### 1.chmod - изменяет права доступа к файлу

* **u** - владелец файла;
* **g** - группа файла;
* **o** - все остальные пользователи;
* r (чтение) = 4
* w (запись) = 2
* х (выполнить) = 1
* — (без разрешения) = 0

| **Число** | **разрешение** |
| --- | --- |
| 0 | — |
| 1 | выполнение |
| 2 | запись |
| 3 (то есть 2 + 1) | запись и выполнение |
| 4 | чтение |
| 5 (то есть 4 + 1) | чтение и выполнение |
| 6 (то есть 4 + 2) | запись и чтение |
| 7 (то есть 4 + 2 + 1) | запись, чтение и выполнение |

2. Анализ файлов - неотъемлемая часть работы с ними. Иногда возникает необходимость подсчитать количество строк или слов в тексте. С этой задачей эффективно справляется команда **wc** Linux.

|  |  |  |
| --- | --- | --- |
| Параметр | Длинный вариант | Значение |
| -c | --bytes | Отобразить размер объекта в байтах |
| -m | --count | Показать количесто символов в объекте |
| -l | --lines | Вывести количество строк в объекте |
| -w | --words | Отобразить количество слов в объекте |

3. Для навигации с помощью команды **cd** есть два формата указания пути к нужной папке: абсолютный и относительный.

**Абсолютный путь** — это полный путь от корневой папки сервера **«/»**. Например, вот такой:

/var/www/data/username/data/www/example.com/

Также к абсолютным относятся пути, в которых используется **~** — указание на домашнюю папку текущего пользователя. Например, при доступе к файлу **~/.bash\_profile** он всегда будет открываться из одного и того же места: вместо **~** будет использоваться значение переменной окружения $HOME, которое не меняется.

**Относительный путь** — это неполный путь. Он указывается относительно какого-то объекта: например,  текущей рабочей директории. То есть в начале отсутствует символ корневой папки **«/»**.

./docs/files/file.txt

4.Чтобы отобразить список содержимого папки, можно использовать команду **ls** (**l**i**s**t). Если дополнить её ключами **-al**, она покажет типы данных, разрешения и скрытые файлы.

* **-a** - отображать все файлы, включая скрытые, это те, перед именем которых стоит точка;
* **-A** - не отображать ссылку на текущую папку и корневую папку . и ..;
* **--author** - выводить создателя файла в режиме подробного списка;
* **-b** - выводить Escape последовательности вместо непечатаемых символов;
* **--block-size** - выводить размер каталога или файла в определенной единице измерения, например, мегабайтах, гигабайтах или килобайтах;
* **-B** - не выводить резервные копии, их имена начинаются с ~;
* **-c** - сортировать файлы по времени модификации или создания, сначала будут выведены новые файлы;
* **-C** - выводить колонками;
* **--color** - включить цветной режим вывода, автоматически активирована во многих дистрибутивах;
* **-d**- выводить только директории, без их содержимого, полезно при рекурсивном выводе;
* **-D** - использовать режим вывода, совместимый с Emacs;
* **-f** - не сортировать;
* **-F** - показывать тип объекта, к каждому объекту будет добавлен один из специализированных символов \*/=>@|;
* **--full-time** - показывать подробную информацию, плюс вся информация о времени в формате ISO;
* **-g** - показывать подробную информацию, но кроме владельца файла;
* **--group-directories-first** - сначала отображать директории, а уже потом файлы;
* **-G** - не выводить имена групп;
* **-h** - выводить размеры папок в удобном для чтения формате;
* **-H** - открывать символические ссылки при рекурсивном использовании;
* **--hide** - не отображать файлы, которые начинаются с указанного символа;
* **-i** - отображать номер индекса inode, в которой хранится этот файл;
* **-l** - выводить подробный список, в котором будет отображаться владелец, группа, дата создания, размер и другие параметры;
* **-L** - для символических ссылок отображать информацию о файле, на который они ссылаются;
* **-m** - разделять элементы списка запятой;
* **-n** - выводить UID и GID вместо имени и группы пользователя;
* **-N** - выводить имена как есть, не обрабатывать контролирующие последовательности;
* **-Q** - брать имена папок и файлов в кавычки;
* **-r** - обратный порядок сортировки;
* **-R** - рекурсивно отображать содержимое поддиректорий;
* **-s** - выводить размер файла в блоках;
* **-S** - сортировать по размеру, сначала большие;
* **-t**- сортировать по времени последней модификации;
* **-u** - сортировать по времени последнего доступа;
* **-U** - не сортировать;
* **-X** - сортировать по алфавиту;
* **-Z** - отображать информацию о расширениях SELinux;
* **-1** - отображать один файл на одну строку.
* ls, dir, vdir - выводит содержимое каталога
* Программа **ls** сначала выводит список всех *файлов* (не каталогов), перечисленных в командной строке, а затем выводит список всех файлов, находящихся в каталогах, перечисленных в командной строке. Если не указано ни одного файла, то по умолчанию аргументом назначается `.' (текущий каталог). Опция -d заставляет **ls** не считать аргументы-каталоги каталогами. Будут отображаться только файлы, которые не начинаются с `.' или все файлы, если задана опция -a.

5. Для создания новых директорий используется команда **mkdir** с указанием пути до создаваемой папки. Пример ниже создаст папку **/new** в существующей директории **/home/user**:

mkdir /home/user/new

Если нужно создать сразу несколько вложенных папок, можно использовать параметр **-p**. В таком случае все несуществующие директории, указанные в пути, будут созданы автоматически. Представим, что нам нужно создать вложенные папки **docs/photos/2020-08-17** в текущей директории — это будет выглядеть так:

mkdir -p docs/photos/2020-08-17

Если нужно создать несколько папок за раз, можно просто перечислить их через пробел:

mkdir docs pictures games

6.Удалить папку можно с помощью команды **rmdir** с указанием пути. Папка будет удалена только в случае, если она пуста (можно использовать параметр **--ignore-fail-on-non-empty**, чтобы удалить папку в любом случае). Для удаления нескольких вложенных папок можно аналогично использовать параметр **-p**. Пример ниже удалит папку **/photos** и все каталоги внутри неё:

rmdir -p docs/photos

7. [rm](https://baks.dev/search.php?tags=rm)утилита командной строки для удаления файлов и каталогов В отличие rmdir от rm команды можно удалять как пустые, так и непустые каталоги.

По умолчанию при использовании без какой-либо опции rm не удаляются каталоги. Чтобы удалить пустой каталог, используйте опцию -d ( --dir ) и удалите непустой каталог, а все его содержимое используйте опцию -r

Если каталог или файл в каталоге защищен от записи, вам будет предложено подтвердить удаление. Чтобы удалить каталог без запроса, используйте -f параметр: -i Параметр указывает rm на запрос на подтверждение удаления каждого подкаталога и файла. Если каталог содержит много файлов, это может немного раздражать, поэтому вы можете рассмотреть возможность использования -I опции, которая предложит вам только один раз, прежде чем продолжить удаление. С помощью rm и find вы можете удалять каталоги на основе различных критериев быстро и эффективно.

**rm** удаляет каждый заданный *файл*. По умолчанию каталоги не удаляются, но если заданы опции -r или -R, то будет удаляться все дерево каталогов ниже заданного каталога, включая и его самого (без ограничения на глубину этого дерева). Будет выдано сообщение об ошибке, если последний компонент *файла* -- это `.' или `..' (чтобы избежать неприятных сюрпризов при задании команды `rm -r .\*' или ей подобных).

Если задана опция -i или файл является недоступным на запись и при этом стандартный вывод -- это терминал и не задана опция **-f,** то **rm** выводит на стандартный вывод ошибок запрос на подтверждение удаления этого файла и читает ответ из стандартного ввода. Если ответ не утвердительный, то файл пропускается.

8. В терминале есть несколько способов создания новых файлов.

Самый простой, одновременно создающий и файл, и недостающие папки — команда **touch**. Достаточно передать ей полный путь к файлу. Для примера создадим в папке **docs** папку **copies**, а внутри неё — файл **copies-list.txt**:

touch docs/copies/copies-list.txt

Несуществующая папка **copies** создастся автоматически.

Если нужно создать сразу несколько файлов, можно просто перечислить их через пробел:

touch file1.txt file2.txt file3.txt

Ещё один способ создания новых файлов — вызвать какой-нибудь консольный текстовый редактор с указанием полного пути к файлу. Однако все папки придётся создать заранее. Тогда редактор автоматически создаст файл и откроет его для редактирования. Если не сохранить файл в редакторе, даже пустой, он не будет создан. Команда ниже добавит новый файл с помощью редактора **vi**:

vi new-file.txt

Для сохранения и выхода из файла в **vi** нужно нажать клавишу Esc, ввести сочетание :wq и нажать Enter. Для выхода без сохранения нужно также нажать Esc, ввести сочетание :q! и нажать Enter.

Ещё пара способов создания пустых файлов:

cat > new-file.txt

echo > new-file.txt

такой тип написания: touch testfile{1..6}

9. echo - показывает текстовую строку

Опций всего несколько, давайте рассмотрим их, чтобы вы могли лучше ориентироваться в работе утилиты:

* **-n** - не выводить перевод строки;
* **-e** - включить поддержку вывода Escape последовательностей;
* **-E** - отключить интерпретацию Escape последовательностей.

Это все опции, если включена опция -e, то вы можете использовать такие Escape последовательности для вставки специальных символов:

* **/c** - удалить перевод строки;
* **/t** - горизонтальная табуляция;
* **/v** - вертикальная табуляция;
* /b - удалить предыдущий символ;
* **/n** - перевод строки;
* **/r** - символ возврата каретки в начало строки.

10. cat - объединяет файлы и направляет их на стандартный вывод .Она часто применяется опытными пользователями во время работы с терминалом. С помощью этой команды можно очень просто посмотреть содержимое небольшого файла, склеить несколько файлов и многое другое.Объединяет ФАЙЛ(ы) или поток стандартного ввода и записывает (направляет) на стандартный вывод. Её также часто используют для вывода на экран содержимого небольших файлов. При попытке вывода больших файлов, если не использовать сочетания клавиш <Ctrl+s> (для приостановки) и <Ctrl+q> (для продолжения), их содержимое быстро промелькнёт перед глазами. В этом случае лучше использовать для просмотра больших файлов команды more и less, которые выполняют поэкранный вывод информации.

* **-b** - нумеровать только непустые строки;
* **-E** - показывать символ $ в конце каждой строки;
* **-n** - нумеровать все строки;
* **-s** - удалять пустые повторяющиеся строки;
* **-T** - отображать табуляции в виде ^I;
* **-h** - отобразить справку;
* **-v** - версия утилиты.

11.pwd - выводит путь текущего каталога  Выводит полный путь текущего каталога.

Опций всего две, это -P и -L. Они влияют на то, как будут обрабатываться символы возврата в предыдущую директорию и символические ссылки:

* **-P** - позволяет следовать по символическим ссылкам перед тем, как будут обработаны все переходы ".."; - ,без символических ссылок
* **-L** - переходит по символическим ссылкам только после того, как были обработаны "..";
* **-e** - если папку, в которую нужно перейти не удалось найти - выдает ошибку.

12ю More – Команда— одна из самых примитивных команд для работы с текстом. Её ближайшая родственница — команда less — обладает куда большим набором опций и дополнительных возможностей. -10 строк

Список опций команды:

* **-d** — вывод информации в конце страницы о клавишах, использующихся для продолжения работы, завершения её или получения инструкций;
* **-l** — игнорирование в тексте символа разрыва страницы;
* **-f** — подсчёт числа логических строк вместо экранных;
* **-p** — очистка экрана терминала для того, чтобы пользователю не пришлось пользоваться прокруткой перед выводом следующей порции текста;
* **-c** — устранение потребности в прокрутке (как и **-p)**— отображение текста, начиная с верха экрана, и стирание при этом предыдущего вывода построчно;
* **-s** — замена нескольких пустых строк, расположенных подряд, одной пустой строкой;
* **-u** — удаление подчёркивания;
* **-n** — отображение n-го количества строк;
* **+n** — отображение текста, начиная со строки с номером n;
* **+/строка** — поиск в файле указанной строки и начало вывода текста именно с неё;
* **--help** — вызов справки;
* **-v (--version)** — вывод на экран текущей версии утилиты.

13. cp - копирование файлов и каталогов . **cp** копирует файлы (или, если попросить, каталоги). Вы можете либо копировать один файл в другой, заданный файл, либо копировать сколько угодно файлов в заданный каталог.Есkи последний аргумент является существующим каталогом, то **cp** копирует каждый исходный *файл*в этот каталог (сохраняя имена). В противном случае, если задано только два файла, то **cp** копирует первый файл во второй. Если задано более двух аргументов, которые не являются опциями, а последний аргумент не является именем какого-либо каталога, то это приведет к ошибке.

* **--attributes-only** - не копировать содержимое файла, а только флаги доступа и владельца;
* **-f, --force** - перезаписывать существующие файлы;
* **-i, --interactive** - спрашивать, нужно ли перезаписывать существующие файлы;
* **-L** - копировать не символические ссылки, а то, на что они указывают;
* **-n** - не перезаписывать существующие файлы;
* **-P** - не следовать символическим ссылкам;
* **-r** - копировать папку Linux рекурсивно;
* **-s** - не выполнять копирование файлов в Linux, а создавать символические ссылки;
* **-u** - скопировать файл, только если он был изменён;
* **-x** - не выходить за пределы этой файловой системы;
* **-p** - сохранять владельца, временные метки и флаги доступа при копировании;
* **-t** - считать файл-приемник директорией и копировать файл-источник в эту директорию.

**14.** mv - перемещает (переименовывает) файлы   .

* **-b** / **—backup** / **—backup=МЕТОД** — создает копию файлов, которые были перемещены или перезаписаны.
* **-f** — при активации не будет спрашивать разрешение у владельца файла, если речь идет о перемещении или переименовании файла.
* **-i** — наоборот, будет спрашивать разрешение у владельца.
* **-n** — отключает перезапись уже существующих объектов.
* **—strip-trailing-slashes** — удаляет завершающий символ / у файла при его наличии.
* **-t ДИРЕКТОРИЯ** — перемещает все файлы в указанную директорию.
* **-u** — осуществляет перемещение только в том случае, если исходный файл новее объекта назначения.
* **-v** — отображает сведения о каждом элементе во время обработки команды.

### **Переместить файл**

Переместим файл myfile1.txt в директорию mydir/ (директория должна существовать). Для этого выполняем команду без каких-либо аргументов:

* mv myfile1.txt mydir/

Если при переименовании *исходного\_файла* в *файл\_назначения*, этот *файл\_назначения* существует и при этом задана опция -i или если произвести запись в *файл назначения* невозможно, а стандартным выводом является терминал и не задана опция **-f**, то **mv** спрашивает у пользователя разрешение на замену этого файла, которое выдается на стандартный вывод ошибок, и читает ответ из стандартного ввода. Если ответ не утвердительный, то файл пропускается.

**15.** ln - создает ссылки между файлами

**СИМВОЛИЧЕСКИЕ ССЫЛКИ**

Символические ссылки более всего похожи на обычные ярлыки. Они содержат адрес нужного файла в вашей файловой системе. Когда вы пытаетесь открыть такую ссылку, то открывается целевой файл или папка. Главное ее отличие от жестких ссылок в том, что при удалении целевого файла ссылка останется, но она будет указывать в никуда, поскольку файла на самом деле больше нет.

Вот основные особенности символических ссылок:

* Могут ссылаться на файлы и каталоги;
* После удаления, перемещения или переименования файла становятся недействительными;
* Права доступа и номер inode отличаются от исходного файла;
* При изменении прав доступа для исходного файла, права на ссылку останутся неизменными;
* Можно ссылаться на другие разделы диска;
* Содержат только имя файла, а не его содержимое.

Теперь давайте рассмотрим жесткие ссылки.

**ЖЕСТКИЕ ССЫЛКИ**

Этот тип ссылок реализован на более низком уровне файловой системы. Файл размещен только в определенном месте жесткого диска. Но на это место могут ссылаться несколько ссылок из файловой системы. Каждая из ссылок - это отдельный файл, но ведут они к одному участку жесткого диска. Файл можно перемещать между каталогами, и все ссылки останутся рабочими, поскольку для них неважно имя. Рассмотрим особенности:

* Работают только в пределах одной файловой системы;
* Нельзя ссылаться на каталоги;
* Имеют ту же информацию inode и набор разрешений что и у исходного файла;
* Разрешения на ссылку изменяться при изменении разрешений файла;
* Можно перемещать и переименовывать и даже удалять файл без вреда ссылке.

Рассмотрим опции утилиты:

* **-d** - разрешить создавать жесткие ссылки для директорий суперпользователю;
* **-f** - удалять существующие ссылки;
* **-i** - спрашивать нужно ли удалять существующие ссылки;
* **-P** - создать жесткую ссылку;
* **-r** - создать символическую ссылку с относительным путем к файлу;
* **-s** - создать символическую ссылку***.  ls -li***

**ln** делает ссылки между файлами. По умолчанию делаются жесткие ссылки; при указании опции **-s ,** делаются символьные (или мягкие) ссылки.

16. head - вывести первые строки

Печатает первые 10 строк каждого ФАЙЛА на стандартный вывод. Если задано несколько ФАЙЛОВ, сначала печатает заголовок с именем файла. Если ФАЙЛ не задан или задан как -, читает стандартный ввод.

Обязательные аргументы, используемые для длинных опций, обязательны также и для коротких опций.

**-c**, **--bytes**=*[-]N*

вывод первыx N байт каждого файла; с предваряющим символом '-' вывод последних N байт каждого файла

**-n**, **--lines**=*[-]N head – n 4*

вывод первыx N строк каждого файла вместо первых 10; с предваряющим символом '-' вывод последних N строк каждого файла

**-q**, **--quiet**, **--silent**

не печатать заголовки, содержащие имена файлов

**-v**, **--verbose**

всегда печатать заголовки, содержащие имена файлов

**--help**

вывести справку и закончить работу

**--version**

вывести информацию о версии и закончить работу

17. tail - вывести последние строки

Печатает последние 10 строк каждого из ФАЙЛОВ на стандартный вывод. Если задано несколько ФАЙЛОВ, сначала печатается заголовок с именем файла. Если ФАЙЛ не задан или задан как /-, читает стандартный ввод.

* **-c** - выводить указанное количество байт с конца файла;
* **-f** - обновлять информацию по мере появления новых строк в файле;
* **-n** - выводить указанное количество строк из конца файла;
* **--pid** - используется с опцией -f, позволяет завершить работу утилиты, когда завершится указанный процесс;
* **-q** - не выводить имена файлов;
* **--retry** - повторять попытки открыть файл, если он недоступен;
* **-v** - выводить подробную информацию о файле;

**18.** sort - сортирует строки текстовых файлов  Отправляет результат сортировки содержимого указанных *ФАЙЛОВ* (всех *ФАЙЛОВ*) на стандартный поток вывода. Все указываемые в командной строке файлы объединяются.

Основные параметры:

Обязательные аргументы для длинных форм параметров являются также обязательными и для коротких форм.

**-b** - не учитывать пробелы

* **-d** - использовать для сортировки только буквы и цифры
* **-i** - сортировать только по ASCII символах
* **-n** - сортировка строк linux по числовому значению
* **-r** - сортировать в обратном порядке
* **-с** - проверить был ли отсортирован файл
* **-o** - вывести результат в файл
* **-u** - игнорировать повторяющиеся строки
* **-m** - объединение ранее отсортированных файлов
* **-k** - указать поле по которому нужно сортировать строки, если не задано, сортировка выполняется по всей строке. Колонка ls -l | sort -k9
* **-f** - использовать в качестве разделителя полей ваш символ вместо пробела.

Вот несколько принципов, по которым команда sort linux сортирует строки:

* Строки с цифрами размещаются выше других строк
* Строки, начинающиеся с букв нижнего регистра размещаются выше
* Сортировка выполняется в соответствии алфавиту
* Строки сначала сортируются по алфавиту, а уже вторично по другим правилам.

19. **grep** - поиск образца в файле . Команда grep означает «глобальная печать регулярных выражений», и это одна из самых мощных и часто используемых команд в Linux. grep ищет в одном или нескольких входных файлах строки, соответствующие заданному шаблону, и записывает каждую соответствующую строку в стандартный вывод. Если файлы не указаны, grep считывает из стандартного ввода, которое обычно является выводом другой команды.

Иногда может понадобится найти файл, в котором содержится определённая строка или найти строку в файле, где есть нужное слово. В Linux всё это делается с помощью одной очень простой, но в то же время мощной утилиты **grep**. С её помощью можно искать не только строки в файлах, но и фильтровать вывод команд, и много чего ещё.

Grep ‘r$’ последний символ . Grep ‘^r’ первый

|  |  |
| --- | --- |
| **-b** | Предваряет каждую строку номером блока, в котором она была найдена. Это может пригодиться при поиске блоков по контексту (блоки нумеруются с 0). |
| **-c** | Выдает только количество строк, содержащих образец. |
| **-h** | Предотвращает выдачу имени файла, содержащего сопоставившуюся строку, перед собственно строкой. Используется при поиске по нескольким файлам. |
| **-i** | Игнорирует регистр символов при сравнениях. |
| **-l** | Выдает только имена файлов, содержащих сопоставившиеся строки, по одному в строке. Если образец найден в нескольких строках файла, имя файла не повторяется. |
| **-n** | Выдает перед каждой строкой ее номер в файле (строки нумеруются с 1). |
| **-s** | Подавляет выдачу сообщений о не существующих или недоступных для чтения файлах. |
| **-v** | Выдает все строки, за исключением содержащих образец. |
| **-w** | Ищет выражение как слово, как если бы оно было окружено метасимволами \< и \>. |

Fgrep это команда grep F, которая обрабатывает переданный шаблон как список фиксированных данных строкового типа. Эта команда полезна, когда в шаблоне используются зарезервированные для регулярных выражений символы, которые при обычно grep пришлось бы экранировать.

Три стандартных потока ввода/вывода

Для того, чтобы понять то, о чём мы будем тут говорить, важно знать, откуда берутся данные, которые можно перенаправлять, и куда они идут. В Linux существует три стандартных потока ввода/вывода данных.  
  
Первый — это стандартный поток ввода (standard input). В системе это — поток №0 (так как в компьютерах счёт обычно начинается с нуля). Номера потоков ещё называют дескрипторами. Этот поток представляет собой некую информацию, передаваемую в терминал, в частности — инструкции, переданные в оболочку для выполнения. Обычно данные в этот поток попадают в ходе ввода их пользователем с клавиатуры.  
  
Второй поток — это стандартный поток вывода (standard output), ему присвоен номер 1. Это поток данных, которые оболочка выводит после выполнения каких-то действий. Обычно эти данные попадают в то же окно терминала, где была введена команда, вызвавшая их появление.  
  
И, наконец, третий поток — это стандартный поток ошибок (standard error), он имеет дескриптор 2. Этот поток похож на стандартный поток вывода, так как обычно то, что в него попадает, оказывается на экране терминала. Однако, он, по своей сути, отличается от стандартного вывода, как результат, этими потоками, при желании, можно управлять раздельно. Это полезно, например, в следующей ситуации. Есть команда, которая обрабатывает большой объём данных, выполняя сложную и подверженную ошибкам операцию. Нужно, чтобы полезные данные, которые генерирует эта команда, не смешивались с сообщениями об ошибках. Реализуется это благодаря раздельному перенаправлению потоков вывода и ошибок.  
  
Теперь все становится на свои места. Для каждого сеанса взаимодействия пользователя и компьютера создается отдельный терминал, внутри терминала работает специальная программа - командный интерпретатор. При помощи консоли пользователь передает командному интерпретатору или запущенной с его помощью программе входящие данные и получает назад результат их работы. Осталось разобраться каким именно образом это происходит.

Для взаимодействия запускаемых в терминале программ и пользователя используются **стандартные потоки ввода-вывода**, имеющие зарезервированный номер (дескриптор) зарезервированный на уровне операционной системы. Всего существует три стандартных потока:

* **stdin** (*standard input, 0*) - стандартный ввод, по умолчанию нацелен на устройство ввода текущей консоли (клавиатура)
* **stdout** (*standard output, 1*) - стандартный вывод, по умолчанию нацелен на устройство вывода текущей консоли (экран)
* **stderr** (*standard error, 2*) - стандартный вывод ошибок, специальный поток для вывода сообщения об ошибках, также направлен на текущее устройство вывода (экран)

Как мы помним, в основе философии Linux лежит понятие - **все есть файл**. Стандартные потоки не исключение, с точки зрения любой программы - это специальные файлы, которые открываются либо на чтение (поток ввода), либо на запись (поток вывода). Это вызывает очевидный вопрос, а можно ли вместо консоли использовать файлы? Да, можно и здесь мы вплотную подошли к понятию **перенаправления потоков**.

Немного особняком стоит стандартный поток вывода ошибок, допустим мы хотим получить список файлов несуществующей директории с перенаправлением вывода в файл, но сообщение об ошибке мы все равно получим на экран.

[](https://interface31.ru/tech_it/assets_c/2021/10/linux-standard-stream-pipeline-009-13741.html)Почему так? Да потому что вывод ошибок производится в отдельный поток, который мы никуда не перенаправляли. Если мы хотим подавить вывод сообщений об ошибках на экран, то можно использовать конструкцию:

ls -l dir3 > result 2>/dev/null

В данном примере весь вывод стандартного потока ошибок будет перенаправлен в **пустое устройство** /dev/null.

Но можно пойти и другим путем, перенаправив поток ошибок в стандартный поток вывода:

ls -l dir3 > result 2>&1

Для того, чтобы перенаправить вывод одной программы на вход другой используйте знак |, на жаргоне "труба".

При такоэкрм перенаправлении вывода файл-приемник каждый раз будет создаваться заново, т.е. будет перезаписан. Это очень важный момент, сразу и навсегда запомните > - **всегда перезаписывает файл**!

Каждому процессу позволено иметь до девяти открытых дескрипторов файлов. Оболочка bash резервирует первые три дескриптора с идентификаторами 0, 1 и 2. Вот что они означают.

* 0, STDIN — стандартный поток ввода.
* 1, STDOUT — стандартный поток вывода.
* 2, STDERR — стандартный поток ошибок.

Эти три специальных дескриптора обрабатывают ввод и вывод данных в сценарии.  
Вам нужно как следует разобраться в стандартных потоках. Их можно сравнить с фундаментом, на котором строится взаимодействие скриптов с внешним миром. Рассмотрим подробности о них.

STDIN

STDIN — это стандартный поток ввода оболочки. Для терминала стандартный ввод — это клавиатура. Когда в сценариях используют символ перенаправления ввода — <, Linux заменяет дескриптор файла стандартного ввода на тот, который указан в команде. Система читает файл и обрабатывает данные так, будто они введены с клавиатуры.  
  
Многие команды bash принимают ввод из STDIN, если в командной строке не указан файл, из которого надо брать данные. Например, это справедливо для команды cat.  
  
Когда вы вводите команду cat в командной строке, не задавая параметров, она принимает ввод из STDIN. После того, как вы вводите очередную строку, cat просто выводит её на экран.

STDOUT

STDOUT — стандартный поток вывода оболочки. По умолчанию это — экран. Большинство bash-команд выводят данные в STDOUT, что приводит к их появлению в консоли. Данные можно перенаправить в файл, присоединяя их к его содержимому, для этого служит команда >>.  
  
Итак, у нас есть некий файл с данными, к которому мы можем добавить другие данные с помощью этой команды:

pwd >> myfile

**ТИПЫ ФАЙЛОВ В LINUX**

Файлы в операционной системе Linux можно поделить на три основных типа:

* Обычные файлы, для хранения информации
* Специальные файлы - для устройств и туннелей
* Директории

Дальше рассмотрим более подробно эти типы файлов linux.

**ОБЫЧНЫЕ ФАЙЛЫ**

Это файлы, с которыми мы привыкли работать каждый день, они могут содержать текст, исполняемые инструкции для программ, изображения или другую информацию. Это самый распространенный тип файлов, которые вы можете найти в системе Linux. Рассмотрим небольшой список относящихся сюда файлов:

* Текстовые файлы
* Исполняемые файлы
* Файлы изображений
* Файлы архивов
* Файлы библиотек программ
* И другие подобные типы

### СПЕЦИАЛЬНЫЕ ФАЙЛЫ

* Специальные файлы намного интереснее, они предназначены для обмена информации с ядром, работы с устройствами или общения между программами. Такие файлы могут тоже быть нескольких типов, в зависимости от назначения.
* **Блочные файлы -**это файлы устройств, которые обеспечивают буферизованный доступ к аппаратным компонентам. При записи данных на жесткий диск или на флешку нет смысла записывать данные сразу же после их поступления. Так мы будем только понапрасну расходовать ресурс устройства и энергию. Можно подождать пока наберется достаточное количество данных а потом записать их за один раз. Эти данные и собираются в буфере. С помощью таких файлов, файловая система и другие утилиты могут обращаться к драйверам аппаратных устройств. Такие файлы могут передать большой блок данных за небольшой один раз. Утилита ls обозначает блочные файлы буквой b
* **Символьные файлы**обеспечивают не буферизованный доступ к аппаратным компонентам и ядру. Поскольку у них нет буфера, они позволяют передавать только по одному символу за один раз. А в остальном, это такие же файлы устройств, как и блочные файлы.
* Вы также можете отфильтровать их с помощью ls. Символьные файлы обозначаются буквой c (character):

**Символические сылки** - это файлы, которые указывают на другие файлы в системе по их имени. Они могут указывать как на обычные файлы, таки на каталоги или другие типы файлов в linux. По сути, это те же ярлыки Windows. В Linux еще есть жесткие ссылки, но они не имеют отношения к типу файлов, потому что реализованы на уровне файловой системы и считаются обычными файлами. Поскольку они указывают на одно и то же место на диске, это два разных файла, с одинаковым содержимым.

Но вернемся к символическим ссылкам. Утилита ls обозначает их буквой l (link)

**Файлы сокетов** - это файлы, обеспечивающие прямую связь между процессами, они могут передавать информацию между процессами, запущенными в разных средах или даже разных машинах. Это значит, что с помощью сокетов программы могут обмениваться данными даже по сети. По сути, сокет работает так же как туннели, но только в обе стороны.

Файлы сокетов обозначаются буквой s: ls -l /dev/ | grep "^s"

**КАТАЛОГИ**

Это специальные файлы, которые позволяют объединять другие и каталоги в группы для более простой навигации и поиска. Естественно, они могут содержать как обычные, так и специальные файлы, одним словом любые типы файлов ос linux. В системе Linux, файлы организуются в папки начиная от корня (/)

Обозначаются каталоги буквой d (directory):