**Краткая шпаргалка по Linux**

При простом вводе **ls** мы получим список файлов и каталогов из текущей папки.

Команда **ls** может принимать параметр **-l**, который позволяет отобразить более детальную информацию о файлах и каталогах, в частности, права доступа, дату создания и т.д.

Команда **ls** в качестве аргумента получает путь к каталогу, содержимое которого надо получить. (**ls /usr**)

По умолчанию результат выполнения команды отображается на консоли. Используя же перенаправление, вы можете отправить его куда угодно, например, в файл. Например, вывдеем результат команды ls -l в файл folder.txt:

**ls -l > folder.txt**

Если такой файл есть, но его надо не перезаписать, а добавить результат команды к уже имеющемуся содержимому. Тогда можно использовать оператор **>>** :

**ls -l >> folder.txt**

В процессе выполнения команды могут возникать ошибки. Для вывода ошибок применяется оператор **2>**, например:

**ls -l > folder.txt 2> errors.txt**

В данном случае вывод команды записывается в файл "folder.txt". Но если будут ошибки, то они записываются в "errors.txt".

Чтобы увидеть всю эту историю, можно открыть данный файл, либо можно ввести в терминале команду **history**:

Linux позволяет повторить команду, использовав ее номер в истории. Например, в выводе выше под номером **4** идет команда **ls -l /usr**. И чтобы повторить эту команду, достаточно ввести **!4** (перед номером ставится восклицательный знак).

Также можно очистить всю историю с помогает команды:

**history –c**

Команда **pwd** отображает текущий каталог.

Команда **cd** позволяет перемещаться по каталогам. При использовании **cd** следует учитывать некоторые особенности файловой системы Linux:

* Имена файлов и каталогов Linux чувствительны к регистру. То есть bin и BIN — это не одно и то же!
* Для перехода в каталог, который находится непосредственно под корневым каталогом, надо поставить косую черту **/** перед именем каталога. Без косой черты эта команда попытается найти указанный каталог как подкаталог в текущем каталоге.

Команда **cd** имеет только один аргумент - имя каталога, в который надо перейти. Например, перейдет в каталог **/usr/bin**, который находится непосредственно в корневом каталоге файловой системы.

Для перехода к домашнему каталогу применяется символ **~**. Например, перейдем к папке "Documents" в домашнем каталоге:

**cd ~/Documents**

В данном случае путь "~/Documents" идентичен "/home/[имя]/Documents"

Для перехода на один уровень вверх применяется путь **..** :

**cd ..**

Соответственно для перехода на два уровня вверх применяется путь "**../../**", на три - "**../../../**" и так далее.

Для создания папки применяется команда **mkdir**.

**mkdir ~/Documents/test**

Также при создании папки можно указать полный путь относительно корневого каталога. Например, создадим папку "test" в каталоге "tmp" (каталог временных файлов):

**eugene@Eugene:~$ mkdir /tmp/test**

Однако по умолчанию команда **mkdir** завершится с ошибкой, если не существует родительского каталога, в котором надо создать папку. Каталога "test1" не существует, поэтому и создать в нем папку нельзя.

Однако команде mkdir можно передать опцию **-p**, которая принудительно также создает родительский каталог, если его не существует.

**mkdir -p ~/Documents/test1/files**

Команда **rmdir** удаляет каталог. Однако она работает только с уже пустыми каталогами. Например, удалим выше созданный каталог "~/Documents/test1/files":

**rmdir ~/Documents/test1/files**

Если же в каталоге есть файлы и/или подкаталоги, применяется команда **rm -r** или **rm -rf**:

**rm -r ~/Documents/test1**

Команда **ls** отображает список файлов и каталогов в определенном каталоге. Она имеет следующую форму: **ls [options] [FILE]**

В качестве необязательного аргумента **FILE** команда принимает каталог, содержимое которого надо вывести. Если этот аргумент не указан, то выводится содержимое текущего каталога.

Вместо конкретной папки можно указать шаблон файлов, в котором символ звездочка **\*** представляет заменитель для произвольных символов. Например:

**ls /etc/\*a.\***

В данном случае получаем все файлы из папки "/etc", которые заканчиваются на "a" и имеют произвольное расширение.

Другой пример - найдем все каталоги и файлы, названия которых начинаются либо на a, либо на b, либо на c:

**ls /etc/[abc]\***

Выражение [abc] указывает, что символ должен представлять букву a, b или c, а звездочка после - [abc]\* указывает, что дальше может идти произвольное количество символов. Единственно надо учитывать, что если название каталога соответствует данному фильтру, то выводится все содержимое каталога.

Символ звездочка **\*** позволяет вывести также содержимое подкаталогов из текущей папки:

**ls \***

Опция -R позволяет рекурсивно получить содержимое подкаталогов. Например, получим содержимое папки Pictures из домашнего каталога пользователя:

ls -R ~/Pictures

/home/eugene/Pictures:

Screenshots

Вывод только каталогов.

Для вывода только каталогов применяется опция **–d** :

**ls -d \*/**

Вывод размера файлов.

Для рядом с файлами размера в киллобайтах применяется опция **-s** (нижний регистр):

**ls -s**

Сортировка файлов.

Для сортировки файлов по размеру применяется опция **-S** (верхний регистр):

**ls -S**

Для сортировки файлов по времени доступа применяется опция **–t**:

**ls –t**

Для вывода свойств каталогов и файлов применяется опция **–l**.

Эта команда отображает данные в виде следующих столбцов:

* Тип файла: самая первая буква показывает тип файла. Если отображается дефис -, это обычный файл. Если отображается буква d, то это каталог.
* Разрешения: сразу после типа файла идут разрешения, назначенные этому файлу. Есть девять позиций, которые показывают права доступа к файлам, например, rwxrwxr-x. Далее мы подробнее рассмотрим разрешения.
* Количество ссылок на содержимое
* Владение: в Linux у каждого файла есть владелец пользователя и владелец группы. В следующем примере выше всех файлов и каталогов в качестве владельца установлен текущей пользователь - в моем случае "eugene", а в качестве группы - одноименная группа этого пользователя.
* Размер файла: рядом с владельцем группы отображается размер файла в байтах.
* Дата и время создания: для каждого файла также отображаются дата и время создания.
* Имя файла: в последнем столбце вывода можно увидеть имя файла.

Вывод скрытых файлов и каталогов

Использование опции **-a** выводит скрытые файлы:

**ls -a**

Соответственно для вывода подробной информацию эту опцию можно комбинировать с опцией "**-l**":

**ls -a -l**

Вывод команды в файл

Нередко применяется вывод команды в файл. Например выведем результат команды в файл "output.txt":

**ls -a -l > output.txt**

Для создания файла применяется команда **touch**.

**touch ~/Documents/hello.txt**

Для удаления файлов (а также каталогов) применяется команда **rm**.

**rm ~/Documents/hello.txt**

Команду **rm** можно также использовать для удаления каталогов. В этом случае необходимо использовать опцию **-r**. Например, удаление папки "test":

**rm -r ~/Documents/test**

Копирование файлов и каталогов

Для копирования файлов/каталогов применяется команда **cp**. Она имеет следующую форму:

**cp source destination**

Первый аргумент команды - **source** обычно представляет имя каталога, файлов или шаблон файла, которые надо скопировать. Второй аргумент - **destination** представляет путь, куда надо скопировать. Например, скопируем файл "Documents/test/hello.txt" в папку "Documents":

**cp ~/Documents/test/hello.txt ~/Documents/hello.txt**

Другой пример - скопируем файл "Documents/test/hello.txt" в папку "Documents/test2":

**mkdir ~/Documents/test2**

**cp ~/Documents/test/hello.txt ~/Documents/test2**

В данном случае важно, что папка "test2" уже существует перед копированием. Иначе будет создаваться текстовый файл "test2" с содержимым из файла "hello.txt".

Для копирования **каталогов** применяется флаг **-r**. Например, скопируем содержимое папки "Documents/test" в папку "Documents/test4"

**cp -r ~/Documents/test/ ~/Documents/test4**

При этом папка test4 создается автоматически.

Шаблон файла может применять символ звездочки **\*** для указания всех файлов. Например, команда **cp ~/\* /tmp** копирует все файлы из домашнего каталога в каталог **/tmp**

Чтобы скопировать скрытые файлы, применяется символ звездочки. Например, команда:

**mkdir ~/Documents/test5**

**cp .\* ~/Documents/test5**

cкопирует все скрытые файлы из текущего каталога в папку "Documents/test5", поскольку первый аргумент относится ко всем файлам, имена которых начинаются с точки (папка test5 должна быть предварительно создана).

Если надо скопировать еще и скрытые каталоги и их содержимое, применяется флаг **-r**:

**cp -r .\* ~/Documents/test5**

Опции:

Команда принимает множество опций. Отмечу некоторые из них:

**-a**: указывает, что свойства копируемых файлов, например, разрешения, также копируются

**-b**: резервное копирование. Если команда **cp** перезапишет существующий файл назначения, эта опция гарантирует, что сначала будет создана резервная копия этого файла назначения.

**-f**: если файл в месте назначения запрещает копировать файл, эта опция приведет к принудительному копированию. Это означает, что файл назначения перезаписан.

**-i**: при использовании этой опции **cp** запрашивает перезапись, если указанный файл уже существует в месте назначения.

**-p**: гарантирует, что атрибуты файла, такие как владельцы и разрешения, также копируются.

**-r**: гарантирует, что каталоги копируются рекурсивно.

**-u**: копируется только в том случае, если целевой файл старше исходного файла или если целевой файл не существует.

Перемещение файлов/каталогов

Команда **mv** перемещает файлы/каталоги в новое место. Она имеет следующую форму:

**mv source destination**

Первый аргумент команды - **source** представляет каталоги/файлы, которые надо переместить. Второй аргумент - **destination** представляет путь, куда надо переместить. Например, переместим файл "Documents/test/hello.txt" в папку "Documents/test6":

**mkdir ~/Documents/test6**

**mv ~/Documents/test/hello.txt ~/Documents/test6**

В данном примере предварительно создается каталог test6, в который затем перемещается файл. Однако если бы каталога test6 не существовало, то создавался бы файл "test6", в который помещалось бы содержимое из файла "Documents/test/hello.txt"

Аналогично можно перемещать каталоги. Например:

**mkdir ~/Documents/test7**

**touch ~/Documents/test7/helworld.txt**

**mv ~/Documents/test7 ~/Documents/test8**

Для теста здесь создается папка test7, в которой создается файл helworld.txt. Далее мы перемещаем каталог "Documents/test7" по пути "Documents/test8". Что конкретно произойдет, зависит от того, существует ли каталог "test8" или нет. Если он существует, то в нем создается подкаталог "test7" со всем содержимым из "Documents/test7". Если "test8" НЕ существует, то создается катлог "test8", в который помещается все содержимое из "Documents/test7"

Последний случай показывает, что мы можем применять команду mv для переименования. Например:

**mv ~/Documents/test8 ~/Documents/test9**

Если test9 не существует, то фактически test8 будет переименован в test9.

Количество слов в файле

Специальная команда **wc** позволяет подсчитать количество слов в файле:

**wc ~/Documents/test/hello.txt**

**32 274 1572 /home/eugene/Documents/test/hello.txt**

Данная команда последовательно выводит количество строк, слов и символов.

Ссылки

Ссылка (link) в Linux представляет указатель на файл. Ссылка облегчает поиск нужных файлов. Символическая ссылка ссылается на имя файла. Ее можно использовать для ссылки на файл, который находится где угодно. Однако символическая ссылка зависит от исходного файла. Например, если исходный файл будет удален, символическая ссылка больше не будет работать.

Для создания символической ссылки применяется команду **ln** с опцией **-s**. Например, для создния ссылки с именем "myhost" в домашнем каталоге, которая ссылается на файл /etc/hosts, используется следующая команда:

**ln -s /etc/hosts ~/myhost**

В результате в домашнем каталоге будет создан ярлык с именем "myhost". Этот ярлык относится к файлу "/etc/hosts". И при каждом открытии файла "myhost" фактически будет открываться файл /etc/hosts.

Для просмотра информации о ссылке можно использовать команду ls -l:

**ls -l ~/myhost**

**lrwxrwxrwx 1 eugene eugene 10 Mar 11 20:15 /home/eugene/myhost -> /etc/hosts**

Первая буква **l** является сокращением от "link" и указывает, что это ссылка. Кроме того, в выводе выражение /home/eugene/myhost -> /etc/hosts указывает, на какой именно файл ссылается ссылка.

**find**

Команда **find** применяется для поиска файлов и каталогов по различным критериям, таким как имя, размер, тип и время изменения. Например, найдем все файлы с расширением .txt в каталоге Documents текущего пользователя и его подкаталогах:

**find ~/Documents -name "\*.txt"**

Первый аргумент команды — это начальный каталог поиска. А параметр -**name** используется для указания соответствующего шаблона имени файла. Символ **\*** — это подстановочный знак, который соответствует любому количеству символов, а **\*.txt** соответствует всем файлам, оканчивающимся на ".txt".

Команда **find** позволяет не только находить файлы по их именам. Например, можно найти файлы по их размеру, владельцу, разрешениям и многим другим параметрам. Опции для поиска файлов:

-amin n: находит все файлы, к которым последний раз обращались менее n минут назад. Например, find -amin 5 выдаст все файлы, к которым обращались менее пяти минут назад.

-executable: находит все исполняемые файлы.

-group name: показывает все файлы, владельцем группы которых является пользователь с логином "name".

-mmin n: показывает все файлы, которые последний раз были изменены менее n минут назад.

-newer file показывает все файлы новее, чем файл file.

-nogroup, -nouser: показать все файлы, у которых нет группы или владельца.

-perm mode: находит все файлы, для которых установлен режим разрешений .mode

-size n: находит все файлы размера n. С помощью этого параметра также можно найти файлы больше или меньше определенного размера. Например, команда find / -size +2G позволит найти все файлы размером более 2 гигабайт. При использовании этого параметра также применяются K, M и G для килобайтов, мегабайтов и гигабайтов соответственно. Знак + позволяет задать поиск файлов, размер которых превышает определенный.

-type t: находит файлы определенного типа. Наиболее интересные типы файлов, которые вы можете искать с помощью этой опции, — это d для каталога или f для обычного файла (то есть любого файла, не являющегося каталогом).

Также можно комбинировать разные варианты, например, найти все файлы пользователя eugene, размером более 100 МБ:

**find ~ -user eugene -size +100M**

**whereis**

Команда whereis применяется для поиска бинарных, исходных файлов и файлов страниц руководств для определенной команды. Например, найдем местоположение команды find. Или найдем местоположение команды "gcc" (компилятора языка Си). (**whereis find** / **whereis gcc**)

Команда **mount** в общем случае предназначена для ручного монтирования устройств (например, подключаемых по USB) в систему. Для отключения устройства применяется команда **umount**. Эта команда может принимать два аргумента: имя устройства или имя каталога, в котором устройство смонтировано. Например:

**umount /dev/sda1**

В этом случае для завершения процессов, связанных с устройством, можно использовать команду fuser -km, которой передается название устройства

**fuser -km /dev/sda1**

Команда **mount** позволяет вывести информацию обо всех примонтированных устройствах.

Для обзора реальных смонтированных файловых систем применяется команда **df –h**.

Команда **lsblk** отображает все устройства, достпные в текущий момент на компьютере.

Основным элементом жесткого диска/ssd являются разделы, которые представляют собой некотрые участки диска. Список всех разделов на дисках хранится в файле **/proc/partitions**:

**cat /proc/partitions**

Для управления дисками и разделами применяется ряд утилит, в частности, **fdisk** и **gdisk**. С помощью параметра **-l** можно посмотреть информацию по определенному диску, в частности, его разделы. Например, посмотрим информацию по какому-нибудь диску:

**sudo fdisk -l /dev/nvme1n1**

Здесь видно, что на диске 2 раздела: /dev/nvme1n1p1 и /dev/nvme1n1p2. Можно увидеть начало и конец раздела, количество секторов, размер, тип системы и т.д.

Аналогичную информацию можно получить с помощью утилиты gdisk:

**sudo gdisk -l /dev/nvme1n1**

Текстовые файлы играют важную роль в Linux. Это могут быть различные файлы конфигурации, логи и т.д. И Linux предоставляет инструментарий для работы с текстовыми файлами. Распространенные команды для вывода содержимого файлов:

**cat**: отображает содержимое файла.

**tac**: делает то же самое, что и cat, но отображает содержимое в обратном порядке.

**tail**: показывает только последние строки текстового файла.

**head**: отображает первые строки файла.

**less**: открывает расширенную программу просмотра файлов.

**more**: то же самое, что и less, только с меньшими возможностями.

Например, выведем с помощью команды cat файл /etc/hosts:

**cat /etc/hosts**

Команда **tac** отобразит все строки в обратном порядке:

**tac /etc/hosts**

Команда **tail** отображает последние строки файла (по умолчанию 10 последних). Но также ей можно передать конкретное количество строк (с конца). Например, вывелем три последние строки:

**tail -3 /etc/hosts**

Команда **head** аналогична **tail**, только отображает первые строки файла (по умолчанию 10 первых). И она также может принимать конкретное количество строк (с конца). Например, вывелем три первых строки:

**head -3 /etc/hosts**

Комбинируя команды **head** и **tail**, можно получить определенную строку или опеределенный набор строк с начала или конца. Например, получим только 2-ю строку:

**head -n 2 /etc/hosts | tail -n 1**

В данном случае выражение **head -n 2 /etc/hosts** сначала выбирает из файла 2 первых строки. Далее эти 2 строки передаются команде **tail -n 1**, которая выбирает из них последнюю (то есть фактически 2-ю строку).

Команда **less** открывает файл в специальном окружении, которое напоминает встроенный текстовый редактор vi:

**less /etc/hosts**

Для выхода из окружения просмотра файла нажмем на **q**.

Стоит отметить, что вывод других команд также по сути представляет собой текст, и этот текст мы также можем обрабатывать с помощью текстовых команд. Например, при выводе списока файлов и каталогов получим только последние 2 строки:

**ls -l | tail -n 2**

Команда **tr** применяется для преобразования или удаления символов из файла. Распространенный сценарий использования **tr** — перевод строчных букв в прописные.

**cat test.txt | tr a-z A-Z**

Как видно, команда tr применяется к строкам из файла test.txt, которые считываются командой cat. В данном случае команда tr преобразует все символы диапазона a–z в соответствующие символы из диапазона A–Z

Преобразованный результат затем можно записать в другой файл:

**cat test.txt | tr a-z A-Z > test2.txt**

Для сортировки строк в файле применяется команда **sort**:

**cat test.txt | sort**

Аналогично можно отсортировать вывод других команд, например, список файлов и каталогов:

**ls | sort**

По умолчанию команда сортирует строки в лексикографическом порядке. Однако с числами подобная сортировка может давать некорректные результаты. И в этом случае мы можем использовать флаг -n:

**cat test.txt | sort -n**

Команда **cut**. Получение определенной информации

Команда **cut** позволяет получить из структурированного файла определенный кусок информации. Это похоже на работу с таблицами, где мы хотим получить данные определенных столбцов. Данная команда принимает ряд опций, из которых отмечу основные:

* -d: представляет разделитель данных в строке
* -f: представляет список номеров полей для вывода. Список полей можно задать с помощью одного из следующих форматов:
* N: выводит поле с номером N (нумерация начинается с 1)
* N-: определяет диапазон полей, который начинается с номера N и заканчивается последним полем
* N-M: определяет диапазон полей от N до M
* -M: определяет диапазон полей, который начинается с первого поля и заканчивается полем с номером M

Условно говоря, каждая строка содержит данные пользователя - имя, возраст, компанию, то есть три **поля**. И каждое поле отделено от других двоеточием. И допустим, нам надо получить имена пользователей:

**cut -d : -f 1 test.txt**

Здесь опция -d указывает, что разделитель - двоеточие, а опция -f указывает, что надо получить 1-е поле.

Выведем 1 и 3 -е поля:

**cut -d : -f 1,3 test.txt**

Команда grep выполняет поиск в файле с использованием регулярных выражений. Она имеет следующий формат:

**grep [OPTIONS] PATTERNS [FILE]**

В качестве обязательного аргумента PATTERNS она принимает шаблон текста, который надо найти. Если команде передан аргумент FILE, то он используется в качестве файла для поиска текста. И необязательный параметр OPTIONS позволяет настроить применение шаблона для поиска текста.

Например, найдем текст "127.0." в файле /etc/hosts:

**grep "127.0." /etc/hosts**

Другой пример - найдем в домашнем каталоге пользователя папки и файлы, названия которых начинаются на "Do":

**ls ~ | grep "Do"**

Основные элементы регулярных выражений:

**^:** указывает, что соответствие шаблону должно находиться в начале строки. Например, найдем строки, которые начинаются на символ "s":

**ls ~ | grep "^s"**

**$:** указывает, что соответствие шаблону должно находиться в конце строки. Например, найдем все файлы с расширением "txt":

**ls ~ | grep "\.txt$"**

**.:** соответствует любому символу.

Найдем все строки, которые содержат "Tim" или "Tom":

grep "T.m" test.txt

**[]:** указывает символы в регулярном выражении, которые следует интерпретировать как альтернативы. Например, найдем все строки, которые содержат "Tum" или "Tom":

**grep "T[ou]m" test.txt**

**[^ ]:** игнорирует все символы после знака ^. Например, найдем все строки, которые НЕ содержат "Tum" или "Tom":

**grep "T[^ou]m" test.txt**

**-:** определяет диапазон символов, которому должен соответствовать символ. Например, найдем все строки, которые начинаются с цифрового символа:

**grep "^0-9" test.txt**

**\:** позволяет экранировать спец. символы. Например, символ точки . рассматривается как заменитель произвольного символа. Но если нам надо указать, что текст должен содержать именно точку как стандартный символ, то нам надо использовать "\.". Например, найдем в папке загрузок все файлы, которые имеют расширение tar.gz:

**ls ~/Downloads | grep "\.tar\.gz"**

**\*:** указывает, что предыдущее регулярное выражение может встречаться один раз, несколько раз или не встречаться вообще

**?:** указывает, что предыдущее регулярное выражение может встречаться один раз или может отсутствовать

**+:** указывает, что предыдущее регулярное выражение может встречаться один или более раз

**\{n\}:** указывает, что предыдущий символ может встречаться как минимум n раз. Например, найдем строки, где есть два буквы o подряд:

**grep "o\{2\}" test.txt**

Утилита **awk** представляет еще один мощный инструмент для работы с текстовыми файлами, предоставляя множество возможностей. Команда awk имеет следующий формат:

**awk '/pattern/{action}' file**

Сначала команде **awk** передается параметр **pattern** - шаблон (регулярное выражение), которому должен сооответствовать текст. Обычно шаблоны заключаются между косыми чертами. Далее идет параметр **action**, который задает действия над найденным соответствием. Действия заключаются в фигурные скобки. Поскольку **awk** работает с регулярными выражениями, то шаблон разумно помещать в одинарные кавычки, чтобы оболочка случайно не интерпретировала их.

Если шаблон не указан, то действие выполняется над каждой строкой файла. Если действие не указано, awk просто показывает строки, соответствующие шаблону.

Условно говоря, каждая строка содержит данные пользователя - имя, возраст, компанию, то есть три поля. И каждое поле отделено от других двоеточием. То есть каждая строка имеет формат имя:возраст:компания

Допустим, нам надо получить все строки, где имя пользователя - "Tom":

**awk '/Tom/' users.txt**

Утилита **awk** предоставляет ряд интересных возможностей, в частности, **awk** позволяет работать с файлами в табличной структурой, как наш файл. Например, для **awk** разделителем полей в строке по умолчанию является пробел. Однако в нашем файле таким разделителем является двоеточие. С помощью опции **-F** можно установить символ, который будет применяться в качестве разделителя.

Например, выведем только компанию, в которой работают пользователи с именем "Tom":

**awk -F : '/Tom/{print $3}' users.txt**

Здесь опции -F передается разделитель - двоеточие. Кроме того, здесь добавлено действие - встроенная команда awk - print, которая вывыодит передаенное ей значение. Ей передается значение $3 - это значение 3-го столбца. В нашем случае 1-й столбец представляет имя, 2-й столбец - возраст и 3-й - компанию. Специальное значение - $0 означает всю строку.

Если бы мы хотели вывести компанию для всех пользователей, то можно было опустить шаблон:

**awk -F : '{print $3}' users.txt**

Поскольку awk может ссылаться на определенные поля, можно также сравнивать поля друг с другом. Для этой цели доступны следующие операторы:

==: Равно (ищет поле с тем же значением)

!=: Не равно

<: меньше, чем

<=: меньше или равно

>: Больше, чем

>=: больше или равно

С помощью этих операторов вы можете выполнять некоторые вычисления с текстовыми файлами. Например, выведем пользователей, у которых возраст больше 30:

**awk -F : '$2 > 30 {print $1}' users.txt**

**gunzip и gzip**

Команды **gunzip** (GNU Unzip) и **gzip** (GNU Zip) используются для сжатия и восстановления файлов. Команда **gzip** используется для сжатия файлов и создания сжатого файла с расширением **.gz**, а также для восстановления исходного файла, а команда **gunzip** используется только для восстановления этих файлов **.gz**.

Команда **gzip** имеет следующую форму:

**gzip [OPTION]... [FILE]...**

Команде передается набор опций и набор файлов/каталогов для сжатия/восстановления. Например:

**gzip book.pdf**

В данном случае архивируем файл "book.pdf". По умолчанию архивируемый файл удаляется, а вместо него создается сжатый, который называется как оригинальный, только с расширением **.gz**. В моем случае это файл "book.pdf.gz".

Аналогично можно сжать сразу несколько файлов:

**gzip book.pdf image.png**

В этом случае для каждого файла будет создаваться свой сжатый файл.

Чтобы оригинальный файл не удалялся, применяется опция **-k**

**gzip -k book.pdf**

**gzip** также предоставляет еще ряд опций, из которых следует отметить следующие:

-d, --decompress: распаковать архив

-l, --list: выводит список заархивированных файлов и каталогов

-r, --recursive : рекурсивно архивирует содержимое каталогов

-1, --fast: применяет быстрое архивирование (в ущерб размеру)

-9, --best: применяет лучшее архивирование с упором на наименьший размер (в ущерб скорости)

Например, посмотрим информацию о созданном архиве с помощью опции **-l**:

**gzip -l book.pdf.gz**

Восстановление вышесозданного архива "book.pdf.gz":

**gzip -d book.pdf.gz**

При этом оригинальный архив также удаляется. Чтобы его сохранить также применяется опция **-k**

**gzip -k -d book.pdf.gz**

И также можно распаковать с помощью команды **gunzip**, которой просто передается имя архива. Если файл уже существует, то команда спросит, надо ли перезаписывать его:

**gunzip -k book.pdf.gz**

По умолчанию **gunzip** также удаляет файл архива после распаковки. Чтобы этого не произошло, команде также передается опция **-k**

**bzip2** и **bunzip2.**

Команда **bzip2** используется для сжатия файла (применяет алгоритм сжатия текста с сортировкой блоков Берроуза-Уиллера (Burrows-Wheeler) с последующим кодированием Хаффмана), а команда **bunzip2** - для восстановления файла, сжатого с помощью **bzip2**. Создаваемые архивы имеют расширение .**bz2**. Считается, что **bzip2** чуть более эффективнее сжимает, чем **gzip** - сжатые файлы получаются примерно на 10% меньше

Команде **bzip2** передается набор опций и файлов для сжатия. Основные опции утилиты **bzip2**:

-d --decompress: выполняет распаковку

-z --compress: выполняет сжатие

-v --verbose: выводит сводку об операции

-k --keep: сохраняет (не удаляет) сжимаемые файлы

-f --force: перезаписывает уже существующие компоненты в сжатом файле

-t --test: проверяет целостность сжатого файла

-1 .. -9: уровень сжатия, где 1 означает наибыстрое сжатие, но ниабольный размер выходного файла. Тогда как 9 означает масимальное сжатие - наименьший размер, но время сжатия может увеличиться. Значения от 2 до 8 позволяют применить промежуточные уровни.

--fast: аналог -1

--best: аналог -9

Например, сожмем два файла:

**bzip2 -k book.pdf image.png**

Здесь сжимаются файлы "book.pdf" и "image.png". И свой архив создается для каждого входного файла. То есть на выходе получим файлы "book.pdf.bz2" и "image.png.bz2".

По умолчанию сжимаемые файлы удаляются. И чтобы их сохранить, применяется опция -k

**bzip2 -k book.pdf image.png**

Чтобы увидеть подробную сводку по сжатию, применяется опция -v:

**bzip2 -v book.pdf image.png**

Используя опцию -d, можно восстановить ранее сжатые файлы:

**bzip2 -d -v book.pdf.bz2 image.png.bz2**

Команда **bunzip2** используется для восстановления файлов, сжатых с помощью **bzip2**. Когда сжатый файл распаковывается с помощью **bunzip2**, исходный файл восстанавливается с тем же именем и путем к каталогу, что и сжатый файл, но без расширения .**bz2**. Стоит отметить, что команда **bunzip2** также может распаковывать файлы, сжатые с помощью команды **gzip**.

Использование **bunzip2** во многом аналогично **bzip2** за тем исключением, что по умолчанию идет не сжатие, а восстановление файлов. Например, восстановим ранее сжатые файлы:

**bunzip2 book.pdf.bz2 image.png.bz2**

**XZ** — это инструмент командной строки, используемый для сжатия и распаковки данных. Он основан на алгоритме цепочки Лемпеля-Зива-Маркова (Lempel-Ziv-Markov или LZMA), который известен своей высокой степенью сжатия и низким использованием памяти. **XZ** используется для создания сжатых архивов больших файлов и каталогов, особенно в системах Linux. Он также используется для сжатия ядра Linux во время установки. **XZ** обеспечивает значительное уменьшение размера файла при сохранении целостности исходного файла. Кроме того, **XZ** можно использовать для сжатия и распаковки файлов «на лету», что полезно в некоторых сценариях.

Общий формат команды:

**xz [OPTION]... [FILE]...**

Вначале команде могут передаваться опции, а затем набор файлов. Команда принимает следующие опции:

-z, --compress: сжатие файлов

-d, --decompress: восстановление файлов

-t, --test: проверка целостность сжатого файла

-l, --list: выводит информацию о файлах в архиве

-k, --keep: сохраняет (не удаляет) входные файлы

-f, --force: принудительно перезаписывает выходной файл

-0...-9: задает уровень сжатия; по умолчанию — 6

-e, --extreme: попытаться улучшить степень сжатия, используя больше процессорного времени; не влияет на требования к памяти декомпрессора

-T, --threads=NUM: устанавливает максимальное число потоков в NUM; значение по умолчанию — 1; установите значение 0, чтобы использовать столько потоков, сколько имеется ядер процессора.

-v, --verbose: выводит подробную сводку операции

Например, сожмем файлы:

**xz -v book.pdf image.png**

В итоге для каждого файла создается свой сжатый файл с расширением .xz. Таким образом, в данном случае для файла "book.pdf" создается файл "book.pdf.xz", а для файла "image.png" - файл "image.png.xz"

Для распаковки сжатых файлов применим флаг -d:

**xz -d -v book.pdf.xz image.png.xz**