

ASTRA LINUX

GUIDE

ОПЕРАЦИИ С ФАЙЛАМИ И АРХИВАМИ

ASTRA LINUX GUIDE

TOUCH

Команда touch - это простой, но полезный инструмент в Unix-подобных системах, который позволяет пользователям создавать файлы и изменять временные метки непосредственно из терминала. Это одна из самых фундаментальных терминальных команд в Linux, и сисадмины часто используют ее для различных целей. Перед вами краткое руководство, содержащее 15 практических примеров ее использования.

Практическое использование команды touch для пользователей Linux

Приведенные ниже примеры помогут вам получить полное представление о команде. Однако, вы должны попробовать использовать эти команды сами, если хотите освоить этот инструмент в целом. Итак, запустите ваш терминал Linux и начните выполнять команды, приведенные ниже, по мере того, как вы о них будете читать.

1. Создание пустого файла

При самом простом использовании touch создает простой файл, который ничего не содержит. Он очень полезен в ряде ситуаций, например, когда вы экспериментируете с файловыми операциями или какими-либо другими командами терминала. Для этого просто передайте имя файла после команды touch.

```
$ touch test
```

Так в текущей рабочей директории создается простой пустой файл, который называется test. Вы можете проверить это с помощью команды ls. Вы также можете использовать команду touch для создания пустого файла в любой другой директории, просто передав правильный путь перед именем файла.

```
$ touch /путь/к/файлу
```

2. Создание нескольких пустых файлов

С помощью данной команды можно также создавать несколько файлов одновременно из одного терминала. Все, что вам нужно сделать, это передать все имена файлов, которые вы хотите создать один за другим. Взгляните на приведенную ниже команду, чтобы увидеть, как это работает.

```
$ rm test  
$ touch test1 test2 test3 test4 test5
```

Вышеуказанная команда создаст все пять файлов одновременно. Проверить это можно с помощью команды ls -l в терминале. Вы также можете использовать фигурные скобки для создания нескольких файлов одновременно с помощью touch, как показано в примере, приведенном ниже.

```
$ rm test1 test2 test3 test4 test5      # сначала удалите файлы  
$ touch test{1,2,3,4,5}
```

Вышеуказанная команда сделает ту же работу, но гораздо более аккуратным способом.

ASTRA LINUX GUIDE

3. Создание и заполнение файла

В приведенном ниже примере показано, как создать простой файл с помощью сенсорного ввода и заполнить его некоторым содержимым. Для этой цели мы используем команду seq в Linux. Но вы можете прибегнуть и к другим методам, если хотите.

```
$ rm test1 test2 test3 test4 test5      # сначала удалите файлы  
$ seq 100 > test
```

Теперь тестовый файл не пустой, а содержит числа от 1 до 10. Вы можете проверить это с помощью команды Linux cat.

4. Повторение метки времени последнего доступа к файлу

Команда touch позволяет пользователям повторять метки времени времени последнего доступа разных файлов. Для этого необходимо использовать ключ -r. Мы создадим новый файл под названием new, и вместо использования текущего времени мы возьмем значения времени из тестового файла, созданного ранее.

```
$ touch -r test new  
$ ls -l
```

Вы можете проверить это с помощью команды ls. Вы должны увидеть, что файл new имеет временную метку, более раннюю, чем текущее время, и такую же, как и файл test. Для этого ключа также существует длинная форма, называемая -reference.

5. Изменение метки времени последнего доступа к файлу

Администраторы Linux часто используют команду для изменения метки времени последнего доступа к файлам, созданным ранее. Следующая команда показывает нам, как это сделать. Для этого нам нужно использовать ключ -a. Сначала проверьте метку времени файла test, используя нижеприведенную команду.

```
$ ls -l test --time=atime
```

Теперь воспользуйтесь ключом -a, чтобы изменить эту метку времени с помощью команды touch. Обратите внимание, что ключ -time=atime из ls показывает нам время доступа к файлу.

```
$ touch -a test
```

Теперь, если вы еще раз проверите временную метку файла test, вы заметите, как она изменилась после выполнения вышеуказанной команды. Это один из наиболее распространенных видов использования в Linux.

6. Изменение метки времени последнего изменения файла

Вы также можете изменить время последнего изменения файла в Linux с помощью этой команды. Для этого необходимо использовать ключ -m. Внимательно посмотрите на пример ниже, чтобы увидеть, как это работает.

```
$ ls -l test  
$ touch -m test
```

ASTRA LINUX GUIDE

Во-первых, мы просмотрели последнее время последнего изменения файла test с использованием команды ls. Затем мы использовали ключ -t вместе с командой touch, чтобы изменить это значение на текущее время. Проверить это можно, выполнив еще раз команду ls.

7. Изменение даты и времени на текущее время

Вы можете использовать ключи -a и -t вместе для изменения как метки времени последнего доступа к файлу, так и времени последнего изменения файла в Linux. Для демонстрации этого необходимо использовать старый файл. Вы можете просто запустить приведенную ниже команду для одного такого файла, чтобы посмотреть, работает ли это так, как ожидалось или нет.

```
$ stat /путь/к/старому/файлу
```

Вы можете просмотреть всю информацию о времени, относящуюся к старому файлу, с помощью команды stat. Она покажет время последнего доступа, время последнего изменения и время последнего изменения атрибутов вашего файла. Теперь используйте следующую команду для изменения даты и времени на текущие.

```
$ touch -am /путь/к/старому/файлу
```

Проверьте изменение даты и времени еще раз с помощью команды stat.

```
$ stat /путь/к/старому/файлу
```

8. Пропустить создание файла

Ключ -c или --no-create позволяет пропустить создание файла. Хотя вы можете считать это непродуктивным, но многие пользователи используют этот ключ для проверки существования файла.

```
$ touch -c new-file  
$ touch --no-create new-file
```

При выполнении вышеописанных команд в Вашей рабочей директории не произойдет никаких изменений. Вы можете запустить команду ls еще раз, чтобы проверить это.

9. Изменение даты файла

Мы также можем использовать команду touch, чтобы указать дату файла вручную. Для этого необходимо использовать ключ -d или -date. Приведенная ниже команда показывает, как изменить дату файла на более раннюю с помощью touch.

```
$ ls -l test  
$ touch -d '15 Mar' test
```

Теперь, если вы снова запустите команду ls, вы увидите, что она показывает дату файла - 15 Марта. Но вы же создали этот файл сегодня, верно? Если Вы посмотрите внимательно, то увидите, что теперь ls показывает только дату, а не время. Вы даже можете использовать эту команду, чтобы установить дату в будущем.

ASTRA LINUX GUIDE

10. Изменение даты и времени на пользовательские значения

Ключ `-t` позволяет нам изменять как дату, так и время файла на персонализированные значения. Таким образом, мы можем изменить несколько параметров, включая год, месяц, дату, час, минуты и секунды. Внимательно посмотрите на следующий пример, чтобы понять, как это работает.

```
$ ls -l test          # покажет время и дату файла  
$ touch -t 2025 02 01 04 22 test
```

Теперь снова запустите команду `ls`, чтобы проверить параметры даты и времени. Аргумент к ключу `-t` должен быть в формате `YYMMDDhhmm`. Так как мы установили значения, которые будут в будущем, команда `ls` не сможет показать параметр времени.

11. Изменение даты с помощью слова

Также можно изменять дату файла в формате слов, например, "yesterday" с помощью команды `touch`. Для этого просто передайте аргумент в виде строки к ключу `-date`. Команда ниже показывает, как изменить дату файла на предыдущий день.

```
$ touch --date="yesterday" test
```

Используйте следующую команду для изменения даты на следующий день.

```
$ touch --date="tomorrow" test
```

Проверьте даты с помощью команды `ls` и убедитесь, что они работают, как ожидалось.

12. Копирование метки времени последнего доступа к файлу для символьных ссылок
Ключ `-h` или `-no-dereference` позволяет пользователям копировать время последнего доступа к файлу по символьским ссылкам. Однако, этот ключ будет работать только на системах, которые позволяют изменять время символьских ссылок.

```
$ touch -h link
```

Эта команда не работает во многих дистрибутивах Linux из-за различных проблем с безопасностью.

13. Просмотр справочной информации

Страница справки содержит сводную информацию обо всех возможных вариантах команды. Это очень полезно, так как пользователи могут просто взглянуть на эту страницу и вспомнить, что должен делать тот или иной вариант команды.

```
$ touch --help
```

Данная команда покажет страницу справки для linux-команды `touch` в вашем терминале. Это избавит Вас от необходимости запоминать использование каждого ключа.

14. Просмотр мануала по команде `touch`

Страница `man` или руководство содержит подробную информацию обо всех доступных вариантах команд терминала Linux. Вы можете просмотреть эту страницу в любое время, выполнив следующую простую команду.

```
$ man touch
```

ASTRA LINUX GUIDE

на покажет вам подробную документацию о том, как работать с командой touch в Linux системах. Обращайтесь к этой документации всякий раз, когда у вас возникнет путаница в отношении использования ключей.

15. Просмотр информации о версии

Вы можете просмотреть, какая версия команды touch установлена в вашей системе, с помощью приведенной ниже команды. Она поможет вам определить различную информацию об установленном пакете.

```
$ touch --version
```

Команда touch - одна из самых простых команд терминала Linux. Она очень проста в изучении благодаря ограниченному количеству ключей. Однако, она может быть действительно полезна в ряде ситуаций, когда системные администраторы используют эту команду для управления временными метками файлов.

ASTRA LINUX GUIDE

СР

Одна из команд, которые вы должны знать в Linux, — ср. Это сокращение от copy, и она делает именно то, что предполагает ее название: она копирует. ср используется для копирования файлов из одного местоположения в другое. ср также можно использовать для копирования всех каталогов в новое место. Вы можете использовать его для копирования нескольких файлов и каталогов.

1. Как скопировать файл

Самый простой пример — скопировать файл. Для этого вам просто нужно указать исходный файл и целевой каталог или файл.

```
cp source_file target_directory / target_file
```

В приведенном выше примере, если target_file не существует в target_directory, он создаст файл target_file.

Однако, если новый_файл уже существует, он будет перезаписывать его, не спрашивая. Это означает, что содержимое существующего файла цели будет изменено с содержимым исходного файла.

Мы покажем вам, как бороться с перезаписью файлов позже в этой статье.

Примечание

Имейте в виду: по умолчанию команда cp перезаписывает, если целевой файл уже существует. Это поведение может быть изменено с помощью опции -n или -i, описанной ниже.

2. Как скопировать несколько файлов

Если вы хотите скопировать несколько файлов одновременно в новое место, вы можете сделать это следующим образом:

```
cp file1 file2 file3 fileN target_directory
```

Эта команда скопирует все указанные файлы в целевой каталог. Если в целевом каталоге есть файл(ы), соответствующий имени исходного файла (ов), он будет перезаписан.

3. Несколько способов обработки перезаписи при копировании файлов.

Вероятно, вы не всегда хотите, чтобы ваши существующие файлы целей были перезаписаны, и это абсолютно логично.

Чтобы предотвратить перезапись существующих файлов, вы можете использовать опцию -n. Таким образом, ср выиграл ‘перезаписать существующие файлы’.

```
cp -n source_file target_directory
```

Но, возможно, вы хотите перезаписать некоторые файлы. Вы можете использовать интерактивную опцию -i, и она спросит вас, хотите ли вы перезаписать существующие файлы.

```
cp -i source_file target_directory  
cp: overwrite 'target_directory/source_file'?
```

ASTRA LINUX GUIDE

Вы можете ввести `u` для перезаписи существующего файла или `n`, чтобы не переписать его.

Существует также возможность создания автоматических резервных копий. Если вы используете опцию `-b` с командой `cp`, она перезапишет существующие файлы, но до этого она создаст резервную копию перезаписанных файлов.

```
cp -b file.txt target_dir / file.txt  
ls target_dir  
file.txt file.txt ~
```

Резервная копия файла заканчивается на `~`.

Вы также можете использовать параметр обновления `-u`, когда имеете дело с перезаписью. С параметром `-u` исходные файлы будут скопированы только в новое место, если исходный файл более новый, чем существующий, или если он не существует в целевом каталоге.

Подвести итоги:

- `-i`: Подтвердить перед перезаписью
- `-n`: Нет перезаписи
- `-b`: Перезапись с резервным копированием
- `-u`: Перезаписать, если целевой файл устарел или не существует

4. Как скопировать каталог

Вы также можете использовать команду `cp` для копирования всего каталога, включая все его файлы и подкаталоги. Вы должны использовать параметр `-r` здесь, который означает рекурсивный.

```
cp -r source_dir target_dir
```

Это скопирует весь `source_dir` в `target_dir`. Теперь `source_dir` будет подкаталогом `target_dir`.

```
ls target_dir  
source_dir
```

5. Как скопировать только содержимое каталога, а не самого каталога

В предыдущем примере вы скопировали весь каталог в новое место.

Но если вы просто хотите скопировать содержимое исходного каталога в целевой каталог, вы должны добавить `/` в конце исходного каталога. Это укажет на то, что вы хотите скопировать содержимое исходного каталога.

Посмотрим на пример:

```
ls source_dir  
source_file1 source_file2
```

Теперь скопируйте содержимое исходного каталога:

```
cp -r source_dir/ target_dir
```

ASTRA LINUX GUIDE

Если вы проверите содержимое целевого каталога сейчас, вы увидите, что было скопировано только содержимое исходного каталога.

```
ls target_dir  
source_file1 source_file2
```

6. Как скопировать несколько каталогов

Вы также можете скопировать несколько каталогов одновременно с помощью команды cp в Linux.

Просто используйте его так же, как и для одного каталога.

```
cp -r source_dir1 source_dir2 source_dir3 target_dir
```

Это всегда последний аргумент в команде, который принимается как целевой каталог.

Если вы хотите скопировать только содержимое нескольких каталогов одновременно, вы также можете это сделать:

```
cp -r source_dir1 ./source_dir2 ./source_dir3 ./target_dir
```

Фактически, вы можете смешивать каталоги, их содержимое и файлы в целом.

```
cp -r source_dir1 source_dir2 ./source_file target_dir
```

Совет

Вы можете использовать подробный режим с опцией -v, чтобы посмотреть, какие файлы копируются.

7. Как сохранить атрибуты при копировании

Когда вы копируете файл в новое место, его атрибуты, такие как права доступа к файлам и временные метки файла, изменяются.

Если вы хотите сохранить атрибуты исходного файла, вы можете скопировать файлы с помощью опции -p.

Давайте посмотрим на пример.

```
ls -l /etc/services
```

```
-rw-r - r-- 1 root root 19183 12 Jan 2018 /etc/services
```

Если я попытаюсь скопировать этот файл, его атрибуты будут изменены:

```
ls -l /etc/services  
-rw-r--r-- 1 root root 19183 Jan 12 2018 /etc/services
```

Но если мы используем параметр p, скопированный файл сохранит режим, право собственности и отметку времени.

```
cp /etc/services .  
ls -l services  
-rwxrwxrwx 1 andreyex andreyex 19183 Nov 27 23:16 services
```

ASTRA LINUX GUIDE

Если вы проверите содержимое целевого каталога сейчас, вы увидите, что было скопировано только содержимое исходного каталога.

```
ls target_dir  
source_file1 source_file2
```

6. Как скопировать несколько каталогов

Вы также можете скопировать несколько каталогов одновременно с помощью команды cp в Linux.

Просто используйте его так же, как и для одного каталога.

```
cp -r source_dir1 source_dir2 source_dir3 target_dir
```

Это всегда последний аргумент в команде, который принимается как целевой каталог.

Если вы хотите скопировать только содержимое нескольких каталогов одновременно, вы также можете это сделать:

```
cp -r source_dir1 ./source_dir2 ./source_dir3 ./target_dir
```

Фактически, вы можете смешивать каталоги, их содержимое и файлы в целом.

```
cp -r source_dir1 source_dir2 ./source_file target_dir
```

Совет

Вы можете использовать подробный режим с опцией -v, чтобы посмотреть, какие файлы копируются.

7. Как сохранить атрибуты при копировании

Когда вы копируете файл в новое место, его атрибуты, такие как права доступа к файлам и временные метки файла, изменяются.

Если вы хотите сохранить атрибуты исходного файла, вы можете скопировать файлы с помощью опции -p.

Давайте посмотрим на пример.

```
ls -l /etc/services
```

```
-rw-r - r-- 1 root root 19183 12 Jan 2018 /etc/services
```

Если я попытаюсь скопировать этот файл, его атрибуты будут изменены:

```
ls -l /etc/services  
-rw-r--r-- 1 root root 19183 Jan 12 2018 /etc/services
```

Но если мы используем параметр p, скопированный файл сохранит режим, право собственности и отметку времени.

```
cp /etc/services .  
ls -l services  
-rwxrwxrwx 1 andreyex andreyex 19183 Nov 27 23:16 services
```

ASTRA LINUX GUIDE

КОМАНДА CHOWN LINUX

Основа философии Linux - все объекты операционной системы - это файлы, для предоставления доступа к тем или иным возможностям системы мы просто даем доступ пользователю к нужным файлам или убираем. Я более подробно рассказывал обо всех правах в статье права доступа к файлам в Linux, здесь же скажу только что у каждого файла есть три группы прав: для владельца, группы и всех остальных.

При создании файла ему тот пользователь, от имени которого он был создан становится его владельцем, а группой устанавливается основная группа владельца. Но владельца файла и группу можно менять, для этого используются команды chown и chgrp.

1. СИНТАКСИС И ОПЦИИ

Синтаксис chown, как и других подобных команд linux очень прост:

```
$ chown пользователь опции /путь/к/файлу
```

В поле пользователь надо указать пользователя, которому мы хотим передать файл. Также можно указать через двоеточие группу, например, пользователь:группа. Тогда изменится не только пользователь, но и группа. Вот основные опции, которые могут вам понадобиться:

- -c, --changes - подробный вывод всех выполняемых изменений;
- -f, --silent, --quiet - минимум информации, скрыть сообщения об ошибках;
- --dereference - изменять права для файла к которому ведет символическая ссылка вместо самой ссылки (поведение по умолчанию);
- -h, --no-dereference - изменять права символьических ссылок и не трогать файлы, к которым они ведут;
- --from - изменять пользователя только для тех файлов, владельцем которых является указанный пользователь и группа;
- -R, --recursive - рекурсивная обработка всех подкаталогов;
- -H - если передана символическая ссылка на директорию - перейти по ней;
- -L - переходить по всем символическим ссылкам на директории;
- -P - не переходить по символическим ссылкам на директории (по умолчанию).

Утилита имеет ещё несколько опций, но это самые основные и то большинство из них вам не понадобится. А теперь давайте посмотрим как пользоваться chown.

2. ИСПОЛЬЗОВАНИЕ CHOWN

Например, у нас есть несколько папок dir и их владелец пользователь sergiy:

```
ls
```

Давайте изменим владельца папки dir1 на root:

```
chown root ./dir1
```

Если вы хотите поменять сразу владельца и группу каталога или файла запишите их через двоеточие, например, изменим пользователя и группу для каталога dir2 на root:

```
chown root:root ./dir2
```

ASTRA LINUX GUIDE

Если вы хотите чтобы изменения применялись не только к этому каталогу, но и ко всем его подкаталогам, добавьте опцию -R:

```
chown -R root:root ./dir3
```

Дальше давайте изменим группу и владельца на www-data только для тех каталогов и файлов, у которых владелец и группа root в каталоге /dir3:

```
chown --from=root:root www-data:www-data -cR ./
```

Для обращения к текущему каталогу используйте путь `./`. Мы его использовали и выше. Далее указываем нужную группу с помощью опции `--from` и просим утилиту выводить изменения, которые она делает в файловой системе с помощью опции `-c`.

КОМАНДА CHMOD LINUX

Система полномочий в Linux имеет очень важное значение, поскольку благодаря ей можно разделять привилегии между пользователями, ограничить доступ к нежелательным файлам или возможностям, контролировать доступные действия для сервисов и многое другое. В Linux существует всего три вида прав - право на чтение, запись и выполнение, а также три категории пользователей, к которым они могут применяться - владелец файла, группа файла и все остальные.

Эта команда имеет типичный для команд linux синтаксис, сначала команда, затем опции, а в конце файл или папка, к которой ее нужно применить:

```
$ chmod опции права /путь/к/файлу
```

Сначала рассмотрим какими бывают права доступа linux и как они устанавливаются. Пред этим рекомендую прочитать статью про права, ссылка на которую есть выше. Есть три основных вида прав:

- r - чтение;
- w - запись;
- x - выполнение;
- s - выполнение от имени суперпользователя (дополнительный);

Также есть три категории пользователей, для которых вы можете установить эти права на файл linux:

- u - владелец файла;
- g - группа файла;
- o - все остальные пользователи;

Синтаксис настройки прав такой:

группа_пользователей действие вид_прав

В качестве действий могут использоваться знаки "+" - включить или "-" - отключить. Рассмотрим несколько примеров:

- u+x - разрешить выполнение для владельца;
- ugo+x - разрешить выполнение для всех;
- ug+w - разрешить запись для владельца и группы;
- o-x - запретить выполнение для остальных пользователей;
- ugo+rwx - разрешить все для всех;

ASTRA LINUX GUIDE

Но права можно записывать не только таким способом. Есть еще восьмеричный формат записи, он более сложен для понимания, но пишется короче и проще. Я не буду рассказывать как считать эти цифры, просто запомните какая цифра за что отвечает, так проще:

- 0 - никаких прав;
- 1 - только выполнение;
- 2 - только запись;
- 3 - выполнение и запись;
- 4 - только чтение;
- 5 - чтение и выполнение;
- 6 - чтение и запись;
- 7 - чтение запись и выполнение.

Права на папку linux такие же, как и для файла. Во время установки прав сначала укажите цифру прав для владельца, затем для группы, а потом для остальных.

Например :

- 744 - разрешить все для владельца, а остальным только чтение;
- 755 - все для владельца, остальным только чтение и выполнение;
- 764 - все для владельца, чтение и запись для группы, и только чтение для остальных;
- 777 - всем разрешено все.

Каждая из цифр не зависит от предыдущих, вы выбираете именно то, что вам нужно.

Теперь давайте рассмотрим несколько опций команды, которые нам понадобятся во время работы:

- -c - выводить информацию обо всех изменениях;
- -f - не выводить сообщения об ошибках;
- -v - выводить максимум информации;
- --preserve-root - не выполнять рекурсивные операции для корня "/";
- --reference - взять маску прав из указанного файла;
- -R - включить поддержку рекурсии;
- --version - вывести версию утилиты;

Теперь, когда вы знаете опции и как настраиваются права доступа chmod, давайте рассмотрим несколько примеров как работает команда chmod linux.

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ CHMOD

Я не буду приводить много примеров, так как там и так все более-менее понятно после пояснения правил создания выражений установки прав. Сначала самый частый случай - разрешить выполнения скрипта владельцу:

```
chmod u+x file
```

Или можно воспользоваться цифровой записью:

```
chmod 766 file  
ls -l file
```

ASTRA LINUX GUIDE

Недостаток цифровой записи в том, что вы не можете модифицировать уже существующие права доступа linux. Например, в первом варианте вы просто добавили флаг выполнения для владельца файла, а все остальные права оставили неизменными. В восьмеричном варианте мы полностью заменили текущие права новыми - все для владельца и чтение/запись для группы и остальных. Как видите, права установлены как и ожидалось. Теперь отключим выполнение владельцем:

```
chmod u-x file
```

И снова проверяем:

```
ls -l file
```

Дальше разрешим только чтение всем, кроме владельца:

```
chmod 744 file  
ls -l file
```

Или отберем все права:

```
chmod ugo-rwx file
```

Файлы с правами 000 недоступны никаким пользователям, кроме суперпользователя и владельца. Вернем права обратно:

```
chmod 755 file
```

```
ls -l file
```

Такая же ситуация с файлами, владельцем которых вы не являетесь, если вы хотите изменить их права - используйте sudo. Из модификаторов прав вы можете конструировать любые последовательности, я думаю тут нет смысла разбирать их все. Для того чтобы поменять права на все файлы в папке используйте опцию -R:

```
chmod -R ug+rw dir
```

```
ls -l dir/
```

Также вы можете смотреть подробную информацию про вносимые изменения, например:

```
chmod -Rv ug+rw dir
```

ASTRA LINUX GUIDE

Описание команды rm

rm

Удаление файлов и директорий.

По умолчанию команда rm не удаляет директории. Чтобы удалить директорию и все ее содержимое, включая вложенные директории, нужно использовать опцию -r (рекурсивное удаление).

Синтаксис

rm опции файл(ы)

файл(ы) — один или несколько файлов и/или директорий, записанных через пробел.

Можно использовать шаблон (например, *.txt).

Опции

-f или --force

Игнорировать несуществующие файлы и аргументы. Никогда не выдавать запросы на подтверждение удаления.

-i

Выводить запрос на подтверждение удаления каждого файла.

-I

Выдать один запрос на подтверждение удаления всех файлов, если удаляется больше трех файлов или используется рекурсивное удаление. Опция применяется, как более «щадящая» версия опции -i

--interactive[=КОГДА]

Вместо КОГДА можно использовать:

never — никогда не выдавать запросы на подтверждение удаления.

once — выводить запрос один раз (аналог опции -I).

always — выводить запрос всегда (аналог опции -i).

Если значение КОГДА не задано, то используется always

--one-file-system

Во время рекурсивного удаления пропускать директории, которые находятся на других файловых системах.

--no-preserve-root

Если в качестве директории для удаления задан корневой раздел /, то считать, что это обычная директория и начать выполнять удаление.

--preserve-root

Если в качестве директории для удаления задан корневой раздел /, то запретить выполнять команду rm над корневым разделом. Данное поведение используется по умолчанию.

-r или -R или --recursive

Удаление директорий и их содержимого. Рекурсивное удаление.

-d или --dir

Удалять пустые директории.

-v или --verbose

Выводить информацию об удаляемых файлах.

Примечание

Если нужно удалить файл, название которого начинается с символа -, например, файл -myfile, то используется следующая команда:

ASTRA LINUX GUIDE

```
rm -- myfile
```

Или:

```
rm ./myfile
```

Примеры использования команды rm

Удалить файл

Чтобы удалить файл с помощью команды rm достаточно указать название файла:

```
rm myfile.txt
```

Можно удалить несколько файлов, перечислив их имена через пробел:

```
rm myfile1.txt myfile2.txt myfile3.txt
```

Принудительно удалить защищенный файл

Если файл защищен от записи, то по умолчанию будет выдан запрос на подтверждение операции удаления. Чтобы запрос не выводился, и происходило удаление защищенных файлов, используется опция -f

```
rm -f myfile.txt
```

Удалить все файлы в текущей директории

Чтобы удалить все файлы в текущей директории можно использовать шаблонную запись звездочка *

```
rm *
```

Выдавать запрос перед удалением каждого файла

Чтобы перед удалением каждого файла выводилось подтверждение на удаление данного файла, используется опция -i

```
rm -i myfile*.txt
```

Выдать запрос на удаление один раз

При использовании опции -l выводится всего один запрос на подтверждение удаления файлов, причем, только если удаляемых файлов четыре и более. Если файлов 3, или 2, или 1, то запрос не выводится.

```
rm -l myfile*.txt
```

Удалить директорию и ее содержимое

Для удаления директорий и их содержимого используется опция -r. Удалим директорию mydir и все файлы и директории внутри нее:

```
rm -r mydir
```

ASTRA LINUX GUIDE

КОМАНДА TAR В LINUX

В качестве инструмента для архивации данных в Linux используются разные программы. Например архиватор Zip Linux, приобретший большую популярность из-за совместимости с ОС Windows. Но это не стандартная для системы программа. Поэтому хотелось бы осветить команду tar Linux — встроенный архиватор.

Синтаксис команд для создания и распаковки архива практически не отличается (в том числе с утилитами сжатия bzip2 или gzip). Так, чтобы создать новый архив, в терминале используется следующая конструкция:

```
tar опции архив.tar файлы_для_архивации
```

Для его распаковки:

```
tar опции архив.tar
```

Функции, которые может выполнять команда:

- A --concatenate - Присоединить существующий архив к другому
- c --create - Создать новый архив
- d --diff - Проверить различие между архивами
- delete - Удалить из существующего архива файл
- r --append - Присоединить файлы к концу архива
- t --list - Сформировать список содержимого архива
- u --update - Обновить архив более новыми файлами с тем же именем
- x --extract - Извлечь файлы из архива

При определении каждой функции используются параметры, которые регламентируют выполнение конкретных операций с tar-архивом:

- C dir --directory=DIR - Сменить директорию перед выполнением операции на dir
- f file --file - Вывести результат в файл (или на устройство) file
- j --bzip2 - Перенаправить вывод в команду bzip2
- p --same-permissions - Сохранить все права доступа к файлу
- v --verbose - Выводить подробную информацию процесса
- totals - Выводить итоговую информацию завершенного процесса
- z --gzip - Перенаправить вывод в команду gzip

А дальше рассмотрим примеры того, как может применяться команда tar Linux.

1. СОЗДАНИЕ АРХИВА TAR

С помощью следующей команды создается архив archive.tar с подробным выводом информации, включающий файлы file1, file2 и file3:

```
tar --totals --create --verbose --file archive.tar file1 file2 file3
```

Но длинные опции и параметры можно заменить (при возможности) однобуквенными значениями:

```
tar --totals -cvf archive.tar file1 file2 file3
```

ASTRA LINUX GUIDE

2. ПРОСМОТР СОДЕРЖИМОГО АРХИВА

Следующая команда выводит содержимое архива, не распаковывая его:

```
tar -tf archive.tar
```

3. РАСПАКОВКА АРХИВА TAR LINUX

Распаковывает архив test.tar с выводом файлов на экран:

```
tar -xvf archive.tar
```

Чтобы сделать это в другой каталог, можно воспользоваться параметром -C:

```
tar -C "Test" -xvf archive.tar
```

4. РАБОТА СО СЖАТЫМИ АРХИВАМИ

Следует помнить, что tar только создаёт архив, но не сжимает. Для этого используются упомянутые компрессорные утилиты bzip2 и gzip. Файлы, сжатые с их помощью, имеют соответствующие расширения .tar.bz2 и .tar.gz. Чтобы создать сжатый архив с помощью bzip2, введите:

```
tar -cjvf archive.tar.bz2 file1 file2 file3
```

Синтаксис для gzip отличается одной буквой в параметрах, и меняется окончание расширения архива:

```
tar -czvf archive.tar.gz file1 file2 file3
```

При распаковке tar-архивов с таким расширением следует указывать соответствующую опцию:

```
tar -C "Test" -xjvf archive.tar.bz2
```

или:

```
tar -xzvf archive.tar.gz
```

ASTRA LINUX GUIDE

Команда gzip

Команда gzip предназначена для сжатия данных без потерь с помощью одноименной утилиты, использующей алгоритм Лемпела-Зива (LZ77) с кодированием Хаффмана. Целью использования данной утилиты является экономия дискового пространства. Упомянутый алгоритм позволяет достичь худшей степени сжатия данных, чем те, которые реализованы в рамках утилит bzip2 и xz. При этом данный алгоритм является стандартным алгоритмом утилиты zip и используется по умолчанию в архивах формата ZIP. Кроме того, он является менее ресурсоемким, чем алгоритмы, которые реализованы в рамках утилит bzip2 и xz. Последнее обстоятельство обуславливает актуальность данной утилиты для низкопроизводительных систем.

Базовый синтаксис команды выглядит следующим образом:

```
$ gzip [параметры] <имена файлов>
```

Чаще всего gzip используется вообще без каких-либо параметров, причем в качестве аргументов может передаваться неограниченное количество имен файлов, которые следует сжать. По умолчанию оригинальные версии файлов заменяются на их сжатые версии с соответствующими метаданными (то есть, меткой времени модификации, правами доступа, именами владельца и группы владельцев и так далее). Если вас не устраивает такое положение дел, вы можете воспользоваться параметром -k для сохранения оригинальных версий файлов. Параметры из диапазона от -1 до -9 позволяют задать степень сжатия (от самой низкой до самой высокой соответственно), при этом чем выше степень сжатия, тем больше системных ресурсов требуется утилите. Параметр -t предназначен для тестирования целостности сжатого файла и не оказывает какого-либо влияния на него (следует комбинировать его с параметром -v для подробного вывода). Параметр -l также не оказывает никакого влияния на сжатый файл и предназначен для получения информации об архиве (размеров сжатого и не сжатого файлов, степени сжатия и имени оригинального файла). Параметр -d позволяет восстановить оригинальные версии файлов с переданными именами на основе их сжатых версий.

Если же вам нужно создать архив с несколькими файлами внутри, одной утилиты gzip будет явно мало. Для этой цели также понадобится утилита tar, с помощью которой можно создать архив с файлами, после чего сжать этот архив с помощью утилиты gzip. Например, вы можете использовать следующую последовательность команд для создания архива с именем archive.tar.gz:

```
$ tar -cf archive.tar <имена файлов>
$ gzip archive.tar
```

Параметры -c и -f утилиты tar предназначены для указания на необходимость добавления всех файлов в один архив (-c) и чтения имени файла архива из следующего аргумента (-f). Альтернативным вариантом является замена последней команды на параметр -z утилиты tar, позволяющий автоматически сжать полученный архив с помощью gzip:

```
$ tar -cfz archive.tar.bz2 <имена файлов>
```

ASTRA LINUX GUIDE

1. Сжатие одного файла

```
$ gzip text.txt
```

В результате оригинальный файл text.txt будет заменен на свою сжатую версию text.txt.gz.

2. Одновременное сжатие нескольких файлов

```
$ gzip text1.txt text2.txt text3.txt
```

В этом случае также все оригинальные версии файлов (text1.txt, text2.txt, text3.txt) будут заменены на сжатые версии (text1.txt.gz, text2.txt.gz, text3.txt.gz). Добавление нескольких файлов в единый файл архива будет рассмотрено ниже.

3. Сжатие одного файла с сохранением оригинала

```
$ gzip -k text.txt
```

Теперь оригинальный файл text.txt будет оставлен в директории вместе со сжатой версией text.txt.gz.

4. Восстановление оригинальной версии файла из сжатой версии

```
$ gzip -d text.txt.gz
```

В результате сжатая версия файла text.txt.gz будет заменена на его оригинальную версию text.txt. Если вам нужно сохранить сжатую версию, следует воспользоваться параметром -k таким же образом, как было показано выше.

5. Сжатие файла с указанием степени сжатия

Степень сжатия файла может регулироваться с помощью параметров из диапазона от -1 (минимальная) до -9 (максимальная). Команда для сжатия файла с минимальной степенью сжатия:

```
$ gzip -1 text.txt
```

С максимальной степенью сжатия:

```
$ gzip -9 text.txt
```

При этом параметр -1 может заменяться на параметр --fast, а параметр -9 — на параметр --best.

6. Создание сжатого архива со всеми файлами из директории

```
$ tar -cfz etc.tar.bz2 /etc/
```

В результате будет создан архив etc.tar.gz с файлами из директории /etc/.

ASTRA LINUX GUIDE

7. Проверка целостности сжатой версии файла

```
$ gzip -tv text.txt.gz
```

text.txt.gz: OK

Для проверки целостности сжатых версий файлов используются контрольные суммы (CRC). В случае повреждения сжатой версии файла выводится сообщение об ошибке, в противном случае — слово «OK».

8. Получение информации о сжатом файле

```
$ gzip -l enums.c.gz
```

compressed	uncompressed	ratio	uncompressed_name
253	371	38.8%	enums.c

Очевидно, что в столбце compressed выводится размер сжатого файла в байтах, в столбце uncompressed — размер несжатого файла в байтах, в столбце ratio — степень сжатия файла, а в столбце uncompressed_name — имя оригинального файла. Вся эта информация хранится в заголовке сжатого файла.