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-211БВ-24

Студент: Рыбин В.В.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 22.11.25

Москва, 2025

Постановка задачи

Вариант 18.

Функция 1:

Подсчёт количества простых чисел на отрезке [a, b] (a, b – натуральные):
Сигнатура функции: int prime_count(int a, int b);

- Реализация №1: Наивный алгоритм.
  Проверить делимость текущего числа на все предыдущие числа.
- Реализация №2: Решето Эратосфена

Функция 2:

Перевод числа x из десятичной системы счисления в другую:
Сигнатура функции: char *convert(int x);

- Реализация №1: Перевод в двоичную
- Реализация №2: Перевод в троичную

Общий метод и алгоритм решения

- void *dlopen(const char *filename, int flags); – загружает в память и открывает динамическую библиотеку.
- void *dlsym(void *handle, const char *symbol); – возвращает адрес функции или переменной из загруженной библиотеки.
- int dlclose(void *handle); – выгружает из памяти ранее загруженную динамическую библиотеку.

Я создал две библиотеки, которые реализуют два контракта двух функций. Также я сделал два вида подключения этой библиотеки в программу: статическая библиотека, динамическая библиотека. Первый способ заключается в том, что подключаю библиотеку во время линковки программы. Второй способ заключается в том, что я подключаю библиотеку в runtime через интерфейс ОС для работы с динамическими библиотеками.

Код программы

Library_1.c

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

void reverse(char *str) {
    // реверс строки
    int n = strlen(str);
    for (int i = 0; i < n/2; ++i) {
        char temp = str[n - i - 1];
        str[n - i - 1] = str[i];
        str[i] = temp;
    }
```

```c
}

int prime_count(int a, int b) {
    // подсчет колва простых чисел на отрезке [a, b]
    // наивная реализация
    const char msg[] = "-- Log[library_1]: function prime_count naive impl\n";
    write(STDOUT_FILENO, msg, sizeof(msg));

    int count = 0;
    for (int num = a; num <= b; ++num) {
        int is_prime = 1;
        for (int del = 2; del < num; ++del) {
            if (num % del == 0) {
                is_prime = 0;
                break;
            }
        }
        if (is_prime) ++count;
    }
    return count;
}

char *convert(int x) {
    // перевод из десятичной системы счисления
    // реализация 1: перевод в двоичную
    const char msg[] = "-- Log[library_1]: function convert to 2 c/c\n";
    write(STDOUT_FILENO, msg, sizeof(msg));


    const int mask = 1;
    char *result = (char*)malloc(sizeof(char) * 32);
    if (!result) return NULL;

    int i = 0;
    if (x == 0) {
        result[i] = '0';
    }

    int is_negative = 0;
    if (x < 0) {
        is_negative = 1;
        x *= -1;
    }

    while (x != 0) {
        result[i] = (mask & x) + '0';
        ++i;
        x >>= mask;
    }

    if (is_negative) {
        result[i] = '-';
        ++i;
    }
```

```
    result[i] = '\0';
    reverse(result);
    return result;
}
```

Library_2.c:

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

void reverse(char *str) {
    // реверс строки
    int n = strlen(str);
    for (int i = 0; i < n/2; ++i) {
        char temp = str[n - i - 1];
        str[n - i - 1] = str[i];
        str[i] = temp;
    }
}


int prime_count(int a, int b) {
    // реализация 2: решето Эратосфена
    const char msg[] = "-- Log[library_2]: function prime_count using sieve\n";
    write(STDOUT_FILENO, msg, sizeof(msg));

    int numbers[b + 1];
    // 0 - простое
    // 1 - составное
    for (int i = 0; i <= b; ++i) {
        numbers[i] = 0;
    }


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

    int result = 0;
    for (int i = a; i <= b; ++i) {
        if (!numbers[i]) ++result;
    }
    return result;
}

char *convert(int x) {
    // перевод из 10 с/с в 3 с/с
    const char msg[] = "-- Log[library_2]: function convert to 3 c/c\n";
    write(STDOUT_FILENO, msg, sizeof(msg));
```

```
    char *result = (char*)malloc(sizeof(char) * 100);
    if (!result) return NULL;
    int index = 0;

    if (x == 0) {
        result[index] = '0';
        ++index;
    }

    int is_negative = 0;
    if (x < 0) {
        is_negative = 1;
        x *= -1;
    }

    while (x != 0) {
        result[index] = (x % 3) + '0';
        ++index;
        x /= 3;
    }

    if (is_negative) {
        result[index] = '-';
        ++index;
    }
    result[index] = '\0';
    reverse(result);
    return result;
}
```

dynamic.c:

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

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

#include <dlfcn.h>
#include <unistd.h>

typedef int prime_count_func(int , int);
typedef char* convert_func(int);

static prime_count_func* f_prime_count;
static convert_func* f_convert;

void print_usage(char *name) {
    char msg[1024];
    sprintf(msg, "%s [n] [path to library_1] [path to library_2] ... [path to
library_n]\n", name);
    write(STDERR_FILENO, msg, strlen(msg));
}
```

```c
int validate_arguments(int argc, char**argv) {
    if (argc < 2) {
        print_usage(argv[0]);
        return -1;
    }
    char *endptr;
    int n = strtol(argv[1], &endptr, 10);
    if (*endptr != '\0') {
        print_usage(argv[0]);
        return -1;
    }

    if (n != argc - 2) {
        print_usage(argv[0]);
        return -1;
    }

    if (n < 1) {
        print_usage(argv[0]);
        return -1;
    }

    return 0;
}


int main(int argc, char**argv) {
    int is_validate = validate_arguments(argc, argv);
    if (is_validate) {
        const char msg[] = "-- Error: invalid arguments\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        return -1;
    }

    char* endptr;
    const int n = strtol(argv[1], &endptr, 10);
    char **paths_to_libraries = (char**)malloc(sizeof(char**) * n);
    if (!paths_to_libraries) {
        const char msg[] = "-- Error: fail to alloc memory\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        return -1;
    }
    for (int i = 0; i < n; ++i) {
        paths_to_libraries[i] = argv[i + 2];
    }

    int index_cur_library = 0;
    void *library = dlopen(paths_to_libraries[index_cur_library], RTLD_LOCAL |
RTLD_LAZY);
    if (library == NULL) {
        const char msg[] = "-- Error: error while open library\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        return -1;
```

```
    }

    const char info[] = "0 - change library\n1 start_range end_range - count prime
numbers\n2 number - convert number from 10 c/c\n";
    write(STDOUT_FILENO, info, sizeof(info));

    const char enter_info[] = "Please, enter the command: \n";
    write(STDOUT_FILENO, enter_info, sizeof(enter_info));

    char buffer[1024];
    size_t bytes = read(STDIN_FILENO, buffer, sizeof(buffer));
    if (bytes == -1) {
        const char msg[] = "-- Error: error while read bytes\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        return -1;
    }

    buffer[bytes - 1] = '\0';

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

    int choice = strtol(token, &endptr, 10);
    if (*endptr != '\0') {
        const char msg[] = "-- Error: invalid command\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        return -1;
    }

    while (choice >= 0 && choice < 3) {
        if (choice == 0) {
            dlclose(library);
            index_cur_library = (index_cur_library + 1) % n;
            library = dlopen(paths_to_libraries[index_cur_library], RTLD_LOCAL |
RTLD_LAZY);
            if (library == NULL) {
                const char msg[] = "-- Error: error while change library\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }
            const char msg[] = "-- Info: library successfully changed\n";
            write(STDOUT_FILENO, msg, sizeof(msg));
        }

        if (choice == 1) {
            token = strtok(NULL, " ");
            if (token == NULL) {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            int start_range = strtol(token, &endptr, 10);
```

```
            if (*endptr != '\0') {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            token = strtok(NULL, " ");
            if (token == NULL) {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            int end_range = strtol(token, &endptr, 10);
            if (*endptr != '\0') {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            f_prime_count = dlsym(library, "prime_count");
            if (f_prime_count == NULL) {
                const char msg[] = "-- Error: failed to find prime_count
implementation\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }
            int result = f_prime_count(start_range, end_range);
            char msg[1024];
            sprintf(msg, "result: %d\n", result);
            write(STDOUT_FILENO, msg, strlen(msg));
        }

        if (choice == 2) {
            token = strtok(NULL, " ");
            if (token == NULL) {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            int number = strtol(token, &endptr, 10);
            if (*endptr != '\0') {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            f_convert = dlsym(library, "convert");
```

```
            if (f_convert == NULL) {
                const char msg[] = "-- Error: failed to find convert
implementation\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }
            char *result = f_convert(number);
            char msg[1024];
            sprintf(msg, "result: %s\n", result);
            write(STDOUT_FILENO, msg, strlen(msg));
            free(result);
        }

        write(STDOUT_FILENO, enter_info, sizeof(enter_info));
        bytes = read(STDIN_FILENO, buffer, sizeof(buffer));
        if (bytes == -1) {
            const char msg[] = "-- Error: error while read bytes\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            return -1;
        }
        buffer[bytes - 1] = '\0';
        token = strtok(buffer, " ");
        choice = strtol(token, &endptr, 10);
    }

    const char msg[] = "-- Info: programm sucessfully closed\n";
    write(STDOUT_FILENO, msg, sizeof(msg));
    dlclose(library);
}
```

static.c:

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

#include <dlfcn.h>
#include <unistd.h>

int prime_count(int a, int b);
char *convert(int x);

int main() {
    const char info[] = "1 start_range end_range - count prime numbers\n2 number -
convert number from 10 c/c\n";
    write(STDOUT_FILENO, info, sizeof(info));

    const char enter_info[] = "Please, enter the command: \n";
    write(STDOUT_FILENO, enter_info, sizeof(enter_info));

    char buffer[1024];
    size_t bytes = read(STDIN_FILENO, buffer, sizeof(buffer));
    if (bytes == -1) {
        const char msg[] = "-- Error: error while read bytes\n";
```

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

    buffer[bytes - 1] = '\0';

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

    char *endptr;
    int choice = strtol(token, &endptr, 10);
    if (*endptr != '\0') {
        const char msg[] = "-- Error: invalid command\n";
        write(STDERR_FILENO, msg, sizeof(msg));
        return -1;
    }

    while (choice > 0 && choice < 3) {
        if (choice == 1) {
            token = strtok(NULL, " ");
            if (token == NULL) {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            int start_range = strtol(token, &endptr, 10);
            if (*endptr != '\0') {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            token = strtok(NULL, " ");
            if (token == NULL) {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            int end_range = strtol(token, &endptr, 10);
            if (*endptr != '\0') {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            int result = prime_count(start_range, end_range);
            char msg[1024];
            sprintf(msg, "result: %d\n", result);
```

```
            write(STDOUT_FILENO, msg, strlen(msg));
        }

        if (choice == 2) {
            token = strtok(NULL, " ");
            if (token == NULL) {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            int number = strtol(token, &endptr, 10);
            if (*endptr != '\0') {
                const char msg[] = "-- Error: invalid arguments for function 1
(search prime numbers)\n";
                write(STDERR_FILENO, msg, sizeof(msg));
                return -1;
            }

            char *result = convert(number);
            char msg[1024];
            sprintf(msg, "result: %s\n", result);
            write(STDOUT_FILENO, msg, strlen(msg));
            free(result);
        }

        write(STDOUT_FILENO, enter_info, sizeof(enter_info));
        bytes = read(STDIN_FILENO, buffer, sizeof(buffer));
        if (bytes == -1) {
            const char msg[] = "-- Error: error while read bytes\n";
            write(STDERR_FILENO, msg, sizeof(msg));
            return -1;
        }
        buffer[bytes - 1] = '\0';
        token = strtok(buffer, " ");
        choice = strtol(token, &endptr, 10);
    }

    const char msg[] = "-- Info: programm sucessfully closed\n";
    write(STDOUT_FILENO, msg, sizeof(msg));
}
```

Протокол работы программы

```
0 - change library
1 start_range end_range - count prime numbers
2 number - convert number from 10 c/c
Please, enter the command:
1 1 10
-- Log[library_1]: function prime_count naive impl
result: 5
Please, enter the command:
2 10
-- Log[library_1]: function convert to 2 c/c
result: 1010
Please, enter the command:
2 -5
-- Log[library_1]: function convert to 2 c/c
result: -101
Please, enter the command:
0
-- Info: library successfully changed
Please, enter the command:
1 1 15
-- Log[library_2]: function prime_count using sieve
result: 7
Please, enter the command:
2 18
-- Log[library_2]: function convert to 3 c/c
result: 200
```

Вывод

В ходе лабораторной работы я научился создавать динамические библиотеки, создавать программы, использующие динамические библиотеки. Также я научился делать статические библиотеки и использовать их при компиляции моей программы.