

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-209БВ-24

Студент: Попов П.А.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 23.12.25

Москва, 2025

Постановка задачи

Вариант 15.

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 обеих программ должен быть организован следующим образом:

1. 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»;

2. «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 её выполнения;

3. «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 её выполнения.

Вариант 15:

1. Функция 1:

- №2
- Описание: расчет производной функции $\cos(x)$ в точке A с приращением δx
- Сигнатура: `float Derivative(float A, float deltaX)`
- Реализация 1: $f'(x) = (f(A + \delta x) - f(A))/\delta x$
- Реализация 2: $f'(x) = (f(A + \delta x) - f(A - \delta x))/(2 * \delta x)$

2. Функция 2:

- №9
- Описание: отсортировать целочисленный массив
- Сигнатура: `int * Sort(int * array)`
- Реализация 1: пузырьковая сортировка
- Реализация 2: сортировка Хоара

Общий метод и алгоритм решения

Использованные системные вызовы:

- `void *dlopen(const char *filename, int flags);` – загрузка динамической библиотеки (*.so) в память.
- `void *dlsym(void *handle, const char *symbol);` – получение адреса функции по её имени из загруженной библиотеки.
- `int dlclose(void *handle);` – выгрузка библиотеки из памяти.
- `int openat(int dirfd, const char *pathname, int flags, mode_t mode);` – открытие файла библиотеки для чтения.
- `void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);` – отображение кода библиотеки в память процесса.
- `int munmap(void *addr, size_t length);` – освобождение отображённой памяти.
- `ssize_t write(int fd, const void *buf, size_t count);` – вывод результатов на экран.
- `ssize_t read(int fd, void *buf, size_t count);` – чтение пользовательского ввода с клавиатуры.
- `int close(int fd);` – закрытие файлового дескриптора после работы с библиотекой.

Алгоритм работы программы №1 (статическая линковка):

1. Инициализация:
 - Подключает заголовочный файл `implementation_1.h`.
 - Выводит приветствие и формат команд.
2. Основной цикл:
 - Выводит приглашение "Введите команду: ".
 - Считывает строку ввода (`fgets`).
 - Если ввод завершён (`Ctrl+D`) → выводит "Завершение работы..." и выходит.
3. Обработка команд:
 - Команда "1":
 - Парсирует два числа `A` и `deltaX`.
 - Проверяет, что `deltaX != 0`.
 - Вызывает `Derivative(A, deltaX)`.
 - Выводит результат.
 - Команда "2":
 - Парсирует первое число как `size` (размер массива).
 - Проверяет, что $1 \leq size \leq 100$.
 - Считывает следующие `size` чисел.
 - Формирует массив `{size, n1, n2, ...}`.
 - Вызывает `Sort(array)`.
 - Выводит исходный и отсортированный массивы.
 - Команда "0":
 - Выводит сообщение: "Переключение реализаций недоступно в программе 1."
4. Завершение:

- Возвращает 0.

Алгоритм работы программы №2 (динамическая загрузка):

1. Инициализация:
 - Загружает библиотеку `lib1.so` с помощью `dlopen()`.
 - Получает указатели на функции `Derivative` и `Sort` через `dlsym()`.
 - Выводит приветствие и формат команд.
2. Основной цикл:
 - Выводит приглашение "Введите команду: ".
 - Считывает строку ввода.
 - Если ввод завершён (Ctrl+D) → выводит "Завершение работы..." и выходит.
3. Обработка команд:
 - Команда "0":
 1. Закрывает текущую библиотеку (`dlclose`).
 2. Переключает `current_lib` ($1 \leftrightarrow 2$).
 3. Загружает новую библиотеку `lib{current_lib}.so`.
 4. Получает новые указатели на функции.
 5. Выводит сообщение о переключении и новые формулы.
 - Команда "1":
 1. Парсирует `A` и `deltaX`.
 2. Проверяет `deltaX != 0`.
 3. Вызывает `Derivative` через полученный указатель.
 4. Выводит результат.
 - Команда "2":
 1. Парсирует `size` и следующие `size` чисел.
 2. Формирует массив `{size, n1, n2, ...}`.
 3. Вызывает `Sort` через полученный указатель.
 4. Выводит исходный и отсортированный массивы с указанием алгоритма.
4. Завершение:
 - Закрывает библиотеку (`dlclose`).
 - Возвращает 0.

Код программы

implementation_1.h

```
#ifndef IMPLEMENTATION_1_H
#define IMPLEMENTATION_1_H
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
float Derivative(float A, float deltaX);
```

```
int* Sort(int* array);
```

```
#ifdef __cplusplus  
}  
#endif
```

```
#endif
```

implementation_1.c

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

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

```
float Derivative(float A, float deltaX) {  
    return (cosf(A + deltaX) - cosf(A)) / deltaX;  
}
```

```
int* Sort(int* array) {  
    int size = array[0];  
    int* data = array + 1;  
  
    int* copy = (int*)malloc(size * sizeof(int));  
    if (!copy) return NULL;  
  
    for (int i = 0; i < size; i++) {  
        copy[i] = data[i];  
    }  
  
    for (int i = 0; i < size - 1; i++) {  
        for (int j = 0; j < size - i - 1; j++) {  
            if (copy[j] > copy[j + 1]) {  
                int temp = copy[j];  
                copy[j] = copy[j + 1];  
                copy[j + 1] = temp;  
            }  
        }  
    }  
}
```

```
    }

    return copy;
}
```

implementation_2.h

```
#ifndef IMPLEMENTATION_2_H
#define IMPLEMENTATION_2_H

#ifdef __cplusplus
extern "C" {
#endif

float Derivative(float A, float deltaX);
int* Sort(int* array);

#ifdef __cplusplus
}
#endif

#endif
```

implementation_2.c

```
#include <math.h>
#include <stdlib.h>

float Derivative(float A, float deltaX) {
    return (cosf(A + deltaX) - cosf(A - deltaX)) / (2 * deltaX);
}

static int partition(int* arr, int low, int high) {
    int pivot = arr[low];
```

```

int i = low - 1;

int j = high + 1;

while (1) {
    do {
        i++;
    } while (arr[i] < pivot);

    do {
        j--;
    } while (arr[j] > pivot);

    if (i >= j) {
        return j;
    }

    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

}

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

int* Sort(int* array) {
    int size = array[0];
    int* data = array + 1;

```

```

int* copy = (int*)malloc(size * sizeof(int));

if (!copy) return NULL;

for (int i = 0; i < size; i++) {
    copy[i] = data[i];
}

if (size <= 1) return copy;

quick_sort_hoare(copy, 0, size - 1);

return copy;
}

```

main_1.c

```

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "implementation_1.h"

int main() {
    char line[256];

    printf("=== Программа 1: Статическая линковка ===\n");
    printf("Функция 1: Производная cos(x) (f'(x) = (f(A+Δx)-f(A))/Δx)\n");
    printf("Функция 2: Пузырьковая сортировка\n");
    printf("Формат ввода:\n");
    printf("  1 A deltaX          - производная cos(x) в точке A с шагом deltaX\n");
    printf("  2 size n1 n2 ...      - сортировка массива (size = количество чисел)\n");
    printf("  Ctrl+D                - завершить программу\n");
    printf("Пример: 2 5 3 8 1 2    - сортирует 5 чисел: 3,8,1,2\n");
    printf("=====\n");
}

```



```

while (1) {

    printf("Введите команду: ");

    if (!fgets(line, sizeof(line), stdin)) {

        printf("\nЗавершение работы...\n");

        break;

    }

    line[strcspn(line, "\n")] = 0;

    if (line[0] == '0') {

        printf("Переключение реализаций недоступно в программе 1.\n");

    } else if (line[0] == '1') {

        float A, deltaX;

        if (sscanf(line, "1 %f %f", &A, &deltaX) == 2) {

            if (deltaX == 0.0f) {

                printf("Ошибка: шаг не может быть нулём.\n");

            } else {

                float res = Derivative(A, deltaX);

                printf("Производная cos(x) в точке %g с шагом %g = %g\n", A,
deltaX, res);

            }

        } else {

            printf("Ошибка ввода. Используйте: 1 A deltaX\n");

        }

    } else if (line[0] == '2') {

        int size;

        int numbers[101];

        char* copy = strdup(line + 2);

        if (!copy) {

            printf("Ошибка памяти.\n");

            continue;

        }

```

```

char* token = strtok(copy, " ");

if (!token) {
    printf("Ошибка: не указан размер массива.\n");
    free(copy);
    continue;
}

size = atoi(token);
if (size <= 0 || size > 100) {
    printf("Ошибка: размер должен быть от 1 до 100.\n");
    free(copy);
    continue;
}

numbers[0] = size;

int i;
for (i = 1; i <= size; i++) {
    token = strtok(NULL, " ");
    if (!token) {
        printf("Ошибка: недостаточно чисел. Ожидается %d чисел после
размера.\n", size);
        free(copy);
        break;
    }
    numbers[i] = atoi(token);
}

free(copy);

if (i <= size) continue;

printf("Исходный массив: ");

```

```

        for (int j = 1; j <= size; j++) {
            printf("%d ", numbers[j]);
        }
        printf("\n");

        int* sorted = Sort(numbers);
        if (!sorted) {
            printf("Ошибка сортировки.\n");
            continue;
        }

        printf("Отсортированный массив (пузырьковая): ");
        for (int j = 0; j < size; j++) {
            printf("%d ", sorted[j]);
        }
        printf("\n");

        free(sorted);
    } else {
        printf("Неизвестная команда. Используйте 1 или 2.\n");
    }
}

return 0;
}

```

main_2.c

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dlfcn.h>

typedef float (*DerivativeFunc)(float, float);
typedef int* (*SortFunc)(int*);

```

```

void* lib_handle = NULL;
DerivativeFunc Derivative = NULL;
SortFunc Sort = NULL;
int current_lib = 1;

void load_library(const char* lib_name) {
    if (lib_handle) dlclose(lib_handle);

    lib_handle = dlopen(lib_name, RTLD_LAZY);
    if (!lib_handle) {
        fprintf(stderr, "Ошибка загрузки: %s\n", dlerror());
        return;
    }

    Derivative = (DerivativeFunc)dlsym(lib_handle, "Derivative");
    Sort = (SortFunc)dlsym(lib_handle, "Sort");

    if (!Derivative || !Sort) {
        fprintf(stderr, "Ошибка загрузки функций: %s\n", dlerror());
        dlclose(lib_handle);
        lib_handle = NULL;
        return;
    }
}

int main() {
    char line[256];

    printf("=== Программа 2: Динамическая загрузка ===\n");
    printf("Начальные реализации:\n");
    printf("  Функция 1: Производная cos(x) ( $f'(x) = (f(A+\Delta x) - f(A)) / \Delta x$ )\n");
    printf("  Функция 2: Пузырьковая сортировка\n");
    printf("Формат ввода:\n");
    printf("  0          - переключить реализацию\n");
    printf("  1 A deltaX - производная cos(x) в точке A с шагом deltaX\n");
    printf("  2 size n1 n2 ... - сортировка массива (size = количество чисел)\n");
    printf("  Ctrl+D      - завершить программу\n");
    printf("Пример: 2 5 3 8 1 2 - сортирует 5 чисел: 3,8,1,2\n");
    printf("=====\n");

    load_library("./lib1.so");

    while (1) {
        printf("Введите команду: ");
        if (!fgets(line, sizeof(line), stdin)) {
            printf("\nЗавершение работы...\n");
            break;
        }
        line[strcspn(line, "\n")] = 0;
    }
}

```

```

if (line[0] == '0') {
    current_lib = (current_lib == 1) ? 2 : 1;
    char lib_name[20];
    snprintf(lib_name, sizeof(lib_name), "./lib%d.so", current_lib);
    load_library(lib_name);
    printf("--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---\n");
    if (current_lib == 1) {
        printf("Текущие реализации:\n");
        printf("  Функция 1:  $f'(x) = (f(A+\Delta x) - f(A)) / \Delta x$ \n");
        printf("  Функция 2: Пузырьковая сортировка\n");
    } else {
        printf("Текущие реализации:\n");
        printf("  Функция 1:  $f'(x) = (f(A+\Delta x) - f(A-\Delta x)) / (2 * \Delta x)$ \n");
        printf("  Функция 2: Сортировка Хоара (быстрая сортировка)\n");
    }
} else if (line[0] == '1') {
    if (!Derivative) {
        printf("Библиотека не загружена.\n");
        continue;
    }

    float A, deltaX;
    if (sscanf(line, "1 %f %f", &A, &deltaX) == 2) {
        if (deltaX == 0.0f) {
            printf("Ошибка: шаг не может быть нулём.\n");
        } else {
            float res = Derivative(A, deltaX);
            printf("Производная cos(x) в точке %g с шагом %g = %g\n", A,
deltaX, res);
        }
    } else {
        printf("Ошибка ввода. Используйте: 1 A deltaX\n");
    }
} else if (line[0] == '2') {
    if (!Sort) {
        printf("Библиотека не загружена.\n");
        continue;
    }

    int size;
    int numbers[101];

    char* copy = strdup(line + 2);
    if (!copy) {
        printf("Ошибка памяти.\n");
        continue;
    }
}

```

```

char* token = strtok(copy, " ");
if (!token) {
    printf("Ошибка: не указан размер массива.\n");
    free(copy);
    continue;
}

size = atoi(token);
if (size <= 0 || size > 100) {
    printf("Ошибка: размер должен быть от 1 до 100.\n");
    free(copy);
    continue;
}

numbers[0] = size;

int i;
for (i = 1; i <= size; i++) {
    token = strtok(NULL, " ");
    if (!token) {
        printf("Ошибка: недостаточно чисел. Ожидается %d чисел после
размера.\n", size);
        free(copy);
        break;
    }
    numbers[i] = atoi(token);
}

free(copy);

if (i <= size) continue;

printf("Исходный массив: ");
for (int j = 1; j <= size; j++) {
    printf("%d ", numbers[j]);
}
printf("\n");

int* sorted = Sort(numbers);
if (!sorted) {
    printf("Ошибка сортировки.\n");
    continue;
}

"Хоара"); printf("Отсортированный массив (%s): ", current_lib == 1 ? "пузырьковая" :
for (int j = 0; j < size; j++) {
    printf("%d ", sorted[j]);
}
printf("\n");

```

```

        free(sorted);
    } else {
        printf("Неизвестная команда. Используйте 0, 1 или 2.\n");
    }
}

if (lib_handle) dlclose(lib_handle);
return 0;
}

```

Makefile

```

CC = gcc
CFLAGS = -fPIC -Wall -Wextra
LDFLAGS = -lm -ldl

all: lib1.so lib2.so program1 program2

lib1.so: implementation_1.c implementation_1.h
        $(CC) $(CFLAGS) -shared -o lib1.so implementation_1.c -lm

lib2.so: implementation_2.c implementation_2.h
        $(CC) $(CFLAGS) -shared -o lib2.so implementation_2.c -lm

program1: main_1.c implementation_1.c implementation_1.h
        $(CC) -o program1 main_1.c implementation_1.c -lm

program2: main_2.c lib1.so lib2.so
        $(CC) -o program2 main_2.c $(LDFLAGS)

run1: program1
        ./program1

run2: program2
        LD_LIBRARY_PATH=. ./program2

clean:
        rm -f lib1.so lib2.so program1 program2 *.o

.PHONY: all clean run1 run2

```

Протокол работы программы

Тестирование:

\$./program1

=== Программа 1: Статическая линковка ===

Функция 1: Производная $\cos(x)$ ($f'(x) = (f(A+\Delta x) - f(A)) / \Delta x$)

Функция 2: Пузырьковая сортировка

Формат ввода:

- 1 A deltaX - производная $\cos(x)$ в точке A с шагом deltaX
- 2 size n1 n2 ... - сортировка массива (size = количество чисел)
- Ctrl+D - завершить программу

Пример: 2 5 3 8 1 2 - сортирует 5 чисел: 3,8,1,2

=====

Введите команду: 1 0.5 0.001

Производная $\cos(x)$ в точке 0.5 с шагом 0.001 = -0.479817

Введите команду: 2 5 0 1 6 7 4

Исходный массив: 0 1 6 7 4

Отсортированный массив (пузырьковая): 0 1 4 6 7

Введите команду:

Завершение работы...

\$./program2

=== Программа 2: Динамическая загрузка ===

Начальные реализации:

Функция 1: Производная $\cos(x)$ ($f'(x) = (f(A+\Delta x) - f(A))/\Delta x$)

Функция 2: Пузырьковая сортировка

Формат ввода:

- 0 - переключить реализацию
- 1 A deltaX - производная $\cos(x)$ в точке A с шагом deltaX
- 2 size n1 n2 ... - сортировка массива (size = количество чисел)
- Ctrl+D - завершить программу

Пример: 2 5 3 8 1 2 - сортирует 5 чисел: 3,8,1,2

=====

Введите команду: 1 0.5 0.001

Производная $\cos(x)$ в точке 0.5 с шагом 0.001 = -0.479817

Введите команду: 2 5 0 1 6 7 4

Исходный массив: 0 1 6 7 4

Отсортированный массив (пузырьковая): 0 1 4 6 7

Введите команду: 0

--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---

Текущие реализации:

Функция 1: $f'(x) = (f(A+\Delta x) - f(A-\Delta x))/(2*\Delta x)$

Функция 2: Сортировка Хоара (быстрая сортировка)

Введите команду: 1 0.5 0.001

Производная $\cos(x)$ в точке 0.5 с шагом 0.001 = -0.4794

Введите команду: 2 5 0 1 6 7 4

Исходный массив: 0 1 6 7 4

Отсортированный массив (Хоара): 0 1 4 6 7

Введите команду: 0

--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---

Текущие реализации:

Функция 1: $f'(x) = (f(A+\Delta x) - f(A))/\Delta x$

Функция 2: Пузырьковая сортировка

Введите команду:

Завершение работы...

Strace:

```
$ strace -f ./program1
```

```
execve("./program1", [ "./program1" ], 0x7ffce6c48508 /* 28 vars */) = 0

brk(NULL)                               = 0x5b5dabb64000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ac89f93a000

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=41635, ...}) = 0

mmap(NULL, 41635, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ac89f92f000

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"..., 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0

mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ac89f846000

mmap(0x7ac89f856000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7ac89f856000

mmap(0x7ac89f8d5000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x7ac89f8d5000

mmap(0x7ac89f92d000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x7ac89f92d000

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\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"..., 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"..., 784, 64)
= 784

mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ac89f600000

mmap(0x7ac89f628000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7ac89f628000

mmap(0x7ac89f7b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7ac89f7b0000

mmap(0x7ac89f7ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7ac89f7ff000
```

```

mmap(0x7ac89f805000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7ac89f805000

close(3) = 0

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ac89f843000

arch_prctl(ARCH_SET_FS, 0x7ac89f843740) = 0

set_tid_address(0x7ac89f843a10) = 627

set_robust_list(0x7ac89f843a20, 24) = 0

rseq(0x7ac89f844060, 0x20, 0, 0x53053053) = 0

mprotect(0x7ac89f7ff000, 16384, PROT_READ) = 0

mprotect(0x7ac89f92d000, 4096, PROT_READ) = 0

mprotect(0x5b5d719cc000, 4096, PROT_READ) = 0

mprotect(0x7ac89f972000, 8192, PROT_READ) = 0

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

munmap(0x7ac89f92f000, 41635) = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

getrandom("\x79\x78\xa9\xda\xd4\x68\x02\x2c", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0x5b5dabb64000

brk(0x5b5dabb85000) = 0x5b5dabb85000

write(1, "=== \320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260
1: \320\241\321\202\320\260"... , 70=== Программа 1: Статическая линковка ===

) = 70

write(1, "\320\244\321\203\320\275\320\272\321\206\320\270\321\217 1:
\320\237\321\200\320\276\320\270\320\267\320\262\320\276"... , 78Функция 1: Производная
cos(x) (f'(x) = (f(A+Δx)-f(A))/Δx)

) = 78

write(1, "\320\244\321\203\320\275\320\272\321\206\320\270\321\217 2:
\320\237\321\203\320\267\321\213\321\200\321\214\320\272"... , 62Функция 2: Пузырьковая
сортировка

) = 62

write(1, "\320\244\320\276\321\200\320\274\320\260\321\202
\320\262\320\262\320\276\320\264\320\260:\n", 25Формат ввода:

) = 25

write(1, " 1 A deltaX - \320\277\321\200\320\276\320\270"... , 91 1 A deltaX
- производная cos(x) в точке A с шагом deltaX

) = 91

```

```

write(1, " 2 size n1 n2 ... - \321\201\320\276\321\200\321\202"... , 101 2 size n1
n2 ... - сортировка массива (size = количество чисел)

) = 101

write(1, " Ctrl+D - \320\267\320\260\320\262\320\265"... , 62 Ctrl+D
- завершить программу

) = 62

write(1, "\320\237\321\200\320\270\320\274\320\265\321\200: 2 5 3 8 1 2 -
\321\201"... , 71Пример: 2 5 3 8 1 2 - сортирует 5 чисел: 3,8,1,2

) = 71

write(1, "===== "... ,
43=====

) = 43

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 1 0.5 0.001

"1 0.5 0.001\n", 1024) = 12

write(1,
"\320\237\321\200\320\276\320\270\320\267\320\262\320\276\320\264\320\275\320\260\321\217
cos(x) \320\262"... , 80Производная cos(x) в точке 0.5 с шагом 0.001 = -0.479817

) = 80

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 2 5 0 1 6 7 4

"2 5 0 1 6 7 4\n", 1024) = 14

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: 0"... , 42Исходный массив: 0 1 6 7 4

) = 42

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"... , 81Отсортированный массив (пузырьковая): 0 1 4 6 7

) = 81

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, "", 1024) = 0

write(1, "\n", 1

) = 1

```

```

write(1,
"\320\227\320\260\320\262\320\265\321\200\321\210\320\265\320\275\320\270\320\265
\321\200\320\260\320\261\320\276\321\202\321"... , 37Завершение работы...

) = 37

exit_group(0)                                = ?

+++ exited with 0 +++

```

Strace

```

$ strace -f ./program2

execve("./program2", [ "./program2" ], 0x7ffd9b2e2f38 /* 28 vars */) = 0

brk(NULL)                                = 0x59c8fa0d5000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7a3af8bd0000

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=41635, ...}) = 0

mmap(NULL, 41635, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a3af8bc5000

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\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) = 0x7a3af8800000

mmap(0x7a3af8828000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x7a3af8828000

mmap(0x7a3af89b0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7a3af89b0000

mmap(0x7a3af89ff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7a3af89ff000

mmap(0x7a3af8a05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7a3af8a05000

close(3)                                = 0

```

```

mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7a3af8bc2000

arch_prctl(ARCH_SET_FS, 0x7a3af8bc2740) = 0

set_tid_address(0x7a3af8bc2a10)          = 643

set_robust_list(0x7a3af8bc2a20, 24)      = 0

rseq(0x7a3af8bc3060, 0x20, 0, 0x53053053) = 0

mprotect(0x7a3af89ff000, 16384, PROT_READ) = 0

mprotect(0x59c8d8f04000, 4096, PROT_READ) = 0

mprotect(0x7a3af8c08000, 8192, PROT_READ) = 0

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

munmap(0x7a3af8bc5000, 41635)            = 0

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

getrandom("\x2f\xca\x96\x01\x6f\xa2\xfc\x5a", 8, GRND_NONBLOCK) = 8

brk(NULL)                                = 0x59c8fa0d5000

brk(0x59c8fa0f6000)                      = 0x59c8fa0f6000

write(1, "=== \320\237\321\200\320\276\320\263\321\200\320\260\320\274\320\274\320\260
2: \320\224\320\270\320\275"... , 72=== Программа 2: Динамическая загрузка ===

) = 72

write(1, "\320\235\320\260\321\207\320\260\320\273\321\214\320\275\321\213\320\265
\321\200\320\265\320\260\320\273\320\270\320\267\320"... , 41Начальные реализации:

) = 41

write(1, " \320\244\321\203\320\275\320\272\321\206\320\270\321\217 1:
\320\237\321\200\320\276\320\270\320\267\320\262"... , 80 Функция 1: Производная cos(x)
(f'(x) = (f(A+Δx)-f(A))/Δx)

) = 80

write(1, " \320\244\321\203\320\275\320\272\321\206\320\270\321\217 2:
\320\237\321\203\320\267\321\213\321\200\321\214"... , 64 Функция 2: Пузырьковая сортировка

) = 64

write(1, "\320\244\320\276\321\200\320\274\320\260\321\202
\320\262\320\262\320\276\320\264\320\260:\n", 25Формат ввода:

) = 25

write(1, " 0 - \320\277\320\265\321\200\320\265"... , 68 0
- переключить реализацию

) = 68

write(1, " 1 A deltaX - \320\277\321\200\320\276\320\270"... , 91 1 A deltaX
- производная cos(x) в точке A с шагом deltaX

) = 91

```

```

write(1, " 2 size n1 n2 ... - \321\201\320\276\321\200\321\202"... , 101 2 size n1
n2 ... - сортировка массива (size = количество чисел)

) = 101

write(1, " Ctrl+D - \320\267\320\260\320\262\320\265"... , 62 Ctrl+D
- завершить программу

) = 62

write(1, "\320\237\321\200\320\270\320\274\320\265\321\200: 2 5 3 8 1 2 -
\321\201"... , 71Пример: 2 5 3 8 1 2 - сортирует 5 чисел: 3,8,1,2

) = 71

write(1, "===== "...,
45=====

) = 45

openat(AT_FDCWD, "./lib1.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"... , 832) = 832

fstat(3, {st_mode=S_IFREG|0777, st_size=15568, ...}) = 0

getcwd("/mnt/c/Dev/Projects/OS_Labs/lab4", 128) = 33

mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a3af8bcb000

mmap(0x7a3af8bcc000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7a3af8bcc000

mmap(0x7a3af8bcd000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7a3af8bcd000

mmap(0x7a3af8bce000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7a3af8bce000

close(3) = 0

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

fstat(3, {st_mode=S_IFREG|0644, st_size=41635, ...}) = 0

mmap(NULL, 41635, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a3af8bb7000

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"... , 832) = 832

fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0

mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a3af8ace000

mmap(0x7a3af8ade000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7a3af8ade000

mmap(0x7a3af8b5d000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x7a3af8b5d000

```

```

mmap(0x7a3af8bb5000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x7a3af8bb5000

close(3) = 0

mprotect(0x7a3af8bb5000, 4096, PROT_READ) = 0

mprotect(0x7a3af8bce000, 4096, PROT_READ) = 0

munmap(0x7a3af8bb7000, 41635) = 0

fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}) = 0

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 1 0.5 0.001

"1 0.5 0.001\n", 1024) = 12

write(1,
"\320\237\321\200\320\276\320\270\320\267\320\262\320\276\320\264\320\275\320\260\321\217
cos(x) \320\262"... , 80Производная cos(x) в точке 0.5 с шагом 0.001 = -0.479817

) = 80

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 2 5 0 1 6 7 4

"2 5 0 1 6 7 4\n", 1024) = 14

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: 0"... , 42Исходный массив: 0 1 6 7 4

) = 42

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"... , 81Отсортированный массив (пузырьковая): 0 1 4 6 7

) = 81

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 0

"0\n", 1024) = 2

munmap(0x7a3af8bcb000, 16416) = 0

munmap(0x7a3af8ace000, 950296) = 0

openat(AT_FDCWD, "./lib2.so", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\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=15648, ...}) = 0

getcwd("/mnt/c/Dev/Projects/OS_Labs/lab4", 128) = 33

mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a3af8bcb000

```

```
mmap(0x7a3af8bcc000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7a3af8bcc000
```

```
mmap(0x7a3af8bcd000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7a3af8bcd000
```

```
mmap(0x7a3af8bce000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7a3af8bce000
```

```
close(3) = 0
```

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

```
fstat(3, {st_mode=S_IFREG|0644, st_size=41635, ...}) = 0
```

```
mmap(NULL, 41635, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a3af8bb7000
```

```
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\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) = 0x7a3af8ace000
```

```
mmap(0x7a3af8ade000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x10000) = 0x7a3af8ade000
```

```
mmap(0x7a3af8b5d000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8f000) = 0x7a3af8b5d000
```

```
mmap(0x7a3af8bb5000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe7000) = 0x7a3af8bb5000
```

```
close(3) = 0
```

```
mprotect(0x7a3af8bb5000, 4096, PROT_READ) = 0
```

```
mprotect(0x7a3af8bce000, 4096, PROT_READ) = 0
```

```
munmap(0x7a3af8bb7000, 41635) = 0
```

```
write(1, "---\320\240\320\225\320\220\320\233\320\230\320\227\320\220\320\246\320\230\320\230\320\237\320\225\320\240\320"... , 52--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---
```

```
) = 52
```

```
write(1, "\320\242\320\265\320\272\321\203\321\211\320\270\320\265\321\200\320\265\320\260\320\273\320\270\320\267\320\260\321\206\320"... , 37Текущие реализации:
```

```
) = 37
```

```
write(1, " \320\244\321\203\320\275\320\272\321\206\320\270\321\217 1: f'(x) = (f(A"... , 56 Функция 1: f'(x) = (f(A+Δx)-f(A-Δx))/(2*Δx)
```

```
) = 56
```



```

write(1, " \320\244\321\203\320\275\320\272\321\206\320\270\321\217 2:
\320\241\320\276\321\200\321\202\320\270\321\200"... , 90 Функция 2: Сортировка Хоара
(быстрая сортировка)

) = 90

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 1 0.5 0.001

"1 0.5 0.001\n", 1024) = 12

write(1,
"\320\237\321\200\320\276\320\270\320\267\320\262\320\276\320\264\320\275\320\260\321\217
cos(x) \320\262"... , 78Производная cos(x) в точке 0.5 с шагом 0.001 = -0.4794

) = 78

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 2 5 0 1 6 7 4

"2 5 0 1 6 7 4\n", 1024) = 14

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: 0"... , 42Исходный массив: 0 1 6 7 4

) = 42

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"... , 69Отсортированный массив (Хоара): 0 1 4 6 7

) = 69

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

read(0, 0

"0\n", 1024) = 2

munmap(0x7a3af8bcb000, 16416) = 0

munmap(0x7a3af8ace000, 950296) = 0

openat(AT_FDCWD, "./lib1.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=15568, ...}) = 0

getcwd("/mnt/c/Dev/Projects/OS_Labs/lab4", 128) = 33

mmap(NULL, 16416, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7a3af8bcb000

mmap(0x7a3af8bcc000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1000) = 0x7a3af8bcc000

mmap(0x7a3af8bcd000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) =
0x7a3af8bcd000

```

```

mmap(0x7a3af8bce000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2000) = 0x7a3af8bce000

close(3) = 0

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

fstat(3, {st_mode=S_IFREG|0644, st_size=41635, ...}) = 0

mmap(NULL, 41635, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7a3af8bb7000

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\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) = 0x7a3af8ace000

mmap(0x7a3af8ade000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7a3af8ade000

mmap(0x7a3af8b5d000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x7a3af8b5d000

mmap(0x7a3af8bb5000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x7a3af8bb5000

close(3) = 0

mprotect(0x7a3af8bb5000, 4096, PROT_READ) = 0

mprotect(0x7a3af8bce000, 4096, PROT_READ) = 0

munmap(0x7a3af8bb7000, 41635) = 0

write(1, "---
\320\240\320\225\320\220\320\233\320\230\320\227\320\220\320\246\320\230\320\230
\320\237\320\225\320\240\320"..., 52--- РЕАЛИЗАЦИИ ПЕРЕКЛЮЧЕНЫ ---

) = 52

write(1, "\320\242\320\265\320\272\321\203\321\211\320\270\320\265
\321\200\320\265\320\260\320\273\320\270\320\267\320\260\321\206\320"..., 37Текущие
реализации:

) = 37

write(1, " \320\244\321\203\320\275\320\272\321\206\320\270\321\217 1: f'(x) =
(f(A"... , 48 Функция 1: f'(x) = (f(A+Δx)-f(A))/Δx

) = 48

write(1, " \320\244\321\203\320\275\320\272\321\206\320\270\321\217 2:
\320\237\321\203\320\267\321\213\321\200\321\214"..., 64 Функция 2: Пузырьковая сортировка

) = 64

write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265
\320\272\320\276\320\274\320\260\320\275\320\264\321\203: ", 31Введите команду: ) = 31

```

```

read(0, "", 1024)                = 0

write(1, "\n", 1

)                                = 1

write(1,
"\320\227\320\260\320\262\320\265\321\200\321\210\320\265\320\275\320\270\320\265
\321\200\320\260\320\261\320\276\321\202\321"... , 37Завершение работы...

) = 37

munmap(0x7a3af8bcb000, 16416)      = 0

munmap(0x7a3af8ace000, 950296)     = 0

exit_group(0)                     = ?

+++ exited with 0 +++

```

Вывод

В ходе работы были реализованы две программы, использующие общие библиотечные функции разными способами. Программа со статической линковкой содержит код библиотеки внутри исполняемого файла, что упрощает распространение и ускоряет запуск. Программа с динамической загрузкой загружает библиотеки во время выполнения, обеспечивая возможность переключения реализаций в реальном времени без перекомпиляции. Оба подхода имеют свои преимущества и выбираются в зависимости от требований к гибкости и производительности приложения.