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 и Translate_Binary

libtranslation.so: PrimeCountSieve и Translate_Ternary

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

На этапе компиляции указываются библиотеки (-llight)

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_light.c — библиотека с наивной реализацией подсчёта простых чисел и перевода в двоичную систему. Функция PrimeCount() 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 translation() реализует перевод числа в двоичную систему счисления через вспомогательную функцию translate_to_base().

lib_hard.c — библиотека с продвинутой реализацией подсчёта простых чисел и перевода в троичную систему. Функция PrimeCount() реализует Решето Эратосфена: выделяет массив флагов для диапазона [2, B], инициализирует все флаги как "простое затем для каждого простого числа i отмечает его кратные как составные. Функция translation() реализует перевод числа в троичную систему счисления через вспомогательную функцию translate_to_base().

program1.c — программа с статической линковкой, использующая библиотеки на этапе компиляции. Для функции Translate программа выводит вариант Binary, так как линкована с liblight. 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 двоичную систему счисления, "help"для справки и "exit"для выхода.

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

impl.primefunc=PrimeCount, impl.translatefunc=TranslateBinary, impl.primefunc=PrimeCou

ntSieve, impl.translatefunc=TranslateTernary. Проверка успеха загрузки и наличия всех символов критична для дальнейшей работы. Переключение реализаций выполняется командой "0 которая вызывает handlecmd0(), меняя currentimpl между 0 и 1. Последующие

команды используют новые реализации.

Результаты

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

Выводы

Поставленные в лабораторной работе цели полностью достигнуты. Реализованы две полнофункциональные динамические библиотеки (liblight.so и libhard.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 загруз-

ку 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);
char* translation(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("ss\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;
  }

  int argc = 0;
  char* token = strtok(copy, " ");
  while (token && argc < MAX_ARRAY_SIZE + 1) {
    (*argv_ptr)[argc] = (char*)malloc(strlen(token) + 1);
    if (!(*argv_ptr)[argc]) {
      free(copy);
      return -1;
    }
    strcpy((*argv_ptr)[argc], token);
    argc++;
    token = strtok(NULL, " ");
  }

  free(copy);
  return argc;
}

static inline void free_argv(int argc, char** argv) {
  if (!argv) return;
  for (int i = 0; i < argc; i++)
    if (argv[i]) free(argv[i]);
  free(argv);
}

static inline char* read_line(char line[BUFFER_SIZE]) {
  if (!fgets(line, BUFFER_SIZE, stdin)) return NULL;
  size_t len = strlen(line);
  if (len > 0 && line[len - 1] == '\n') line[len - 1] = '\0';
  return line;
}
```

Листинг 2: *Заголовочный файл с утилитами*

```
#include <stdio.h>

#ifdef _WIN32

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

#else

#include <dlfcn.h>

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;
24 }
25 
26 static inline const char* get_last_error(void) { return dlerror(); }
27 
28 #define LIB_EXT ".so"
29 #define LIB_PATH_PREFIX "./lib/"
30 
31 #endif
```

Листинг 3: *Кроссплатформенный интерфейс для динамической загрузки библиотек*

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  
5  #include "contract.h"
6  
7  static int is_prime(int n) {
8    if (n < 2) {
9      return 0;
10   }
11   if (n == 2) {
12     return 1;
13   }
14   if (n  2 == 0) return 0;for (int i = 3; i * i <= n; i += 2) if (n  i == 0) {
15       return 0;
16     }
17   }
18   return 1;
19 }
20 
21 int PrimeCount(int A, int B) {
22   if (A > B) {
23     return 0;
24   }
25   if (A < 2) {
26     A = 2;
27   }
28 
```

```
  int count = 0;
  for (int i = A; i <= B; i++) {
    if (is_prime(i)) {
      count++;
    }
  }
  return count;
}

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 *
    translation(long x)return translate_to base(x, 2);
```

Листинг 4: *Библиотека с первыми реализациями*

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

#include "contract.h"

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

  int max_num = B;
  if (max_num < 2) {
    return 0;
  }

  char* sieve = (char*)malloc(max_num + 1);
  if (!sieve) {
    fprintf(stderr, "Error: Memory allocation failed\n");
    return 0;
  }

  memset(sieve, 1, max_num + 1);
  sieve[0] = sieve[1] = 0;

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

  int count = 0;
  for (int i = A; i <= B; i++) {
    if (sieve[i]) {
      count++;
    }
  }

  free(sieve);
  return count;
}

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] = ' -'; returnresult; char *
    translation(longx)returntranslate_to_base(x, 3);
```

Листинг 5: *Библиотека со вторыми реализациями*

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

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

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 argv[1]);return;if
    (error_b)fprintf(stderr,"Error : InvalidargumentB'argv[2]);return;intresult =
    PrimeCount(A, B);printf("PrimeCount(staticvoidhandle_cmd_2(intargc, char *
    *argv)if(argc < 2)fprintf(stderr,"Error : Translateneeds1arg : number");return;char * endptr;longn
```

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* (*TranslationFunc)(long);

typedef struct {
  DynamicLib lib;
  PrimeCountFunc prime_func;
  TranslationFunc translation_func;
} LibImpl;

static LibImpl impl[MAX_LIBS];
static int current_impl = 0;

static void handle_cmd_0(void) {
  int new_impl = 1 - current_impl;

  if (!impl[new_impl].prime_func || !impl[new_impl].translation_func) {
    fprintf(stderr, "Error: Alternative implementation not available\n");
    return;
  }

  current_impl = new_impl;
  printf("Switched to impl d (s)\n\n", current_impl + 1,
         current_impl == 0 ? "Light" : "Hard");
}

```

```c
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 argv[1]);return;if
    (error_b)fprintf(stderr,"Error : InvalidargumentB'argv[2]);return;if(!impl[current_impl].prime_func),
    impl[current_impl].prime_func(A, B);printf("PrimeCount(current_impl == 0?"naive" :
    "sieve"); staticvoidhandle_cmd_2(intargc, char *
    *argv)if(argc < 2)fprintf(stderr,"Error : Translateneeds1arg : number"); return;char * endptr; longn
```

Листинг 7: *Программа с динамической загрузкой*

```
execve("./bin/program1", ["./bin/program1"], 0x7ffc4fb031d0 /* 35 vars */) = 0
brk(NULL)                               = 0x55f1f9031000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe09626010) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x7ff262c06000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=30076, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 30076, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff262bfe000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\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
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"...,
    48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\
    266\235"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) =
     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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff2629d5000
mprotect(0x7ff2629fd000, 2023424, PROT_NONE) = 0
mmap(0x7ff2629fd000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x28000) = 0x7ff2629fd000
mmap(0x7ff262b92000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
    0x1bd000) = 0x7ff262b92000
mmap(0x7ff262beb000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_DENYWRITE, 3, 0x215000) = 0x7ff262beb000
```

```
mmap(0x7ff262bf1000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
    MAP_ANONYMOUS, -1, 0) = 0x7ff262bf1000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x7ff2629d2000
arch_prctl(ARCH_SET_FS, 0x7ff2629d2740) = 0
set_tid_address(0x7ff2629d2a10)         = 41198
set_robust_list(0x7ff2629d2a20, 24)     = 0
rseq(0x7ff2629d30e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7ff262beb000, 16384, PROT_READ) = 0
mprotect(0x55f1c00e4000, 4096, PROT_READ) = 0
mprotect(0x7ff262c40000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY})
     = 0
munmap(0x7ff262bfe000, 30076)           = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x9), ...},
    AT_EMPTY_PATH) = 0
getrandom("\x7e\x6d\x41\x89\x41\xd2\x19\x02", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x55f1f9031000
brk(0x55f1f9052000)                     = 0x55f1f9052000
write(1, "\n", 1)                       = 1
write(1, "=== Program 1 (Static Linking - "..., 58) = 58
write(1, "Commands: 1 A B | 2 number | hel"..., 41) = 41
write(1, "\n", 1)                       = 1
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x9), ...},
    AT_EMPTY_PATH) = 0
read(0, "1 1 20\n", 1024)               = 7
write(1, "PrimeCount(1, 20) = 8 (naive)\n", 30) = 30
read(0, "2 5\n", 1024)                  = 4
write(1, "Decimal: 5\n", 11)            = 11
write(1, "Binary: 101\n", 12)           = 12
read(0, "exit\n", 1024)                 = 5
write(1, "Goodbye!\n", 9)               = 9
exit_group(0)                           = ?
+++ exited with 0 +++
```

Листинг 8: *Strace логи к program1*

```
execve("./bin/program2", ["./bin/program2"], 0x7ffe741eb1e0 /* 35 vars */) = 0
brk(NULL)                               = 0x557b24908000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe8f1c9920) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0
    x7f20aad5d000
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
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=30076, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 30076, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f20aad55000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\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
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, AT_EMPTY_PATH) = 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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f20aab2c000
mprotect(0x7f20aab54000, 2023424, PROT_NONE) = 0
mmap(0x7f20aab54000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f20aab54000
mmap(0x7f20aace9000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7f20aace9000
mmap(0x7f20aad42000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x215000) = 0x7f20aad42000
mmap(0x7f20aad48000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f20aad48000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f20aab29000
arch_prctl(ARCH_SET_FS, 0x7f20aab29740) = 0
set_tid_address(0x7f20aab29a10)         = 41577
set_robust_list(0x7f20aab29a20, 24)     = 0
rseq(0x7f20aab2a0e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7f20aad42000, 16384, PROT_READ) = 0
mprotect(0x557b230a1000, 4096, PROT_READ) = 0
mprotect(0x7f20aad97000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f20aad55000, 30076)           = 0
getrandom("\x36\x9b\x80\x86\x8e\xb4\x1f\xc9", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x557b24908000
brk(0x557b24929000)                     = 0x557b24929000
openat(AT_FDCWD, "./lib/liblight.so", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=15848, ...}, AT_EMPTY_PATH) = 0
getcwd("/home/kolko/OS/cross_platform_lab/lab4/build", 128) = 45
mmap(NULL, 16448, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f20aad58000
mmap(0x7f20aad59000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f20aad59000
mmap(0x7f20aad5a000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f20aad5a000
mmap(0x7f20aad5b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f20aad5b000
close(3)                                = 0
mprotect(0x7f20aad5b000, 4096, PROT_READ) = 0
openat(AT_FDCWD, "./lib/libhard.so", O_RDONLY|O_CLOEXEC) = 3
```

```
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=15912, ...}, AT_EMPTY_PATH) = 0
getcwd("/home/kolko/OS/cross_platform_lab/lab4/build", 128) = 45
mmap(NULL, 16464, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f20aab24000
mmap(0x7f20aab25000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f20aab25000
mmap(0x7f20aab26000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f20aab26000
mmap(0x7f20aab27000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f20aab27000
close(3)                                = 0
mprotect(0x7f20aab27000, 4096, PROT_READ) = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x9), ...}, AT_EMPTY_PATH) = 0
write(1, "\n", 1)                       = 1
write(1, "=== Program 2 (Dynamic Loading -"..., 65) = 65
write(1, "Commands: 0 | 1 A B | 2 number |"..., 45) = 45
write(1, "\n", 1)                       = 1
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x9), ...}, AT_EMPTY_PATH) = 0
read(0, "0\n", 1024)                    = 2
write(1, "Switched to impl 2 (Hard)\n\n", 27) = 27
read(0, "1 1 20\n", 1024)               = 7
write(1, "PrimeCount(1, 20) = 8 (sieve)\n", 30) = 30
read(0, "2 27\n", 1024)                 = 5
write(1, "Decimal: 27\n", 12)           = 12
write(1, "Ternary: 1000\n", 14)         = 14
read(0, "0\n", 1024)                    = 2
write(1, "Switched to impl 1 (Light)\n\n", 28) = 28
read(0, "2 27\n", 1024)                 = 5
write(1, "Decimal: 27\n", 12)           = 12
write(1, "Binary: 11011\n", 14)         = 14
read(0, "exit\n", 1024)                 = 5
munmap(0x7f20aad58000, 16448)           = 0
munmap(0x7f20aab24000, 16464)           = 0
write(1, "Goodbye!\n", 9)               = 9
exit_group(0)                           = ?
+++ exited with 0 +++
```

Листинг 9: *Strace логи к program2*