Права доступа к файлам и каталогам

Чтобы получить доступ к файлу или каталогу или изменить его, необходимо установить правильные разрешения. Существует три различных разрешения, которые могут быть помещены в файл или каталог: чтение, запись и выполнение.

Способ применения этих разрешений отличается для файлов и каталогов, как показано на приведенной ниже диаграмме:

Разрешение	Эффект для файла	Эффект для каталога
read (r)	Позволяет чтение или копирования содержимого файла.	Без разрешения на выполнение для каталога, допускает не детальный список файлов. С разрешением на выполнение ls –l может предоставит подробный список.
write (w)	Позволяет изменять или перезаписывать содержимое. Позволяет добавлять или удалять файлы из каталога.	Чтобы это разрешение работало, каталог также должен иметь разрешение на выполнение.
execute (x)	Позволяет запускать файл как процесс, хотя для файлов сценариев также требуется разрешение на чтение.	Позволяет пользователю перейти в каталог, если родительские каталоги также имеют разрешение на запись.

При выводе файла с помощью команды **ls -l** вывод включает информацию о правах доступа:

Ниже приводится переполнение полей, связанных с разрешениями.

```
-rwxr-xr-x. 1 root root 118644 Apr  4  2013 /bin/ls
```

Первый символ этого вывода указывает тип файла. Если первый символ -, то это обычный файл. Если бы символ был **d**, это был бы каталог.

Если первым символом вывода команды ls -l является l, то файл является символической ссылкой. Файл символьной ссылки «указывает» на другой файл в файловой системе.

Чтобы получить доступ к файлу, на который указывает символическая ссылка, необходимо иметь соответствующие разрешения как для файла символической ссылки, так и для файла, на который он указывает.

Поле разрешений

```
-rwxr-xr-x. 1 root root 118644 Apr  4  2013 /bin/ls
```

После символа типа файла отображаются разрешения. Разрешения разбиты на три набора по три символа:

rwxr-xr-x: первый набор для пользователя, которому принадлежит файл.

rwx**r-x**r-x: второй набор для группы, которой принадлежит файл.

rwxr-x**r-x**: последний набор для всех остальных, что означает «любой, кто не является пользователем, которому принадлежит файл, или член группы, которой принадлежит файл».

Владелец файла

```
-rwxr-xr-x. 1 root root 118644 Apr  4  2013 /bin/ls
```

После подсчета ссылок отображается владелец файла.

Поле владельца группы

```
-rwxr-xr-x. 1 root root 118644 Apr  4  2013 /bin/ls
```

После поля владельца пользователя отображается владелец группы файлов.

Смена собственника

Смена владельца пользователя

Первоначально владельцем файла является пользователь, который его создает. Владелец может быть изменен только пользователем с привилегиями **root** с помощью команды **chown**. Выполнив **chown <newuser> pathname**, пользователь **root** может изменить владельца файла **<pathname>** на учетную запись **<newuser>.**

Смена владельца группы

Когда пользователь создает файл или каталог, его основная группа обычно будет владельцем группы. Команда **id** показывает личность текущего пользователя, текущую первичную группу и все членства в группах.

Чтобы создать файл или каталог, который будет принадлежать группе, отличной от вашей текущей основной группы, один из вариантов - изменить текущую основную группу на другую, к которой вы принадлежите, с помощью команды **newgrp**. Например, чтобы изменить текущую начальную группу со `student` на среднюю группу, называемую `circles`, выполните команду: **newgrp `circles`.** После выполнения этой команды любые новые созданные файлы или каталоги будут принадлежать группе **`circles`**.

Команда **newgrp** открывает новую оболочку и назначает первичную группу для этой оболочки указанной группе. Чтобы вернуться к основной группе по умолчанию, используйте команду **exit**, чтобы закрыть новую оболочку, запущенную командой **newgrp**.

Примечание. Вы должны быть участником группы, в которую хотите перейти, кроме того, администраторы могут защитить группы паролем.

Другой вариант заключается в том, чтобы пользователь-владелец файла мог изменить владельца группы с помощью команды **chgrp**. Например, если вы забыли изменить свою основную группу на `circles` до того, как создали **myfile**, вы можете выполнить команду **chgrp circle myfile**.

Команда **chgrp** не меняет текущую первичную группу, поэтому при создании большого количества файлов более эффективно просто использовать команду **newgrp** вместо выполнения команды **chgrp** для каждого файла.

Примечание. Только владелец файла и пользователь root могут изменять групповое владение файлом.

Разрешения

Информация о пользователе и владельце группы является важным фактором при определении того, какие разрешения будут эффективны для конкретного пользователя. Чтобы определить, какой набор применяется к конкретному пользователю, сначала проверьте личность пользователя. Используя команду **whoami** или команду **id**, вы можете отобразить вашу личность.

Чтобы понять, какие разрешения будут применяться к вам, рассмотрите следующее:

rwxr-xr-x: если ваша текущая учетная запись является владельцем файла, тогда будет применен первый набор из трех разрешений, а остальные разрешения не будут действовать.

rwx**r-x**r-x: если ваша текущая учетная запись не является владельцем файла и вы являетесь участником группы, которой принадлежит файл, то будут применяться разрешения группы, а другие разрешения не будут действовать.

rwxr-x**r-x**: если вы не являетесь пользователем, которому принадлежит файл, или не являетесь участником группы, которой принадлежит файл, к вам применяется третий набор разрешений. Этот последний набор разрешений известен как разрешения для «других»; иногда его называют «разрешениями на мир».

Например, используйте команду **ls -l /etc** и проверьте отображаемые разрешения.

Ответьте на следующие вопросы:

Кто владеет файлами в каталоге / etc?

Могут ли члены корневой группы изменять (записывать) эти файлы?

Могут ли пользователи, не являющиеся root, и члены корневой группы изменять эти файлы?

Например, рассмотрим следующий вывод команды **ls -l:**

```
-r--rw-rwx. 1 bob staff 999 Apr 10 2013 /home/bob/test
```

В этом случае пользователь **bob** получает меньше доступа к этому файлу, чем члены группы **staff** или все остальные. Пользователь **bob** имеет только права доступа r--. Неважно, является ли Боб членом группы **staff**; после того, как право собственности пользователя установлено, применяются только разрешения владельца пользователя.

Изменение основных прав доступа к файлам

Команда **chmod** используется для изменения прав доступа к файлу или каталогу. Только пользователь root или пользователь, которому принадлежит файл, могут изменять права доступа к файлу.

С помощью команды **chmod** существует два метода изменения прав доступа: символьный и восьмеричный (также называемый числовым методом). Символьный метод удобен для изменения одного набора разрешений за раз. Восьмеричный или числовой метод требует знания восьмеричного значения каждого из разрешений и требует, чтобы каждый раз указывались все три набора разрешений (пользователь, группа, другие). При изменении более чем одного набора разрешений восьмеричный метод, вероятно, является лучшим методом.

Символический метод

Чтобы использовать символический метод **chmod**, необходимо использовать следующие символы для представления того, какой набор разрешений вы меняете:

Символ	Значение
u	пользователь: пользователь, которому принадлежит файл
g	группа: группа, которой принадлежит файл

Символ	Значение
o	другие: люди, отличные от владельца пользователя или члена группы
a	все: для обозначения пользователя, группы и других

Используйте вышеуказанные символы для того, кого вы указываете вместе с символом действия:

Символ	Значение
+	Добавить разрешение, если необходимо
=	Указать точное разрешение
-	Удалить разрешение, если это необходимо

После символа действия необходимо указать одно или несколько разрешений (r = чтение, w = запись и x = выполнение), затем пробел и пути к файлам для назначения этих разрешений. Чтобы указать разрешения для более чем одного набора, используйте запятые (и без пробелов) между каждым набором.

Примеры

Добавить разрешение на выполнение для владельца пользователя:

```
chmod u+x myscript
```

Удалить разрешение на запись от владельца группы:

```
chmod g-w file
```

Назначьте другим пользователям только разрешение на чтение, удалите разрешение на запись у владельца группы и добавьте разрешение на выполнение для владельца пользователя:

```
chmod o=r,g-w,u+x myscript
```

Назначьте всем без разрешения:

```
chmod a=- file
```

В следующем примере файл создается с помощью команды **touch**, а затем его разрешения изменяются с помощью символического метода:

```
touch sample
ls -l sample
-rw-rw-r-- 1 root root 0 Oct 17 18:05 sample
chmod g-w,o-r sample
ls -l sample
-rw-r----- 1 root root 0 Oct 17 18:05 sample
```

Восьмеричный метод

Использование восьмеричного метода требует указания разрешений для всех трех наборов. Для этого сложите восьмеричное значение для разрешения на чтение, запись и выполнение для каждого набора:

Разрешение	Восьмеричное значение
read (`r`)	`4`
write (`w`)	`2`
execute (`x`)	`1`

Например, установите права доступа к файлу rwxr-xr-x. Используя значения, найденные в предыдущей таблице, получим 7 для разрешения владельца пользователя, 5 для прав владельца группы и 5 для других:

Эти цифры легко получить, просто сложив восьмеричное значение для указанных разрешений. Таким образом, для чтения, записи и выполнения добавьте 4 + 2 + 1, чтобы получить 7. Или, для чтения, а не записи и выполнения, добавьте 4 + 0 + 1, чтобы получить 5.

Примеры

Измените разрешения на rwxrw-r--

```
chmod 764 myscript
```

Измените разрешения на rw-r - r--:

```
chmod 644 myfile
```

Измените разрешения на rwxr - r--:

```
chmod 744 myscript
```

Измените разрешения на ---------:

```
chmod 000 myfile
```

Что можно сделать с файлом, если для него нет разрешений? Владелец файла всегда может использовать команду **chmod** в какой-то момент в будущем для предоставления прав доступа к файлу. Кроме того, с правами записи и выполнения **-wx** в каталоге, содержащем этот файл, пользователь также может удалить его с помощью команды **rm.**

В следующем примере разрешения образца файла, созданного ранее, изменяются с использованием восьмеричного метода:

```
ls -l sample
-rw-r----- 1 root root 0 Oct 17 18:05 sample
chmod 754 sample
ls -l sample
-rwxr-xr-- 1 root root 0 Oct 17 18:05 sample
```

Сценарии разрешений

Прежде чем обсуждать изменение или настройку разрешений, необходимо понять, как на самом деле работают разрешения. Многие начинающие пользователи Linux зацикливаются на кратких описаниях «read», «write» и «execute», но эти слова не определяют, что пользователь может делать с файлом или каталогом.

Чтобы понять, как ведут себя разрешения, будет представлено несколько сценариев. При рассмотрении этих сценариев имейте в виду следующую таблицу из предыдущего слайда:

Разрешение	Эффект для файла	Эффект для каталога
read (r)	Позволяет чтение или копирования содержимого файла.	Без разрешения на выполнение для каталога, допускает не детальный список файлов. С разрешением на выполнение ls –l может предоставит подробный список.
write (w)	Позволяет изменять или перезаписывать содержимое. Позволяет добавлять или удалять файлы из каталога.	Чтобы это разрешение работало, каталог также должен иметь разрешение на выполнение.
execute (x)	Позволяет запускать файл как процесс, хотя для файлов сценариев также требуется разрешение на чтение.	Позволяет пользователю перейти в каталог, если родительские каталоги также имеют разрешение на запись.

Для каждого сценария ниже представлена диаграмма, описывающая иерархию каталогов. На этой диаграмме информация о ключах предоставляется для каждого файла и каталога. Например, рассмотрим следующее:

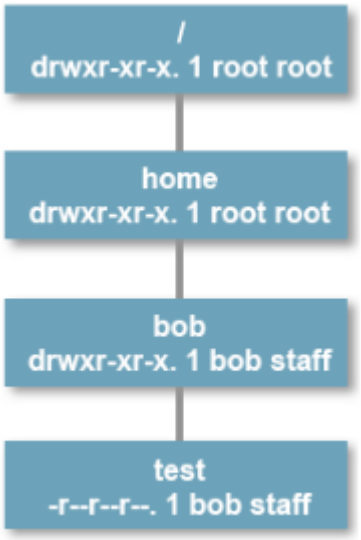
Первое поле описывает **каталог /.** Права доступа: rwxr-xr-x, владелец пользователя - root, а владелец группы - root.

Второй блок описывает **каталог etc**, который является подкаталогом в каталоге /. Третий блок описывает файл **passwd**, который находится в каталоге etc.

Сценарий № 1

Вопрос: Учитывая следующую диаграмму, что пользователь bob может сделать с файлом **/home/bob/test**?

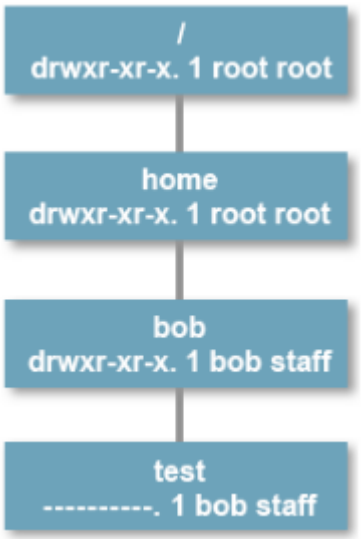
Ответ: пользователь bob может просматривать и копировать файл, потому что у этого пользователя есть разрешения **r--** на файл.



Сценарий № 2

Вопрос: Принимая во внимание следующую диаграмму, может ли пользователь bob удалить файл **/home/bob/test**?

Ответ: да. Хотя может показаться, что пользователь bob не сможет удалить этот файл, поскольку у него нет прав доступа к самому файлу, для удаления файла пользователю необходимо иметь разрешение на запись в каталог, в котором хранится файл. Права доступа к файлу не используются при удалении файла.



Сценарий № 3

Вопрос: Учитывая приведенную ниже диаграмму, может ли пользователь sue просмотреть содержимое файла **/home/bob/test**?

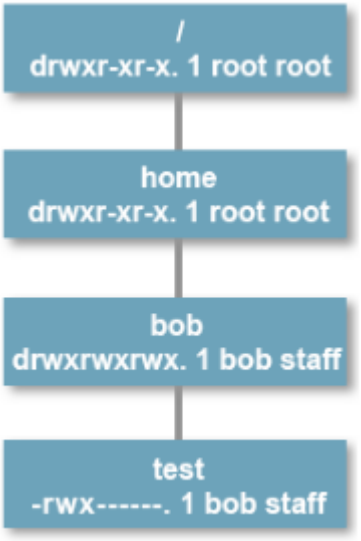
Ответ: Нет. Изначально может показаться, что пользователь sue должен иметь возможность просматривать содержимое файла **/home/bob/test,** поскольку все пользователи имеют разрешение на чтение этого файла.

Однако при проверке разрешений важно проверить разрешения для каталогов «над» рассматриваемым файлом. В этом случае только пользователь bob может «войти» в каталог **/home/bob**, потому что только пользователь **bob** имеет разрешение на выполнение этого каталога. Если вы не можете войти в каталог, не имеет значения, какие разрешения для файлов в этом каталоге установлены.

Сценарий № 4

Вопрос: Принимая во внимание следующую диаграмму, кто может удалить файл /home/bob/test?

Ответ: Все пользователи в системе могут. Напомним, что **для удаления** файла пользователю **необходимы** разрешения **на запись и выполнение в каталоге**, в котором хранится файл. Все пользователи имеют права на запись и выполнение в каталоге **/home/bob**. Это огромная ошибка, потому что теперь все пользователи могут удалять все файлы в домашнем каталоге Боба.



Изменение расширенных прав доступа к файлам

Следующие разрешения считаются расширенными, поскольку обычно они устанавливаются только администратором (пользователем root) и выполняют очень специализированные функции. Их можно установить с помощью команды **chmod**, используя символьный или восьмеричный метод.

Permission	Symbol	Octal Value	Purpose
setuid on a file	An `s` replaces the `x` for the user owner permissions Set with `u+s`	`4000`	Заставляет исполняемый файл выполняться под именем владельца пользователя вместо пользователя, выполняющего команду.
setgid on a file	An `s` replaces the `x` for the group owner permissions Set with `g+s`	`2000`	Заставляет исполняемый файл выполняться под именем владельца группы вместо пользователя, выполняющего команду.
setgid on a directory	An `s` replaces the `x` for the group owner permissions Set with `g+s`	`2000`	Заставляет новые файлы и каталоги, созданные внутри, принадлежать группе, которой принадлежит каталог.
sticky on a directory	**A `t` replaces the `x` for the others permissions** **Set with `o+t`**	`1000`	**Обеспечивает возможность удаления файлов внутри каталога только владельцем или пользователем root.**

Ведущий 0, например 0755, удалит все специальные разрешения из файла или каталога.

Разрешение setuid

Разрешение **setuid** используется для исполняемых файлов, чтобы позволить пользователям, которые не являются пользователем **root**, выполнять эти файлы, как если бы они были пользователем **root**.

Например, команда **passwd** имеет разрешение **setuid**. Команда **passwd** изменяет файл **/etc/shadow**, чтобы обновить значение пароля пользователя. Этот файл обычно не может быть изменен обычным пользователем; на самом деле обычные пользователи обычно не имеют прав доступа к файлу.

Поскольку команда **/usr/bin/passwd** принадлежит корневому пользователю и имеет разрешение **setuid**, она выполняется с «личным состоянием дуэли», что позволяет ей получать доступ к файлам как от лица, выполняющего команду, так и от имени пользователя **root**. Когда команда **passwd** пытается обновить файл **/etc/shadow**, она использует учетные данные пользователя **root** для изменения файла (термин «учетные данные» похож на «полномочия»).

```
ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 47032 Feb 17  2014 /usr/bin/passwd
```

Обратите внимание на листинг, четвертый символ - это **s**, где обычно будет x, если этот файл просто исполняемый. Когда этот символ представляет собой строчные буквы **s**, это означает, что установлены разрешения **setuid** и **execute**. Заглавная буква **S** в позиции четвертого символа означает, что файл не имеет разрешения на выполнение, только разрешение **setuid**. Без разрешения на выполнение для владельца пользователя разрешение **setuid** неэффективно.

Назрешение setgid

На файл

Разрешение **setgid** для файла работает очень похоже на разрешение **setuid**, за исключением того, что вместо выполнения в качестве пользователя, которому принадлежит файл, разрешение **setgid** будет выполняться как группа, владеющая файлом. Далее отображается исполняемый файл **setgid /usr/bin/wall**, если он указан командой **ls -l**:

```
ls -l /usr/bin/wall
-rwxr-sr-x 1 root tty 19024 Jun  3 20:54 /usr/bin/wall
```

Обратите внимание на листинг, седьмой символ - это символ **s,** где обычно есть x для разрешения на выполнение **группы**. Строчные **s** указывают, что у этого файла есть и **setgid**, и набор разрешений на выполнение. **S** вместо **s** означает, что у файла нет разрешения на выполнение для группы. Без разрешения на выполнение для группы разрешение **setgid** будет неэффективным.

Способ, которым ядро Linux ограничивает исполняемые файлы **setuid** и **setgid,** более строг, чем некоторые реализации UNIX. Хотя Linux не запрещает пользователям устанавливать эти разрешения для файлов сценариев, ядро Linux **не будет выполнять сценарии** с разрешениями **setuid** или **setgid**.

Чтобы эффективно установить эти разрешения, ядро Linux будет учитывать их только для исполняемых файлов в двоичном формате. Это повышает безопасность системы.

Для каталога

В большинстве случаев после завершения установки дистрибутива Linux файлы, для которых требуется разрешение **setuid** и **setgid**, должны автоматически устанавливать эти разрешения. Хотя некоторые права **setuid и setgid** для файлов могут быть удалены из

соображений безопасности, очень редко создаются новые файлы setuid или setgid. С другой стороны, администраторы часто добавляют **setgid** в каталоги.

Использование **setgid** для каталога может быть чрезвычайно полезным для пользователей, пытающихся поделиться каталогом файлов с другими пользователями, которые находятся в разных группах. Чтобы понять почему, рассмотрим следующий сценарий.

Пользователи из разных групп в вашей организации попросили администратора создать каталог, в котором они могут обмениваться файлами. В этом примере будет три пользователя: Джо, который является членом группы **staff**, Майя, который является членом группы **payroll**, и Стив, который является членом группы **acct**. Для этого администратор предпринимает следующие шаги:

Шаг №1Ж Создайте новую группу для трех пользователей:

```
groupadd -r common
```

Шаг № 2: Добавьте пользователей в группу:

```
usermod -aG common joe
usermod -aG common maya
```

Шаг № 3: Создайте каталог и сделайте общим владельца группы:

```
mkdir /shared
chgrp common /shared
```

Шаг № 4: Сделать каталог доступным для записи для группы:

```
chmod 770 /shared
```

Это решение почти идеально: пользователи joe, maya и steve теперь могут получить доступ к каталогу /shared и добавлять новые файлы. Однако есть потенциальная проблема: например, когда пользователь joe создает новый файл в каталоге **/shared**, он, скорее всего, будет выглядеть следующим образом при отображении с помощью команды ls -l:

```
-rw-rw----. 2 joe staff 8987 Jan 10 09:08 data.txt
```

Проблема в том, что ни maya уи steve не являются членами группы staff. В результате их разрешения ---, что означает, что они не могут получить доступ к содержимому этого файла.

Пользователь **joe** мог использовать команду **newgrp** для переключения в общую группу перед созданием файла. Или, после создания файла, пользователь **joe** мог использовать команду **chgrp,** чтобы изменить владельца группы на общую группу. Тем не менее, пользователи не всегда будут помнить запуск этих команд; некоторые могут даже не знать, что эти команды существуют.

Вместо того, чтобы обучать пользователей использованию команд **newgrp** и **chgrp**, администратор может создать каталог с разрешением **setgid.** Если для директории установлен **setgid**, то все новые файлы, созданные или скопированные в директорию, будут автоматически принадлежать группе, которой принадлежит эта директория. Это означает, что пользователям не нужно использовать команды **newgrp** или **chgrp**, поскольку владение группой будет управляться автоматически.

Таким образом, лучшим решением для этого сценария будет использование следующей команды для шага № 4:

```
chmod 2775 /shared
```

Вывод сведений о каталоге после выполнения предыдущей команды приведет к следующему выводу:

```
drwxrwsr-x. 2 root common 4096 Jan 10 09:08 /shared
```

После выполнения этих шагов пользователи joe, maya и steve теперь смогут легко создавать файлы в каталоге **/shared**, которые будут автоматически принадлежать группе common.

Разрешение sticky bit

Последним расширенным разрешением для обсуждения является разрешение sticky bit", которое иногда называют «закреплением». Установка этого разрешения может быть важна для **предотвращения удаления пользователями файлов других пользователей**.

Напомним, что для возможности удаления файла необходимы только разрешения на запись и выполнение в каталоге. Разрешение на запись и выполнение также необходимо для добавления файлов в каталог. Таким образом, если администратор хочет создать каталог, куда любой пользователь может добавить файл, требуемые разрешения означают, что пользователь также может удалить файл любого другого пользователя в этом каталоге.

Разрешение sticky каталога изменяет разрешение на запись в каталог, так что только определенные пользователи могут удалять файл в каталоге:

- Пользователь, которому принадлежит файл
- Пользователь, которому принадлежит каталог, в котором установлен бит закрепления (обычно это пользователь root)
- Пользователь root

В типичной установке Linux обычно имеет две директории, для которых по умолчанию установлено разрешение sticky: директории **/tmp** и **/var/tmp**. Помимо домашних каталогов пользователей, эти два каталога являются единственными каталогами, которые обычные пользователи имеют разрешение на запись по умолчанию.

Как следует из их названия, эти каталоги предназначены для временных файлов. Любой пользователь может скопировать файлы в каталог **/tmp** или **/var/tmp,** чтобы поделиться ими с другими.

Поскольку каталоги **/tmp** или **/var/tmp** имеют разрешение sticky, не нужно беспокоиться о том, что пользователи удаляют общие файлы. Однако есть причина, по которой эти каталоги считаются временными: эти каталоги автоматически очищают свое содержимого по регулярному расписанию. Если вы поместите в них файл, он будет удален автоматически в будущем.

Когда каталог имеет разрешение sticky, символ **t** заменяет **x** в наборе разрешений для других. Например, ниже показан вывод команды ls -ld /tmp:

```
ls -ld /tmp
```

Каталог **/tmp** имеет восьмеричный режим разрешений **1777**, или полные разрешения для всех, плюс закрепленное разрешение для каталога. Если набор разрешений «другие» включает в себя разрешение на выполнение, то **t** будет в нижнем регистре. Если есть верхний регистр **T**, это означает, что разрешение на выполнение не предоставляется другим. Это не означает, что каталог не имеет действующего разрешения sticky, однако это означает, что пользователи, на которых распространяется разрешение «другие», не могут перейти в этот каталог или создать файлы в каталоге.

Существуют случаи, когда пользователи могут не захотеть выполнять их для других, но все еще имеют разрешение «закрепить бит» в каталоге. Например, при создании общих каталогов с разрешением **setgid**. Рассмотрим следующую команду:

```
mkdir /shared
chmod 3770 /shared
```

Обратите внимание, что специальные разрешения **setgid** и **stickybit** могут быть добавлены вместе, разрешение **2000** плюс разрешение **1000** дают **3000** для специальных разрешений. Добавьте это к основным разрешениям для пользователя, группы и других; Владелец и группа имеют полный доступ, а другие не получают. Вывод списка каталога после внесения изменений с помощью команды **ls -ld /shared** приводит к следующему выводу:

ls -ld /shared

```
root@L-FW:~# ls -ld /shared
drwxrws--T 2 root root 4096 May  9 22:29 /shared
```

Верхний регистр **T** в позиции разрешений на выполнение для других указывает, что для других нет разрешения на выполнение. Однако, поскольку несколько пользователей группы по-прежнему имеют доступ, разрешение закрепления действует для общей группы.

Команда **stat** может отображать разрешения как в символической, так и в восьмеричной нотации, а также информацию о владельце пользователя и владельца группы. Первая строка, которая начинается с Access, демонстрирует это:

stat /shared

```
root@L-FW:~# stat /shared/
  File: /shared/
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 801h/2049d      Inode: 32769       Links: 2
Access: (3770/drwxrws--T)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2019-05-09 22:29:35.602867202 -0400
Modify: 2019-05-09 22:29:35.602867202 -0400
Change: 2019-05-09 22:30:23.462866781 -0400
 Birth: -
```

Разрешения для файлов по умолчанию

В отличие от других операционных систем, где разрешения на новый каталог или файл могут быть унаследованы от родительского каталога, Linux устанавливает разрешения по умолчанию для этих новых объектов на основе значения параметра **UMASK** создателя. Команда **umask** используется как для установки значения **UMASK**, так и для его отображения.

Команда **umask** автоматически выполняется при запуске оболочки. Чтобы иметь постоянную настройку для **UMASK**, в файл **~ /.bashrc** можно добавить пользовательскую команду **umask**.

Чтобы настроить значение UMASK, важно понимать, что делает значение UMASK. Значение значения UMASK влияет только на разрешения, предоставляемые новым файлам и каталогам во время их создания. Это влияет только на основные разрешения для владельца пользователя, владельца группы и других. Значение UMASK не влияет на специальные расширенные права доступа **setuid, setgid** или **sticky bit**.

UMASK - это восьмеричное значение, основанное на тех же значениях, которые были рассмотрены в этом разделе:

read	4
write	2
execute	1
none	0

При использовании команды **chmod** с восьмеричными значениями сложите вместе значения для пользователя, группы и других. Однако восьмеричное значение UMASK используется для указания разрешений, которые будут удалены. Другими словами, **восьмеричное значение,** установленное для **UMASK**, **вычитается из** максимально **возможного разрешения** для определения разрешений, которые устанавливаются при создании файла или каталога.

UMASK для файлов

По умолчанию максимальные разрешения, которые будут помещены в новый файл, равны rw-rw-rw-, которые могут быть представлены в восьмеричном виде как 666. Бит выполнения отключен по соображениям безопасности. Значение UMASK может

использоваться для указания того, какие из этих разрешений по умолчанию удаляются для новых файлов. Будут предоставлены три восьмеричных значения: значение разрешений для удаления для владельца пользователя, владельца группы и других.

Например, чтобы установить значение UMASK для новых файлов, которое привело бы к полным разрешениям для владельца (результат: rw-), удалите или замаскируйте разрешения на запись для владельца группы (результат: r--) и удалите все разрешения от других (результат: ---), вычислите значение UMASK следующим образом:

Первой цифрой для владельца пользователя будет 0, что не будет маскировать какие-либо разрешения по умолчанию.

Вторая цифра будет 2, которая будет маскировать только разрешение на запись.

Третья цифра будет 6, которая будет маскировать права на чтение и запись.

В результате значение UMASK 026 приведет к тому, что новые файлы будут иметь права доступа rw-r -----.

Другой пример: чтобы установить значение UMASK для новых файлов, которые будут удалять разрешения на запись для владельца (результат: r--) и удалять разрешения на чтение и запись для группы и других (результат: ---), рассчитайте значение UMASK как следующим образом:

Первой цифрой для владельца пользователя будет 2, что будет маскировать только разрешение на запись.

Вторая цифра будет 6, которая будет маскировать права на чтение и запись.

Третья цифра будет 6, которая будет маскировать права на чтение и запись.

В результате значение 266 UMASK приведет к тому, что новые файлы будут иметь права r --------.

UMASK для каталогов

Способ применения UMASK к обычным файлам отличается от файлов каталогов. По соображениям безопасности обычные файлы не могут быть исполняемыми во время их создания. Было бы опасно разрешать запуск файла как процесса без явного назначения пользователем разрешения на выполнение. Таким образом, независимо от того, включено ли разрешение на выполнение в значение UMASK, оно не будет применяться к обычным файлам.

Для каталогов разрешение на выполнение крайне важно для правильного доступа к каталогу. Без разрешения на выполнение вы не сможете перейти в каталог, и разрешение на запись не работает. По сути, каталоги не очень полезны вообще без разрешения на выполнение, поэтому новые каталоги могут быть исполняемыми по умолчанию. Разрешения по умолчанию для новых каталогов - rwxrwxrwx или 777.

Например, если вам нужно значение UMASK для новых каталогов, которое приведет к полным разрешениям для владельца пользователя (результат: rwx), удалите или замаскируйте разрешения на запись для группы, в которой состоит владелец (результат: rx) и удалите все разрешения от других (результат : ---), тогда вы можете рассчитать значение UMASK следующим образом:

Первой цифрой для владельца пользователя будет 0, что не будет маскировать какие-либо разрешения по умолчанию.

Вторая цифра будет 2, которая будет маскировать только разрешение на запись.

Третья цифра будет 7, которая будет маскировать права на чтение и запись.

В результате значение UMASK 027 приведет к тому, что новые каталоги будут иметь разрешения r-xr-x---

Очень важно: хотя значение UMASK влияет на разрешения для новых файлов и каталогов по-разному (из-за разных максимальных разрешений), отдельного значения UMASK для файлов и каталогов не существует; одно значение UMASK применяется к

обоим, как вы можете видеть из следующей таблицы часто используемых значений UMASK:

umask	File Permissions	Directory Permissions	Description
002	664 or rw-rw-r--	775 or rwxrwxr-x	По умолчанию для обычных пользователей
022	644 or rw-r--r--	755 or rwxr-xr-x	По умолчанию для пользователя root
007	660 or rw-rw----	770 or rwxrwx---	Нет доступа для других
077	600 or rw-------	700 or rwx------	Приватный для пользователя

Следующие команды отобразят текущее значение UMASK, установят другое значение и отобразят новое значение: