275\321\213\320\271 \320"..., 66Отсортированный массив: 0 1 2 3 4 5 6 7 8 9
) = 66
read(0, exit
"exit\n", 1024)                 = 5
exit_group(0)                           = ?
+++ exited with 0 +++
/mnt/d/MAI/3semestr/OSI/laba4 →
```

```
/mnt/d/MAI/3semestr/OSI/laba4 → ./prog2
Загружена библиотека 1
Функция e: (1 + 1/x)^x
Сортировка: Пузырьковая
1 10
e(10) = 2.593743 (библиотека 1)
1 1
e(1) = 2.000000 (библиотека 1)
2 10 1 2 3 4 6 5 9 8 7 0
Исходный массив: 1 2 3 4 6 5 9 8 7 0
Отсортированный массив: 0 1 2 3 4 5 6 7 8 9  (библиотека 1)
```

```
0
Загружена библиотека 2
Функция e: Сумма ряда 1/n!
Сортировка: Быстрая сортировка Хоара
1 1
e(1) = 2.000000 (библиотека 2)
1 10
e(10) = 2.718282 (библиотека 2)
2 10 1 2 3 6 5 4 9 8 7 0
Исходный массив: 1 2 3 6 5 4 9 8 7 0
Отсортированный массив: 0 1 2 3 4 5 6 7 8 9  (библиотека 2)
0
Загружена библиотека 1
Функция e: (1 + 1/x)^x
Сортировка: Пузырьковая
exit


/mnt/d/MAI/3semestr/OSI/laba4 → strace ./prog2
execve("./prog2", ["./prog2"], 0x7ffc757c0c80 /* 30 vars */) = 0
brk(NULL)                               = 0x56128aef6000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7da3c2a08000
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=34311, ...}) = 0
mmap(NULL, 34311, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7da3c29ff000
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) = 0x7da3c2600000
mmap(0x7da3c2628000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7da3c2628000
mmap(0x7da3c27b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7da3c27b0000
mmap(0x7da3c27ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7da3c27ff000
mmap(0x7da3c2805000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7da3c2805000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7da3c29fc000
arch_prctl(ARCH_SET_FS, 0x7da3c29fc740) = 0
set_tid_address(0x7da3c29fca10)         = 212658
set_robust_list(0x7da3c29fca20, 24)     = 0
rseq(0x7da3c29fd060, 0x20, 0, 0x53053053) = 0
mprotect(0x7da3c27ff000, 16384, PROT_READ) = 0
mprotect(0x5612724f0000, 4096, PROT_READ) = 0
```

```
mprotect(0x7da3c2a40000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7da3c29ff000, 34311)           = 0
getrandom("\x0e\xa7\x91\x78\x3c\x6a\xfe\xdb", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x56128aef6000
brk(0x56128af17000)                     = 0x56128af17000
openat(AT_FDCWD, "./l1/libfirst.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|0777, st_size=15552, ...}) = 0
getcwd("/mnt/d/MAI/3semestr/OSI/laba4", 128) = 30
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7da3c2a03000
mmap(0x7da3c2a04000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7da3c2a04000
mmap(0x7da3c2a05000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7da3c2a05000
mmap(0x7da3c2a06000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7da3c2a06000
close(3)                                = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=34311, ...}) = 0
mmap(NULL, 34311, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7da3c29f3000
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\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) = 0x7da3c290a000
mmap(0x7da3c291a000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7da3c291a000
mmap(0x7da3c2999000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x7da3c2999000
mmap(0x7da3c29f1000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x7da3c29f1000
close(3)                                = 0
mprotect(0x7da3c29f1000, 4096, PROT_READ) = 0
mprotect(0x7da3c2a06000, 4096, PROT_READ) = 0
munmap(0x7da3c29f3000, 34311)           = 0
write(1, "\320\227\320\260\320\263\321\200\321\203\320\266\320\265\320\275\320\260
\320\261\320\270\320\261\320\273\320\270\320\276\321"..., 1173Загружена библиотека 1
Функция e: (1 + 1/x)^x
Сортировка: Пузырьковая
) = 117
read(0, 1 10
"1 10\n", 1024)                 = 5
write(1, "e(10) = 2.593743
(\320\261\320\270\320\261\320\273\320\270\320\276\321\202"..., 42e(10) = 2.593743
(библиотека 1)
) = 42
read(0, 1 1
"1 1\n", 1024)                  = 4
write(1, "e(1) = 2.000000
(\320\261\320\270\320\261\320\273\320\270\320\276\321\202\320"..., 41e(1) = 2.000000
(библиотека 1)
) = 41
read(0, 2 1 2 3 6 5 4 9 8 7 0
"2 1 2 3 6 5 4 9 8 7 0\n", 1024) = 22
```

```
write(1, "\320\230\321\201\321\205\320\276\320\264\320\275\321\213\320\271
\320\274\320\260\321\201\321\201\320\270\320\262: 2"..., 34Исходный массив: 2
) = 34
write(1,
"\320\236\321\202\321\201\320\276\321\200\321\202\320\270\321\200\320\276\320\262\320\260\32
0\275\320\275\321\213\320\271 \320"..., 73Отсортированный массив: 2  (библиотека 1)
) = 73
read(0, 0
"0\n", 1024)                    = 2
munmap(0x7da3c2a03000, 16408)           = 0
munmap(0x7da3c290a000, 950296)          = 0
openat(AT_FDCWD, "./l2/libsecond.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|0777, st_size=15312, ...}) = 0
getcwd("/mnt/d/MAI/3semestr/OSI/laba4", 128) = 30
mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7da3c2a03000
mmap(0x7da3c2a04000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7da3c2a04000
mmap(0x7da3c2a05000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7da3c2a05000
mmap(0x7da3c2a06000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7da3c2a06000
close(3)                                = 0
mprotect(0x7da3c2a06000, 4096, PROT_READ) = 0
write(1, "\320\227\320\260\320\263\321\200\321\203\320\266\320\265\320\275\320\260
\320\261\320\270\320\261\320\273\320\270\320\276\321"..., 154Загружена библиотека 2
Функция e: Сумма ряда 1/n!
Сортировка: Быстрая сортировка Хоара
) = 154
read(0, 1 1
"1 1\n", 1024)                  = 4
write(1, "e(1) = 2.000000
(\320\261\320\270\320\261\320\273\320\270\320\276\321\202\320"..., 41e(1) = 2.000000
(библиотека 2)
) = 41
read(0, 2 3 6 5 2
"2 3 6 5 2\n", 1024)            = 10
write(1, "\320\230\321\201\321\205\320\276\320\264\320\275\321\213\320\271
\320\274\320\260\321\201\321\201\320\270\320\262: 6"..., 38Исходный массив: 6 5 2
) = 38
write(1,
"\320\236\321\202\321\201\320\276\321\200\321\202\320\270\321\200\320\276\320\262\320\260\32
0\275\320\275\321\213\320\271 \320"..., 77Отсортированный массив: 2 5 6  (библиотека 2)
) = 77
read(0, 0
"0\n", 1024)                    = 2
munmap(0x7da3c2a03000, 16400)           = 0
openat(AT_FDCWD, "./l1/libfirst.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|0777, st_size=15552, ...}) = 0
getcwd("/mnt/d/MAI/3semestr/OSI/laba4", 128) = 30
mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7da3c2a03000
mmap(0x7da3c2a04000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7da3c2a04000
mmap(0x7da3c2a05000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7da3c2a05000
mmap(0x7da3c2a06000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7da3c2a06000
```

```
close(3)                                = 0
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=34311, ...}) = 0
mmap(NULL, 34311, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7da3c29f3000
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) = 0x7da3c290a000
mmap(0x7da3c291a000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7da3c291a000
mmap(0x7da3c2999000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x7da3c2999000
mmap(0x7da3c29f1000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x7da3c29f1000
close(3)                                = 0
mprotect(0x7da3c29f1000, 4096, PROT_READ) = 0
mprotect(0x7da3c2a06000, 4096, PROT_READ) = 0
munmap(0x7da3c29f3000, 34311)           = 0
write(1, "\320\227\320\260\320\263\321\200\321\203\320\266\320\265\320\275\320\260
\320\261\320\270\320\261\320\273\320\270\320\276\321"..., 1173Загружена библиотека 1
Функция e: (1 + 1/x)^x
Сортировка: Пузырьковая
) = 117
read(0, exit
"exit\n", 1024)                 = 5
munmap(0x7da3c2a03000, 16408)           = 0
munmap(0x7da3c290a000, 950296)          = 0
exit_group(0)                           = ?
+++ exited with 0 +++
/mnt/d/MAI/3semestr/OSI/laba4 →
```

Вывод

Научился создавать динамические библиотеки и реализовывать программы, которые взаимодействуют с такими библиотеками.