Лабораторная работа №3
Работа с файлами и каталогами. FHS

Введение

Из этой лабораторной работы вы узнаете, что такое стандарт FHS, почему «всё есть файл», и познакомитесь с необычными видами файлов в Linux. Вы узнаете, как устроена файловая система Linux изнутри и с помощью каких команд наиболее эффективно управлять объектами файловой системы.

Основные отличия в работе с файлами (Windows и Linux)

Всё есть файл

Первым существенным отличием является то, что в Linux файлы — это не то, чем они кажутся. Файлы используются не только для хранения информации на диске. С их помощью реализованы, например, каталоги и ссылки. А еще интерфейс доступа к файлам используется для взаимодействия с устройствами, каналами и сокетами.

Такой подход по обеспечению доступа ко всем возможностям операционной системы через унифицированный интерфейс файловой системы получил название «всё есть файл». Этот интерфейс, конечно, может быть не всегда оптимальным для решения поставленной задачи, зато он всегда одинаковый. Для примера рассмотрим каталог `/proc`. В эту точку монтируется виртуальная файловая система, которая предоставляет доступ к информации о системе и её процессах, поэтому с помощью команды `cat /proc/meminfo` можно увидеть параметры использования памяти.

Если в Linux придерживаются концепции «всё есть файл», то в Windows следуют концепции «всё есть объект», то есть у всех системных ресурсов — файлов, каталогов, аппаратных устройств — есть свойства и методы, как в объектно-ориентированном программировании.

Единое пространство имен

Вторым отличием от Windows является единое пространство имен файловой системы. Если в Windows каждому блочному устройству назначается собственная буква и абсолютный путь к локальному файлу имеет вид `C:\pagefile.sys`, то в *nix-подобных системах все диски встраиваются в единое пространство имен, которое начинается с косой черты `/` (корня).

Например, в операционной системе Astra Linux при подключении внешнего usb-накопителя его файловая система монтируется в точку `/run/user/<UID>/media/<ID устройства>` с помощью модуля FUSE (*от англ. filesystem in userspace — «файловая система в пользовательском пространстве»*). Более того, любые

каталоги, например, `/home` или `/var`, могут физически располагаться на других устройствах, в том числе сетевых, и монтироваться средствами операционной системы.

Подход к хранению файлов программ

Третьим отличием является подход к хранению файлов. В системе Windows служебные файлы операционной системы находятся в каталоге `C:\Windows`, а файлы приложений в каталоге `C:\Program Files`, причем для каждого приложения создается отдельная папка, в которой хранятся все его файлы, организованные любым способом на усмотрение разработчиков приложения.

В Linux для организации файлов придерживаются так называемого стандарта иерархии файловых систем (Filesystem Hierarchy Standard, FHS), который унифицирует размещение файлов по их назначению. Например, исполняемые файлы приложений должны находиться в каталогах `/bin` и `/sbin`` (*от англ. binary - бинарные*), конфигурационные файлы находятся в папке `/etc` (*от англ. «et cetera» - прочие*), а изменяемые файлы, такие как журналы, должны находиться в каталоге `/var` (*от англ. variable - изменчивый*).

Зависимость от регистра символов

Четвертое отличие состоит в том, что файловые системы Linux традиционно учитывают регистр символов, то есть в одном каталоге одновременно могут находиться файлы `test.txt`, `Test.txt` и `TEST.txt`, которые будут считаться совершенно разными файлами.

Если же примонтировать в Linux диск, отформатированный в NTFS или exFAT, то мы сможем создать только один из этих файлов и обращаться к нему по любому из предложенных вариантов именования, как в Windows. То есть эта особенность является особенностью файловой системы, а не операционной.

Подход к использованию ссылок

Рассмотрим пятое отличие, которое заключается в использовании ссылок.

Несмотря на то, что команда `mklink` для создания ссылок появилась в системе Windows еще со времен NT, большинство пользователей Windows ограничиваются использованием ярлыков, которые представляют собой обычные файлы с расширением `*.lnk` и содержат настройки для быстрого доступа к файлам и папкам. Полным аналогом lnk-файлов в Linux являются файлы `*.desktop`.

В системе Linux жесткие (hard link) и мягкие ссылки (symbolic link) получили широкое распространение, поэтому изменения в одном файле могут отражаться сразу в нескольких местах, что сначала может сильно сбивать с толку.

Например, с помощью команд `stat /run/cups/cups.sock` и `stat /var/run/cups/cups.sock` вы можете убедиться, что эти файлы указывают на айноду (*от англ. information node, inode — информационный узел*) с одним и тем же номером, а все потому, что папка `/var/run` является мягкой ссылкой на папку `/run`, что можно проверить командой `ls -l /var/run`. Чуть позднее мы узнаем, что сделано так было для обеспечения обратной совместимости.

Еще интереснее, когда на один и тот же файл указывают сразу несколько жестких ссылок. Например, с помощью команд `stat /usr/bin/gunzip` и `stat /usr/bin/uncompress` вы можете легко убедиться, что это действительно две разные жесткие ссылки, которые указывают на одну и ту же айноду, поэтому до тех пор, пока мы не удалим обе ссылки, этот файл будет физически оставаться на диске.

Прямая косая черта в качестве разделителя

В завершение напомним, что в Linux в качестве разделителя между именами объектов файловой системы используют символ прямой косой черты «/» (слэш), например, `/etc/hosts`. В Windows можно использовать слэш, но более традиционным считается все же использование обратной косой черты «\» (бэкслэш), например, `C:\Windows\System32\drivers\etc\hosts`.

Устройство файловой системы Linux

Корневая файловая система и стандарт FHS

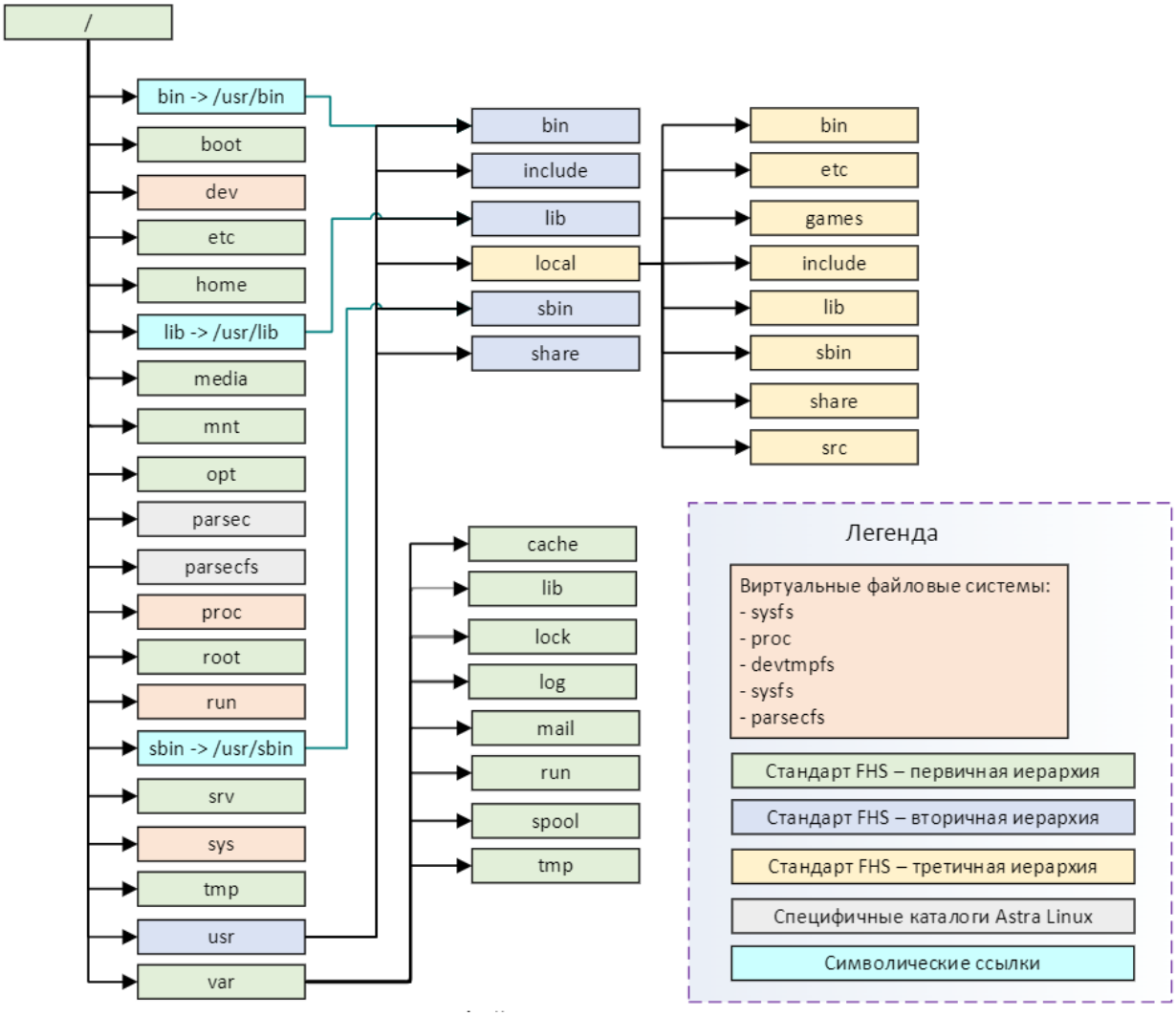
Системный диск подключается в Linux непосредственно к операционной системе, и точкой монтирования выступает та самая косая черта, которую называют еще корнем, а файловую систему системного диска, соответственно, называют корневой файловой системой. Именно к папкам корневой файловой системы в дальнейшем монтируются файловые системы других устройств.

Для того чтобы мы всегда знали, в каких каталогах искать нужные нам файлы, разработчикам потребовалось упорядочить структуру корневой системы, и они создали единый стандарт файловой системы (Filesystem Hierarchy Standard, FHS). Это очень странный стандарт, которого придерживаются практически все, но которому в полной мере не следует никто.

Причина такого положения вещей заключается в том, что некоторые идеи FHS уже устарели, но сообщество опасается делать резкие движения и продолжает двигаться по инерции.

Поэтому, изучая FHS, некоторые моменты лучше принять «как есть», а что-то станет понятным только в сугубо историческом контексте. А если вы думаете, что в структуре

каталогов Windows нет подобных атавизмов, то вы просто еще не смотрели на нее глазами разработчиков!



Корневая файловая система в Astra Linux

Рассмотрим основные каталоги системы:

- `/` – это корневой каталог системы, в эту точку монтируется системный диск.
- `/boot` – содержит конфигурационные файлы загрузчика GRUB, образы ядра системы, файлы временной файловой системы initrd, используемые ядром Linux при начальной загрузке.
- `/dev` – это каталог псевдофайловой системы devtmpfs, который содержит файлы устройств (*от англ. device - устройство*). Этот каталог является примером концепции «всё есть файл», поэтому в Windows подобного каталога не существует.

Все устройства можно разделить на две группы:

- **Символьные устройства** — обеспечивают последовательный поток ввода/вывода. Например, мышь является символьным устройством, и каждое нажатие кнопки приводит к отправке символа в файл `/dev/input/mouse0`, в чем можно убедиться, если сделать cat этого файла.

- **Блочные устройства** — предоставляют произвольный доступ к данным. Например, жесткий диск является блочным устройством, так как обеспечивает чтение/запись данных в произвольном порядке.

- `/media` – точка для автоматического монтирования сменных носителей, таких как FDD, CD-ROM, DVD-ROM. В эту категорию не попадают флешки – напомним, что внешние USB-носители монтируются в точку `/run/user/<UID>/media/` средствами модуля FUSE (*от англ. filesystem in userspace — «файловая система в пользовательском пространстве»*).

- `/mnt` – точка для монтирования внешних носителей вручную (*от англ. mount - монтировать*).

- `/proc` – в эту точку монтируется виртуальная файловая система procfs, с помощью которой можно получать информацию о состоянии ядра операционной системы и запущенных процессах (*от англ. process - процесс*). Например, команда `ps` берет свои данные именно через файлы procfs.

Отметим, что некоторыми файлами procfs можно манипулировать, чтобы изменять конфигурацию ядра. Например, если записать новое значение в файл `/proc/sys/kernel/hostname`, то можно изменить имя хоста «на лету».

- `/sys` - в эту точку монтируется виртуальная файловая система sysfs, которая является новой реализацией procfs и постепенно заменяет ее функции (*от англ. system - система*).

- `/run` – каталог содержит pid-файлы процессов, файлы временных сокетов и другие данные, описывающие систему с момента ее загрузки. Каталог размещается в TMPFS, поэтому очищается автоматически при перезагрузке системы.

Примечание

Ранее для тех же целей использовался каталог `/var/run`, который все еще остался в системе для обеспечения обратной совместимости, но является теперь символической ссылкой на каталог `/run`.

- `/bin` – содержит бинарные файлы стандартных утилит Linux, которые должны быть доступны всем пользователям системы (*от англ. binaries - двоичные файлы*). В системе Windows такие файлы имеют расширение `*.exe` и находятся в каталоге `C:\Windows\System32`. Каталог является мягкой ссылкой на каталог `/usr/bin`.

- `/sbin`– каталог содержит набор системных утилит, которые необходимы для загрузки системы и ее восстановления в случае сбоя, например, init, iptables, ifconfig (*от англ. system binaries - системные двоичные файлы*). Запускать эти утилиты имеет право только root. Каталог является мягкой ссылкой на каталог `/usr/sbin`.

- `/lib` – в этом каталоге находятся динамические библиотеки, которые нужны для работы приложений из каталогов `/bin/` и `/sbin` (*от англ. library - библиотека*). Каталог является мягкой ссылкой на каталог `/usr/lib`.

• `/usr/bin`, `/usr/sbin` и `/usr/lib` – каталоги, на которые ссылаются `/bin`, `/sbin` и `/lib`. Причину, по которой эти каталоги существуют в двух экземплярах, можно понять только в контексте развития вычислительной техники.

Давным-давно, когда жесткие диски были размером всего в пару мегабайт, разработчики операционной системы Unix довольно быстро подошли к этому пределу, и им пришлось перенести часть файлов из каталогов bin, sbin и lib на пользовательский диск (поэтому точка монтирования и называется `/usr`, от слова user). Критерий разнесения файлов по каталогам был прост — те файлы, которые не требовались системе до монтирования пользовательского диска, переносились в каталог `/usr`.

В дальнейшем размер дисков кратно увеличился, и это решение потеряло актуальность, но для обеспечения обратной совместимости каталоги оставили и сделали на них мягкие ссылки. Вот и получается, что этому атавизму уже более 35 лет. И в ближайшее время вряд ли что-то изменится.

• `/etc` – содержит конфигурационные файлы операционной системы и всех ее служб (от лат. «et cetera» – «и другие», «и тому подобное»). Данный каталог можно сравнить с **реестром Windows**, который был разработан для преодоления ограничений FAT16, связанных с отсутствием возможности разграничения прав доступа и медленным доступом к большому количеству маленьких INI-файлов с настройками приложений.

• `/var`– содержит файлы, которые подвергаются наиболее частому изменению (*от англ. variable - переменный*):

• `/var/cache` - каталог предназначен для хранения файлов кэша, генерация которых требует значительного количества вычислительных ресурсов и/или операций ввода-вывода. Удаление этих файлов не должно приводить к потере данных.

• `/var/lib` – содержит файлы, которые используются для передачи информации о состоянии приложения между его вызовами или для обмена информацией между несколькими экземплярами приложения, которые работают одновременно. В этом каталоге, например, находятся файлы службы SSSD после ее установки.

• `/var/lock` – файлы блокировки, указывающие на занятость устройств и других ресурсов. Эти файлы необходимы для возможности использования одного и того же ресурса несколькими приложениями.

• `/var/log` – каталог содержит файлы журналов.

• `/var/mail` – каталог содержит почтовые ящики локальных пользователей в устаревшем формате mbox, который представляет собой обычный текстовый файл. Для отправки сообщений локальным пользователям достаточно, чтобы в системе была установлена утилита `sendmail`, а, если в системе настроен модуль pam_mail, то при входе в терминал пользователи будут получать уведомления «you have new mail». Локальная почта в какой-то степени является аналогом Windows-утилиты msg.exe.

• `/var/run` – устаревший каталог, который оставлен для обратной совместимости и является теперь символической ссылкой на каталог `/run`.

• `/var/spool` – задачи, ожидающие обработки (например, очередь печати или очередь входящих/исходящих писем).

• `/var/tmp` – временные файлы, которые должны сохраняться между перезагрузками системы.

• `/opt` – каталог предназначен для размещения дополнительных приложений, которые отклоняются от стандарта FHS и хранят все свои файлы в одном месте (*от англ. optional - дополнительный*).

Каждому приложению выделяется отдельный каталог `/opt/<package>`, внутри которого могут быть каталоги bin, lib, man и др. для хранения исполняемых файлов, библиотек и справочных страниц соответственно. Установка таких приложений сводится к копированию одной папки и может выполняться даже без использования пакетных менеджеров. Таким образом, катало `/opt` является ближайшим аналогом для каталога `C:\Program files` системы Windows.

• `/srv` – каталог предназначен для хранения файлов сетевых служб (*от англ. service - служба*). Например, если хост является веб-сервером, то в этом каталоге может размещаться папка www с файлами обслуживаемых веб-сайтов.

• `/tmp` – каталог, в котором хранятся временные файлы. Linux, в отличие от Windows, следит за временными файлами и регулярно очищает этот каталог.

• `/home` – содержит домашние каталоги обычных пользователей, в которых хранятся личные файлы, настройки программ и прочее. В системе Windows для тех же целей предназначен каталог `C:\Users`.

• `/root` – домашний каталог суперпользователя. И тут встает вопрос: почему каталог `/root` лежит в корне, а не в папке `/home`? Когда жесткие диски были небольших размеров, одного жесткого диска хватало впритык для хранения файлов операционной системы, поэтому каталог `/home` почти всегда монтировали с отдельного жесткого диска. Но чтобы суперпользователь не терял доступ к своему домашнему каталогу даже в тех случаях, когда пользовательский диск не подключен, его домашний каталог был вынесен отдельно в корень системы.

• `/usr` – каталог предназначен для хранения прикладных программ и может быть смонтирован по сети. В нем размещаются общие файлы, которые могут совместно использоваться множеством компьютеров сразу, поэтому в нем не должно быть файлов, специфичных для конкретного хоста, а у пользователей не должно быть прав на запись в этот каталог. В Windows для получения того же эффекта мы можем подключить сетевую папку с архивом полезных приложений, не требующих установки.

Примечание

В корне Astra Linux есть еще два каталога, специфичных для этой операционной системы: parsec и parsecfs. Они связаны с подсистемой безопасности PARSEC, входящей в состав ядра ОС.

Организации хранения данных на диске

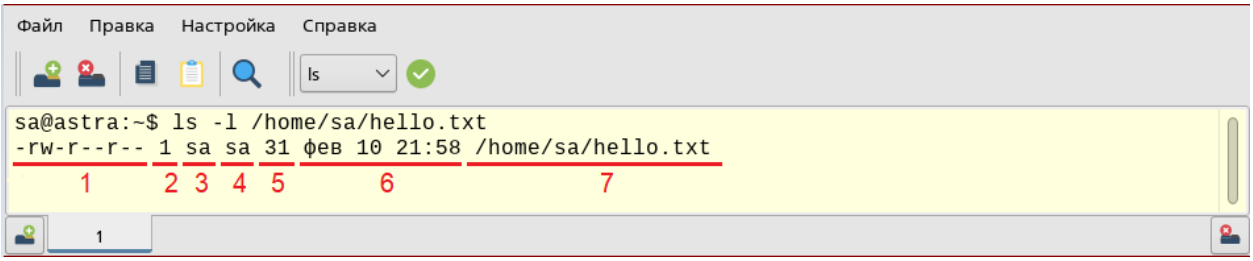
Файловые системы Windows и Linux используют немного разные подходы к организации хранения данных на диске, понимание которых позволит избежать проблем в будущем.

В Windows основной файловой системой является NTFS. Диск разбивается на кластеры, а для хранения метаинформации о файлах используется специальная база данных, которая называется главной файловой таблицей (Master File Table, MFT). Строкам этой таблицы или так называемым записям (MFT record) соответствуют файлы, а в столбцах таблицы хранятся значения конкретных атрибутов файлов.

По мере создания новых файлов размер MFT увеличивается, и файл этой базы может стать фрагментирован. Для того чтобы избежать деградации производительности системы в связи с фрагментированностью MFT, в момент форматирования диска система резервирует под MFT 12,5% дискового пространства и удерживает его до тех пор, пока не будет исчерпано все свободное пространство диска.

Наиболее популярной файловой системой Linux является EXT4. Диск разбивается на блоки, которые являются аналогом кластеров NTFS. А для хранения метаинформации о файлах используется структура данных, которая называется айнодой и является аналогом записей MFT.

Вывод команды `ls -l /home/sa/hello.txt` основан на данных из айноды:



1. Права на файл.
2. Количество жестких ссылок на файл.
3. Пользователь, являющийся владельцем файла.
4. Группа, являющаяся владельцем.
5. Размер файла в байтах.
6. Дата последнего изменения файла.
7. Полный путь к файлу.

А с помощью ключа `-i` утилита `ls` может показать номер индексного дескриптора:

```
sa@astra:~$ ls -i /home/sa/hello.txt
943173 /home/sa/hello.txt
```

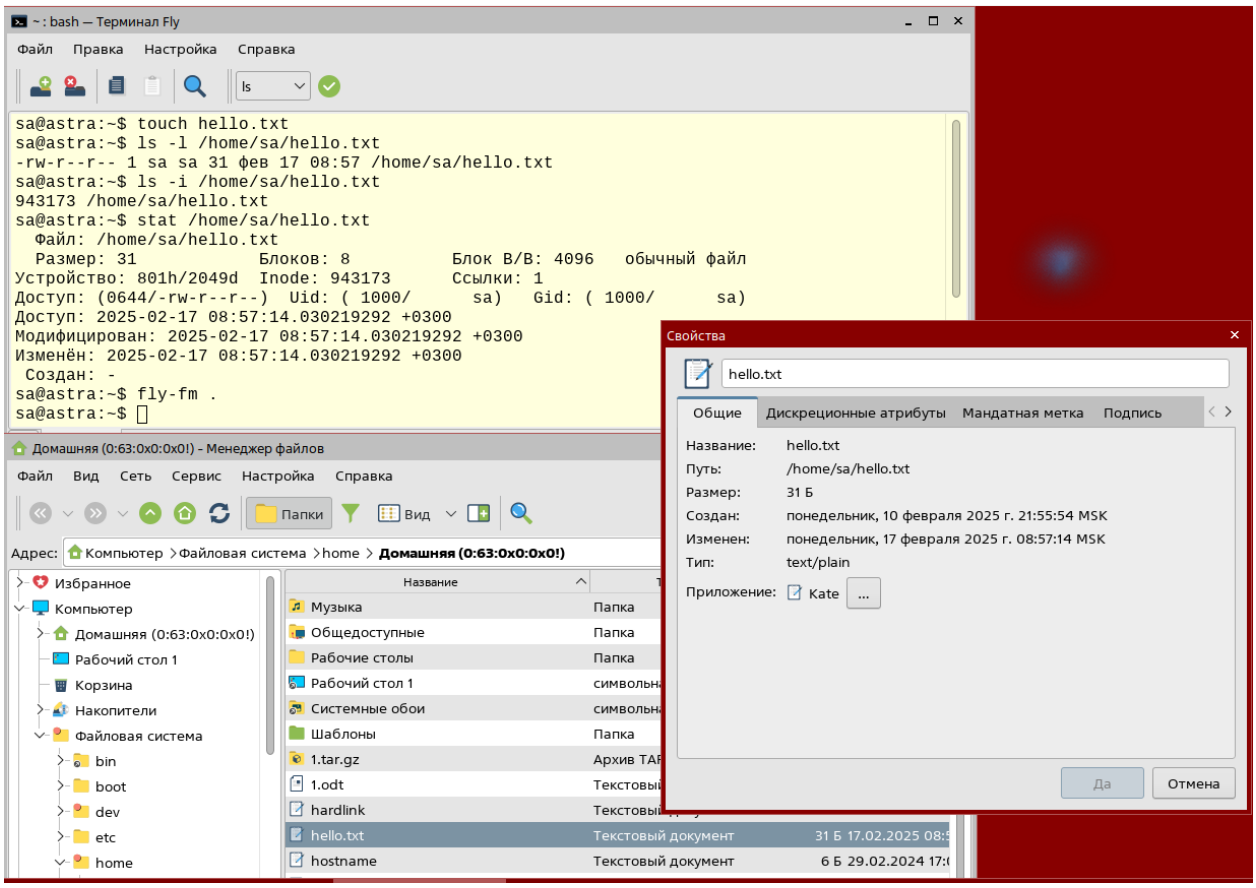
Используя команду `stat`, можно получить ряд дополнительных атрибутов файла, хранящихся в его айноде. Одним из преимуществ команды `stat` в сравнении с `ls` является то, что она показывает права доступа к файлу в числовом формате, например, в приведенном ниже примере на файл hello.txt установлены права 0644:

```
sa@astra:~$ stat /home/sa/hello.txt
  Файл: hello.txt
  Размер: 31            Блоков: 8          Блок В/В: 4096   обычный файл
Устройство: 801h/2049d  Inode: 943173      Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/sa)   Gid: ( 1000/sa)
Доступ: 2025-02-17 08:57:15.030319292 +0300
Модифицирован: 2025-02-17 08:57:15.030319292  +0300
Изменён: 2025-02-17 08:57:15.030319292 +0300
 Создан: -
```

Примечание

Утилита `stat` в текущих редакциях ALSE не показывает дату создания файла, но вы без труда найдете ее в графической утилите «Менеджер файлов».

Кстати открыть fly-fm в текущей директории можно командой: `fly-fm .`, где символ точки – это текущий каталог, который будет отправлен как аргумент по умолчанию. Также можно открыть текущий каталог в VS Code `code -n .`.



Однако, в отличие от таблицы MFT, которая может изменять свой размер динамически, количество индексных дескрипторов статично и определяется только в момент форматирования диска. С этой особенностью связана одна из распространенных проблем Linux, которая возникает, когда какое-нибудь приложение выжигает все доступные дескрипторы, создавая большое количество маленьких файлов, например, в каталоге для журналов. В этом случае свободного места остается много, но создать новый файл не получится. Давайте посмотрим, что можно с этим сделать.

Если подключить к виртуальной машине дополнительный диск объемом 32 Гб, затем создать на диске раздел с помощью утилиты `fdisk` и отформатировать этот раздел в ext4 утилитой `mkfs` с параметрами по умолчанию, то мы получим 8 млн. блоков по 4Кб и 2 млн. индексных дескрипторов.

```
$ lsblk
...
$ fdisk /dev/sdb
...
$ mkfs -t ext4 /dev/sdb1
...
Creating filesystem with 8388352 4k blocks and 2097152 inodes
...
```

По умолчанию размер блока составляет 4Кб, а количество дескрипторов определяется из расчета 1 дескриптор на каждые 16 384 байт дискового пространства, см. параметры blocksize и inode_ratio в конфигурационном файле `/etc/mke2fs.conf`. Вы можете изменить этот конфигурационный файл или явно задать размер блока и желаемое количество дескрипторов в момент форматирования диска с помощью ключей `-b` и `-N`:

```
$ mkfs -t ext4 -b 1024 -N 5000000 /dev/sdb1
...
Creating filesystem with 33553408 1k blocks and 5000000 inodes
...
```

Примечание

Файловая система XFS (X File System) позволяет динамически выделять блоки индексных дескрипторов, что снимает проблему их исчерпания. Если же вы используете более традиционную систему EXT4, то на этапе планирования вам следует определить оптимальные параметры диска и периодически следить за статистикой использования айнод.

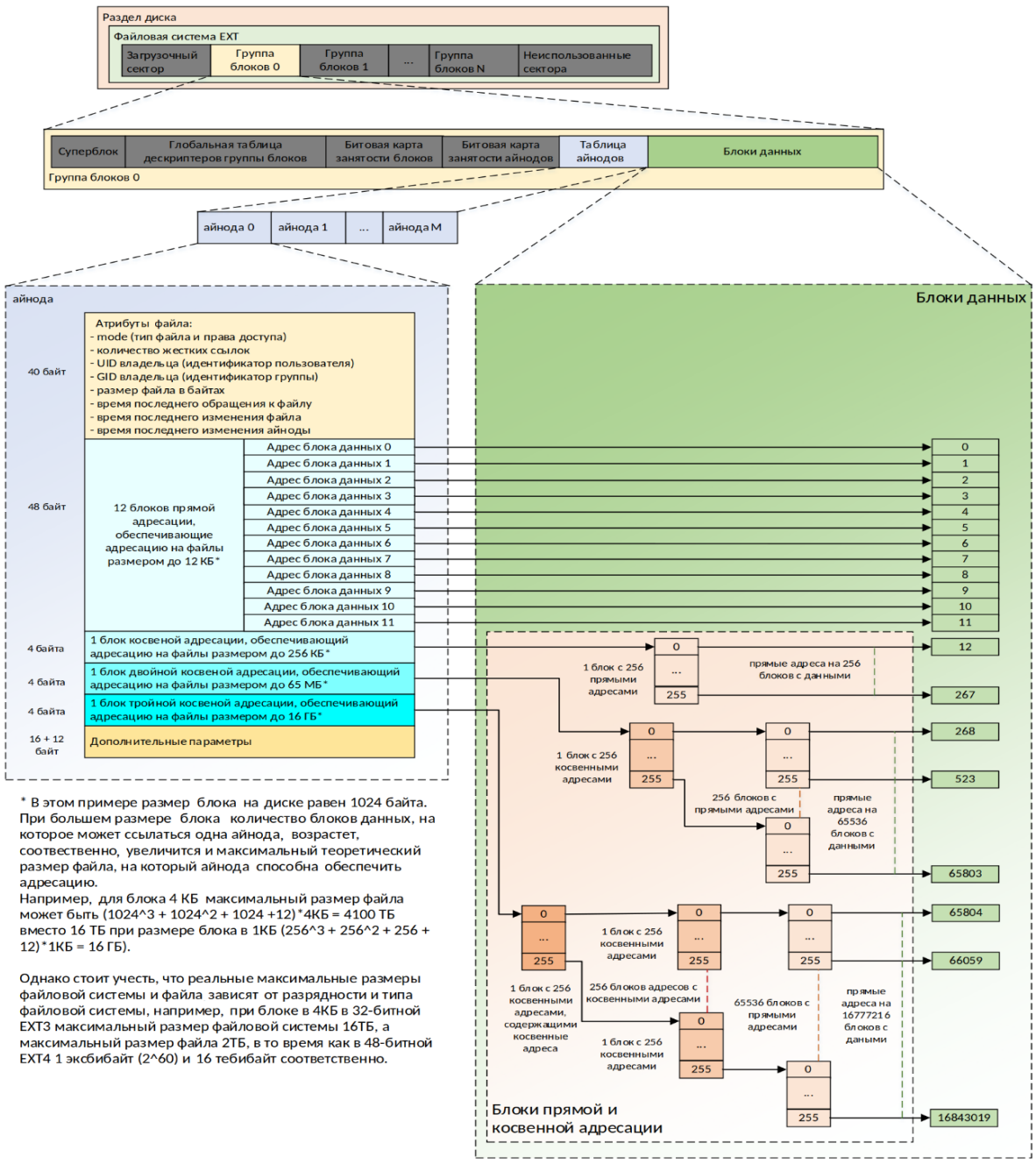
Узнать количество занятых и свободных айнод можно командой `df -i`.

```
sa@astra:~$ df -i
Файловая система Iнодов IИспользовано ICвободно IИспользовано% Смонтировано в
udev              240770           484    240286             1% /dev
tmpfs             253151           678    252473             1% /run
/dev/sda1         987360        226603    760757            23% /
tmpfs             253151             3    253148             1% /dev/shm
tmpfs             253151             4    253147             1% /run/lock
tmpfs             253151            19    253132             1% /sys/fs/cgroup
tmpfs             253151            10    253141             1% /run/user/107
tmpfs             253151            38    253113             1% /run/user/1000
```

Еще одним отличием является способ хранения имен файлов. Если в Windows имя файла является частью записи MFT, то в Linux хранение имен реализовано с помощью жестких ссылок.

Дело в том, что индексный дескриптор представляет собой структуру данных фиксированного размера (для ext4 это 256 байт), внутри которой может находиться только ограниченный набор атрибутов: тип файла, размер, дата создания/изменения, информация

о владельце и группе, базовые права доступа rwx, ну и, конечно же, ссылки на блоки данных.



* В этом примере размер блока на диске равен 1024 байта. При большем размере блока количество блоков данных, на которое может ссылаться одна айнода, возрастет, соотвественно, увеличится и максимальный теоретический размер файла, на который айнода способна обеспечить адресацию.
Например, для блока 4 КБ максимальный размер файла может быть (1024^3 + 1024^2 + 1024 +12)*4КБ = 4100 ТБ вместо 16 ТБ при размере блока в 1КБ (256^3 + 256^2 + 256 + 12)*1КБ = 16 ГБ).

Однако стоит учесть, что реальные максимальные размеры файловой системы и файла зависят от разрядности и типа файловой системы, например, при блоке в 4КБ в 32-битной EXT3 максимальный размер файловой системы 16ТБ, а максимальный размер файла 2ТБ, в то время как в 48-битной EXT4 1 эксбибайт (2^60) и 16 тебибайт соответственно.

Структура индексного дескриптора и схема адресации[3]

Поэтому дескриптор плохо подходит для хранения имен произвольной длины. Но для этого отлично подходят специальные файлы каталогов.

В файлы каталогов записываются так называемые жесткие ссылки, которые связывают имя файла с номером индексного дескриптора. У дескриптора таких ссылок может быть сразу несколько, но все они являются равнозначными, поэтому файл будет удален с диска только в том случае, если будут удалены все его жесткие ссылки.

Для того чтобы можно было легко принимать решение об удалении файла, внутри айноды есть атрибут, с помощью которого файловая система ведет подсчет количества жестких ссылок. При создании новой ссылки значение счетчика увеличивается на единицу, при удалении — уменьшается, а если счетчик станет равен нулю, то файл будет удален.

Давайте рассмотрим процесс, как файловая система работает с айнодами. Если мы захотим прочитать файл, то система возьмет номер дескриптора из жесткой ссылки, считает метаинформацию о файле, выполнит проверку прав доступа и вернет содержимое файла, прочитав его с диска по ссылкам на блоки данных. В тех случаях, когда размер файла укладывается в 12 блоков, системе будет достаточно так называемых прямых ссылок. Для более крупных файлов есть несколько ссылок косвенной адресации, которые позволяют использовать блоки данных для хранения ссылок. Такой подход позволяет адресовать файлы размером до 16 терабайт при стандартном размере блока в 4 Кб. Структура индексного дескриптора и схема адресации блоков данных представлена на рисунке выше.

Типы индексных дескрипторов

В Linux существует семь типов айнод, пять из которых относят к специальным файлам, см. таблицу ниже.

Тип файла	**Обозначение в ls**	**Назначение**
Обычные файлы	**-**rwxrwxrwx	Содержат различные данные, например, тексты, картинки, архивы, скрипты, библиотеки, исполняемые файлы и т. д.
Каталоги	**d**rwxrwxrwx	Содержат жесткие ссылки, связывающие имена файлов с индексными дескрипторами.
Специальные файлы:		
Символические ссылки	**l**rwxrwxrwx	Предоставляют альтернативное имя для доступа к файлам, которые расположены в любом месте файловой системы, в том числе на других носителях (аналог mklink в Windows).
Блочные устройства	**b**rwxrwxrwx	Предоставляют интерфейс для взаимодействия с аппаратным обеспечением компьютера.
Символьные устройства	**c**rwxrwxrwx	
Каналы	**p**rwxrwxrwx	Обеспечивают в Linux работу механизмов межпроцессного взаимодействия.
Сокеты	**s**rwxrwxrwx	

Обычные файлы

К обычным файлам можно отнести все файлы с данными, например, тексты, картинки, архивы, скрипты, исполняемые файлы, библиотеки и т. д. Кратким обозначением обычных файлов является символ дефиса «-».

Откроем терминал и введем команду `ls -l`, знакомую нам по лабораторной работе №2, где мы создали 2 файла – script и hello.txt:

```
sa@astra:~$ ls -l
итого 104
-rw-r--r-- 1 sa sa 16623 фев 29  2024 1.odt
-rw-r--r-- 1 sa sa 15087 фев 29  2024 1.tar.gz
lrwxrwxrwx 1 sa sa    17 фев 17 08:20 Desktop -> Desktops/Desktop1
drwx------ 6 sa sa  4096 мар 22  2023 Desktops
-rw-r--r-- 2 sa sa     6 фев 29  2024 hardlink
-rw-r--r-- 1 sa sa    31 фев 17 08:57 hello.txt
-rw-r--r-- 2 sa sa     6 фев 29  2024 hostname
-rw------- 1 sa sa     1 фев 10 18:26 nano.save
-rwxr-xr-x 1 sa sa   109 фев 11 08:26 script
-rw-r--r-- 1 sa sa    29 фев 10 17:21 script.save
-rwxr--r-- 1 sa sa   118 фев 10 18:17 script.save.1
lrwxrwxrwx 1 sa sa     8 фев 29  2024 simlink -> hostname
lrwxrwxrwx 1 sa sa    21 мар 22  2023 SystemWallpapers -> /usr/share/wallpapers
-rw------- 1 sa sa    23 фев 10 22:03 test.txt.save
drwxr-xr-x 2 sa sa  4096 фев 29  2024 tmp
drwxr-xr-x 2 sa sa  4096 мар 22  2023 Видео
drwxr-xr-x 2 sa sa  4096 мар 22  2023 Документы
drwxr-xr-x 2 sa sa  4096 мар 22  2023 Загрузки
drwxr-xr-x 3 sa sa  4096 мар 22  2023 Изображения
drwxr-xr-x 2 sa sa  4096 мар 22  2023 Музыка
drwxr-xr-x 2 sa sa  4096 мар 22  2023 Общедоступные
drwxr-xr-x 2 sa sa  4096 мар 22  2023 Шаблоны
```

В выводе команды мы можем увидеть несколько типов файлов, что можно определить по цвету строки и первому символу. Как вы можете заметить, символьные ссылки выделяются бирюзовым цветом, каталоги - синим, исполнимые файлы - зеленым, а обычные файлы отображаются цветом по умолчанию.

Каталоги

В Linux каталог - это особый файл, который хранит список жестких ссылок, связывающих имя дочернего файла с номером индексного дескриптора. С помощью обычных текстовых редакторов мы можем только посмотреть содержимое этих файлов, т.е. прямое редактирование файлов каталогов запрещается. Редактированием этих файлов занимается ядро операционной системы, получая команды создания, удаления, переименования файлов. Для просмотра содержимого каталога можно воспользоваться утилитами `ls`, `dir` или `tree`.

Примечание

Утилита `tree` выводит структуру каталогов в псевдографическом виде, который похож на вывод tree в cmd. Для её установки введите команду `sudo apt install tree`.

Символьные ссылки

Символьная или символическая ссылка – это специальный файл, внутри которого содержится ссылка на адрес другого файла по его имени, а не индексному дескриптору. Ссылки будут подробно рассмотрены чуть дальше по тексту.

Символьные и блочные устройства

Файлы устройств предназначены для обращения к аппаратному обеспечению компьютера (дискам, принтерам, терминалам и др.). Когда происходит обращение к файлу устройства, ядро операционной системы передает запрос драйверу этого устройства.

К символьным устройствам обращение происходит последовательно. Примером символьного устройства является мышь. Блочные устройства позволяют работать с данными в произвольном порядке. Примером блочного устройства является жесткий диск.

Сокеты и каналы

И каналы, и сокеты предназначены для организации взаимодействия между процессами. Ключевое отличие канала от сокета заключается в том, что канал является однонаправленным, а сокет позволяет передавать данные сразу в двух направлениях. Примером сокетов является файл диспетчера печати `/usr/lib/systemd/system/cups.socket`.

Жесткие и символические ссылки

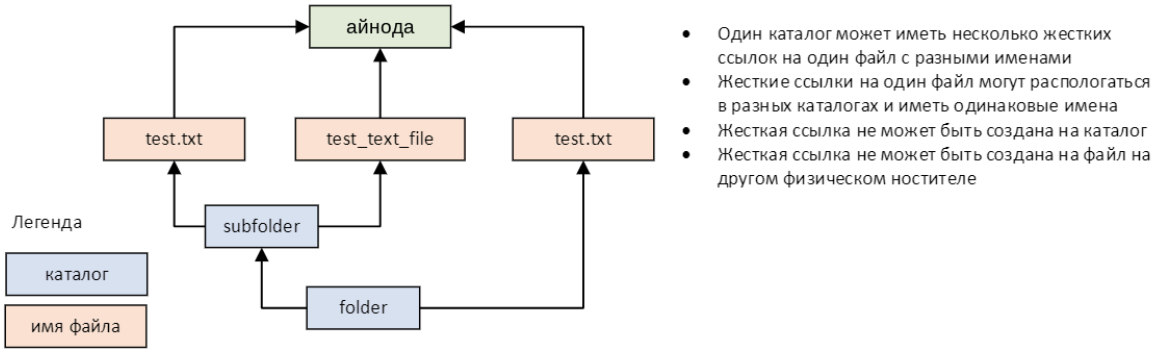
Как мы уже знаем, в Linux есть два типа ссылок: жесткие (*англ. hard link*) и символические (*англ. symbol link, символьная ссылка, «симлинк» или мягкая ссылка*). Кроме того, в графической оболочке Fly используются ярлыки `*.desktop`, которые абсолютно аналогичны ярлыкам `*.lnk` Windows.

Примечание

Стоит сказать, что файловая система NTFS поддерживает жесткие и символические ссылки, а также точки соединения (*англ. directory hard link, junction point*). Для их создания необходимо использовать `fsutil`, `mklink`, PowerShell. Однако в Windows такие ссылки используются крайне редко.

Жесткая ссылка

В Linux жесткая ссылка представляет собой запись в файле каталоге, которая связывает имя файла с номером его айноды.



Схема, отображающая связь между именами файлов и индексным дескриптором

При создании любого файла или каталога для него создается жесткая ссылка, и счетчик ссылок становится равен единице (0 + 1 = 1). Выполним команду `mkdir test` для создания папки test, перейдем в эту папку командой `cd test` и выполним команды:

- `touch test1` для создания файла test1
- `ls -il` для просмотра количества ссылок на объекты в текущем каталоге и номеров их индексных дескрипторов.

```
sa@astra:~$ mkdir test
sa@astra:~$ cd test
sa@astra:~/test$ touch test1
sa@astra:~/test$ ls -il
итого 0
1075458 -rw-r--r-- 1 sa sa 0 фев  17 10:28 tese1
```

Теперь создадим для этого файла жесткую ссылку командой `ln test1 hardlink_test1`, и просмотрим ссылки еще раз `ls -il`:

```
sa@astra:~/test$ ln test1 hardlink_test1
sa@astra:~/test$ ls -il
итого 0
1075458 -rw-r--r-- 2 sa sa 0 фев  17 10:28 hardlink_test1
1075458 -rw-r--r-- 2 sa sa 0 фев  17 10:28 test1
```

Как видим, счетчик увеличился на единицу. Создадим подкаталог командой `mkdir subtest` и создадим в нем еще одну жёсткую ссылку на этот файл командой: `ln hardlink_test1 subtest/hardlink2_test1`

При обращении к файлу по жесткой ссылке мы можем использовать любое имя, так как все ссылаются на одну и ту же айноду и нельзя сказать, какая из жестких ссылок была создана раньше, а какая позже. Убедиться в этом можно, выполнив команду `ls -il subtest`.

Теперь обновим время последнего изменения файла командой `touch hardlink_test1` и еще раз посмотрим свойства объектов в test и subtest:

```
sa@astra:~/test$ mkdir subtest
sa@astra:~/test$ ln hardlink_test1 subtest/hardlink2_test1
sa@astra:~/test$ ls -il subtest
итого 0
1075458 -rw-r--r-- 3 sa sa 0 фев  17 10:28 hardlink2_test1
sa@astra:~/test$ touch hardlink_test1
sa@astra:~/test$ ls -il
итого 4
1075458 -rw-r--r-- 3 sa sa    0 фев  17 10:36 hardlink_test1
1075460 drwxr-xr-x 2 sa sa 4096 фев  17 10:34 subtest
1075458 -rw-r--r-- 3 sa sa    0 фев  17 10:36 test1
sa@astra:~/test$ ls -il subtest
итого 0
1075458 -rw-r--r-- 3 sa sa 0 фев  17 10:36 hardlink2_test1
```

При удалении файла командой `rm` происходит удаление жесткой ссылки на этот файл и уменьшение счётчика жестких ссылок на единицу, но с блоками данных ничего не происходит. Если счётчик ссылок станет равен нулю, то система удалит такой файл и через какое-то время задействует освобожденные блоки данных для хранения других файлов.
Так как жесткая ссылка представляет собой просто ссылку на айноду файла, в Linux запрещено создавать жесткие ссылки на айноды за пределами устройства, на котором находится эта айнода, потому что идентификатор айноды уникален только в пределах текущей файловой системы.

Те самые «.» и «..» в относительных путях являются жесткими ссылками на текущий и родительский каталог, созданные самой операционной системой. Запрещено создавать дополнительные жесткие ссылки на каталоги, так как это может привести к зацикливанию, когда один каталог ссылается на второй, а тот в свою очередь на первый.

```
sa@astra:~/test$ ln subtest hardlink_subtest
ln: subtest: не допускается создавать жёсткие ссылки на каталоги
```

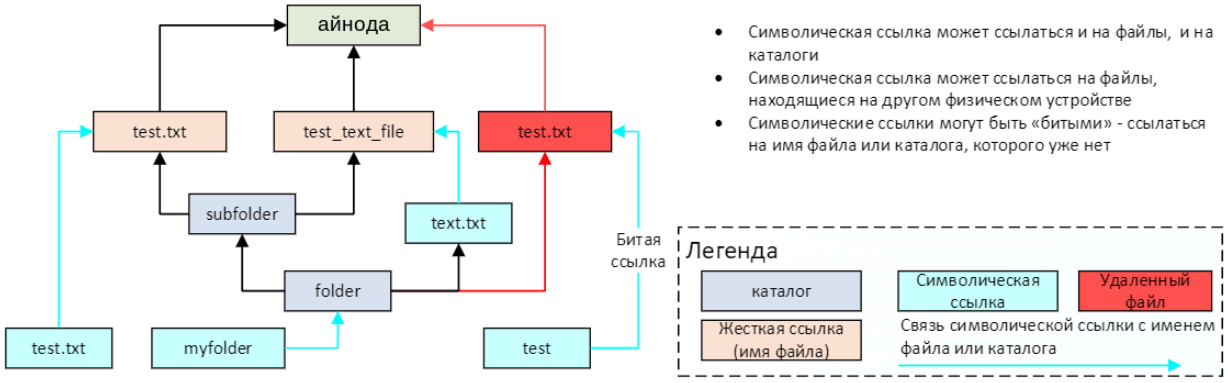
Примечание

У команды `ln` есть специальный ключ `-d`, который позволяет на ряде систем с правами суперпользователя создать жесткую ссылку на каталог, однако в Astra Linux эта операция не разрешена:

```
sa@astra:~/test$ sudo ln -d subtest hardlink_subtest
ln: не удалось создать жёсткую ссылку 'hardlink_subtest' => 'subtest': Операция не позволена
```

Символическая ссылка

Символическая ссылка, в отличие от жесткой, позволяет ссылаться на файлы и каталоги на любом устройстве, доступном в корневой файловой системе, потому что представляет собой файл, в котором хранится полный путь к файлу, на который ведет эта ссылка. Айноды символических ссылок не дублируют атрибуты тех файлов, на которые они указывают, поэтому на символические ссылки можно выставить свои собственные права доступа. На дополненной схеме связи между именами файлов и индексным дескриптором представлена связь между символическими ссылкам и именами файлов.



Расширенная схема, отображающая связь между символьными ссылками, именами файлов и индексным дескриптором

Символическая ссылка ссылается на путь к файлу или папке. Если объект, на который она ссылается, будет удален (перемещен), то ссылка будет вести на несуществующий объект, то есть станет разорванной (*англ. broken link*) или «битой».

Создадим несколько символических ссылок на файлы и папку с помощью команды `ln` с ключом `-s` (`--symbolic`):

- `ln -s ~/test/subtest ~/subtest` – создаем ссылку subtest в домашнем каталоге на каталог subtest в каталоге test.

```
sa@astra:~$ cd ~/test
sa@astra:~/test$ ln -s ~/test/subtest ~/subtest
sa@astra:~/test$ ls -l ~/subtest
lrwxrwxrwx 1 sa sa 21 фев  17 10:42 /home/sa/subtest ->/home/sa/test/subtest
sa@astra:~/test$ ln -s test1 test1_link
sa@astra:~/test$ ln -s hardlink_test1 hardlink_test1_link
sa@astra:~/test$ ls -l
итого 4
-rw-r--r-- 3 sa sa    0 фев  17 10:36 hardlink_test1
lrwxrwxrwx 1 sa sa   14 фев  17 10:45 hardlink_test1_link -> hardlink_test1
drwxr-xr-x 2 sa sa 4096 фев  17 10:43 subtest
-rw-r--r-- 3 sa sa    0 фев  17 10:36 test1
lrwxrwxrwx 1 sa sa    5 фев  17 10:45 test1_link -> test1
```

Попробуем получить доступ к файлам и каталогам через символьные ссылки. Выполним команды `ls ~/subtest`, `echo "test text" > hardlink_test1_link` и `cat test1_link`:

```
sa@astra:~/test$ ls ~/subtest
hardlink2_test1  subtest
sa@astra:~/test$ echo "test text" > hardlink_test1_link
sa@astra:~/test$ cat test1_link
test text
sa@astra:~/test$ cat test1
test text
```

Как видим, с ссылками можно работать точно так же, как и с обычными файлами.

Удалим жесткую ссылку hardlink_test1 командой `rm hardlink_test1` и проверим доступ до файла через жесткую и символическую ссылки командами `cat hardlink_test1_link`, `cat test1_link`, `cat test1`:

```
sa@astra:~/test$ rm hardlink_test1
sa@astra:~/test$ ls -l
итого 8
lrwxrwxrwx 1 sa sa   14 фев  17 10:45 hardlink_test1_link -> hardlink_test1
drwxr-xr-x 2 sa sa 4096 фев  17 10:43 subtest
-rw-r--r-- 2 sa sa   10 фев  17 10:48 test1
lrwxrwxrwx 1 sa sa    5 фев  17 10:45 test1_link -> test1
sa@astra:~/test$ cat hardlink_test1_link
cat: hardlink_test1_link: Нет такого файла или каталога
sa@astra:~/test$ cat test1_link
test text
sa@astra:~/test$ cat test1
test text
```

Как видим, после удаления жесткой ссылки вывод команды `ls` подсвечивает разорванную ссылку красным цветом, и при попытке прочитать файл мы получим сообщение, что такого файла не существует. Однако по символической ссылке на другое имя файла этот файл по-прежнему доступен, как и по другой жесткой ссылке.

В завершение отметим, что при удалении символической ссылки жесткая ссылка на файл и сами данные не удаляются.

Правила именования файлов и каталогов

Правила именования файлов и каталогов (помним, что каталог — это тоже файл) идентичны. Имя не может превышать 255 байт, а общая длина пути (включая все компоненты) не может превышать 4096 байт.

Минимальный безопасный набор символов для использования в именах файлов:

- буквы (как латиницы, так и кириллицы в любом регистре);
- цифры;
- символ «.» (точка);
- символ «_» (подчеркивание);
- символ «-» (тире).

Важно

Если название файла или каталога начинается с символа точки «.», то в Linux он будет считаться скрытым. Например, в рабочей папке пользователя есть такие файлы, как `.bashrc`, `.profile` и др., которые обычно не отображаются. Чтобы увидеть все файлы, включая скрытые, следует использовать команду `ls` с ключом `-a`.

Использование спецсимволов, например, пробела, символов «*», «?», «$», «’», «”» не рекомендуется, так как их наличие в именах файлов может вызвать некорректную работу некоторых программ, хотя допустимо использовать любые символы, кроме «/», например, команда `touch test\ file.txt` создаст в текущем каталоге файл, в имени которого будет содержаться пробел:

```
sa@astra:~$ touch test\ file.txt
sa@astra:~$ ls
...'test file.txt'...
```

Для решения проблем со спецсимволами в именах файлов при работе в командной строке есть несколько способов:

• **Экранирование спецсимволов.** Для экранирования специальных символов перед ними нужно поставить символ обратной косой черты «\» или взять все имя файла в одинарные кавычки, например, `cat test\\ file.txt` или `cat 'test file.txt'`:

```
sa@astra:~$ echo text > test\ file.txt
sa@astra:~$ cat 'test file.txt'
text
```

• **Автодополнение** по нажатию клавиши `Tab` дополнит путь и автоматически сделает экранирование символов:

```
sa@astra:~$ cat test\ file.txt
```

• **Работа с файлами с символом тире в начале имени.** Если в начале имени файла стоит символ тире `-`, то командный интерпретатор будет воспринимать его как параметр команды. Тут не поможет стандартное экранирование символа. Вместо него можно использовать или относительный путь к файлу, или специальный параметр `--`, который сообщает команде, что это не ключ, а параметр.

```
sa@astra:~$ echo text > '-test.txt'

sa@astra:~$ cat -test.txt
cat: неверный ключ — «.»

sa@astra:~$ cat \-test.txt
cat: неверный ключ — «.»

sa@astra:~$ cat '-test.txt'
cat: неверный ключ — «.»

sa@astra:~$ cat -- -test.txt
text

sa@astra:~$ cat ./-test.txt
text
```

Расширения файлов

В именах файлов Linux, также как в Windows, могут использоваться расширения или так называемые суффиксы, например, `*.txt` или `*.doc`. Но если в Windows расширения играют ключевую роль в определении типа файла, то в Linux они игнорируются, поэтому иногда говорят, что в Linux нет расширений.

Тип файла в Linux определяется по его сигнатуре, то есть нескольким байтам в начале файла, которые однозначно указывают на тип файла и являются, таким образом, его подписью. По этой причине переименование изображения `logo.png` в текстовый файл `logo.txt` никак не сможет обмануть утилиту `file`, которая продолжит считать его цветным PNG изображением:

```
sa@astra:~$ wget https://astralinux.ru/logo.png
--2025-02-17 10:59:28--  https://astralinux.ru/logo.png
Распознаётся astralinux.ru (astralinux.ru)… 178.170.196.116
Подключение к astralinux.ru (astralinux.ru)| 178.170.196.116|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа… 200 OK
Длина: 35129 (34K) [image/png]
Сохранение в: «logo.png»

logo.png       100%[===================================>]  34,31K  --.-KB/s    за 0,001s

2025-02-17 10:59:28 (35,9 MB/s) - «logo.png» сохранён [35129]

sa@astra:~$ file logo.png
logo.png: PNG image data, 2969 x 789, 8-bit/color RGBA, non-interlaced
sa@astra:~$ mv logo.png logo.txt
sa@astra:~$ file logo.txt
logo.txt: PNG image data, 2969 x 789, 8-bit/color RGBA, non-interlaced
```

Однако стоит заметить, что файловый менеджер Astra Linux работает с расширениями аналогично тому, как это делает проводник Windows, поэтому рекомендуется все же

придерживаться общепринятых соглашений, чтобы файлы можно было открывать из графики двойным кликом.

Абсолютный и относительный путь

Путь к объектам файловой системы представляет собой цепочку символов, которые показывают место расположения файла или каталога в файловой системе. Путь может быть абсолютным и относительным. Абсолютный или полный путь указывает на одно и то же место в файловой системе вне зависимости от текущего рабочего каталога или других обстоятельств, поэтому полный путь всегда начинается с корневого каталога. Относительный путь представляет собой путь по отношению к текущему рабочему каталогу пользователя или активных приложений.

Относительные пути могут использовать символы подстановки:

• Символ точки «.» в начале пути означает текущий каталог, например, введем команду `ls ./Desktop`, и она выведет содержимое каталога `/home/sa/Desktop`. В этом можно убедиться, введя команду с абсолютным путем `ls /home/sa/Desktop`:

```
sa@astra:~$ ls ./Desktop
firefox.desktop  fly-help.desktop  mycomp.desktop  mytrash.desktop vlc.desktop
sa@astra:~$ ls /home/sa/Desktop
firefox.desktop  fly-help.desktop  mycomp.desktop  mytrash.desktop vlc.desktop
```

• Два символа точки «..» указывают, что системе для поиска файла нужно подняться на один уровень выше. Например, если эти символы будут в самом начале пути, то они укажут на родительский каталог. Введем команду `ls ..`, и она выведет содержимое родительского каталога `/home/`:

```
sa@astra:~$ ls ..
sa
```

• Символ «~» подставляет полный путь к домашнему каталогу пользователя. Введем команду `cd /` , чтобы перейти в корень, а затем введем команду `cd ~`, чтобы перейти обратно в домашний каталог. Командой `pwd` можно проверить, в каком каталоге мы находимся, однако это не обязательно – в строке приглашения мы можем видеть текущий каталог (в данном случае символ «~» был изменен на «/», а затем обратно на «~»):

```
sa@astra:~$ cd /
sa@astra:/$ pwd
/
sa@astra:/$ cd ~
sa@astra:~$ pwd
/home/sa
```

Работа с файлами и каталогами

Список основных команд для работы с каталогами

К основным командам можно отнести следующие:

• `pwd` — вывести полный путь к текущему каталогу (*от англ. print working directory - распечатать рабочий каталог*).

• `cd [<directory>]` — перейти в указанный каталог (*от англ. change directory - сменить каталог*).

• `mkdir <directory>` — создать каталог.

• `rmdir <directory>` — удалить пустой каталог (*от англ. remove - удалить*).

• `rm -rf <directory>` — удалить каталог со всем его содержимым.

• `du [<directory>]` — вывести размер каталога (*от англ. disk usage - использование диска*).

Рассмотрим их более подробно с примерами.

Команда pwd

Выводит полный путь к текущему каталогу, например:

```
sa@astra:~$ pwd
/home/sa
```

Команда ls

Выводит список содержимого каталога. У команды довольно много ключей, полный их список можно найти в справке `man ls`.

Рассмотрим основные:

• `ls -a` (`ls --all`) при выводе не скрывает файлы, начинающиеся с точки «.».

• `ls -A` (`ls --almost-all`) выводит список всех файлов (в том числе и скрытых), кроме «.» и «..».

• `ls -l` – в дополнение к имени каждого файла выводится дополнительная информация: тип файла, права доступа к файлу, количество ссылок на файл, имя владельца, имя группы, размер файла в байтах и временной штамп (время последней модификации файла, если не задано другое).

• `ls -R` – выводит информацию рекурсивно из всех подкаталогов.

Примеры использования:

```
sa@astra:~/test$ ls -a
.  ..  hardlink_test1_link  subtest  test1  test1_link
sa@astra:~/test$ ls -A
hardlink_test1_link  subtest  test1  test1_link
sa@astra:~/test$ ls -l
итого 52
lrwxrwxrwx 1 sa sa   14 фев  17 10:45 hardlink_test1_link hardlink_test1
drwxr-xr-x 2 sa sa 4096 фев  17 10:43 subtest
-rw-r--r-- 2 sa sa   10 фев  17 10:48 test1
lrwxrwxrwx 1 sa sa    5 фев  17 10:45 test1_link -> test1
-rw-r--r-- 1 sa sa    5 фев 17 10:55 'test file.txt'
-rw-r--r-- 1 sa sa    5 фев 17 10:56  -test.txt
sa@astra:~/test$ ls -la
итого 60
drwxr-xr-x  3 sa sa  4096 фев 17 11:02  .
drwx------ 22 sa sa  4096 фев 17 10:42  ..
lrwxrwxrwx  1 sa sa    14 фев 17 10:45  hardlink_test1_link -> hardlink_test1
```

```
-rw-r--r--  1 sa sa 35129 дек 25  2023  logo.txt
drwxr-xr-x  2 sa sa  4096 фев 17 10:43  subtest
-rw-r--r--  2 sa sa    10 фев 17 10:48  test1
lrwxrwxrwx  1 sa sa     5 фев 17 10:45  test1_link -> test1
-rw-r--r--  1 sa sa     5 фев 17 10:55 'test file.txt'
-rw-r--r--  1 sa sa     5 фев 17 10:56  -test.txt
sa@astra:~/test$ ls -R
.:
hardlink_test1_link  subtest  test1  test1_link
./subtest:
hardlink2_test1
```

Команда cd

Переходит в указанный каталог. Удобно использовать относительные пути и символ «-», означающий предыдущий каталог, из которого ранее был совершён переход.

- `cd` – переход в домашний каталог пользователя
- `cd ..` – переход в родительский каталог

Примеры использования:

```
sa@astra:~/test$ cd
sa@astra:~$ cd test
sa@astra:~/test$ cd -
/home/sa
sa@astra:~$ cd /
sa@astra:/$ cd ~/test/subtest
sa@astra:~/test/subtest$ cd ..
sa@astra:~/test$
```

Команда mkdir

Команда mkdir создает каталоги

- `mkdir test2` – создаст каталог test2 в текущем каталоге.
- `mkdir -p test3/subtest/subsubtest` – создаст каталог subsubtest и все недостающие родительские каталоги в пути до него.

Примеры использования:

```
sa@astra:~$ ls -l | grep ^d | grep test
drwxr-xr-x 2 sa sa    4096 фев 17 10:43 test
sa@astra:~$ mkdir test2
sa@astra:~$ ls -l | grep ^d | grep test
drwxr-xr-x 3 sa sa    4096 фев 17  10:43 test
drwxr-xr-x 2 sa sa    4096 фев 17  11:23 test2
sa@astra:~$ mkdir ~/test3/subtest/subsubtest
mkdir: невозможно создать каталог «/home/sa/test3/subtest/subsubtest»: Нет такого файла или
каталога
sa@astra:~$ mkdir -p ~/test3/subtest/subsubtest
sa@astra:~$ tree test3
test3
└── subtest
    └── subsubtest
2 directories, 0 files
```

Команда rmdir

Удаляет каталоги, если они пустые.

• `rmdir test2` – удалит папку test2, если она пустая. В противном случае выдаст ошибку.

Примеры использования:

```
sa@astra:~$ rmdir test2
sa@astra:~$ rmdir test3
rmdir: не удалось удалить 'test3': Каталог не пуст
```

Команда rm -rf

Команда `rm -rf <путь>` удаляет папку вместе с её содержимым. Ключ `-r` указывает, что требуется рекурсивно удалить вложенные каталоги, ключ `-f` указывает, что не требуется запрашивать подтверждение от пользователя. Например, команда `rm -rf test3` удалит каталог *test3* и все его содержимое без дополнительных подтверждений.

Примеры использования:

```
sa@astra:~$ rm -rf test3
```

Внимание

Это одна из самых опасных команд в Linux. При неосторожном использовании особенно с правами суперпользователя можно безвозвратно удалить часть данных или всю корневую файловую систему, включая информацию на примонтированных носителях (`rm -rf /`).

Список основных команд для работы с файлами

К основным командам можно отнести следующие:

• `touch <file>` — создать файл или изменить временные метки у существующего файла.

• `realpath <file>` — узнать абсолютный путь к файлу.

• `file <file>` — определить тип файла.

• `stat <file>` — получить информацию о файле (размер файла, дата создания файла и т. д.) и проверить существование файла.

• `cp <file1> <file2>`— копировать файл «file1» в файл с именем «file2» (*от англ. copy - копировать*). Если файл с таким же названием уже существует, то он будет заменён.

• `mv <file1> <file2>` — переименовать файл «file1» в «file2» (*от англ. move - переместить*).

• `mv <file> <directory>`— переместить файл в каталог.

• `rm <file>` — удалить файл (*от англ. remove - удалить*).

• `find <file>` — найти файл.

• `sh <file>` — запустить файл со сценарием Bash.

• `./file` — запустить исполняемый файл.

Рассмотрим их более подробно с примерами.

Команда touch

Основная функция команды – обновить временные метки у файла. Однако, если требуется создать новый пустой файл, это можно сделать в том числе и этой командой.

- `touch` – обновляет временные метки существующего файла или создает новый файл.
- `touch -a` – изменяет только время последнего доступа к файлу (время, когда файл был прочитан в последний раз).
- `touch -m` – изменяет только время последнего изменения файла (время последнего изменения содержимого файла).
- `touch -d "STRING"` – может задать время, в том числе в свободной форме, например, «next Wednesday», подробнее `man touch`.
- `touch -r <referencefile> <newfile>` – копирует значения даты и времени у указанного файла.

Примеры использования:

Подготовка для демонстрации тестового каталога

```
sa@astra:~$ mkdir test4
sa@astra:~$ cd test4
sa@astra:~/test4$ ls
```

Создание файла

```
sa@astra:~/test4$ touch test.txt
sa@astra:~/test4$ ls -l
итого 0
-rw-r--r-- 1 sa sa 0 фев 17 11:30 test.txt
sa@astra:~/test4$ stat test.txt
 Файл: test.txt
 Размер: 0             Блоков: 0          Блок В/В: 4096   пустой обычный файл
Устройство: 801h/2049d  Inode: 442812      Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/      sa)   Gid: ( 1000/      sa)
Доступ: 2025-02-17 11:30:03.818347276 +0300
Модифицирован: 2025-02-17 11:30:03.818347276 +0300
Изменён: 2025-02-17 11:30:03.818347276 +0300
Создан: -
```

Обновление временных меток файла:

```
sa@astra:~/test4$ touch test.txt
sa@astra:~/test4$ stat test.txt
  Файл: test.txt
  Размер: 0             Блоков: 0          Блок В/В: 4096   пустой обычный файл
Устройство: 801h/2049d  Inode: 442812      Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/sa)   Gid: ( 1000/sa)
Доступ: 2025-02-17 11:30:03.818347276 +0300
Модифицирован: 2025-02-17 11:30:03.818347276 +0300
Изменён: 2025-02-17 11:30:03.818347276 +0300
Создан: -
```

Обновление временной метки последнего доступа:

```
sa@astra:~/test4$ touch -a test.txt
sa@astra:~/test4$ stat test.txt
 Файл: test.txt
Размер: 0             Блоков: 0          Блок В/В: 4096   пустой обычный файл
Устройство: 801h/2049d  Inode: 1075480     Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/      sa)   Gid: ( 1000/      sa)
Доступ: 2025-02-17 11:34:26.982348761 +0300
Модифицирован: 2025-02-17 11:32:44.298348181 +0300
Изменён: 2025-02-17 11:34:26.982348761 +0300
Создан: -
```

Обновление временной метки последней модификации:

```
sa@astra:~/test4$ touch -m test.txt
sa@astra:~/test4$ stat test.txt
  Файл: test.txt
 Размер: 0             Блоков: 0          Блок В/В: 4096   пустой обычный файл
Устройство: 801h/2049d  Inode: 1075480     Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/      sa)   Gid: ( 1000/      sa)
Доступ: 2025-02-17 11:34:26.982348761 +0300
Модифицирован: 2025-02-17 11:35:38.174349163 +0300
Изменён: 2025-02-17 11:35:38.174349163 +0300
Создан: -
```

Обновление временной метки последней модификации с выбором даты в свободной форме:

```
sa@astra:~/test4$ touch -d "next Wednesday" test.txt
sa@astra:~/test4$ stat test.txt
 Файл: test.txt
Размер: 0             Блоков: 0          Блок В/В: 4096   пустой обычный файл
Устройство: 801h/2049d  Inode: 1075480     Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/      sa)   Gid: ( 1000/      sa)
Доступ: 2025-02-17 11:36:26.914349438 +0300
Модифицирован: 2025-02-17 11:36:26.914349438 +0300
Изменён: 2025-02-17 11:36:26.914349438 +0300
Создан: -
```

Примечание

Можно было бы предположить, что по аналогии с командой `mkdir` должна существовать специальная команда `mkfile`. Действительно, например, в ОС Solaris есть такая команда, однако в Linux такой команды нет. Вместо неё принято использовать команду `touch`. Пустой файл можно создать также, например, с помощью оператора > (> file) или через текстовый редактор.

```
sa@astra:~/test$  > emptyfile
sa@astra:~/test$ ls -l
итого 0
-rw-r--r-- 1 sa sa 0 фев 17 11:36  -d
-rw-r--r-- 1 sa sa 0 фев 17 11:38  emptyfile
-rw-r--r-- 1 sa sa 0 фев 17 11:36 'next Wednesday'
-rw-r--r-- 1 sa sa 0 фев 17 11:36  test.txt
```

Команда realpath

Команда `realpath <file>` – выводит полный путь до файла.

Примеры использования:

Вывод полного пути до файла:

```
sa@astra:~/test4$ ls
test.txt
sa@astra:~/test4$ realpath test.txt
/home/sa/test4/test.txt
```

Команда file

Команда `file <file>` – определяет тип файла.

Примеры использования:

```
sa@astra:~$ file hello.txt
hello.txt: UTF-8 Unicode text

sa@astra:~$ file script
script: ASCII text

sa@astra:~$ file /usr/bin/ls
/usr/bin/ls: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically ...
```

Команда stat

Команда `stat` выводит атрибуты индексного дескриптора или статус файловой системы.

- `stat <file>` – выводит содержимое атрибутов индексного дескриптора.
- `stat -f <file>` – выводит статус файловой системы, на которой расположен файл.

Примеры использования:

Вывод информации из дескриптора файла:

```
sa@astra:~/test4$ stat test.txt
 Файл: test.txt
 Размер: 0             Блоков: 0          Блок В/В: 4096   пустой обычный файл
Устройство: 801h/2049d  Inode: 442812      Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/sa)   Gid: ( 1000/sa)
Доступ: 2023-06-14 00:00:00.000000000 +0300
Модифицирован: 2023-06-14 00:00:00.000000000 +0300
Изменён: 2023-06-13 18:50:34.144062150 +0300
Создан: -
```

Вывод информации о файловой системе, на которой расположен файл:

```
sa@astra:~/test4$ stat -f test.txt
 Файл: «test.txt»
   ID: 96cbae4e2fc631be Длина имени: 255     Тип: ext2/ext3
Размер блока: 4096       Базисный размер блока: 4096
Блоки: Всего: 4881364    Свободно: 3016414    Доступно: 2762693
Inodes: Всего: 1248480    Свободно: 101823 Команда cp
```

Команда `cp <источник> <назначение>` копирует файлы и каталоги, где *источник* или несколько <*источников*> копируются в указанное <*назначение*>:

- `cp file1 file2` – копирует файл *file1* в файл *file2*.
- `cp file1 destination/` – копирует файл *file1* в каталог *destination*.
- `cp -r source/ destination` (`-r`, `--recursive`) - копирует каталог source и все подкаталоги в каталог destination, если каталог destination существует. Если каталог не существует, то создаст его и скопирует **содержимое** каталога source.
- `cp -rT source destination/` (`-r`, `--recursive`; `-T`, `--no-target-directory`) - копирует содержимое каталога source и все подкаталоги в каталог destination. Если каталога destination не существует, создаст его.
- `cp -r source/* destination/` – копирует содержимое каталога source и все подкаталоги в каталог destination, если каталог destination существует. Если каталога не существует, выдаст ошибку.
- `cp -i file1 destination/` (`-i`, `--interactive`) - копирует файл *file1* в каталог с запросом на подтверждение перезаписи. Каталог должен существовать.
- `cp -n file1 destination` – ключ `-n` запрещает заменять существующие файлы.
- `cp -b file1 destination` – ключ `-b` создает резервную копию файла.
- `cp slink destination` – копирует не символическую ссылку slink, а оригинальный файл.
- `cp hlink destination` – копирует не жесткую ссылку hlink, а оригинальный файл.
- `cp -P slink destination` – копирует символическую ссылку именно как ссылку.
- `cp --preserve=links file1_hlink1 file1_hlink2_file destination` – если file1_hlink1 и file1_hlink2 являются жесткими ссылками на один и тот же индексный дескриптор, то утилита скопирует исходный файл и создаст для него две жесткие ссылки в каталоге destination.
- `cp -p source/\* destination` – копирует содержимое каталога source в каталог destination с сохранением информации о владельцах и прав доступа, хранящейся в индексном дескрипторе.
- `cp --preserve=timestamps file destination` – копирует файл в каталог destination с сохранением временных меток оригинального файла.
- `cp -a source/* destination` – копирует структуру файловой системы 1 в 1, включая временные метки, символические и жесткие ссылки.

Примеры использования:

Подготовка для демонстрации тестового каталога с набором файлов, ссылок и подкаталогом с файлами.

```
sa@astra:~$ mkdir -p ./test_cp/destination
sa@astra:~$ cd test_cp
sa@astra:~/test_cp$ mkdir ./source
```

```
sa@astra:~$ touch ./file{1..3}
sa@astra:~$ touch ./source/file{4..6}
sa@astra:~$ ln ./file1 ./hlink_file1
sa@astra:~$ ln -s $(realpath file2) ./slink_file2
sa@astra:~/test_cp$ tree
.
  ├── destination
  ├── file1
  ├── file2
  ├── file3
  ├── hlink_file1
  ├── slink_file2 -> /home/sa/file2
  └── source
      ├── file4
      ├── file5
      └── file6
2 directory, 8 files
```

Копирование файла в файл

sa@astra:~/test_cp$ cp file1 file1_1 sa@astra:~/test_cp$ ls destination file1 file1_1 file2 file3 hlink_file1 slink_file2

Копирование файла в каталог назначения. Каталог должен существовать.

```
sa@astra:~/test_cp$ cp file1 destination
sa@astra:~/test_cp$ ls ./destination/
file1
```

Копирование каталога в каталог. Каталог должен существовать

```
sa@astra:~/test_cp$ cp -r source destination/
sa@astra:~/test_cp$ ls destination/
file1  source

sa@astra:~/test_cp$ cp -r source destination2/
sa@astra:~/test_cp$ ls destination2
file4  file5  file6

sa@astra:~/test_cp$ cp -r source destination2/
sa@astra:~/test_cp$ ls destination2
file4  file5  file6   source
```

Копирование содержимого каталога в каталог. Каталог должен существовать.

Способ 1.

```
sa@astra:~/test_cp$ cp -rT source destination/
sa@astra:~/test_cp$ ls destination/
file1  file4  file5  file6  source
```

Удаление скопированных файлов

```
sa@astra:~/test_cp$ rm ./destination/file{4..6}
sa@astra:~/test_cp$ ls destination/
file1  source
```

Способ 2.

```
sa@astra:~/test_cp$ cp source/* destination/
sa@astra:~/test_cp$ ls destination/
file1  file4  file5  file6  source
```

Копирование файла в каталог с запросом на подтверждение перезаписи. Каталог должен существовать.

```
sa@astra:~/test_cp$ cp -i file1 destination/
cp: переписать 'destination/file1'? y
sa@astra:~/test_cp$ stat destination/file
stat: не удалось выполнить stat для 'destination/file': Нет такого файла или каталога
sa@astra:~/test_cp$ stat destination/file1
 Файл: destination/file1
 Размер: 0             Блоков: 0          Блок В/В: 4096   пустой обычный файл
Устройство: 801h/2049d  Inode: 453120      Ссылки: 1
Доступ: (0644/-rw-r--r--)  Uid: ( 1000/      sa)   Gid: ( 1000/      sa)
Доступ: 2025-02-17 11:52:53.154355004 +0300
Модифицирован: 2025-02-17 11:55:51.450356010 +0300
Изменён: 2025-02-17 11:55:51.450356010 +0300
Создан: -
```

Создание резервной копии файла

```
sa@astra:~/test_cp$ cp -b file1 destination
sa@astra:~/test_cp$ ls destination
file1  file1~  source
```

Копирование символических ссылок

```
sa@astra:~/test_cp$ cp slink_file2 destination/
sa@astra:~/test_cp$ ls -li slink_file2
1075495 lrwxrwxrwx 1 sa sa 22 фев 17 11:51 slink_file2 -> /home/sa/test_cp/file2
sa@astra:~/test_cp$ ls -li destination/slink_file2
1075506 -rw-r--r-- 1 sa sa 0 фев 17 12:04 destination/slink_file2
sa@astra:~/test_cp$ cp -P slink_file2 destination/
sa@astra:~/test_cp$ ls -li destination/slink_file2
1075506 lrwxrwxrwx 1 sa sa 22 фев 17 12:05 destination/slink_file2 -> /home/sa/test_cp/file
2
```

Копирование жестких ссылок

```
sa@astra:~/test_cp$ rm -rf destination/file1
sa@astra:~/test_cp$ ls -li *file1
1075489 -rw-r--r-- 2 sa sa 0 фев 17 11:50 file1
1075489 -rw-r--r-- 2 sa sa 0 фев 17 11:50 hlink_file1
sa@astra:~/test_cp$ cp file1 hlink_file1 destination/
sa@astra:~/test_cp$ ls -li destination/*file1
1075505 -rw-r--r-- 1 sa sa 0 фев 17 12:01 destination/file1
1075507 -rw-r--r-- 1 sa sa 0 фев 17 12:01 destination/hlink_file1
sa@astra:~/test_cp$ rm -rf destination/*file1
sa@astra:~/test_cp$ cp --preserve=links file1 hlink_file1 destination/
sa@astra:~/test_cp$ ls -li destination/*file1
1075505 -rw-r--r-- 2 sa sa 0 фев 17 12:02 destination/file1
1075505 -rw-r--r-- 2 sa sa 0 фев 17 12:02 destination/hlink_file1
```

Сохранение атрибутов (в этом примере временных меток) при копировании

```
sa@astra:~/test_cp$ cp file2 destination/file2_1
sa@astra:~/test_cp$ cp --preserve=timestamps file2 destination/file2_2
sa@astra:~/test_cp$ ls -li file2
1075490 -rw-r--r-- 1 sa sa 0 фев 17 11:50 file2
sa@astra:~/test_cp$ ls -li destination/file2*
1075507 -rw-r--r-- 1 sa sa 0 фев 17 12:11 destination/file2_1
1075508 -rw-r--r-- 1 sa sa 0 фев 17 11:50 destination/file2_2
```

Полное копирование с сохранением всех атрибутов

```
sa@astra:~/test_cp$ cp -a source destination/source_backup
sa@astra:~/test_cp$ ls -l source
итого 0
- rw-r--r-- 1 sa sa 0 фев 17 11:51 file4
-rw-r--r-- 1 sa sa 0 фев 17 11:51 file5
-rw-r--r-- 1 sa sa 0 фев 17 11:51 file6
sa@astra:~/test_cp$ ls -l destination/source_backup
итого 0
-rw-r--r-- 1 sa sa 0 фев 17 11:51 file4
-rw-r--r-- 1 sa sa 0 фев 17 11:51 file5
-rw-r--r-- 1 sa sa 0 фев 17 11:51 file6
```

Команда mv

Команда mv – перемещает (переименовывает) файлы и каталоги.

- `mv file destination` – перемещает файл file в каталог destination.
- `mv directory destination` – перемещает каталог directory в каталог destination.
- `mv file1 file2` – переименовывает файл file1 в file2.
- `mv directory1 directory2` – переименовывает каталог directory1 в directory2.

Примеры использования:

Подготовка для демонстрации тестового каталога с набором файлов и подкаталогами с файлами.

```
sa@astra:~$ cd
sa@astra:~$ mkdir test_mv
sa@astra:~$ cd test_mv
sa@astra:~/test_mv$ touch ./file{1..3}
sa@astra:~/test_mv$ mkdir destination
sa@astra:~/test_mv$ mkdir source
sa@astra:~/test_mv$ touch ./source/file{4..5}
sa@astra:~/test_mv$ tree
.
├── destination
├── file1
├── file2
├── file3
└── source
    ├── file4
    └── file5
2 directories, 5 files
```

Перемещение файла.

```
sa@astra:~/test_mv$ ls -il file1
154405 -rw-r--r-- 1 sa sa 0 фев 17 12:15 file1
sa@astra:~/test_mv$ mv file1 destination/
sa@astra:~/test_mv$ ls -il destination/file1
154405 -rw-r--r-- 1 sa sa 0 фев 17 12:15 destination/file1
```

Перемещение каталога

```
sa@astra:~/test_mv$ mv source destination/
sa@astra:~/test_mv$ tree
```

```
.
├── destination
│   ├── file1
│   └── source
│       ├── file4
│       └── file5
├── file2
└── file3
2 directories, 5 files
```

Переименование файла

```
sa@astra:~/test_mv$ ls -li file2
154406 -rw-r--r-- 1 sa sa 0 фев 17 12:15 file2
sa@astra:~/test_mv$ mv file2 file4
sa@astra:~/test_mv$ ls -li file4
154406 -rw-r--r-- 1 sa sa 0 фев 17 12:15 file4
```

Команда rm

Команда `rm` – удаляет файлы и каталоги.

- `rm file` – удаляет файл file.
- `rm -d directory` – удаляет каталог directory, если он пуст.
- `rm -i file` – перед удалением файла file запрашивает подтверждение у пользователя.
- `rm -I files` – запрашивает подтверждение при удалении более 3 файлов.
- `rm -f file` – удаляет файл file, игнорирует ключ `-i` и отсутствующие файлы и аргументы. Никогда не запрашивает подтверждения.
- `rm -Ir directory` – рекурсивно удаляет каталоги и их содержимое. Запрашивает подтверждение.
- `rm -rf directory` – **рекурсивно удаляет каталоги и их содержимое. Никогда не запрашивает подтверждения!!!**

Примеры использования:

Подготовка для демонстрации тестового каталога с набором файлов, ссылок и подкаталогом с файлами.

```
sa@astra:~$ cd
sa@astra:~$ mkdir -p test_rm/directory1/subdirectory
sa@astra:~$ cd test_rm/
sa@astra:~/test_rm$ mkdir directory2
sa@astra:~/test_rm$ touch file{1..3}
sa@astra:~/test_rm$ touch directory1/file{4..5}
sa@astra:~/test_rm$ touch directory1/subdirectory/file{6..7}
sa@astra:~/test_rm$ tree
.
├── directory1
│   ├── file4
│   ├── file5
│   └── subdirectory
│       ├── file6
│       └── file7
├── directory2
├── file1
├── file2
└── file3
3 directories, 7 files
```

Удаление простого файла

```
sa@astra:~/test_rm$ rm file1
```

Удаление простого файла с запросом на подтверждение

```
sa@astra:~/test_rm$ rm -i file2
rm: удалить пустой обычный файл 'file2'? y
sa@astra:~/test_rm$ rm -I file3
```

Удаление пустого каталога

```
sa@astra:~/test_rm$ rm directory2
rm: невозможно удалить 'directory2': Это каталог
sa@astra:~/test_rm$ rm -d directory2
```

Удаление непустого каталога

```
sa@astra:~/test_rm$ rm -d directory1/subdirectory
rm: невозможно удалить 'subdirectory': Каталог не пуст
sa@astra:~/test_rm$ rm -dr directory1/subdirectory
sa@astra:~/test_rm$ tree
.
├── directory1
│   ├── file4
│   └── file5
└── file3

1 directory, 3 files
```

Удаление непустого каталога с запросом на подтверждение

```
sa@astra:~/test_rm$ rm -drI directory1
rm: удалить 1 аргумент рекурсивно? n
sa@astra:~/test_rm$ rm -rI directory1
rm: удалить 1 аргумент рекурсивно? n
```

Удаление непустого каталога без запроса на подтверждение

```
sa@astra:~/test_rm$ rm -rf directory1
```

Команда find

Команда `find` - ищет файлы в иерархии каталогов. Это мощное средство с большим количеством критериев поиска и возможностью совершать ряд действий с найденными файлами. Для получения полной справки используйте команду `man find`.

Синтаксис команды:

```
find [каталог] [критерий поиска] [действие]
```

Где:

- **каталог** – это отправная точка поиска,

• **критерий поиска** можно указывать несколько критериев, применяя операторы условий:

Основные критерии поиска:

- `-name` – поиск по имени с учетом регистра
- `-iname` – поиск по имени без учета регистра,
- `-type` – поиск по типу файла (f, d, l, b, c, p, s),
- `-size` – поиск по размеру файла («+» - больше, «-» - меньше, без знака – точное совпадение, «c» - байт, «k» – Кбайт, «M» – Мбайт, «G» - Гбайт), например, «+1G», «-1M», «24K».
- `-empty` – поиск пустых фалов,
- `-cmin` – поиск по времени изменения в минутах, например, для поиска файлов, измененных за последний час «-60», для поиска всех файлов, кроме тех, что были изменены за последний час «+60»,
- `-atime` – поиск файлов по времени доступа к ним в днях, например, «+30» – поиск файлов, к которым не обращались за последние 30 дней,
- `-user` – поиск файлов по владельцу,
- `-perm` – поиск файлов по набору прав доступа, например, «755» или «777».

Основные операторы условий поиска:

- `-and` – для объединения условий логическим «и»,
- `-or` – для объединения условий логическим «или»,
- `-not` – для инвертирования условия.

• **действие** – что необходимо делать с каждым найденным файлом. Основные действия:

- `-delete` – удаляет объекты, соответствующие результатам поиска,
- `-ls` – выводит более подробные результаты поиска,
- `-print` – применяется по умолчанию, если не указать другое действие, показывает полный путь к найденным файлам,
- `-exec` – выполняет указанную команду в каждой строке результатов поиска:

```
-exec command {} \;
```

Где: `command` – команда, например, `cp`, `{}` – файл из результата поиска, а `\;` – завершение команды.

Использование команды:

• `find . -name file` – поиск файлов в текущем каталоге по имени или маске file.

• `find ./directory -name file` – поиск файлов в каталоге directory по имени или маске file.

- `find . -size +1M` – поиск файлов в текущем каталоге и подкаталогах с размером более 1МБ.
- `find . -size +1M -exec du {} \;` – поиск файлов в текущем каталоге и подкаталогах с размером более 1МБ и вывод размеров этих файлов.
- `find ./directory –empty` – поиск пустых файлов и каталогов в каталоге directory и подкаталогах.
- `find ./directory -empty -delete` – поиск пустых файлов и каталогов в каталоге directory и подкаталогах и их удаление.
- `find . -cmin -60 -size +10k` – поиск файлов и каталогов в текущем каталоге и подкаталогах, которые были изменены в течение прошлых 60 минут и у которых размер более 10КБ.
- `find . -name file\* -or -name \*txt` – поиск файлов и каталогов в текущем каталоге и подкаталогах по маске «file*» ИЛИ «*.txt».
- `find . -name file\* -type d` – поиск каталогов в текущем каталоге и подкаталогах по маске «file *».
- `find /etc -type f -exec grep -iH 'nameserver' {} \\;` – поиск файлов в каталоге /etc, которые содержат подстроку nameserver.

Примеры использования:

Поиск файлов в текущем каталоге и подкаталогах по маске «test*.txt»

```
sa@astra:~$ find . -name test\*.txt
./test\.txt
./test4/test3.txt
./test4/test2.txt
./test4/test.txt
./test.txt
```

Поиск файлов в каталоге test4 и подкаталогах по маске «test*.txt»

```
sa@astra:~$ find ./test4 -name test\*.txt
./test4/test.txt
```

Поиск файлов в текущем каталоге и подкаталогах с размером более 1МБ.

```
sa@astra:~$ find . -size +1M
/.local/share/flyhelp/ru/.flyhelp/fts
./.mozilla/firefox/ipkecuu8.default-release/places.sqlite
...
```

Поиск файлов в текущем каталоге и подкаталогах с размером более 1МБ и вывод размеров этих файлов

```
sa@astra:~$ find . -size +1M -exec du {} \;
3492    ./.local/share/flyhelp/ru/.flyhelp/fts
5120    ./.mozilla/firefox/ipkecuu8.default-release/places.sqlite
3060    ./.mozilla/firefox/ipkecuu8.default-release/security_state/data.safe.bin
...
```

Поиск пустых файлов и каталогов в указанном каталоге и его подкаталогах

```
sa@astra:~$ find ./test -empty
/test/test2
./test/test4/test.txt
```

Поиск пустых файлов и каталогов в указанном каталоге и его подкаталогах и их удаление

```
sa@astra:~$ find ./test -empty -delete
sa@astra:~$ find ./test -empty
sa@astra:~$
```

Поиск файлов и каталогов в текущем каталоге и подкаталогах, которые были изменены в течение последних 60 минут

```
sa@astra:~$ find . -cmin -60
/.fly/startmenu/recents
./.fly/startmenu/recents/fly-term.desktop
./.fly/menutime
./.local/share/baloo/index-lock
./.local/share/baloo/index
./test_rm
./test_rm/file3
./test_rm/directory1
./test_rm/directory1/file5
./test_rm/directory1/file4
./.config/rusbitech
...
```

Поиск файлов и каталогов в текущем каталоге и подкаталогах, которые были изменены в течение последних 60 минут и у которых размер более 10КБ

```
sa@astra:~$ find . -cmin -60 -size +10k
./.local/share/baloo/index
./.cache/icon-cache.kcache
./.xsession-errors
```

Поиск файлов и каталогов в текущем каталоге и подкаталогах по маске «test*» ИЛИ «*txt»

```
sa@astra:~$ find . -name test\* -or -name \*txt
./test
./test/test                                                                    file.txt
./test/logo.txt
./test/-test.txt
./test/test1
./test/test1_link
...
```

Поиск каталогов в текущем каталоге и подкаталогах по маске «test*»

```
sa@astra:~$ find . -name test\* -type d
./test_rm
./test
./test4
./test_cp
./test_mv
```

Практические задания

Задание 1.

1. Найдите в директории /usr/share все файлы размером более 500 Кб.
2. Полученный в предыдущем пункте список сохраните в файл /tmp/search.list.
3. Выясните размер файла /tmp/search.list в килобайтах и сохраните это значение в файл /tmp/search-size.txt.
4. Выясните количество строк в файле /tmp/search.list и запишите результат в файл /tmp/search-size.txt, сохранив предыдущие данные.

Задание 2.

1. Создайте одной командой директорию /tmp/block6/task/1/2/3 и переместите в нее так же одной командой файлы /tmp/search.list и /tmp/search-size.txt.
2. Создайте жесткую и символическую ссылки на файл /tmp/block6/task/1/2/3/search-size.txt.
3. Выведите содержимое каталога /tmp/block6/task/1/2/3/ с получением айноды всех файлов.
4. Удалите созданный каталог /tmp/block6/task/1/2/3/ вместе с содержимым.

Вопросы

1. Какой каталог содержит конфигурационные файлы?
2. Можно ли в ОС Linux создать два файла /tmp/test.txt и /tmp/TEST.txt?
3. Какая максимальная длина имени файла в ОС Linux?
4. Каким символом в выводе команды `ls -l` помечаются обычные файлы?
5. Можно ли найти неименованный канал в структуре каталогов?
6. Каким ключом команды `ls` можно отобразить айноду?
7. Можно ли создать жесткую ссылку на файл, расположенный на другой файловой системе?
8. Путь «tmp/1/2.txt» является абсолютным или относительным?
9. Что будет при вызове команды touch с существующим файлом?