

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: Махова А.Б.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 19.12.25

Москва, 2025

Постановка задачи

Вариант 21.

| Номер задачи | Описание | Сигнатура | Реализация 1 | Реализация 2 |
|--------------|---|---------------------------------|---|-------------------|
| 3 | Подсчет простых чисел на отрезке [A, B] | int PrimeCount(int A, int B) | Наивный алгоритм (делимость текущего числа на все предыдущие) | Решето Эратосфена |
| 9 | Отсортировать целочисленный массив | int *Sort(int *array, int size) | Пузырьковая сортировка | Сортировка Хоара |

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

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

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

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа (программа №1), которая использует одну из библиотек, используя знания полученные на этапе компиляции;
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение 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.

Общий метод и алгоритм решения

Использованные системные вызовы:

- void *dlopen(const char *filename, int flag) - загружает динамическую библиотеку, имя которой указано в строке filename, и возвращает прямой указатель на начало динамической библиотеки

- `void *dlsym(void *handle, char *symbol)` - использует указатель на динамическую библиотеку, возвращаемую `dlopen`, и оканчивающееся нулем символьное имя, а затем возвращает адрес, указывающий, откуда загружается этот символ

- `int dlclose(void *handle)` - уменьшает на единицу счетчик ссылок на указатель динамической библиотеки `handle`. Если нет других загруженных библиотек, использующих ее символы и если счетчик ссылок принимает нулевое значение, то динамическая библиотека выгружается

- `const char *dlerror(void);` - логгирование ошибок

В начале создаем библиотеки `lib1` и `lib2`, каждая из которых содержит необходимые две функции: подсчет простых чисел на отрезке $[A, B]$ и сортировка целочисленного массива. После этого создаем две программы: в 1-ой будем использовать библиотеку `lib1`, подключаемую на этапе компиляции (на этапе «линковки»/linking), а во 2-ой будем использовать данные библиотек `lib1` и `lib2` во время исполнения основной программы с помощью вышеперечисленных системных вызовов. Также необходимо во 2-ой программе дать возможность переключаться с одной реализации контрактов на другую. При компиляции на языке Си необходимо соблюдать определенные правила.

Для 1-ой программы (`linking.c`) используем команду `gcc -o linking linking.c lib1/lib1.c -lm`, которая компилирует два исходных файла `linking.c` и `lib1.c` в исполняемый файл с именем `linking` и подключает математическую библиотеку `libm` для использования соответствующих функций в программе. Для 2-ой программы (`runtime.c`) необходимо скомпилировать две динамические библиотеки предварительно, находясь строго в директориях `lab4/programs/lib1(lib2)`. Для начала компилируем программы в объектные файлы с позиционно-независимым кодом, необходимые для создания и компиляции динамических библиотек. Для этого используем команду `gcc -c -fPIC lib1(lib2).c`. Далее с помощью команды `gcc -shared lib1(lib2).o -o lib1(lib2).so -lm` компонуем объектный файл `lib1(lib2).o` в динамическую библиотеку `lib1(lib2).so`, включая математическую библиотеку `libm`. иначе → вывод сообщения об ошибке.

Код программы

`linking.c`

```
#include <stdio.h>
#include <stdlib.h>
#include "lib1/lib1.h"

void hello_world(){
    printf("Please enter: M arg1 arg2 ... argN\n");
    printf("M - Selected function from library\n");
    printf("M = -1 - Complete the execution; no args\n");
    printf("M = 1 - The naive algorithm of searching prime numbers; arg1, arg2 - A, B\n");
```

```
    printf("M = 2 - Bubble Sort; arg1 - size of unsorted array, arg2 ... argN - elements of unsorted  
array\n");  
}
```

```
int main(){  
    hello_world();  
    int M, arg1, arg2;  
    int *array;  
    while (scanf("%d", &M) != EOF) {  
        if (M != 1 && M != 2 && M != -1) {  
            printf("You can only choose '0','1' or '2' for M\n");  
            exit(EXIT_FAILURE);  
        }  
        if (M == 1) {  
            scanf("%d %d", &arg1, &arg2);  
            int res1 = PrimeCount(arg1, arg2);  
            printf("Result of the naive algorithm (The amount of prime numbers) - %d\n", res1);  
  
        } else if (M == 2) {  
  
            scanf("%d", &arg1); // arg1 = size of unsorted array  
            array = malloc(sizeof(int)*arg1);  
            for (int i = 0; i < arg1; i++){  
                scanf("%d", &array[i]);  
            }  
  
            Sort(array, arg1);  
            printf("Bubble Sort results - ");  
            for (int i = 0; i < arg1; i++) {  
                printf("%d ", array[i]);  
            }  
            printf("\n");  
            free(array);  
        } else {  
            printf("The program successfully finished\n");  
            exit(EXIT_SUCCESS);  
        }  
    }  
}
```

runtime.c

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

void hello_world(){
    printf("Please enter: M arg1 arg2 ... argN\n");
    printf("M - Selected function from library\n");
    printf("M = -1 - Complete the execution\n");
    printf("M = 0 - Change the library\n");
    printf("M = 1 - Count the amount of prime numbers in [A, B]; arg1, arg2 - A, B\n");
    printf("M = 2 - Array sorting; arg1 - size of unsorted array, arg2 ... argN - elements of unsorted array\n");
}

int main(){
    hello_world();
    int M,arg1,arg2;
    int* array;

    char* libraries[] = {"/lib1/lib1.so", "/lib2/lib2.so"};
    int selected = 1;
    void *cur_lib = dlopen(libraries[selected], RTLD_LAZY); // Downloading the library
    if (cur_lib == NULL){
        char *error = dlerror();
        printf("2nd dynamic library loading error: %s\n", error);
        exit(EXIT_FAILURE);
    } else {
        printf("\n2nd dynamic library loaded\n");
    }

    int (*PrimeCount)(int, int) = dlsym(cur_lib, "PrimeCount"); // Getting a pointer to a function 1
    if (PrimeCount == NULL) {
        char *error = dlerror();
        printf("Error occurred while trying to find PrimeCount in 2nd lib: %s\n", error);
        exit(EXIT_FAILURE);
    } else {
        printf("Now Eratosthenes Sieve is used for counting prime numbers\n");
    }
}
```

```

void (*Sort)(int*,int) = dlsym(cur_lib, "Sort"); // Getting a pointer to a function 2
if (Sort == NULL) {
    char *error = dlerror();
    printf("Error occurred while trying to find Sort in 2nd lib: %s\n", error);
    exit(EXIT_FAILURE);
} else {
    printf("Now QuickSort is used to sort the array\n\n");
}

while (scanf("%d", &M) != EOF) {
    if (M != 1 && M != 2 && M != -1 && M != 0) {
        printf("You can only choose '-1','0','1' or '2' for M\n");
        dlclose(cur_lib);
        exit(EXIT_FAILURE);
    }
    if (M == 0){
        printf("\n");
        dlclose(cur_lib);
        selected = 1 - selected;
        cur_lib = dlopen(libraries[selected], RTLD_LAZY);
        if (cur_lib == NULL){
            if (selected == 0){
                char *error = dlerror();
                printf("1st dynamic library loading error: %s\n", error);
                exit(EXIT_FAILURE);
            } else {
                char *error = dlerror();
                printf("2nd dynamic library loading error: %s\n", error);
                exit(EXIT_FAILURE);
            }
        }
    } else {
        if (selected == 0) {
            printf("1st dynamic library loaded\n");
        } else {
            printf("2nd dynamic library loaded\n");
        }
    }
    PrimeCount = dlsym(cur_lib, "PrimeCount");
    if (PrimeCount == NULL) {
        if (selected == 0){

```

```

        char *error = dlerror();
        printf("Error occurred while trying to find PrimeCount in 1st lib: %s\n", error);
        exit(EXIT_FAILURE);
    } else {
        char *error = dlerror();
        printf("Error occurred while trying to find PrimeCount in 2nd lib: %s\n", error);
        exit(EXIT_FAILURE);
    }
} else {
    if (selected == 0) {
        printf("Now The naive algorithm is used for counting prime numbers\n");
    } else {
        printf("Now Eratosthenes Sieve is used for counting prime numbers\n");
    }
}
Sort = dlsym(cur_lib, "Sort");
if (Sort == NULL) {
    if (selected == 0){
        char *error = dlerror();
        printf("Error occurred while trying to find Sort in 1st lib: %s\n", error);
        exit(EXIT_FAILURE);
    } else {
        char *error = dlerror();
        printf("Error occurred while trying to find Sort in 2nd lib: %s\n", error);
        exit(EXIT_FAILURE);
    }
} else {
    if (selected == 0) {
        printf("Now BubbleSort is used to sort the array\n");
    } else {
        printf("Now QuickSort is used to sort the array\n");
    }
}
printf("Your library has been changed successfully!!!\n\n");
} else if (M == 1) {
    scanf("%d %d", &arg1, &arg2);
    printf("\n");
    int res1 = PrimeCount(arg1, arg2);
    printf("The amount of prime numbers - %d\n\n", res1);
} else if (M == 2) {

```

```

scanf("%d", &arg1); // arg1 = size of unsorted array
array = malloc(sizeof(int)*arg1);
for (int i = 0; i < arg1; i++){
    scanf("%d", &array[i]);
}

Sort(array, arg1);
printf("Sorted array - ");
for (int i = 0; i < arg1; i++) {
    printf("%d ", array[i]);
}
printf("\n\n");
free(array);
} else {
    dlclose(cur_lib);
    printf("The program successfully finished\n");
    exit(EXIT_SUCCESS);
}
}
}

```

lib1.c

```

#include <stdio.h>
#include "lib1.h"
#include <cstdlib>

int PrimeCount(int a, int b) { // The naive algorithm
    int prime_count = 0;

    if ((a > b) || (a < 0) || (b < 0)) {
        exit(EXIT_FAILURE);
    }

    bool flag = false;
    if (a == 1) {
        flag = true;
    }

    for (int i = a; i <= b; ++i) {
        int counter = 0;

```



```

    for (int j = 2; j < i; ++j) {
        if (i % j == 0) {
            counter++;
            break;
        }
    }

    if (counter == 0) {
        prime_count++;
    }
}

if (flag) {
    prime_count--;
}

return prime_count;
}

void *Sort(int *array, int size) { // Bubble sort

    int i, j;

    for (i = 0; i < size - 1; i++) {

        for (j = i + 1; j < size; j++) {
            if (array[i] > array[j]) {
                int temp = array[i];
                array[i] = array[j];
                array[j] = temp;
            }
        }
    }
}

```

lib1.h

```

#ifndef MYLIBRARY1
#define MYLIBRARY1
#include <stdio.h>
#include <stdbool.h>

```

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

```
int PrimeCount(int a, int b);
```

```
void *Sort(int *array, int size);
```

```
#endif
```

lib2.c

```
#include "lib2.h"
```

```
int PrimeCount(int a, int b) { // Sieve of Eratosthenes
```

```
    if((a > b) || (a < 0) || (b < 0)){  
        exit(EXIT_FAILURE);  
    }
```

```
    if(b == 1){  
        return 0;  
    }
```

```
    int *sieve = (int*)malloc((b + 1) * sizeof(int));  
    for (int i = 0; i <= b; i++) {  
        sieve[i] = 1;  
    }  
    sieve[0] = 0;  
    sieve[1] = 0;
```

```
    for(int i = 2; i * i <= b; ++i){  
        if(sieve[i] == 1){  
            for(int j = i * i; j <= b; j += i){  
                sieve[j] = 0;  
            }  
        }  
    }
```

```
    int counter = 0;  
    for(int i = a; i <= b; ++i){  
        if(sieve[i] == 1){  
            counter++;  
        }  
    }
```

```

    }

    return counter;
}

int min(int a, int b, int c) {
    if (a < b) {
        if (a < c) {
            return a;
        } else {
            return c;
        }
    } else {
        if (b < c) {
            return b;
        } else {
            return c;
        }
    }
}

```

```

int max(int a, int b, int c) {
    if (a > b) {
        if (a > c) {
            return a;
        } else {
            return c;
        }
    } else {
        if (b > c) {
            return b;
        } else {
            return c;
        }
    }
}

```

```

void swap(int *array, int a, int b) {
    if(a == b){
        return;
    }
}

```

```

    }
    int tmp = array[a];
    array[a] = array[b];
    array[b] = tmp;
}

int find_pivot(int *array, int size){
    int a, b, c, pivot;
    a = array[0];
    b = array[size-1];
    c = array[size/2];
    pivot = a + b + c - max(a, b, c) - min(a, b, c);
    return pivot;
}

void partition(int *array, int *kf_i, int *kf_k, int size){
    int pivot = find_pivot(array, size);
    int i = 0;
    int j = 0;
    int k = 0;

    while(j < size){
        if(array[j] < pivot){
            swap(array, i, j);
            if(i != k){
                swap(array, k, j);
            }
            i++;
            k++;
        } else if(array[j] == pivot){
            swap(array, k, j);
            k++;
        }
        j++;
    }

    *kf_k = k;
    *kf_i = i;
}

```

```

void quicksort(int *array, int size){

    if(size <2){
        return;

    } else if(size==2){

        if(array[0]>array[1]){
            swap(array, 0, 1);
        }
        return;
    }

    int i,k;

    partition(array, &i, &k, size);
    quicksort(array, i);
    quicksort(&array[k], size-k);
}

void *Sort(int *array, int size) { // QuickSort
    quicksort(array, size);
}

```

ib2.h

```

#ifndef MYLIBRARY2
#define MYLIBRARY2
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>

int PrimeCount(int a, int b);
int min(int a, int b, int c);
int max(int a, int b, int c);
void swap(int *array, int a, int b);
int find_pivot(int *array, int size);
void partition(int *array, int *kf_i, int *kf_k, int size);
void quicksort(int *array, int size);
void *Sort(int *array, int size);

```

#endif

Протокол работы программы

Тестирование

```
vscode → /workspaces/MAI_OS/lab4/src (main) $ gcc -o linking linking.c lib1/lib1.c -lm
vscode → /workspaces/MAI_OS/lab4/src (main) $ ./linking
Please enter: M arg1 arg2 ... argN
M - Selected function from library
M = -1 - Complete the execution; no args
M = 1 - The naive algorithm of searching prime numbers; arg1, arg2 - A, B
M = 2 - Bubble Sort; arg1 - size of unsorted array, arg2 ... argN - elements of unsorted array
1 1 10
Result of the naive algorithm (The amount of prime numbers) - 4
1 10 100
Result of the naive algorithm (The amount of prime numbers) - 21
2 5 5 4 3 2 1
Bubble Sort results - 1 2 3 4 5
-1
The program successfully finished
```

```
● vscode → /workspaces/MAI_OS/lab4/src (main) $ gcc runtime.c -o runtime
● vscode → /workspaces/MAI_OS/lab4/src (main) $ ./runtime
Please enter: M arg1 arg2 ... argN
M - Selected function from library
M = -1 - Complete the execution
M = 0 - Change the library
M = 1 - Count the amount of prime numbers in [A, B]; arg1, arg2 - A, B
M = 2 - Array sorting; arg1 - size of unsorted array, arg2 ... argN - elements of unsorted array

2nd dynamic library loaded
Now Eratosthenes Sieve is used for counting prime numbers
Now QuickSort is used to sort the array

1 1 10

The amount of prime numbers - 4

1 10 100

The amount of prime numbers - 21

2 5 5 4 3 2 1
Sorted array - 1 2 3 4 5

0

1st dynamic library loaded
Now The naive algorithm is used for counting prime numbers
Now BubbleSort is used to sort the array
Your library has been changed successfully!!!

1 10 100

The amount of prime numbers - 21

1 10 100

The amount of prime numbers - 21

2 5 5 4 3 2 1
Sorted array - 1 2 3 4 5

-1
The program successfully finished
```

Strace:

```
execve("./runtime", ["/runtime"], 0xffffe68c1a18 /* 35 vars */) = 0
```

```
brk(NULL) = 0xaaab1e700000
```

```

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0xffff833c8000

faccessat(AT_FDCWD, "/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

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=15048, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 15048, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffff833c4000

close(3) = 0

openat(AT_FDCWD, "/lib/aarch64-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\267\0\1\0\0\0\340u\2\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1637400, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 1805928, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffff831da000

mmap(0xffff831e0000, 1740392, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff831e0000

munmap(0xffff831da000, 24576) = 0

munmap(0xffff83389000, 40552) = 0

mprotect(0xffff83368000, 61440, PROT_NONE) = 0

mmap(0xffff83377000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x187000) = 0xffff83377000

mmap(0xffff8337d000, 48744, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xffff8337d000

close(3) = 0

set_tid_address(0xffff833c8f50) = 17121

set_robust_list(0xffff833c8f60, 24) = 0

rseq(0xffff833c9620, 0x20, 0, 0xd428bc00) = 0

mprotect(0xffff83377000, 16384, PROT_READ) = 0

mprotect(0xaaae7eb1000, 4096, PROT_READ) = 0

mprotect(0xffff833cd000, 8192, PROT_READ) = 0

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

munmap(0xffff833c4000, 15048) = 0

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

getrandom("\xbc\x01\x6f\x31\x7d\xed\xd1\xa5", 8, GRND_NONBLOCK) = 8

brk(NULL) = 0xaaab1e700000

brk(0xaaab1e721000) = 0xaaab1e721000

```

```

write(1, "Please enter: M arg1 arg2 ... ar"..., 35Please enter: M arg1 arg2 ... argN
) = 35
write(1, "M - Selected function from libra"..., 35M - Selected function from library
) = 35
write(1, "M = -1 - Complete the execution\n", 32M = -1 - Complete the execution
) = 32
write(1, "M = 0 - Change the library \n", 28M = 0 - Change the library
) = 28
write(1, "M = 1 - Count the amount of prim"..., 71M = 1 - Count the amount of prime numbers in
[A, B]; arg1, arg2 - A, B
) = 71
write(1, "M = 2 - Array sorting; arg1 - si"..., 97M = 2 - Array sorting; arg1 - size of unsorted
array, arg2 ... argN - elements of unsorted array
) = 97
openat(AT_FDCWD, "./lib2/lib2.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=12640, ...}, AT_EMPTY_PATH) = 0
getcwd("/workspaces/MAI_OS/lab4/src", 128) = 28
mmap(NULL, 139368, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffff831bd000
mmap(0xffff831c0000, 73832, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff831c0000
munmap(0xffff831bd000, 12288) = 0
munmap(0xffff831d3000, 49256) = 0
mprotect(0xffff831c2000, 61440, PROT_NONE) = 0
mmap(0xffff831d1000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0xffff831d1000
close(3) = 0
mprotect(0xffff831d1000, 4096, PROT_READ) = 0
write(1, "\n", 1
) = 1
write(1, "2nd dynamic library loaded\n", 272nd dynamic library loaded
) = 27
write(1, "Now Eratosthenes Sieve is used f"..., 58Now Eratosthenes Sieve is used for counting
prime numbers

```



```

) = 58
write(1, "Now QuickSort is used to sort th"... , 40Now QuickSort is used to sort the array
) = 40
write(1, "\n", 1
)
= 1
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0
read(0, 1 1 10
"1 1 10\n", 1024) = 7
write(1, "\n", 1
)
= 1
write(1, "The amount of prime numbers - 4\n"... , 33The amount of prime numbers - 4

) = 33
read(0, 2 5 5 4 3 2 1
"2 5 5 4 3 2 1\n", 1024) = 14
write(1, "Sorted array - 1 2 3 4 5 \n", 26Sorted array - 1 2 3 4 5
) = 26
write(1, "\n", 1
)
= 1
read(0, 0
"0\n", 1024) = 2
write(1, "\n", 1
)
= 1
munmap(0xffff831c0000, 73832) = 0
openat(AT_FDCWD, "./lib1/lib1.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\3\0\267\0\1\0\0\0\0\0\0\0\0\0\0\0"... , 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=8096, ...}, AT_EMPTY_PATH) = 0
getcwd("/workspaces/MAI_OS/lab4/src", 128) = 28
mmap(NULL, 135208, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffff831be000
mmap(0xffff831c0000, 69672, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff831c0000
munmap(0xffff831be000, 8192) = 0

```

```

munmap(0xffff831d2000, 53288)      = 0
mprotect(0xffff831c1000, 61440, PROT_NONE) = 0
mmap(0xffff831d0000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0) = 0xffff831d0000
close(3)                                = 0
mprotect(0xffff831d0000, 4096, PROT_READ) = 0
write(1, "1st dynamic library loaded\n", 27) = 27
write(1, "Now The naive algorithm is used for counting prime numbers", 59) = 59
write(1, "Now BubbleSort is used to sort the array", 41) = 41
write(1, "Your library has been changed successfully!!!", 45) = 45
write(1, "\n", 1) = 1
read(0, 1 1 10 "1 1 10\n", 1024) = 7
write(1, "\n", 1) = 1
write(1, "The amount of prime numbers - 4\n", 33) = 33
read(0, 2 5 5 4 3 2 1 "2 5 5 4 3 2 1\n", 1024) = 14
write(1, "Sorted array - 1 2 3 4 5 \n", 26) = 26
write(1, "\n", 1) = 1
read(0, -1 "-1\n", 1024) = 3
munmap(0xffff831c0000, 69672)      = 0
write(1, "The program successfully finished", 33) = 33

```

) = 33

lseek(0, -1, SEEK_CUR) = -1 ESPIPE (Illegal seek)

exit_group(0) = ?

+++ exited with 0 +++

Вывод

В ходе лабораторной работы были изучены и практически реализованы механизмы статической и динамической компоновки библиотек в Linux. Созданы две динамические библиотеки, содержащие различные реализации одной и той же функции, что позволило продемонстрировать разницу между линковкой на этапе компиляции и загрузкой во время выполнения.