



# OS Lab 18

Status	approved
checkbox	✓
class	OS
due date	@Feb 24, 2021

## Task

Напишите программу - аналог команды `ls -ld`. Для каждого своего аргумента эта команда должна распечатывать:

- Биты состояния файла в воспринимаемой человеком форме:
  - `d` если файл является каталогом
  - если файл является обычным файлом
  - `?` во всех остальных случаях
- Три группы символов, соответствующие правам доступа для хозяина, группы и всех остальных:
  - `r` если файл доступен для чтения, иначе `✓`
  - `w` если файл доступен для записи, иначе `✓`
  - `x` если файл доступен для исполнения, иначе `-`
- Количество связей файла
- Имена собственника и группы файла (совет - используйте `getpwuid` и `getgrgid`).
- Если файл является обычным файлом, его размер. Иначе оставьте это поле пустым.
- Дату модификации файла (используйте `ctime`).
- Имя файла (если было задано имя с путем, нужно распечатать только имя).

Желательно, чтобы поля имели постоянную ширину, т.е. чтобы листинг имел вид таблицы.

Совет - используйте `printf`.

## Теория

Управляющая информация и информация о состоянии файла хранится в отдельной структуре данных, называемой инод (**inode**). Каждый инод идентифицируется номером, уникальным в пределах файловой системы. Любой файл единственным образом определяется в системе своим inode-номером и номером устройства файловой системы, содержащей этот файл.

Инод содержит такую информацию, как размер файла, права доступа, владелец, тип файла, различные временные отметки и т. д. Кроме того, файловая система хранит в иноде информацию о размещении блоков файла на носителе, но эта информация пользователю напрямую не доступна

Системный вызов `stat(2)` используется для получения информации о состоянии файла. Нет необходимости в праве доступа на чтение, изменение и исполнение для исследуемого файла.

```
struct stat {
    mode_t      st_mode; /* тип файла и режим (права доступа) */
    ino_t        st_ino; /* номер индексного узла */
    dev_t        st_dev; /* номер устройства (файловой системы) */
    dev_t        st_rdev; /* номер устройства для специальных файлов */
    nlink_t      st_nlink; /* количество ссылок */
    uid_t        st_uid; /* идентификатор пользователя владельца */
    gid_t        st_gid; /* идентификатор группы владельца */
    off_t        st_size; /* размер в байтах, для обычных файлов */
    struct timespec st_atim; /* время последнего обращения к файлу */
    struct timespec st_mtim; /* время последнего изменения файла */
    struct timespec st_ctim; /* время последнего изменения состояния файла */
    blksize_t     st_blksize; /* оптимальный размер блока ввода/вывода */
    blkcnt_t      st_blocks; /* количество занятых дисковых блоков */
};
```

`st_mode` — старшие 12 бит это права доступа, младшие 4 это тип файла

`st_rdev` — идентификатор файловой системы в которой лежит файл. (т.е чтобы убедиться что это один и тот же файл нужно чтобы совпадали номера инодов и `rdev`)

`st_ino` — номер инода. уникален в пределах файловой системы

Однако, все директории в путевом имени файла должны быть доступны на поиск. Системный вызов `fstat(2)` используется для уже открытых файлов. Системный вызов `lstat(2)` используется для символических связей (понятие символической связи рассматривается в разделе «Управление директориями»). Если файл не является символической связью, `lstat(2)` ведет себя так же, как `stat(2)`.

## Тип файла (st\_mode):

```
<sys/stat.h>
#define S_IFMT          0xF000 /* type of file */
#define S_IAMB          0x1FF  /* access mode bits */
#define S_IFIFO         0x1000 /* fifo */
#define S_IFCHR         0x2000 /* character special */
#define S_IFDIR         0x4000 /* directory */
/* XENIX definitions are not relevant to Solaris */
#define S_IFNAM         0x5000 /* XENIX special named file */
#define S_INSEM         0x1     /* XENIX semaphore subtype of IFNAM */
#define S_INSHD         0x2     /* XENIX shared data subtype of IFNAM */
#define S_IFBLK         0x6000 /* block special */
#define S_IFREG         0x8000 /* regular */
#define S_IFLNK         0xA000 /* symbolic link */
#define S_IFSOCK        0xC000 /* socket */
#define S_IFD00R        0xD000 /* door */
#define S_IFPORT        0xE000 /* event port */
#define S_ISUID         0x800   /* set user id on execution */
#define S_ISGID         0x400   /* set group id on execution */
#define S_ISVTX         0x200   /* save swapped text even after use */
#define S_IREAD         00400   /* read permission, owner */
#define S_IWRITE        00200   /* write permission, owner */
#define S_IEXEC         00100   /* execute/search permission, owner */
#define S_ENFMT         S_ISGID /* record locking enforcement flag */
#define S_IRWXU         00700   /* read, write, execute: owner */
#define S_IRUSR         00400   /* read permission: owner */
#define S_IWUSR         00200   /* write permission: owner */
#define S_IXUSR         00100   /* execute permission: owner */
#define S_IRWXG         00070   /* read, write, execute: group */
#define S_IRGRP         00040   /* read permission: group */
#define S_IWGRP         00020   /* write permission: group */
#define S_IXGRP         00010   /* execute permission: group */
#define S_IRWXO         00007   /* read, write, execute: other */
#define S_IROTH         00004   /* read permission: other */
```

Замечание: одновременно к двум типам файл принадлежать не может

Таблица 4.1. Макросы для определения типа файла из <sys/stat.h>

Макроопределение	Тип файла
S_ISREG()	Обычный файл
S_ISDIR()	Каталог
S_ISCHR()	Специальный файл символьного устройства
S_ISBLK()	Специальный файл блочного устройства
S_ISFIFO()	Канал (именованный или неименованный)
S_ISLNK()	Символическая ссылка
S_ISSOCK()	Сокет

`st_dev` Это поле идентифицирует файловую систему, которая содержит заданный файл. Это значение можно использовать как аргумент `ustat(2)`, чтобы получить больше информации о файловой системе.

`st_ino` Inode-номер заданного файла. Файл однозначно определяется с помощью `st_dev` и `st_ino`.

`st_mode` Это поле содержит биты прав доступа к файлу, тип файла и специальные биты. Это поле обсуждается более подробно на следующих страницах раздела. Биты доступа и специальные биты могут быть изменены с помощью системного вызова `chmod(2)`.

`st_nlink` Число жестких связей заданного файла. Другими словами, это число ссылающихся на заданный файл записей в директориях. Это понятие подробнее рассматривается в разделе «Управление директориями». Поле изменяется с помощью системных вызовов `link(2)` и `unlink(2)`.

`st_uid` Пользовательский идентификатор владельца файла. Он изменяется с помощью системных вызовов `chown(2)` и `lchown(2)` (для символических связей).

`st_gid` Идентификатор группы заданного файла. Он также изменяется с помощью системных вызовов `chown(2)` и `lchown(2)` (для символических связей).

`st_rdev` Это поле применяется только для байт- и блок-ориентированных специальных файлов и является идентификатором устройства, на которое этот файл ссылается.

`st_size` Размер файла в байтах. Он изменяется при записи в файл. Для специальных файлов этот размер всегда равен нулю.

`st_atime` Время последнего чтения файла. Для директории это время не изменяется при ее поиске (с помощью `cd`), но изменяется при просмотре содержания директории (с помощью `ls`), так как при этом читается файл директории. Это время изменяется следующими системными вызовами: `creat`, `mknod`, `utime` и `read`.

`st_mtime` Время последней записи в файл. Это время изменяется следующими системными вызовами: `creat`, `mknod`, `utime` и `write`.

`st_ctime` Время изменения состояния файла. Это время не изменяется при чтении и модификации содержимого файла. Это время изменяется следующими системными вызовами: `chmod`, `chown`, `creat`, `link`, `mknod`, `pipe`, `unlink`, `utime` и `write`.

## Из исходников соляриса

```
struct stat {
    __dev_t  st_dev;          /* inode's device */
    __ino_t  st_ino;          /* inode's number */
    __mode_t st_mode;         /* inode protection mode */
    __nlink_t st_nlink;       /* number of hard links */
    __uid_t  st_uid;          /* user ID of the file's owner */
    __gid_t  st_gid;          /* group ID of the file's group */
    __dev_t  st_rdev;         /* device type */
    struct timespec st_atim;  /* time of last access */
    struct timespec st_mtim;  /* time of last data modification */
    struct timespec st_ctim;  /* time of last file status change */
    off_t     st_size;        /* file size, in bytes */
    blkcnt_t  st_blocks;      /* blocks allocated for file */
    blksize_t st_blksize;     /* optimal blocksize for I/O */
    fflags_t  st_flags;       /* user defined flags for file */
    __uint32_t st_gen;        /* file generation number */
    __int32_t  st_lspare;
    struct timespec st_birthtim; /* time of file creation */
};
```



## Получение имени владельца

В структуре stat, информация о пользователе и группе файла хранится в виде числовых идентификаторов uid и gid. Для распечатки информации о файле, удобно было бы перевести эту информацию в имена пользователя и группы. Для этого необходимо обратиться к базе данных учетных записей. В традиционных Unix-системах эта база хранилась в текстовых файлах /etc/passwd и /etc/group (в более современных также в файле /etc/shadow). Современные системы могут также использовать распределенные сетевые базы, такие, как NIS, NIS+ и LDAP. Если ваша программа самостоятельно анализирует файл /etc/passwd, она потребует адаптации для работы на системах, использующих LDAP. Поэтому рекомендуется использовать стандартные библиотечные функции, которые поддерживают все типы и форматы БД учетных записей, поддерживаемые текущей версией системы, и используют именно ту БД, из которой настроены брать информацию стандартные утилиты, такие, как login(1) и su(1).

Стандартные библиотечные функции getpwent(3C), getpwuid(3C) и getpwnam(3C) возвращают указатель на структуру, которая содержит разбитую на поля строку из файла /etc/password или другой БД учетных записей, в зависимости от конфигурации системы. Каждая строка в файле представлена в формате структуры password, определенной следующим образом:

```
struct passwd {
    char          *pw_name;
    char          *pw_passwd;
    uid_t         pw_uid;
    gid_t         pw_gid;
    char          *pw_age;
    char          *pw_comment;
    char          *pw_gecos;
    char          *pw_dir;
    char          *pw_shell;
};
```

Наших учёток на солярисе в passwd нет, поскольку есть понятие PAM — plug-able authentication module (то есть там эти базы данных могут браться из каких то других источников). На нашем сервере данные берутся из NIS — network information service команда getent позволяет вывести БД в том виде в котором её видит система (getent etc/passwd)

```
d.khaetskaya@fit-main:/etc$ cat passwd
root:x:0:0:Super-User:/root:/usr/bin/bash
daemon:x:1:1:/:
bin:x:2:2:/:usr/bin:
sys:x:3:3:/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
dladm:x:15:65:DataLink Admin:/:
netadm:x:16:65:Network Admin:/:
netcfg:x:17:65:Network Configuration Admin:/:
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/var/lib/gdm:
zfsnap:x:51:12:ZFS Automatic Snapshots Reserved UID:/usr/bin/pfsh
upnp:x:52:52:UPnP Server Reserved UID:/var/coherence:/bin/ksh
xvm:x:60:60:xVM User:/:
mysql:x:70:70:MySQL Reserved UID:/:
openldap:x:75:75:OpenLDAP User:/:
websrvd:x:80:80:WebServer Reserved UID:/:
svctag:x:95:12:Service Tag UID:/:
unknown:x:96:96:Unknown Remote UID:/:
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
sshd:x:22:22:sshd privsep:/var/empty:/bin/false
pkg5srv:x:97:97:pkg(5) server UID:/:
```

```
d.khaetskaya@fit-main:/etc$ getent passwd
root:x:0:0:Super-User:/root:/usr/bin/bash
daemon:x:1:1:/:
bin:x:2:2:/:usr/bin:
sys:x:3:3:/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
dladm:x:15:65:DataLink Admin:/:
netadm:x:16:65:Network Admin:/:
netcfg:x:17:65:Network Configuration Admin:/:
smmsp:x:25:25:SendMail Message Submission Program:/:
listen:x:37:4:Network Admin:/usr/net/nls:
gdm:x:50:50:GDM Reserved UID:/var/lib/gdm:
zfsnap:x:51:12:ZFS Automatic Snapshots Reserved UID:/usr/bin/pfsh
upnp:x:52:52:UPnP Server Reserved UID:/var/coherence:/bin/ksh
xvm:x:60:60:xVM User:/:
mysql:x:70:70:MySQL Reserved UID:/:
openldap:x:75:75:OpenLDAP User:/:
websrvd:x:80:80:WebServer Reserved UID:/:
svctag:x:95:12:Service Tag UID:/:
unknown:x:96:96:Unknown Remote UID:/:
nobody:x:60001:60001:NFS Anonymous Access User:/:
noaccess:x:60002:60002:No Access User:/:
nobody4:x:65534:65534:SunOS 4.x NFS Anonymous Access User:/:
sshd:x:22:22:sshd privsep:/var/empty:/bin/false
pkg5srv:x:97:97:pkg(5) server UID:/:
denchenko:x:3710:3252:Denchenko Daria Mikhailovna:/home/students/16200/denchenko:/bin/bash
galios:x:3698:3252:Galios Maksim Vitalevich:/home/students/16200/galios:/bin/bash
v.abrosimov:x:4448:4451:Vladimir Abrosimov:/home/students/19200/v.abrosimov:/bin/bash
shchuka:x:3648:3249:Shchukantasov Emil Kharitonovich:/home/students/15200/shchuka:/bin/bash
ivanishkin:x:3721:3252:Ivanishkin Dmitrii Sergeevich:/home/students/16200/ivanishkin:/bin/bash
n.zelenskikh:x:4229:4229:Zelenskikh Mark:/home/students/2019/n.zelenskikh:/bin/bash
pelmenev:x:3612:3249:Pelmenev Petr Viktorovich:/home/students/15200/pelmenev:/bin/bash
y.olimpiev:x:4487:4490:Yuriy Olimpiev:/home/students/19200/y.olimpiev:/bin/bash
shestakova:x:3989:3253:Shestakova Elizaveta Alekseevna:/home/students/17200/shestakova:/bin/bash
khokhlov:x:3964:3253:Khokhlov Mikhail Grigorevich:/home/students/17200/khokhlov:/bin/bash
b.borchenko:x:4408:4411:Boris Borchenko:/home/students/19200/b.borchenko:/bin/bash
mkarpunin:x:4013:3254:Karpunin Mikhail Pavlovich:/home/students/18200/mkarpunin:/bin/bash
v.kuznetsov:x:4228:4228:Kuznetsov Vladimir:/home/students/2019/v.kuznetsov:/bin/bash
k.galitsheva:x:4453:4456:Kristina Galitsheva:/home/students/19200/k.galitsheva:/bin/bash
kuznetsov:x:3596:3249:Kuznetsov Mikhail Iurevich:/home/students/15200/kuznetsov:/bin/bash
cherdantseva:x:3637:3249:Cherdantseva Marina Andreevna:/home/students/15200/cherdantseva:/bin/bash
bastrykina:x:3696:3252:Bastrykina Alena Alekseevna:/home/students/16200/bastrykina:/bin/bash
redko:x:4006:3254:Redko Anna Iurevna:/home/students/18200/redko:/bin/bash
g.stavnichik:x:4197:4197:Stavnichik Gleb:/home/students/2019/g.stavnichik:/bin/bash
evax:x:3918:3253:Axel:/home/students/17200/evax:/bin/bash
kanaev:x:4037:3254:Kanaev Igor Igorevich:/home/students/18200/kanaev:/bin/bash
n.malakhova:x:4393:4396:Marlya Malakhova:/home/students/19200/n.malakhova:/bin/bash
kuleshov:x:3598:3249:Kuleshov Danil Sergeevich:/home/students/15200/kuleshov:/bin/bash
usova:x:3894:3253:Usova Daria Sergeevna:/home/students/17200/usova:/bin/bash
tignatenko:x:3461:3247:Ignatenko Tatiana Andreevna:/home/students/14200/tignatenko:/bin/bash
makovkin:x:3968:3253:Makovkin Vadim Denisovich:/home/students/17200/makovkin:/bin/bash
```

Замечание: Поле pw\_comment не используется. Другие поля имеют значения, соответствующие описаным в password(4). При первом вызове getpwent(3C) возвращает указатель на первую структуру password в файле. В следующий раз getpwent(3C) вернет указатель на следующую структуру password. Последовательные вызовы могут использоваться для поиска во всем файле.

getpwuid(3C) ищет от начала файла, пока не найдет структуру с полем идентификатора пользователя равным uid. Возвращает указатель на найденную структуру.

getpwnam(3C) ищет от начала файла, пока не найдет структуру с полем регистрационного имени пользователя равным name. Возвращает указатель на найденную структуру.

Вызов setpwent(3C) предоставляет возможность вести последующий поиск с помощью функции getpwent(3C) с начала файла.

endpwent(3C) может быть вызвана, чтобы закрыть файл паролей после завершения обработки.

fgetpwent(3C) возвращает указатель на следующую структуру pasword в потоке f, формат которого соответствует формату /etc/password.

Замечание: эти библиотечные функции возвращают указатель на структуру, которая расположена в сегменте данных. Следовательно, значения в структуре должны быть скопированы перед последующими вызовами этих функций. Если достигнут конец файла или возникнет ошибка чтения, то функция вернет NULL.

В традиционных Unix-системах в поле pw\_passwd хранился хэш пароля. В SVR4 информация о

паролях не хранится больше в /etc/password. Информацию о паролях содержит файл /etc/shadow, который доступен для чтения только root. Это защищает от словарной атаки, позволяющей определить пароли пользователей путем подбора значений с совпадающей хэш-функцией.

## Получение имени группы

Функции getgrent(3C), getgrgid(3C) и getgrnam(3C) возвращают указатель на структуру, которая содержит разбитую на поля строку из файла /etc/group. Каждая строка представлена в формате структуры group, определенной следующим образом:

```
grp.h:
struct group {
    char          *gr_name;
    char          *gr_passwd;
    gid_t         gr_gid;
    char          **gr_mem;
};
```

При первом вызове getgrent(3C) возвращает указатель на первую структуру group в файле. В следующий раз getgrent(3C) вернет указатель на следующую структуру group. Последовательные вызовы могут использоваться для поиска во всем файле.

getgrgid(3C) ищет от начала файла, пока не найдет структуру с полем идентификатора группы равным gid. Возвращает указатель на найденную структуру.


getprnam(3C) ищет от начала файла, пока не найдет структуру с полем имени группы равным name. Возвращает указатель на найденную структуру.


Вызов setgrent(3C) предоставляет возможность вести последующий поиск с помощью функции getgrent(3C) с начала файла.

endgrent(3C) может быть вызвана, чтобы закрыть файл групп после завершения обработки.

fgetgrent(3C) возвращает указатель на следующую структуру group в потоке f, формат которого соответствует формату /etc/group.

Замечание: эти библиотечные функции возвращают указатель на структуру, которая расположена в сегменте данных. Следовательно, значения в структуре должны быть скопированы перед последующими вызовами этих функций. Если достигнут конец файла или возникнет ошибка чтения, то функция вернет NULL.

 Совокупность каталогов и других метаданных, т. е. системных структур данных, отслеживающих размещение файлов на диске и свободное дисковое пространство, называется **файловой системой** - *Иртегов глава 11*

 "**Файл**— это совокупность данных, доступ к которой осуществляется по ее имени". - *Иртегов глава 11*

Любой оконечный каталог (который не содержит других каталогов) в счетчике ссылок всегда хранит число 2, потому что на индексный узел ссылаются две каталожные записи: запись, указывающая на каталог testdir, и запись в этом же каталоге, указывающая на каталог «точка».

```
d.khaetskaya@fit-main:~$ df -v
Mount Dir Filesystem blocks used free %used
rpool/fit-main/ROOT/zbe
/ 49774376 10069064 5760165 64%
/dev /dev 0 0 0 0%
/proc proc 0 0 0 0%
/system/co ctfs 0 0 0 0%
/etc/mntta mnttab 0 0 0 0%
/system/ob objfs 0 0 0 0%
/etc/svc/v swap 1683189 62 1683127 1%
/usr/lib/libc/libc_hwcapi.so.1
/lib/libc. 15829229 10069064 5760165 64%
/dev/fd fd 0 0 0 0%
/tmp swap 2079329 396202 1683127 20%
/var/run swap 1683132 5 1683127 1%
/home 10.4.0.80:/home 231967608 57671488 162466568 27%
```

Символическая ссылка — это косвенная ссылка на файл в отличие от жесткой ссылки, которая является прямым указателем на индексный узел файла. Символические ссылки придуманы с целью обойти ограничения, присущие жестким ссылкам.

- Жесткие ссылки обычно требуют, чтобы ссылка и файл размещались в одной файловой системе.
- Только суперпользователь имеет право создавать жесткие ссылки на каталоги (если это поддерживается файловой системой).

Символические ссылки не имеют ограничений, связанных с файловой системой, и любой пользователь сможет создать символическую ссылку на каталог. Символические ссылки обычно используются для «перемещения» файлов или даже целой иерархии каталогов в другое местоположение в системе.

При использовании функций, которые обращаются к файлам по именам, всегда нужно знать, как они обрабатывают символические ссылки. Если функция следует по символической ссылке, она будет воздействовать на файл, на который указывает символическая ссылка. В противном случае операция будет производиться над самой символической ссылкой, а не над файлом, на который она указывает. В табл. 4.9 приводится перечень описываемых в этой главе функций, которые следуют по символическим ссылкам. В этом списке отсутствуют функции `mkdir`, `mkfifo`, `mknod` и `rmdir` — они возвращают признак ошибки, если им в качестве аргумента передается символическая ссылка. Кроме того, функции, которые принима-

## Notes

```
//
// Created by rey on 2/24/21.
//

#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <pwd.h>
#include <grp.h>
#include <sys/stat.h>
#include <time.h>
#include <unistd.h>
#include <libgen.h>

char* default_args[] = { ".", NULL };

void option_list(char* name, struct stat* fs)
{
    char mode[11];
    char lbuf[FILENAME_MAX] = { 0 };
    char fname[FILENAME_MAX];

    // file mode
    if(S_IFREG == (fs->st_mode & S_IFMT))
        mode[0] = '-';
    else if (S_IFDIR == (fs->st_mode & S_IFMT))
        mode[0] = 'd';
    else
        mode[0] = '?';

    // user
    mode[1] = (fs->st_mode & S_IRUSR) ? 'r' : '-';
    mode[2] = (fs->st_mode & S_IWUSR) ? 'w' : '-';
    mode[3] = (fs->st_mode & S_IXUSR) ? 'x' : '-';

    // group
    mode[4] = (fs->st_mode & S_IRGRP) ? 'r' : '-';
    mode[5] = (fs->st_mode & S_IWGRP) ? 'w' : '-';
    mode[6] = (fs->st_mode & S_IXGRP) ? 'x' : '-';

    // others
    mode[7] = (fs->st_mode & S_IROTH) ? 'r' : '-';
    mode[8] = (fs->st_mode & S_IWOTH) ? 'w' : '-';
    mode[9] = (fs->st_mode & S_IXOTH) ? 'x' : '-';

    mode[10] = 0;
    printf("%s", mode);
    printf("\t%ld", fs->st_nlink);

    struct passwd* uent = getpwuid(fs->st_uid);
    struct group* gent = getgrgid(fs->st_gid);

    if (uent == NULL)
        printf("\t%d", fs->st_uid);
    else
        printf("\t%s", uent->pw_name);

    if (gent == NULL)
        printf("\t%d", fs->st_gid);
    else
        printf("\t%s", gent->gr_name);

    // size, date & name
    strftime(lbuf, FILENAME_MAX, "%H:%M %e %b %Y", localtime(&(fs->st_ctime)));
    printf("\t%lu\t%s\t%s\n", fs->st_size, lbuf, basename(name));
```

```
}

int main(int argc, char** argv)
{
    int i = 0;
    char** dirs = (1 < argc) ? (argv + 1) : default_args;
    for (i = 0; dirs[i] != NULL; i++)
    {
        struct stat fs;
        if(lstat(dirs[i], &fs) == 0)
            option_list(dirs[i], &fs);
        else
            perror(dirs[i]);
    }
}

}
```

```
d.khaetskaya@fit-main: ~/lab.18
File Edit View Search Terminal Help
d.khaetskaya@fit-main:~/lab.18$ ls -ld testDir core
-rw----- 1 d.khaetskaya d.khaetskaya 2259032 Feb 17 07:52 core
drwxr-xr-x 2 d.khaetskaya d.khaetskaya 4096 Feb 23 10:29 testDir
d.khaetskaya@fit-main:~/lab.18$ ./lab18 testDir core
drwxr-xr-x 2 d.khaetskaya d.khaetskaya 4096 10:29 23 Feb 2021 testDir
-rw----- 1 d.khaetskaya d.khaetskaya 2259032 07:52 17 Feb 2021 core
d.khaetskaya@fit-main:~/lab.18$ |
```

```
NAME
    strftime, strftime_l cftime, ascftime - convert date and time to string

SYNOPSIS
    #include <time.h>

    size_t strftime(char *restrict s, size_t maxsize,
                    const char *restrict format,
                    const struct tm *restrict timeptr);
```