Московский Авиационный Институт (Национальный Исследовательский Университет)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра №806 «Вычислительной математики и программирования»

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

Группа: М8О-215Б-23

Студент: Лизунов К. Р.

Преподаватель: Миронов Е. С.

Оценка: Дата:

Москва, 2024

**Вариант 20**

**Постановка задачи**

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

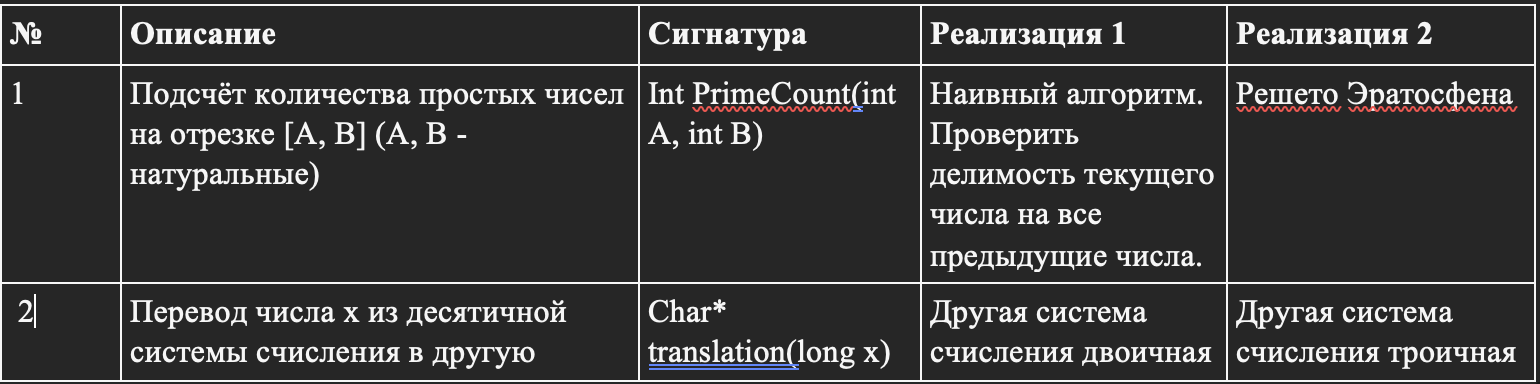
1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

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

* Динамические библиотеки, реализующие контракты, которые заданы вариантом;
* Тестовая программа (программа №1), которая используют одну из библиотек, используя

знания полученные на этапе компиляции;

* Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.



**Общий метод и алгоритм решения**

**Использованные системные вызовы:**

* void\* dlopen(const char\* filename, int flag); – загружает динамическую библиотеку в память.
* int dlclose(void\* handle); – выгружает динамическую библиотеку из памяти.
* void\* dlsym(void\* handle, const char\* symbol); – получает адрес функции из библиотеки.
* char\* dlerror(void); – возвращает строку с описанием последней ошибки.

**Общий алгоритм решения**

**program\_1.c (статическая линковка)**

1. **Меню команд**  
   Программа предлагает пользователю выбрать одну из команд:
   * 0: Выйти из программы
   * 1: Вычислить количество простых чисел в диапазоне
   * 2: Перевести число в новую систему счисления
2. **Выбор команды**  
   Пользователь вводит команду, после чего программа выполняет соответствующее действие.
3. **Выполнение команд**
   * **Команда 1**
     + Пользователь вводит два числа A и B.
     + Вызывается функция PrimeCount(A, B), которая вычисляет количество простых чисел.
     + Результат выводится на экран.
   * **Команда 2**
     + Пользователь вводит число x.
     + Вызывается функция translation(x), которая переводит число в новую систему счисления.
     + Результат выводится на экран.
   * **Команда 0**
     + Программа завершает работу.

**program\_2.c (динамическая загрузка библиотек)**

1. **Загрузка библиотеки**
   * Программа загружает libnaive.so с помощью dlopen().
   * Получает указатели на функции PrimeCount и translation с помощью dlsym().
   * Если загрузка или получение указателей не удалось, программа выводит ошибку.
2. **Меню команд**
   * 0: Переключиться на другую библиотеку (libnaive.so ↔ liboptimized.so).
   * 1: Вычислить количество простых чисел в диапазоне.
   * 2: Перевести число в новую систему счисления.
3. **Выбор команды**
   * Пользователь вводит команду.
   * В зависимости от команды программа выполняет соответствующее действие.
4. **Выполнение команд**
   * **Команда 0 (переключение библиотеки)**
     + Текущая библиотека закрывается с помощью dlclose().
     + Загружается другая библиотека (libnaive.so или liboptimized.so).
     + Если загрузка успешна, программа сообщает о переключении.
   * **Команда 1 (поиск простых чисел)**
     + Пользователь вводит A и B.
     + Вызывается функция PrimeCount(A, B) из загруженной библиотеки.
     + Выводится результат и информация о текущей библиотеке.
   * **Команда 2 (перевод числа)**
     + Пользователь вводит x.
     + Вызывается translation(x), выполняющий перевод числа.
     + Выводится результат и информация о текущей библиотеке.
5. **Завершение программы**
   * При выходе программа закрывает загруженную библиотеку с dlclose().

**Код программы**

Libnaive.c

#include <stdbool.h>

#include <stdlib.h>

int PrimeCount(int A, int B) {

int count = 0;

for (int i = A; i <= B; i++) {

bool is\_prime = true;

if (i < 2) continue;

for (int j = 2; j \* j <= i; j++) {

if (i % j == 0) {

is\_prime = false;

break;

}

}

if (is\_prime) count++;

}

return count;

}

char\* translation(long x) {

char\* result = (char\*)malloc(65);

if (!result) return NULL;

int index = 0;

do {

result[index++] = (x % 2) + '0';

x /= 2;

} while (x > 0);

result[index] = '\0';

for (int i = 0; i < index / 2; i++) {

char temp = result[i];

result[i] = result[index - 1 - i];

result[index - 1 - i] = temp;

}

return result;

}

Liboptimized.c

#include <stdbool.h>

#include <stdlib.h>

#include <string.h>

int PrimeCount(int A, int B) {

if (B < 2) return 0;

bool\* sieve = (bool\*)malloc((B + 1) \* sizeof(bool));

if (!sieve) return 0;

memset(sieve, true, (B + 1) \* sizeof(bool));

sieve[0] = sieve[1] = false;

for (int i = 2; i \* i <= B; i++) {

if (sieve[i]) {

for (int j = i \* i; j <= B; j += i) {

sieve[j] = false;

}

}

}

int count = 0;

for (int i = A; i <= B; i++) {

if (sieve[i]) count++;

}

free(sieve);

return count;

}

char\* translation(long x) {

char\* result = (char\*)malloc(65);

if (!result) return NULL;

int index = 0;

do {

result[index++] = (x % 3) + '0';

x /= 3;

} while (x > 0);

result[index] = '\0';

for (int i = 0; i < index / 2; i++) {

char temp = result[i];

result[i] = result[index - 1 - i];

result[index - 1 - i] = temp;

}

return result;

}

Program1.c

#include <stdio.h>

#include "libnaive.c"

int main() {

int command;

while (1) {

printf("Enter command (0 - exit, 1 - PrimeCount, 2 - translation): ");

scanf("%d", &command);

if (command == 0) {

printf("\nGoodbye!\nHave a nice day!\n");

break;

}

if (command == 1) {

int A, B;

printf(" Enter A and B: ");

scanf("%d %d", &A, &B);

printf(" Prime count: %d\n", PrimeCount(A, B));

}

else if (command == 2) {

long x;

printf(" Enter number: ");

scanf("%ld", &x);

char\* result = translation(x);

if (result) {

printf(" Result: %s\n", result);

free(result);

} else {

printf("Error in translation function.\n");

}

}

else {

printf("Invalid command\n");

}

}

return 0;

}

Program\_2.c

#include <stdio.h>

#include <dlfcn.h>

#include <stdlib.h>

typedef int (\*PrimeCountFunc)(int, int);

typedef char\* (\*TranslationFunc)(long);

int main() {

void\* handle = dlopen("./libnaive.so", RTLD\_LAZY);

if (!handle) {

fprintf(stderr, "Error loading library: %s\n", dlerror());

return 1;

}

PrimeCountFunc PrimeCount = (PrimeCountFunc)dlsym(handle, "PrimeCount");

TranslationFunc translation = (TranslationFunc)dlsym(handle, "translation");

if (!PrimeCount || !translation) {

fprintf(stderr, "Error loading symbols: %s\n", dlerror());

dlclose(handle);

return 1;

}

int is\_naive = 1;

int command;

while (1) {

printf("Enter command (0 - switch library, 1 - PrimeCount, 2 - translation): ");

scanf("%d", &command);

if (command == 0) {

dlclose(handle);

if (is\_naive) {

handle = dlopen("./liboptimized.so", RTLD\_LAZY);

if (!handle) {

fprintf(stderr, "Error switching to liboptimized: %s\n", dlerror());

return 1;

}

printf("\nSwitched to optimized library\n\n");

} else {

handle = dlopen("./libnaive.so", RTLD\_LAZY);

if (!handle) {

fprintf(stderr, "Error switching to libnaive: %s\n", dlerror());

return 1;

}

printf("\nSwitched to naive library\n\n");

}

is\_naive = !is\_naive;

PrimeCount = (PrimeCountFunc)dlsym(handle, "PrimeCount");

translation = (TranslationFunc)dlsym(handle, "translation");

if (!PrimeCount || !translation) {

fprintf(stderr, "Error loading symbols: %s\n", dlerror());

dlclose(handle);

return 1;

}

}

else if (command == 1) {

int A, B;

printf(" Enter A and B: ");

scanf("%d %d", &A, &B);

printf(" Prime count: %d\n", PrimeCount(A, B));

}

else if (command == 2) {

long x;

printf(" Enter number: ");

scanf("%ld", &x);

char\* result = translation(x);

if (result) {

printf(" Result: %s\n", result);

free(result);

}

}

}

dlclose(handle);

return 0;

}

user@MacBook-Air-User lab4 % ./program\_1

Enter command (0 - exit, 1 - PrimeCount, 2 - translation): 1

Enter A and B: 20 25

Prime count: 1

Enter command (0 - exit, 1 - PrimeCount, 2 - translation): 2

Enter number: 2025

Result: 11111101001

Enter command (0 - exit, 1 - PrimeCount, 2 - translation): 0

Goodbye!

Have a nice day!

**Strace**

neo@NeoBot22:~/Lab\_4$ strace ./program\_1

execve("./program\_1", ["./program\_1"], 0x7ffec5b34660 /\* 35 vars \*/) = 0 brk(NULL) = 0x55596fa63000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffde8f359a0) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f033a93b000

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=18507, ...},

AT\_EMPTY\_PATH) = 0

mmap(NULL, 18507, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f033a936000

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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 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)

= 0x7f033a70d000

mprotect(0x7f033a735000, 2023424, PROT\_NONE) = 0

mmap(0x7f033a735000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f033a735000

mmap(0x7f033a8ca000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) =

0x7f033a8ca000

mmap(0x7f033a923000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7f033a923000

mmap(0x7f033a929000, 52816, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f033a929000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f033a70a000

arch\_prctl(ARCH\_SET\_FS, 0x7f033a70a740) = 0 set\_tid\_address(0x7f033a70aa10) = 276976

set\_robust\_list(0x7f033a70aa20, 24) = 0

rseq(0x7f033a70b0e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f033a923000, 16384, PROT\_READ) = 0

mprotect(0x55596f35d000, 4096, PROT\_READ) = 0

mprotect(0x7f033a975000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f033a936000, 18507) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x8), ...},

AT\_EMPTY\_PATH) = 0

getrandom("\xfe\xfd\xa7\xb6\x41\x0d\x20\x21", 8, GRND\_NONBLOCK) = 8 brk(NULL) = 0x55596fa63000

brk(0x55596fa84000) = 0x55596fa84000

newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x8), ...},

AT\_EMPTY\_PATH) = 0

write(1, "Enter command (0 - exit, 1 - Pri"..., 59Enter command (0 - exit, 1 -

PrimeCount, 2 - translation): ) = 59 read(0, 1

"1\n", 1024) = 2

write(1, " Enter A and B: ", 17 Enter A and B: ) = 17 read(0, 20 25

"20 25\n", 1024) = 6

write(1, " Prime count: 1\n", 17 Prime count: 1

) = 17

write(1, "Enter command (0 - exit, 1 - Pri"..., 59Enter command (0 - exit, 1 -

PrimeCount, 2 - translation): ) = 59 read(0, 2

"2\n", 1024) = 2

write(1, " Enter number: ", 16 Enter number: ) = 16 read(0, 2025

"2025\n", 1024) = 5

write(1, " Result: 11111101001\n", 22 Result: 11111101001

) = 22

write(1, "Enter command (0 - exit, 1 - Pri"..., 59Enter command (0 - exit, 1 -

PrimeCount, 2 - translation): ) = 59 read(0, 0

"0\n", 1024) = 2

write(1, "\nGoodbye!\n", 10 Goodbye!

) = 10

write(1, "Have a nice day!\n", 17Have a nice day!

) = 17

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek) exit\_group(0) = ?

+++ exited with 0 +++ neo@NeoBot22:~/Lab\_4$

**Выводы**

В процессе выполнения данной лабораторной работы я освоил работу с динамическими библиотеками, научившись подключать их в runtime программы. Я реализовал базовые алгоритмы для решения поставленных задач, используя библиотеки **libnaive.c** и **liboptimized.c**, и изучил, как динамические библиотеки взаимодействуют с системными вызовами. Это позволило мне лучше понять механизмы интеграции и использования внешних модулей в программах.