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»

Выполнил: А. В. Маркелов
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

Цель работы: Целью является приобретение практических навыков в:

- Создании динамических библиотек

- Использовании функций из динамических библиотек двумя способами

Задание: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.

Контракты и реализации функций

Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные). Int PrimeCount(int A, int B).

- Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.

- Решето Эратосфена.

Перевод числа x из десятичной системы счисления в другую. Char* translation(long x).

- Другая система счисления двоичная.

- Другая система счисления троичная.

Вариант: 20

Метод решения

Разработка проведена в три этапа: создание библиотек с двумя реализациями функций, разработка программ с различными способами подключения библиотек, и анализ различий подходов.

- Библиотеки содержат две реализации:

libprimes.so: PrimeCount_Naive и PrimeCount_Sieve

libtranslation.so: Translate_Binary и Translate_Ternary

- Программа №1 использует статическую линковку (линковка по время компиляции):

На этапе компиляции указываются библиотеки (-lprimes, -ltranslation)

На этапе компоновки линкер разрешает символы функций

Адреса функций определяются при загрузке программы

Все реализации загружаются в память одновременно

- Программа №2 использует динамическую загрузку (загрузка во время выполнения):

Библиотеки загружаются через dlopen() при выполнении программы

Адреса функций получаются через dlsym() по названию символа

Реализации получают адреса, назначенные указателям функций

Переключение реализаций выполняется командой "0"без перекомпиляции

Неиспользуемые реализации остаются в памяти для быстрого переключения

Описание программы

contract.h — заголовочный файл с контрактами (интерфейсами). Определяет сигнатуры функций, которые должны быть реализованы в библиотеках

cli_parser.h — заголовочный файл с утилитами для работы с командной строкой и вводом-выводом. Содержит функции print_translation() — форматированный вывод результата перевода, safe_strtol() для безопасного преобразования строки в число с проверкой overflow/underflow, parse_line() для разбора строки ввода на токены (argc/argv), free_argv() для освобождения выделенной памяти, и read_line() для чтения строки из stdin. Используется обеими программами для унифицированной обработки ввода.

dynamic_loader.h — кроссплатформенный интерфейс для динамической загрузки библиотек. На Linux обеспечивает обёртки вокруг dlopen(), dlsym(), dlclose() из <dlfcn.h>. Макросы LIB_EXT и LIB_PATH_PREFIX позволяют использовать правильные расширения (.so на Linux, .dll на Windows). Используется только в program2.c для загрузки библиотек во время выполнения.

lib_primes.c — библиотека с двумя реализациями подсчёта простых чисел. Функция PrimeCount_Naive() реализует наивный алгоритм: перебирает все числа в диапазоне [A, B] и для каждого проверяет делимость на все числа от 2 до $\sqrt{n}$, помечая число как составное при обнаружении делителя. Функция PrimeCount_Sieve() реализует Решето Эратосфена: выделяет массив флагов для диапазона [start, B], инициализирует все флаги как "простое затем для каждого простого числа i отмечает его кратные как составные.

lib_translation.c — библиотека с двумя реализациями перевода в системы счисления. Вспомогательная функция translate_to_base(long x, int base) преобразует число в любую систему счисления от 2 до 36. Алгоритм сначала подсчитывает количество цифр через деление на base, затем выделяет динамическую память под результат, и заполняет цифры от конца к началу в обратном порядке. Обработка отрицательных чисел выполняется через флаг is_negative. Функция Translate_Binary(long x) вызывает translate_to_base с для перевода в двоичную систему, а Translate_Ternary(long x) вызывает её для перевода в троичную систему.

program1.c — программа с статической линковкой, использующая библиотеки на этапе компиляции. Для функции Translate программа выводит оба варианта (Binary и Ternary одновременно), так как обе реализации уже загружены в память. Все функции вызываются напрямую без использования указателей функций. Валидация входных данных выполняется через safe_strtol() с проверкой на ошибки преобразования. Программа реагирует на команды "1 A B"для подсчёта простых чисел в диапазоне, "2 number"для перевода числа в обе системы счисления, "help"для справки и "exit"для выхода.

program2.c — программа с динамической загрузкой, загружающей библиотеки во время выполнения. Использует ключевую структуру LibSet, которая хранит дескрипторы библиотек, указатели на функции для двух реализаций и индекс текущей активной реализации. Две переменные primes и translates содержат информацию отдельно для каждой функции. При старте main() вызывает open_library() для загрузки libprimes.so и libtranslation.so, затем get_symbol_library() получает адреса функций:

prime_funcs=PrimeCount, prime_funcs=PrimeCount_Sieve, translate_funcs=Translate_Binary,

translate_funcs=Translate_Ternary. Проверка успеха загрузки и наличия всех символов критична для дальнейшей работы. Переключение реализаций выполняется командой "0 которая вызывает switch_impl_primes() и switch_impl_translates(), меняя индексы current_impl между 0 и 1. Последующие команды используют новые реализации.

Результаты

При запуске программы, открытии файлов odd.txt и even.txt и вводе тестовых строк нечётные строки попали в файл odd.txt в перевёрнутом виде, а чётные — в файл even.txt так же в перевернутом виде.

Пример работы:

Ввод:

./bin/program1
1 1 20
2 5
exit
./bin/program1
0
1 1 20
2 27
0
2 27
exit

Вывод:

=== Program 1 (Compile-time Linking) ===
Commands: 1 A B | 2 number | help | exit
PrimeCount(1, 20) = 8
Decimal: 5
Binary: 101
Ternary: 12
Goodbye!
=== Program 2 (Runtime Dynamic Loading) ===
Commands: 0 | 1 A B | 2 number | help | exit
Switched PrimeCount to impl 2 (Sieve)
Switched Translate to impl 2 (Ternary)
PrimeCount(1, 20) = 8
Decimal: 27
Ternary: 1000
Switched PrimeCount to impl 1 (Naive)
Switched Translate to impl 1 (Binary)
Decimal: 27
Binary: 11011

Goodbye!

Выводы

Поставленные в лабораторной работе цели полностью достигнуты. Реализованы две полнофункциональные динамические библиотеки (libprimes.so и libtranslation.so) с двумя реализациями каждой функции, обеспечивающие необходимый функционал для сравнения различных подходов. Созданы две тестовые программы: program1.c демонстрирует статическую линковку библиотек на этапе компиляции, а program2.c показывает динамическую загрузку библиотек во время выполнения с возможностью переключения между реализациями. Вспомогательные заголовочные файлы (contract.h, cli_parser.h, dynamic_loader.h) обеспечивают модульность проекта и разделение ответственности между компонентами.

Статическая линковка обеспечивает максимальную производительность благодаря отсутствию overhead от динамической загрузки и возможности оптимизации на этапе компоновки. Все символы функций разрешаются в момент загрузки программы, что обеспечивает простоту отладки и гарантированную доступность функций. Однако эта схема требует перекомпиляции при изменении выбора реализации, что затрудняет гибкие системы. Размер исполняемого файла может быть больше, так как код библиотек встраивается в программу. На практике статическая линковка используется в системных утилитах, встроенном ПО и критичных по производительности приложениях.

В процессе выполнения лабораторной работы я получил следующие практические навыки:

- Создание динамических библиотек с правильной организации экспортируемых функций

- Освоение двух принципиально различных схем использования библиотек: статическое связывание через линкер и динамическое связывание через системный интерфейс ОС (dlopen, dlsym, dlclose)

- Разработка контрактных интерфейсов в виде заголовочных файлов, обеспечивающих разделение интерфейса от реализации

- Работа с указателями функций и управление ими через структуры для организации переключаемых реализаций

- Реализация кроссплатформенной абстракции через макросы для поддержки различных операционных систем

- Применение методов валидации входных данных и обработки ошибок в критичных местах программы

Анализ типов использования библиотек.

Статическая линковка обеспечивает максимальную производительность благодаря отсутствию временных затрат от динамической загрузки и возможности оптимизации на этапе компоновки. Все символы функций разрешаются в момент загрузки программы, что обеспечивает простоту отладки и гарантированную доступность функций. Однако эта схема требует перекомпиляции при изменении выбора реализации, что затрудняет гибкие системы. Размер исполняемого файла может быть больше, так как код библиотек встраивается в программу.

Динамическая загрузка предоставляет гибкость переключения между реализациями во время выполнения без перекомпиляции программы. Это позволяет выбирать оптимальную реализацию в зависимости от runtime условий, поддерживать плагины и расширения без изменения основного кода. Размер исполняемого файла остаётся небольшим, так как код библиотек остаётся отдельным. Недостатком является небольшие временные затраты при вызове dlopen() и dlsym(), а также зависимость от наличия файлов библиотек в

runtime. Риск обнаружения ошибок только при выполнении программы (неправильные имена символов) требует более тщательного тестирования.

Исходная программа

```
int PrimeCount(int A, int B);
int PrimeCount_Sieve(int A, int B);
char* Translate_Binary(long x);
char* Translate_Ternary(long x);
```

Листинг 1: *Заголовочный файл с контрактами*

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

#define BUFFER_SIZE 256
#define MAX_ARRAY_SIZE 1000

static inline void print_translation(const char* result, const char* prefix) {
  printf("%s%s\n", prefix, result);
}

static inline int safe_strtol(const char* str, int* error) {
  char* endptr;
  errno = 0;

  long val = strtol(str, &endptr, 10);

  if (errno != 0) {
    *error = 1;
    return 0;
  }

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

  if (val > INT_MAX || val < INT_MIN) {
    *error = 1;
    return 0;
  }

  *error = 0;
  return (int)val;
}

static inline int parse_line(const char* line, char*** argv_ptr) {
  char* copy = (char*)malloc(strlen(line) + 1);
  if (!copy) return 0;
  strcpy(copy, line);

  *argv_ptr = (char**)malloc(sizeof(char*) * (MAX_ARRAY_SIZE + 2));
  if (!*argv_ptr) {
    free(copy);
    return 0;
  }
```

```
49
50    int argc = 0;
51    char* token = strtok(copy, " ");
52    while (token && argc < MAX_ARRAY_SIZE + 1) {
53      (*argv_ptr)[argc] = (char*)malloc(strlen(token) + 1);
54      if (!(*argv_ptr)[argc]) {
55        free(copy);
56        return -1;
57      }
58      strcpy((*argv_ptr)[argc], token);
59      argc++;
60      token = strtok(NULL, " ");
61    }
62
63    free(copy);
64    return argc;
65  }
66
67  static inline void free_argv(int argc, char** argv) {
68    if (!argv) return;
69    for (int i = 0; i < argc; i++)
70      if (argv[i]) free(argv[i]);
71    free(argv);
72  }
73
74  static inline char* read_line(char line[BUFFER_SIZE]) {
75    if (!fgets(line, BUFFER_SIZE, stdin)) return NULL;
76    size_t len = strlen(line);
77    if (len > 0 && line[len - 1] == '\n') line[len - 1] = '\0';
78    return line;
79  }
```

Листинг 2: *Заголовочный файл с утилитами*

```
1   #include <stdio.h>
2
3   #ifdef _WIN32
4
5   #define LIB_EXT ".dll"
6   #define LIB_PATH_PREFIX "./lib/"
7
8   #else
9
10  #include <dlfcn.h>
11
12  typedef void* DynamicLib;
13
14  static inline DynamicLib open_library(const char* path) {
15    return dlopen(path, RTLD_LAZY);
16  }
17
18  static inline void* get_symbol_library(DynamicLib lib, const char* symbol) {
19    return dlsym(lib, symbol);
20  }
21
22  static inline int close_library(DynamicLib lib) {
23    return dlclose(lib) == 0 ? 1 : 0;
```

```
}

static inline const char* get_last_error(void) { return dlerror(); }

#define LIB_EXT ".so"
#define LIB_PATH_PREFIX "./lib/"

#endif
```

Листинг 3: *Кроссплатформенный интерфейс для динамической загрузки библиотек*

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

#include "contract.h"

static int count_primes_naive(int A, int B) {
  if (A > B) {
    return 0;
  }
  if (B < 2) {
    return 0;
  }

  if (A < 2) {
    A = 2;
  }

  int count = 0;

  for (int num = A; num <= B; num++) {
    int is_prime = 1;

    for (int i = 2; i * i <= num; i++) {
      if (num % i == 0) {
        is_prime = 0;
        break;
      }
    }

    if (is_prime) {
      count++;
    }
  }

  return count;
}

static int count_primes_sieve(int A, int B) {
  if (B < 2) {
    return 0;
  }

  int start = (A < 2) ? 2 : A;
  int size = B - start + 1;

```

```
  unsigned char* is_prime = (unsigned char*)malloc(size);
  if (is_prime == NULL) {
    fprintf(stderr, "Error: Memory allocation failed\n");
    return 0;
  }

  memset(is_prime, 1, size);

  if (start == 0) {
    is_prime[0] = 0;
  }
  if (start == 1) {
    is_prime[1 - start] = 0;
  }

  for (int i = 2; i * i <= B; i++) {
    int first_multiple = ((start + i - 1) / i) * i;

    for (int j = first_multiple; j <= B; j += i) {
      if (j >= start) {
        is_prime[j - start] = 0;
      }
    }
  }

  int count = 0;
  for (int i = 0; i < size; i++) {
    if (is_prime[i]) {
      count++;
    }
  }

  free(is_prime);
  return count;
}

int PrimeCount(int A, int B) {
  if (A > B) {
    return 0;
  }
  if (B < 2) {
    return 0;
  }

  return count_primes_naive(A, B);
}

int PrimeCount_Sieve(int A, int B) {
  if (A > B) {
    return 0;
  }
  if (B < 2) {
    return 0;
  }

  return count_primes_sieve(A, B);
}
```

Листинг 4: *библиотека с реализациями подсчёта простых чисел*

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

#include "contract.h"

static char* translate_to_base(long x, int base) {
  if (x == 0) {
    char* result = (char*)malloc(2);
    if (result) {
      strcpy(result, "0");
    }
    return result;
  }

  int is_negative = (x < 0);
  long num = (x < 0) ? -x : x;

  char temp[128];
  int len = 0;
  long temp_num = num;
  while (temp_num > 0) {
    temp_num /= base;
    len++;
  }

  if (is_negative) {
    len++;
  }

  char* result = (char*)malloc(len + 1);
  if (!result) {
    fprintf(stderr, "Error: Memory allocation failed\n");
    return NULL;
  }

  result[len] = '\0';
  temp_num = num;
  int pos = len - 1;

  while (temp_num > 0) {
    int digit = temp_num % base;
    result[pos--] = (digit < 10) ? ('0' + digit) : ('A' + digit - 10);
    temp_num /= base;
  }

  if (is_negative) {
    result[0] = '-';
  }

  return result;
}

char* Translate_Binary(long x) {
  if (x == 0) {
    char* result = (char*)malloc(2);
    if (result) {
      strcpy(result, "0");
```

```
    }
    return result;
  }

  return translate_to_base(x, 2);
}

char* Translate_Ternary(long x) {
  if (x == 0) {
    char* result = (char*)malloc(2);
    if (result) {
      strcpy(result, "0");
    }
    return result;
  }

  return translate_to_base(x, 3);
}
```

Листинг 5: *библиотека с реализациями перевода в системы счисления*

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

#include "cli_parser.h"
#include "contract.h"

typedef struct {
  int argc;
  char** argv;
} Args;

static void handle_cmd_1(Args args) {
  if (args.argc < 3) {
    fprintf(stderr, "Error: PrimeCount needs 2 args: A B\n");
    return;
  }

  int error_a, error_b;
  int A = safe_strtol(args.argv[1], &error_a);
  int B = safe_strtol(args.argv[2], &error_b);

  if (error_a) {
    fprintf(stderr, "Error: Invalid argument A '%s' - not a valid integer\n",
            args.argv[1]);
    return;
  }

  if (error_b) {
    fprintf(stderr, "Error: Invalid argument B '%s' - not a valid integer\n",
            args.argv[2]);
    return;
  }

  int result = PrimeCount(A, B);
  printf("PrimeCount(%d, %d) = %d\n", A, B, result);
```

```
}

static void handle_cmd_2(Args args) {
  if (args.argc < 2) {
    fprintf(stderr, "Error: Translate needs 1 arg: number\n");
    return;
  }

  int error;
  long num = strtol(args.argv[1], NULL, 10);
  if (error) {
    fprintf(stderr, "Error: Invalid number '%s'\n", args.argv[1]);
    return;
  }

  char* binary = Translate_Binary(num);
  char* ternary = Translate_Ternary(num);

  if (binary) {
    printf("Decimal: %ld\n", num);
    printf("Binary: %s\n", binary);
    free(binary);
  }
  if (ternary) {
    printf("Ternary: %s\n", ternary);
    free(ternary);
  }
}

int main(void) {
  printf("\n=== Program 1 (Compile-time Linking) ===\n");
  printf("Commands: 1 A B | 2 number | help | exit\n\n");

  char line[BUFFER_SIZE];
  while (read_line(line)) {
    if (strlen(line) == 0) {
      continue;
    }

    Args args;
    int argc = parse_line(line, &args.argv);
    if (argc <= 0) {
      continue;
    }

    args.argc = argc;

    if (!strcmp(args.argv[0], "exit")) {
      free_argv(argc, args.argv);
      break;
    } else if (!strcmp(args.argv[0], "help")) {
      printf("1 A B: PrimeCount in range [A,B]\n");
      printf("2 number: Translate decimal to binary and ternary\n");
    } else if (!strcmp(args.argv[0], "1")) {
      handle_cmd_1(args);
    } else if (!strcmp(args.argv[0], "2")) {
      handle_cmd_2(args);
    } else {
```

```
      printf("Unknown: %s\n", args.argv[0]);
    }

    free_argv(argc, args.argv);
  }

  printf("Goodbye!\n");
  return 0;
}
```

Листинг 6: *Программа с статической линковкой*

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

#include "cli_parser.h"
#include "contract.h"
#include "dynamic_loader.h"

#define MAX_LIBS 2

typedef int (*PrimeCountFunc)(int, int);
typedef char* (*TranslateFunc)(long);

typedef struct {
  DynamicLib libs[MAX_LIBS];
  PrimeCountFunc prime_funcs[MAX_LIBS];
  TranslateFunc translate_funcs[MAX_LIBS];
  int current_impl;
} LibSet;

static LibSet primes = {{0}, {0}, {0}, 0};
static LibSet translates = {{0}, {0}, {0}, 0};

static void switch_impl_primes(void) {
  int new_impl = 1 - primes.current_impl;
  if (primes.prime_funcs[new_impl]) {
    primes.current_impl = new_impl;
    printf("Switched PrimeCount to impl %d (%s)\n\n", new_impl + 1,
           new_impl == 0 ? "Naive" : "Sieve");
  } else {
    fprintf(stderr, "Error: Alternative implementation not available\n");
  }
}

static void switch_impl_translates(void) {
  int new_impl = 1 - translates.current_impl;
  if (translates.translate_funcs[new_impl]) {
    translates.current_impl = new_impl;
    printf("Switched Translate to impl %d (%s)\n\n", new_impl + 1,
           new_impl == 0 ? "Binary" : "Ternary");
  } else {
    fprintf(stderr, "Error: Alternative implementation not available\n");
  }
}

```

```c
static void handle_cmd_0(void) {
  switch_impl_primes();
  switch_impl_translates();
}

static void handle_cmd_1(int argc, char** argv) {
  if (argc < 3) {
    fprintf(stderr, "Error: PrimeCount needs 2 args: A B\n");
    return;
  }

  int error_a, error_b;
  int A = safe_strtol(argv[1], &error_a);
  int B = safe_strtol(argv[2], &error_b);

  if (error_a) {
    fprintf(stderr, "Error: Invalid argument A '%s' - not a valid integer\n",
            argv[1]);
    return;
  }

  if (error_b) {
    fprintf(stderr, "Error: Invalid argument B '%s' - not a valid integer\n",
            argv[2]);
    return;
  }

  if (!primes.prime_funcs[primes.current_impl]) {
    fprintf(stderr, "Error: PrimeCount not loaded\n");
    return;
  }

  int result = primes.prime_funcs[primes.current_impl](A, B);
  printf("PrimeCount(%d, %d) = %d\n", A, B, result);
}

static void handle_cmd_2(int argc, char** argv) {
  if (argc < 2) {
    fprintf(stderr, "Error: Translate needs 1 arg: number\n");
    return;
  }

  char* endptr;
  long num = strtol(argv[1], &endptr, 10);
  if (*endptr != '\0') {
    fprintf(stderr, "Error: Invalid number '%s'\n", argv[1]);
    return;
  }

  if (!translates.translate_funcs[translates.current_impl]) {
    fprintf(stderr, "Error: Translation function not loaded\n");
    return;
  }

  char* result = translates.translate_funcs[translates.current_impl](num);

  if (result) {
    printf("Decimal: %ld\n", num);
```

```
    if (translates.current_impl == 0) {
      printf("Binary: %s\n", result);
    } else {
      printf("Ternary: %s\n", result);
    }
    free(result);
  } else {
    fprintf(stderr, "Error: Translation failed\n");
  }
}

int main(void) {
  char path_primes[256];
  char path_translates[256];
  snprintf(path_primes, sizeof(path_primes), "%slibprimes%s", LIB_PATH_PREFIX,
          LIB_EXT);
  snprintf(path_translates, sizeof(path_translates), "%slibtranslation%s",
          LIB_PATH_PREFIX, LIB_EXT);

  DynamicLib lib_primes_handle = open_library(path_primes);
  if (!lib_primes_handle) {
    fprintf(stderr, "Failed to load libprimes: %s\n", get_last_error());
    return 1;
  }

  DynamicLib lib_translates_handle = open_library(path_translates);
  if (!lib_translates_handle) {
    fprintf(stderr, "Failed to load libtranslation: %s\n", get_last_error());
    close_library(lib_primes_handle);
    return 1;
  }

  primes.libs[0] = lib_primes_handle;
  primes.libs[1] = lib_primes_handle;
  translates.libs[0] = lib_translates_handle;
  translates.libs[1] = lib_translates_handle;

  primes.prime_funcs[0] =
      (PrimeCountFunc)get_symbol_library(lib_primes_handle, "PrimeCount");
  primes.prime_funcs[1] =
      (PrimeCountFunc)get_symbol_library(lib_primes_handle, "PrimeCount_Sieve");

  translates.translate_funcs[0] = (TranslateFunc)get_symbol_library(
      lib_translates_handle, "Translate_Binary");
  translates.translate_funcs[1] = (TranslateFunc)get_symbol_library(
      lib_translates_handle, "Translate_Ternary");

  if (!primes.prime_funcs[0] || !primes.prime_funcs[1] ||
      !translates.translate_funcs[0] || !translates.translate_funcs[1]) {
    fprintf(stderr, "Failed to load symbols: %s\n", get_last_error());
    close_library(lib_primes_handle);
    close_library(lib_translates_handle);
    return 1;
  }

  printf("\n=== Program 2 (Runtime Dynamic Loading) ===\n");
  printf(
      "Commands: 0 | 1 A B | 2 number | help | "
```

```
      "exit\n\n");

  char line[BUFFER_SIZE];
  while (read_line(line)) {
    if (strlen(line) == 0) {
      continue;
    }

    char** argv;
    int argc = parse_line(line, &argv);
    if (argc <= 0) {
      continue;
    }

    if (!strcmp(argv[0], "exit")) {
      free_argv(argc, argv);
      break;
    } else if (!strcmp(argv[0], "help")) {
      printf("0: Switch implementations (Binary <-> Ternary)\n");
      printf("1 A B: PrimeCount in range [A,B]\n");
      printf("2 number: Translate decimal (currently: %s)\n",
            translates.current_impl == 0 ? "Binary" : "Ternary");
      printf("exit: Exit program\n");
    } else if (!strcmp(argv[0], "0")) {
      handle_cmd_0();
    } else if (!strcmp(argv[0], "1")) {
      handle_cmd_1(argc, argv);
    } else if (!strcmp(argv[0], "2")) {
      handle_cmd_2(argc, argv);
    } else {
      printf("Unknown command: %s\n", argv[0]);
    }

    free_argv(argc, argv);
  }

  close_library(lib_primes_handle);
  close_library(lib_translates_handle);

  printf("Goodbye!\n");
  return 0;
}
```

Листинг 7: *Программа с динамической загрузкой*

```
146823 brk(NULL) = 0x5585d2edc000
146823 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x7f8ce47d2000
146823 openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
146823 mmap(NULL, 30076, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f8ce47ca000
146823 openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
146823 mmap(NULL, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0
    x7f8ce45a1000
146823 mprotect(0x7f8ce45c9000, 2023424, PROT_NONE) = 0
146823 mmap(0x7f8ce45c9000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x28000) = 0x7f8ce45c9000
```

```
146823 mmap(0x7f8ce475e000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
     0x1bd000) = 0x7f8ce475e000
146823 mmap(0x7f8ce47b7000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x215000) = 0x7f8ce47b7000
146823 mmap(0x7f8ce47bd000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS, -1, 0) = 0x7f8ce47bd000
146823 mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x7f8ce459e000
146823 mprotect(0x7f8ce47b7000, 16384, PROT_READ) = 0
146823 mprotect(0x5585a0080000, 4096, PROT_READ) = 0
146823 mprotect(0x7f8ce480c000, 8192, PROT_READ) = 0
146823 brk(NULL) = 0x5585d2edc000
146823 brk(0x5585d2efd000) = 0x5585d2efd000
146823 openat(AT_FDCWD, "./lib/libprimes.so", O_RDONLY|O_CLOEXEC) = 3
146823 mmap(NULL, 16456, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f8ce47cd000
146823 mmap(0x7f8ce47ce000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1000) = 0x7f8ce47ce000
146823 mmap(0x7f8ce47cf000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0
    x2000) = 0x7f8ce47cf000
146823 mmap(0x7f8ce47d0000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x2000) = 0x7f8ce47d0000
146823 mprotect(0x7f8ce47d0000, 4096, PROT_READ) = 0
146823 openat(AT_FDCWD, "./lib/libtranslation.so", O_RDONLY|O_CLOEXEC) = 3
146823 mmap(NULL, 16448, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f8ce4599000
146823 mmap(0x7f8ce459a000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x1000) = 0x7f8ce459a000
146823 mmap(0x7f8ce459b000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0
    x2000) = 0x7f8ce459b000
146823 mmap(0x7f8ce459c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x2000) = 0x7f8ce459c000
146823 mprotect(0x7f8ce459c000, 4096, PROT_READ) = 0
146823 +++ exited with 0 +++
```

Листинг 8: *Strace логи*