Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"
Московский институт электроники и математики
им. А. Н. Тихонова НИУ ВШЭ
Департамент компьютерной инженерии

Лабораторная работа № 10
по теме:
«Организация файловой системы»

По дисциплине

«Операционные системы»

Группа: **БПМ 243**

Студент: **Николаев Степан Денисович**

Москва 2026

Цель работы

Познакомиться с устройством файловой системы Linux: узнать предел вложенности символьных ссылок, написать аналоги команд ls и ls -al.

Ход работы

Все программы собирал и запускал в виртуальной машине с Ubuntu в UTM под обычным пользователем marmos, компилятор gcc с ключами -Wall -Wextra, предупреждений нет.

Глубина рекурсии символьных связей

Программа symlink_depth.c создаёт каталог symlink_test (mkdir), заходит в него и создаёт обычный файл a через creat. Затем в цикле создаёт символьную ссылку на предыдущую (link1 на a, link2 на link1 и так далее) и пробует открыть её через open. Цикл заканчивается, когда open возвращает ошибку ELOOP; число успешно открытых ссылок и есть глубина. Имена ссылок взял link1, link2, … вместо aa, ab из задания. Всё лежит в отдельном каталоге, ссылки не удаляются.

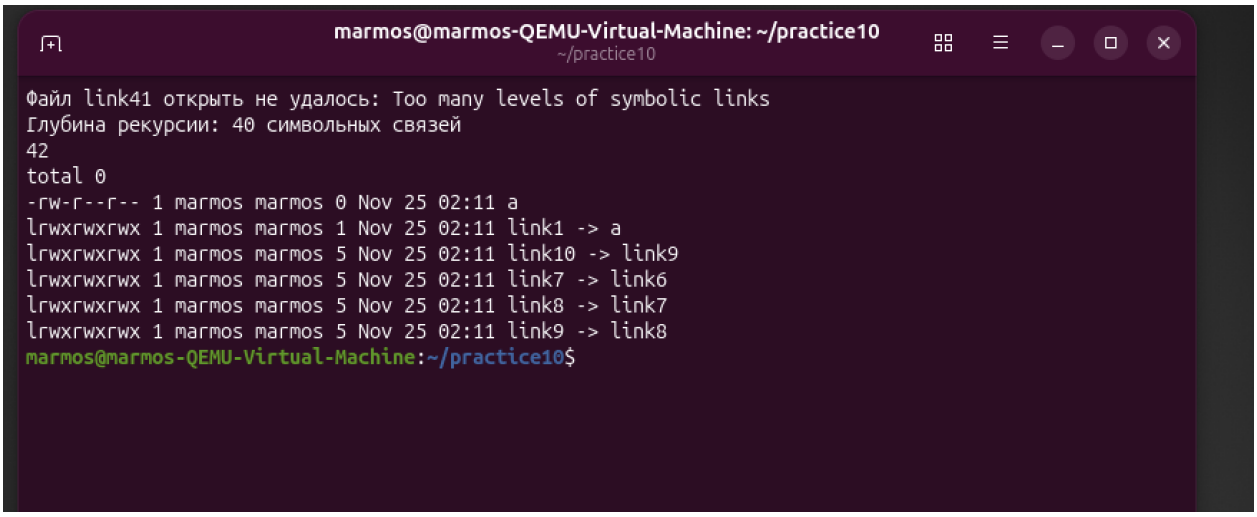

Рис. 1. Определение глубины рекурсии символьных связей.

Глубина получилась 40: link40 ещё открывается, a link41 уже нет (Too many levels of symbolic links). В каталоге 42 файла: a и 41 ссылка, последняя из которых как раз не открылась. Насколько я понял, 40 — ограничение самого ядра Linux (MAXSYMLINKS).

Аналог ls

myls.c принимает каталог аргументом, без аргумента берёт «.». Список читается через scandir (внутри он использует readdir) и сортируется по имени. Для каждой записи вызывается lstat, чтобы ссылки не разыменовывались, тип определяется макросами S_ISREG, S_ISDIR, S_ISLNK и другими. Для ссылок через readlink печатается цель.

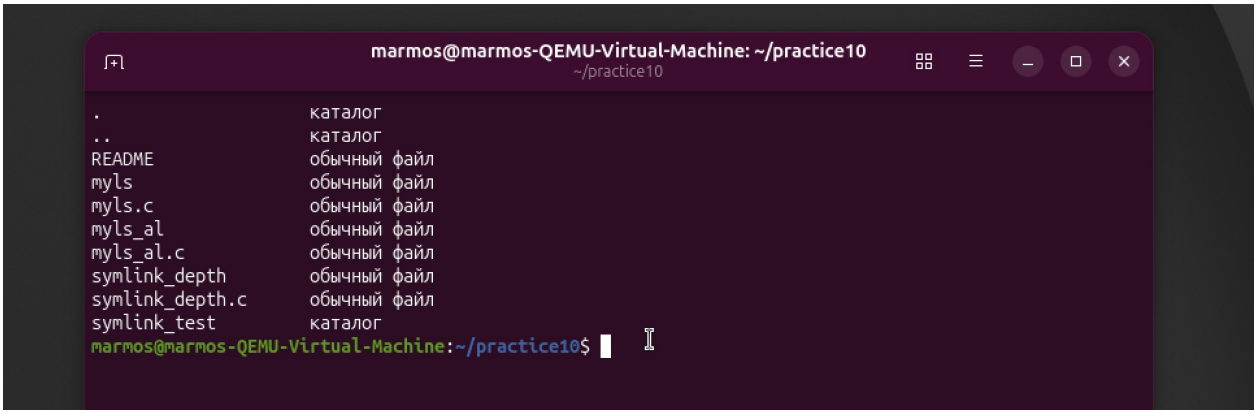

Рис. 2. Запуск ./myls в текущем каталоге.

```
marmos@marmos-QEMU-Virtual-Machine:~/practice10$ ./myls /dev | head -n 25
.                   каталог
..                  каталог
autofs              символьное устройство
block               каталог
btrfs-control       символьное устройство
bus                 каталог
char                каталог
console             символьное устройство
core                символьная ссылка -> /proc/kcore
cpu_dma_latency     символьное устройство
cpu_wakeup_latency  символьное устройство
cuse                символьное устройство
disk                каталог
dma_heap            каталог
dri                 каталог
ecryptfs            символьное устройство
fb0                 символьное устройство
fd                  символьная ссылка -> /proc/self/fd
full                символьное устройство
fuse                символьное устройство
hidraw0             символьное устройство
hidraw1             символьное устройство
hidraw2             символьное устройство
hugepages           каталог
hwrng               символьное устройство
marmos@marmos-QEMU-Virtual-Machine:~/practice10$
```

Рис. 3. Запуск ./myls /dev.

В начале списка /dev видны каталоги, символьные устройства и ссылки с целями (core -> /proc/kcore, fd -> /proc/self/fd). Каталог /dev в системе большой, поэтому здесь и дальше вывод обрезан до 25 строк через head.

Аналог ls -al

myls_al.c выводит всё как ls -al: строка total, тип и права (включая биты s, t), число ссылок, владелец, группа, размер, дата и имя; для ссылок добавляется «-> цель». Для символьных и блочных устройств вместо размера печатаются старший и младший номера. Дата показывается со временем, а для файлов старше полугода — с годом. Если у файла есть расширенные атрибуты (список доступа ACL, capabilities), после прав ставится «+», а у остальных строк каталога пробел, как в ls. Ширина колонок считается по самой длинной записи, поэтому всё выровнено. Описание и отличия от настоящего ls вынес в README (в приложении).

```
marmos@marmos-QEMU-Virtual-Machine:~/practice10$ LC_ALL=C ls -al /dev | head -n 25
total 4
drwxr-xr-x  18 root   root       4020 Nov 25 02:08 .
drwxr-xr-x  20 root   root       4096 Nov 20 14:36 ..
crw-r--r--   1 root   root     10, 235 Nov 25 02:02 autofs
drwxr-xr-x   2 root   root        360 Nov 25 02:08 block
crw-------   1 root   root     10, 234 Nov 25 02:02 btrfs-control
drwxr-xr-x   3 root   root         60 Jan  1  1970 bus
drwxr-xr-x   2 root   root       3680 Nov 25 02:02 char
crw-------   1 root   tty       5,   1 Nov 25 02:02 console
lrwxrwxrwx   1 root   root         11 Nov 25 02:02 core -> /proc/kcore
crw-------   1 root   root     10, 260 Nov 25 02:02 cpu_dma_latency
crw-------   1 root   root     10, 261 Nov 25 02:02 cpu_wakeup_latency
crw-------   1 root   root     10, 203 Nov 25 02:02 cuse
drwxr-xr-x  10 root   root        200 Nov 25 02:02 disk
drwxr-xr-x   2 root   root        100 Jan  1  1970 dma_heap
drwxr-xr-x   3 root   root        100 Nov 25 02:02 dri
crw-------   1 root   root     10, 258 Nov 25 02:02 ecryptfs
crw-rw----   1 root   video    29,   0 Nov 25 02:02 fb0
lrwxrwxrwx   1 root   root         13 Nov 25 02:02 fd -> /proc/self/fd
crw-rw-rw-   1 root   root      1,   7 Nov 25 02:02 full
crw-rw-rw-   1 root   root     10, 229 Nov 25 02:02 fuse
crw-------   1 root   root    234,   0 Nov 25 02:02 hidraw0
crw-------   1 root   root    234,   1 Nov 25 02:02 hidraw1
crw-------   1 root   root    234,   2 Nov 25 02:02 hidraw2
drwxr-xr-x   2 root   root          0 Nov 25 02:02 hugepages
marmos@marmos-QEMU-Virtual-Machine:~/practice10$
```

Рис. 4. Вывод команды ls -al /dev (первые 25 строк).

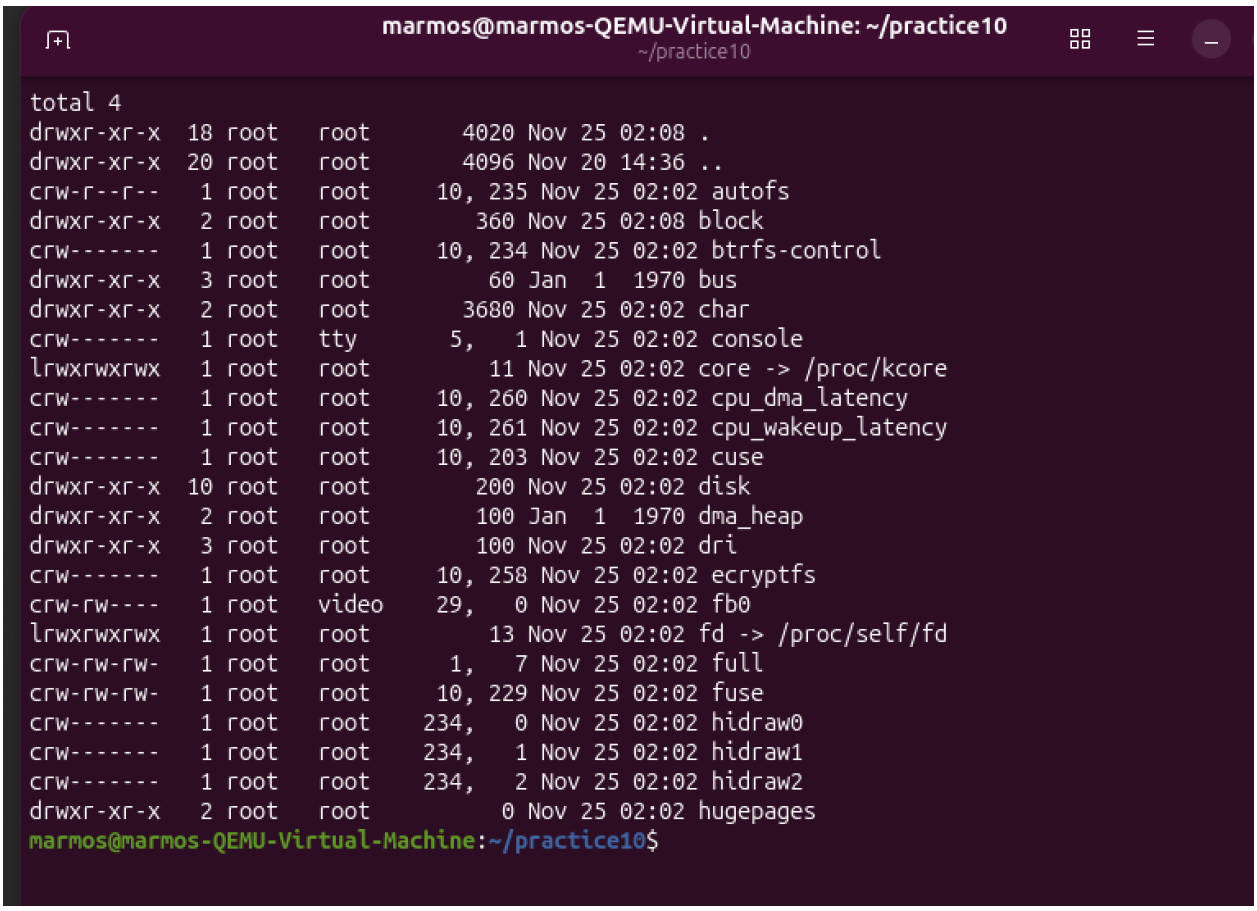

Рис. 5. Вывод ./myls_al /dev (первые 25 строк).

Для точного сравнения прогнал оба варианта через diff на четырёх каталогах. ls запускал с LC_ALL=C, чтобы даты были на английском, как у myls_al. «совпадает» печатается, если различий нет.

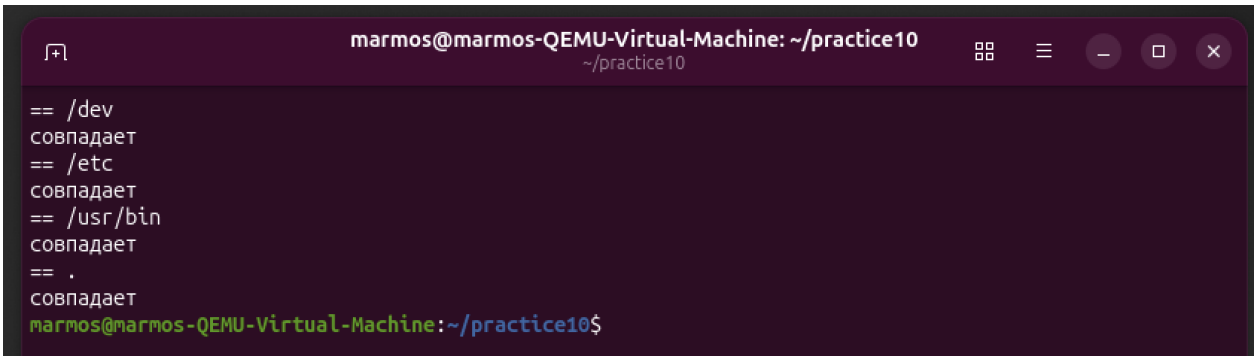

Рис. 6. Сравнение вывода ls -al и myls_al через diff.

Первые прогоны показали две разницы. У части файлов в /dev и /usr/bin есть расширенные атрибуты (ACL у устройств, capabilities у программ вроде ping), и ls ставит у них «+» после прав, а у остальных строк добавляет пробел. Кроме того, ширину колонки размера ls считает по самой длинной паре «major, minor», а не по отдельным максимумам. Добавил в myls_al проверку атрибутов через listxattr и поправил расчёт ширины, после этого diff на /dev, /etc, /usr/bin и в текущем каталоге различий не нашёл. Отличия от настоящего ls остались только в том, чего я не делал: нет значка «.» (контекст SELinux), цветов и разбора ключей, сортировка простым сравнением строк, без учёта локали.

Вывод. Определил, что глубина вложенности символьных ссылок в этой системе равна 40, написал программы, похожие на ls и ls -al. На проверенных каталогах вывод myls_al совпал с ls -al.

Приложение symlink_depth.c:

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/stat.h>

int main() {
    // Всё создаём в отдельной директории, чтобы потом удалить одной командой
    if (mkdir("symlink_test", 0755) == -1) {
        perror("mkdir symlink_test");
        return 1;
    }
    if (chdir("symlink_test") == -1) {
        perror("chdir");
        return 1;
    }

    int fd = creat("a", 0644);
    if (fd == -1) {
        perror("creat");
        return 1;
    }
    close(fd);

    char prev[64] = "a";
    char name[64];
    int depth = 0;

    for (;;) {
        snprintf(name, sizeof(name), "link%d", depth + 1);
        if (symlink(prev, name) == -1) {
            perror("symlink");
            return 1;
        }
        fd = open(name, O_RDONLY);
        if (fd == -1) {
            if (errno == ELOOP) {
                printf("Файл %s открыть не удалось: %s\n", name, strerror(errno));
                break;
            }
            perror("open");
            return 1;
        }
        close(fd);
        depth++;
        strcpy(prev, name);
    }

    printf("Глубина рекурсии: %d символьных связей\n", depth);
    return 0;
}
```

Приложение myls.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <unistd.h>
#include <limits.h>
#include <sys/stat.h>

static const char *file_type(mode_t m) {
    if (S_ISREG(m)) return "обычный файл";
    if (S_ISDIR(m)) return "каталог";
    if (S_ISLNK(m)) return "символьная ссылка";
    if (S_ISCHR(m)) return "символьное устройство";
    if (S_ISBLK(m)) return "блочное устройство";
    if (S_ISFIFO(m)) return "FIFO";
    if (S_ISSOCK(m)) return "сокет";
    return "неизвестно";
}

static int by_name(const struct dirent **a, const struct dirent **b) {
    return strcmp((*a)->d_name, (*b)->d_name);
}

int main(int argc, char *argv[]) {
    const char *dir = argc > 1 ? argv[1] : ".";

    struct dirent **list;
    int n = scandir(dir, &list, NULL, by_name);
    if (n == -1) {
        perror(dir);
        return 1;
    }

    for (int i = 0; i < n; i++) {
        char path[PATH_MAX];
        snprintf(path, sizeof(path), "%s/%s", dir, list[i]->d_name);

        struct stat st;
        if (lstat(path, &st) == -1) {
            perror(path);
            continue;
        }
        printf("%-20s %s", list[i]->d_name, file_type(st.st_mode));
        if (S_ISLNK(st.st_mode)) {
            char target[PATH_MAX];
            ssize_t len = readlink(path, target, sizeof(target) - 1);
            if (len != -1) {
                target[len] = '\0';
                printf(" -> %s", target);
            }
        }
        printf("\n");
        free(list[i]);
    }
    free(list);
    return 0;
}
```

Приложение myls_al.c:

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <dirent.h>
#include <unistd.h>
#include <limits.h>
#include <pwd.h>
#include <grp.h>
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include <sys/xattr.h>

struct entry {
    char name[NAME_MAX + 1];
    struct stat st;
    char target[PATH_MAX];
    int is_link;
    int has_acl;
};

static int cmp(const void *a, const void *b) {
    const struct entry *x = a, *y = b;
    return strcmp(x->name, y->name);
}

static void mode_string(mode_t m, char *s) {
    if (S_ISREG(m)) s[0] = '-';
    else if (S_ISDIR(m)) s[0] = 'd';
    else if (S_ISLNK(m)) s[0] = 'l';
    else if (S_ISCHR(m)) s[0] = 'c';
    else if (S_ISBLK(m)) s[0] = 'b';
    else if (S_ISFIFO(m)) s[0] = 'p';
    else if (S_ISSOCK(m)) s[0] = 's';
    else s[0] = '?';
    const char *rwx = "rwxrwxrwx";
    for (int i = 0; i < 9; i++)
        s[i + 1] = (m & (1 << (8 - i))) ? rwx[i] : '-';
    if (m & S_ISUID) s[3] = (m & S_IXUSR) ? 's' : 'S';
    if (m & S_ISGID) s[6] = (m & S_IXGRP) ? 's' : 'S';
    if (m & S_ISVTX) s[9] = (m & S_IXOTH) ? 't' : 'T';
    s[10] = '\0';
}

static int num_width(unsigned long v) {
    int w = 1;
    while (v >= 10) { v /= 10; w++; }
    return w;
}

int main(int argc, char *argv[]) {
    const char *dir = argc > 1 ? argv[1] : ".";
    DIR *d = opendir(dir);
    if (!d) {
        perror(dir);
        return 1;
    }

    struct entry *e = NULL;
    int n = 0, cap = 0;
    struct dirent *de;
    while ((de = readdir(d)) != NULL) {
        if (n == cap) {
```

```
            cap = cap ? cap * 2 : 64;
            e = realloc(e, cap * sizeof(*e));
            if (!e) {
                perror("realloc");
                return 1;
            }
        }
        struct entry *x = &e[n];
        strcpy(x->name, de->d_name);
        char path[PATH_MAX];
        snprintf(path, sizeof(path), "%s/%s", dir, de->d_name);
        if (lstat(path, &x->st) == -1) {
            perror(path);
            continue;
        }
        x->is_link = S_ISLNK(x->st.st_mode);
        // как в ls Ubuntu: "+" после прав, если у файла есть расширенные атрибуты
        // (ACL, capabilities); у символьной ссылки смотрится файл, на который она
указывает
        char names[1024];
        ssize_t nx = listxattr(path, names, sizeof(names));
        x->has_acl = nx > 0 || (nx == -1 && errno == ERANGE);
        if (x->is_link) {
            ssize_t len = readlink(path, x->target, sizeof(x->target) - 1);
            x->target[len > 0 ? len : 0] = '\0';
        }
        n++;
    }
    closedir(d);
    qsort(e, n, sizeof(*e), cmp);

    // Первый проход: ширины колонок и "total" (в блоках по 1 КиБ)
    long long total = 0;
    int any_acl = 0;
    int w_link = 1, w_user = 0, w_group = 0, w_size = 1, w_major = 0, w_minor = 0,
w_dev = 0;
    for (int i = 0; i < n; i++) {
        struct stat *st = &e[i].st;
        total += st->st_blocks;
        if (e[i].has_acl) any_acl = 1;
        if (num_width(st->st_nlink) > w_link) w_link = num_width(st->st_nlink);
        struct passwd *pw = getpwuid(st->st_uid);
        int lu = pw ? (int)strlen(pw->pw_name) : num_width(st->st_uid);
        struct group *gr = getgrgid(st->st_gid);
        int lg = gr ? (int)strlen(gr->gr_name) : num_width(st->st_gid);
        if (lu > w_user) w_user = lu;
        if (lg > w_group) w_group = lg;
        if (S_ISCHR(st->st_mode) || S_ISBLK(st->st_mode)) {
            int wa = num_width(major(st->st_rdev));
            int wb = num_width(minor(st->st_rdev));
            if (wa > w_major) w_major = wa;
            if (wb > w_minor) w_minor = wb;
            if (wa + 2 + wb > w_dev) w_dev = wa + 2 + wb;
        } else if (num_width(st->st_size) > w_size) {
            w_size = num_width(st->st_size);
        }
    }
    // ширина колонки размера: самое длинное число или самая длинная пара "major,
minor"
    if (w_dev > w_size)
        w_size = w_dev;
    printf("total %lld\n", total / 2);

    time_t now = time(NULL);
    for (int i = 0; i < n; i++) {
        struct stat *st = &e[i].st;
```

```
        char mode[11];
        mode_string(st->st_mode, mode);

        struct passwd *pw = getpwuid(st->st_uid);
        struct group *gr = getgrgid(st->st_gid);
        char user[32], group[32];
        if (pw) snprintf(user, sizeof(user), "%s", pw->pw_name);
        else snprintf(user, sizeof(user), "%u", st->st_uid);
        if (gr) snprintf(group, sizeof(group), "%s", gr->gr_name);
        else snprintf(group, sizeof(group), "%u", st->st_gid);

        char tbuf[32];
        struct tm tm;
        localtime_r(&st->st_mtime, &tm);
        // как в ls: файлы старше полугода показываются с годом вместо времени
        if (st->st_mtime > now || now - st->st_mtime > 15778476)
            strftime(tbuf, sizeof(tbuf), "%b %e  %Y", &tm);
        else
            strftime(tbuf, sizeof(tbuf), "%b %e %H:%M", &tm);

        // если хотя бы у одного файла есть ACL, ls добавляет колонку под "+" у
всех строк
        const char *acl = any_acl ? (e[i].has_acl ? "+" : " ") : "";
        printf("%s%s %*lu %-*s %-*s ", mode, acl, w_link, (unsigned long)st-
>st_nlink,
               w_user, user, w_group, group);
        if (S_ISCHR(st->st_mode) || S_ISBLK(st->st_mode))
            printf("%*u, %*u", w_size - w_minor - 2 > w_major ? w_size - w_minor -
2 : w_major,
                   major(st->st_rdev),
                   w_minor, minor(st->st_rdev));
        else
            printf("%*lld", w_size, (long long)st->st_size);
        printf(" %s %s", tbuf, e[i].name);
        if (e[i].is_link)
            printf(" -> %s", e[i].target);
        printf("\n");
    }
    free(e);
    return 0;
}
```

Приложение README:

```
myls_al - упрощённый аналог команды ls -al

Сборка:
    gcc -Wall -Wextra -o myls_al myls_al.c

Запуск:
    ./myls_al [каталог]
Без аргумента выводится содержимое текущего каталога.

Что выводится (как в ls -al): строка "total", затем для каждой записи,
включая . и .., в порядке сортировки по имени:
    тип и права (-rwxr-xr-x, d, l, c, b, p, s; биты s/S и t/T),
    число жёстких ссылок, владелец, группа,
    размер в байтах (для устройств - старший и младший номера через запятую),
    время изменения (файлы старше полугода - с годом вместо времени),
    имя; для символьных ссылок после " -> " цель ссылки.
Если у файла есть расширенные атрибуты (ACL, capabilities), после прав
ставится "+", а у остальных строк в этом каталоге пробел (как делает ls
в Ubuntu). Ширина колонки размера считается по самому длинному числу
или самой длинной паре "major, minor" у устройств.

Отличия от настоящего ls -al:
    сортировка по strcmp, без учёта локали;
    нет значка "." (контекст SELinux) после прав;
    нет цветов и кавычек вокруг имён с пробелами;
    параметры командной строки не разбираются, принимается только один каталог.

Для сравнения: LC_ALL=C ls -al /dev  и  ./myls_al /dev
(LC_ALL=C нужен, чтобы ls печатал даты по-английски)

Остальные файлы:
    symlink_depth.c - глубина рекурсии символьных связей (каталог symlink_test);
    myls.c          - аналог ls: имена, типы, цели ссылок.
```