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: Ле Шон Лыонг

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 08.12.25

Москва, 2025

Постановка задачи

Вариант 23.

Необходимо разработать программу, демонстрирующую:

- статическую загрузку библиотеки (линковка на этапе сборки);
- динамическую загрузку библиотеки во время выполнения программы с использованием dlopen, dlsym, dlclose;
- корректную работу двух альтернативных реализаций функций GCF(int A, int B) и E(int x), размещённых в библиотеках libimpl1.so и libimpl2.so;
- возможность выбора режима работы программы: без аргументов — статическая версия, с аргументом dynamic — динамическая загрузка и переключение реализации командой 0.

Программа должна принимать команды пользователя и выполнять вычисления с использованием активной реализации библиотеки.

Общий метод и алгоритм решения

Использованные системные вызовы

• **openat()** — открытие файлов библиотек (`.so`).

• **read()** — чтение ELF-заголовков и секций динамической библиотеки.

• **mmap()** — отображение сегментов библиотеки в адресное пространство процесса.

• **mprotect()** — установка прав доступа к загруженным сегментам.

• **fstat()** — получение информации о библиотеке (размер, тип файла).

• **close()** — закрытие файловых дескрипторов, связанных с библиотекой.

• **write()** — вывод результатов работы программы.

• **exit_group()** — завершение процесса.

Алгоритм работы:

Решение состоит из двух частей: статической и динамической загрузки функций GCF(int A, int B) и E(int x).

1. Статическая компоновка

- Библиотека libimpl1.so, содержащая первую реализацию функций GCF и E, компилируется заранее.

- Программа prog_static линкуется с этой библиотекой при сборке:
- gcc prog_static.c -L. -limpl1 -o prog_static
-
- Функции вызываются напрямую, а загрузчик ELF автоматически подгружает библиотеку при запуске программы.
- Основные системные вызовы загрузки:
  openat(), read(), mmap(), mprotect(), close().

  2. Динамическая загрузка
- При запуске программы prog_dynamic с аргументом "dynamic" используются обе библиотеки:
  libimpl1.so и libimpl2.so.
- Программа выполняет:
  - dlopen("./libimpl1.so", RTLD_LAZY)
  - dlopen("./libimpl2.so", RTLD_LAZY)
  - dlsym(handle, "GCF")
  - dlsym(handle, "E")

  - Вызываются функции GCF и E через полученные указатели.
  - Пользователь может переключать активную реализацию библиотек командой 0 без перезапуска программы.
- Основные системные вызовы динамического режима:
  openat(), read(), fstat(), mmap(), mprotect(), close().

  3. Алгоритм программы
- если программа запущена **без аргументов** → используется статическая реализация функций (prog_static);
- если указан аргумент **"dynamic"** → загружаются обе библиотеки, выбирается первая реализация, пользователь может переключать её командой 0;
- команда 1 A B → вычисление GCF(A, B);
- команда 2 x → вычисление E(x);
- команда q или exit → завершение работы.

Код программы

impl1.c

```
#include <math.h>
#include "calc.h"
```

```c
static int abs_int(int x) {
    return x < 0 ? -x : x;
}

int GCF(int A, int B) {
    int a = abs_int(A);
    int b = abs_int(B);

    if (a == 0) return b;
    if (b == 0) return a;

    while (b != 0) {
        int t = a % b;
        a = b;
        b = t;
    }
    return a;
}

float E(int x) {
    if (x <= 0) {
        return NAN;
    }

    double base = 1.0 + 1.0 / (double)x;
    double val  = pow(base, (double)x);
    return (float)val;
}
```

Impl2.c

```c
#include <math.h>
#include "calc.h"

static int abs_int(int x) {
    return x < 0 ? -x : x;
}

int GCF(int A, int B) {
    int a = abs_int(A);
    int b = abs_int(B);

    if (a == 0) return b;
    if (b == 0) return a;

    int min = (a < b) ? a : b;
    int g   = 1;

    for (int d = 1; d <= min; ++d) {
        if (a % d == 0 && b % d == 0) {
            g = d;
```

```
        }
    }
    return g;
}

float E(int x) {
    if (x < 0) {
        return NAN;
    }

    double sum  = 0.0;
    double fact = 1.0;

    for (int n = 0; n <= x; ++n) {
        if (n > 0) {
            fact *= (double)n;
        }
        sum += 1.0 / fact;
    }
    return (float)sum;
}
```

Calc.h

```
#ifndef CALC_H
#define CALC_H

int   GCF(int A, int B);
float E(int x);

#endif
```

prog_dynamic.c

```
#define _POSIX_C_SOURCE 200112L
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
#include "calc.h"

typedef int   (*GCF_fn_t)(int, int);
typedef float (*E_fn_t)(int);

struct Impl {
    void    *handle;
    GCF_fn_t GCF;
    E_fn_t   E;
    const char *path;
};
```

```c
int main(void) {
    struct Impl impls[2] = {
        {NULL, NULL, NULL, "./libimpl1.so"},
        {NULL, NULL, NULL, "./libimpl2.so"}
    };

    for (int i = 0; i < 2; ++i) {
        impls[i].handle = dlopen(impls[i].path, RTLD_LAZY);
        if (!impls[i].handle) {
            fprintf(stderr, "dlopen(%s) failed: %s\n",
                    impls[i].path, dlerror());
            for (int j = 0; j < i; ++j) {
                dlclose(impls[j].handle);
            }
            return 1;
        }

        dlerror();

        impls[i].GCF = (GCF_fn_t)dlsym(impls[i].handle, "GCF");
        const char *err = dlerror();
        if (err) {
            fprintf(stderr, "dlsym(GCF) in %s failed: %s\n",
                    impls[i].path, err);
            for (int j = 0; j <= i; ++j) {
                dlclose(impls[j].handle);
            }
            return 1;
        }

        impls[i].E = (E_fn_t)dlsym(impls[i].handle, "E");
        err = dlerror();
        if (err) {
            fprintf(stderr, "dlsym(E) in %s failed: %s\n",
                    impls[i].path, err);
            for (int j = 0; j <= i; ++j) {
                dlclose(impls[j].handle);
            }
            return 1;
        }
    }

    int current = 0;
    char cmd;

    while (1) {
        if (scanf(" %c", &cmd) != 1) {
            break;
        }

        if (cmd == 'q' || cmd == 'Q') {
            break;
        } else if (cmd == '0') {
            current = 1 - current;
```

```
        } else if (cmd == '1') {
            int A, B;
            if (scanf("%d %d", &A, &B) != 2) {
                break;
            }
            int g = impls[current].GCF(A, B);
            printf("%d\n", g);
        } else if (cmd == '2') {
            int x;
            if (scanf("%d", &x) != 1) {
                break;
            }
            float e_val = impls[current].E(x);
            printf("%f\n", e_val);
        } else {
        }
    }

    for (int i = 0; i < 2; ++i) {
        if (impls[i].handle) {
            dlclose(impls[i].handle);
        }
    }

    return 0;
}
```

prog_static.c

```
#include <stdio.h>
#include "calc.h"

int main(void) {
    char cmd;

    while (1) {
        if (scanf(" %c", &cmd) != 1) {
            break;
        }

        if (cmd == 'q' || cmd == 'Q') {
            break;
        } else if (cmd == '1') {
            int A, B;
            if (scanf("%d %d", &A, &B) != 2) {
                return 1;
            }
            int g = GCF(A, B);
            printf("%d\n", g);
        } else if (cmd == '2') {
            int x;
            if (scanf("%d", &x) != 1) {
```

```
                return 1;
            }
            float e_val = E(x);
            printf("%f\n", e_val);
        } else {
        }
    }

    return 0;
}
```

Протокол работы программы

```
on@DESKTOP-F8Q4L97:/mnt/c/Users/Son/OS_LABS/OS_LABS/lab4/src$ ./prog_static
1 24 18
6
2 10
2.593742
on@DESKTOP-F8Q4L97:/mnt/c/Users/Son/OS_LABS/OS_LABS/lab4/src$ ./prog_dynamic
1 24 18
6
0
2 10
2.718282
```

Strace:

Статическая версия (prog_static):
160 15:03:09.943847 execve("./prog_static", ["./prog_static"], 0x7ffc3a140e28 /* 28 vars */) = 0 <0.006940>
160 15:03:09.952265 brk(NULL) = 0x572f9a252000 <0.000254>
160 15:03:09.953835 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f70a7a35000 <0.000149>
160 15:03:09.955341 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory) <0.000079>
160 15:03:09.956396 openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/ libimpl1.so ", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory) <0.000705>
160 15:03:09.958250 openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v2/ libimpl1.so ", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory) <0.000442>
160 15:03:09.960984 openat(AT_FDCWD, "./libimpl1.so", O_RDONLY|O_CLOEXEC) = 3 <0.001567>
160 15:03:09.963606 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\0\0\0\0\0@5\0\0\0\0\0\0\0\0\0@\08\0\v\0@\0\36\0\35\0\1\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\0\0@\5\0\0\0\0\0\0@\5\0\0\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\5\0\0\0\0\20\0\0\0\0\0\0\20\0\0\0\0\0\0\20\0\0\0\0\0\241\1\0\0\0\0\0\0\241\1\0\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\4\0\0\0\0 \0\0\0\0\0\0\0 \0\0\0\0\0\0"..., 832) = 832 <0.000664>
160 15:03:09.965399 fstat(3, {st_mode=S_IFREG|0777, st_size=15552, ...}) = 0 <0.000564>
160 15:03:09.967091 getcwd("/mnt/c/Users/Son/OS_LABS/OS_LABS/lab4/src", 128) = 42 <0.000068>
160 15:03:09.968261 mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f70a7a30000 <0.000074>

160 15:03:09.969276 mmap(0x7f70a7a31000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x7f70a7a31000 <0.000088>
160 15:03:09.970415 mmap(0x7f70a7a32000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f70a7a32000 <0.000072>
160 15:03:09.971358 mmap(0x7f70a7a33000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x7f70a7a33000 <0.000095>
160 15:03:09.973066 close(3) = 0 <0.000066>
160 15:03:09.974102 openat(AT_FDCWD, "./glibc-hwcaps/x86-64-v3/libc.so.6", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No such file or directory) <0.000890>
... (аналогичные вызовы для libc.so.6 и libm.so.6) ...
160 15:03:10.032982 prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0 <0.000087>
160 15:03:10.033909 munmap(0x7f70a7a1e000, 71671) = 0 <0.000108>
160 15:03:10.036035 fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0 <0.000071>
160 15:03:10.036735 getrandom("\xe1\x9d\x9c\x55\xd1\xd5\x90\xb4", 8, GRND_NONBLOCK) = 8 <0.000077>
160 15:03:10.037334 brk(NULL) = 0x572f9a252000 <0.000065>
160 15:03:10.038088 brk(0x572f9a273000) = 0x572f9a273000 <0.000060>
160 15:03:10.038672 read(0, "1 24 18\n", 1024) = 8 <3.514842>
160 15:03:13.554585 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0 <0.000080>
160 15:03:13.556050 write(1, "6\n", 2) = 2 <0.000151>
160 15:03:13.557489 read(0, "2 10\n", 1024) = 5 <2.726724>
160 15:03:16.285873 write(1, "2.593742\n", 9) = 9 <0.000239>
160 15:03:16.287137 read(0, "", 1024) = 0 <1.304689>
160 15:03:17.593636 exit_group(0) = ?
160 15:03:17.596548 +++ exited with 0 +++

Динамическая версия (prog_dynamic):
182 15:03:25.865626 execve("./prog_dynamic", ["./prog_dynamic", "dynamic"], 0x7fff90508080 /* 28 vars */) = 0 <0.006122>
182 15:03:25.873513 brk(NULL) = 0x618ac06ff000 <0.000067>
182 15:03:25.875161 mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x77d58b448000 <0.000091>
... (загрузка libc.so.6) ...
182 15:03:25.921942 **openat(AT_FDCWD, "./ libimp1.so ", O_RDONLY|O_CLOEXEC) = 3 <0.001787>**
182 15:03:25.924610 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\0\0\0\0\0\0@5\0\0\0\0\0\0\0\0\0@\08\0\v\0@\0\36\0\35\0\1\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\0\0\0\0\0\0\0\0\0\0\0\0\0@\5\0\0\0\0\0\0@\5\0\0\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\5\0\0\0\0\20\0\0\0\0\0\0\20\0\0\0\0\0\0\20\0\0\0\0\0\0\241\1\0\0\0\0\0\241\1\0\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\4\0\0\0\0 \0\0\0\0\0\0\0 \0\0\0\0\0\0"..., 832) = 832 <0.000688>
182 15:03:25.926763 fstat(3, {st_mode=S_IFREG|0777, st_size=15552, ...}) = 0 <0.001055>
182 15:03:25.928817 getcwd("/mnt/c/Users/Son/OS_LABS/OS_LABS/lab4/src", 128) = 42 <0.000163>
182 15:03:25.930052 mmap(NULL, 16408, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x77d58b443000 <0.000098>
182 15:03:25.931146 mmap(0x77d58b444000, 4096, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x77d58b444000 <0.000064>
182 15:03:25.931875 mmap(0x77d58b445000, 4096, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x77d58b445000 <0.000055>
182 15:03:25.932568 mmap(0x77d58b446000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x77d58b446000 <0.000055>
182 15:03:25.933514 close(3) = 0 <0.000054>

... (загрузка limb.so.6) ...

```
182 15:03:25.943938 openat(AT_FDCWD, "./libimp2.so", O_RDONLY|O_CLOEXEC) = 3
<0.000876>
182 15:03:25.945383 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@\0\0\0\0\0\0\0\3504\0\0\0\0\0\0\0\0\0
@\08\0\v\0@\0\32\0\31\0\1\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\0\0\0\0\0\310\4\0\0\0\0\0\0\
310\4\0\0\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\5\0\0\0\0\20\0\0\0\0\0\0\0\20\0\0\0\0\0\0\0\20\0\0\0\0\0\0\301\1\
0\0\0\0\0\0\301\1\0\0\0\0\0\0\0\20\0\0\0\0\0\0\1\0\0\0\4\0\0\0\0 \0\0\0\0\0\0\0 \0\0\0\0\0\0"..., 832) = 832
<0.000247>
182 15:03:25.946108 fstat(3, {st_mode=S_IFREG|0777, st_size=15208, ...}) = 0 <0.000327>
182 15:03:25.946807 getcwd("/mnt/c/Users/Son/OS_LABS/OS_LABS/lab4/src", 128) = 42 <0.000055>
182 15:03:25.947231 mmap(NULL, 16400, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x77d58b43e000 <0.000061>
182 15:03:25.947643 mmap(0x77d58b43f000, 4096, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1000) = 0x77d58b43f000 <0.000067>
182 15:03:25.948061 mmap(0x77d58b440000, 4096, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x77d58b440000 <0.000061>
182 15:03:25.948473 mmap(0x77d58b441000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2000) = 0x77d58b441000 <0.000060>
182 15:03:25.949098 close(3) = 0 <0.000088>
182 15:03:25.949650 mprotect(0x77d58b441000, 4096, PROT_READ) = 0 <0.000060>
182 15:03:25.950210 fstat(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
<0.000099>
182 15:03:25.954697 read(0, "1 24 18\n", 1024) = 8 <2.337602>
182 15:03:28.294643 fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...}) = 0
<0.000279>
182 15:03:28.297122 write(1, "6\n", 2) = 2 <0.000218>
182 15:03:28.298965 read(0, "0\n", 1024) = 2 <3.484956>
182 15:03:31.785632 read(0, "2 10\n", 1024) = 5 <1.421596>
182 15:03:33.209055 write(1, "2.718282\n", 9) = 9 <0.000217>
182 15:03:33.210699 read(0, "", 1024) = 0 <2.072687>
182 15:03:35.284621 munmap(0x77d58b443000, 16408) = 0 <0.000558>
182 15:03:35.285778 munmap(0x77d58b117000, 950296) = 0 <0.000092>
182 15:03:35.286562 munmap(0x77d58b43e000, 16400) = 0 <0.000569>
182 15:03:35.288057 exit_group(0) = ?
182 15:03:35.289187 +++ exited with 0 +++
```

Вывод

В ходе выполнения лабораторной работы была реализована программа с поддержкой статической и динамической загрузки функций через разделяемые библиотеки. Получено практическое понимание механизма работы динамического загрузчика Linux: загрузка .so-файлов выполняется с использованием системных вызовов openat, read, fstat, mmap, mprotect и close.

Были созданы две библиотечные реализации и организовано переключение между ними в процессе выполнения программы. С помощью strace подтверждён факт загрузки

библиотек и показана последовательность системных вызовов, задействованных при инициализации и вызове функций. Программа корректно обрабатывает ввод, выполняет вычисления и переключение реализаций.

Работа позволила закрепить принципы динамической компоновки, работу ELF-загрузчика, а также взаимодействие программы с ОС на уровне системных вызовов.