Изучаем Linux, 101: Управление разрешениями и правами владения файлов

Защита файлов и директорий

Ян Шилдс (ishields@us.ibm.com)
Старший программист
IBM

01.12.2011

Из этой статьи вы узнаете, как управлять разрешениями и правами владения файлов в файловых системах Linux®. Вы узнаете о различных режимах доступа (suid, sgid и бит закрепления в памяти) и о том, как использовать их для защиты файлов и директорий. Вы можете использовать этот материал для подготовки к экзамену LPI 101 программы сертификации на администратора Linux начального уровня или просто для общего развития.

Больше статей из этой серии

Об этой серии

Эта серия статей поможет вам освоить задачи администрирования операционной системы Linux. Вы также можете использовать материал этих статей для подготовки к экзаменам первого уровня сертификации профессионального института Linux (LPIC-1).

Чтобы посмотреть описания статей этой серии и получить ссылки на них, обратитесь к нашему перечню материалов для подготовки к экзаменам LPIC-1. Этот перечень постоянно дополняется новыми статьями по мере их готовности и содержит самые последние (по состоянию на апрель 2009 года) цели экзаменов сертификации LPIC-1. Если какая-либо статья отсутствует в перечне, можно найти ее более раннюю версию, соответствующую предыдущим целям LPIC-1 (до апреля 2009 года), обратившись к нашим руководствам для подготовки к экзаменам института Linux Professional Institute.

Краткий обзор

Из этой статьи вы узнаете, как управлять доступом к файлам и директориям с помощью прав владения и разрешений. Вы научитесь:

- Управлять правами доступа к обычным и к специальным файлам, а также к директориям.
- Обеспечивать защиту, используя режимы доступа suid, sgid и бит закрепления в памяти sticky bit.

- Изменять маску создания файлов.
- Предоставлять доступ к файлам членам групп.

Все примеры этой статьи (если это не оговаривается отдельно) выполнены в операционной системе Fedora 13 с версией ядра 2.6.34. Результаты, полученные в других операционных системах, могут отличаться.

Эта статья поможет вам подготовиться к сдаче экзамена LPI 101 на администратора начального уровня (LPIC-1) и содержит материалы цели 104.5 темы 104. Цель имеет вес 3.

Некоторые принципы управления разрешениями и правами владения файлами изложены в статье "Изучаем Linux 101: управление дисковыми квотами". Текущая статья поможет лучше понять их.

Необходимые условия

Чтобы извлечь наибольшую пользу из наших статей, необходимо обладать базовыми знаниями о Linux и иметь работоспособный компьютер с Linux, на котором можно будет выполнять все встречающиеся команды. Иногда различные версии программ выводят результаты по-разному, поэтому содержимое листингов и рисунков может отличаться от того, что вы увидите на вашем компьютере.

Пользователи и группы

Как связаться с Яном

Ян – один из наших наиболее популярных и плодовитых авторов. Ознакомьтесь со всеми статьями Яна (EN), опубликованными на сайте developerWorks. Вы можете найти контактные данные в профиле Яна и связаться с ним, а также с другими авторами и участниками ресурса My developerWorks.

Вам уже известно о том, что Linux – это многопользовательская система и каждый пользователь является членом одной *основной* группы и, возможно, одной или нескольких дополнительных групп. Кроме того, можно войти в систему под одной учетной записью обычного пользователя и стать суперпользователем, выполнив команду `su` или `sudo -s`. Принадлежность файлов в Linux, а также права доступа к ним тесно связаны с идентификаторами пользователей и групп, поэтому ниже я приведу основные сведения о пользователях и группах.

Кто я такой?

Если вы не сменили ваш идентификатор пользователя, то он остается тем же, что и при входе в систему. Если вы стали другим пользователем, идентификатор пользователя может отображаться в приглашении, как в большинстве примеров этой статьи. Если идентификатор пользователя не отображается в приглашении, то текущий идентификатор можно узнать с помощью команды `whoami`. В листинге 1 приведено несколько примеров, в которых приглашения командной строки (из переменной окружения PS1) отличаются от приглашений в остальных примерах этой статьи. Если идентификатор пользователя отображается в приглашении командной строки, то это может быть удобно.

Листинг 1. Определение идентификатора текущего пользователя

```
/home/ian$ whoami
tom
/home/ian$ exit
exit
$ whoami
ian
```

В какие группы я включен?

Аналогичным образом можно узнать, в какие группы вы включены, выполнив команду `groups`. Команда `id` выводит информацию как о пользователях, так и о группах. Если к командам `groups` и `id` добавить в качестве параметра идентификатор пользователя, то вы получите информацию о пользователе с этим идентификатором, а не о текущем пользователе. В листинге 2 приведено несколько примеров. Обратите внимание на то, что если для команды `id` не указывать идентификатор пользователя, то помимо основной информации также выводится информация о контексте SELinux.

Листинг 2. Определение членства в группах.

```
[ian@echidna ~]$ id
uid=1000(ian) gid=1000(ian) groups=1000(ian),505(development),8093(editor)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[ian@echidna ~]$ id ian
uid=1000(ian) gid=1000(ian) groups=1000(ian),8093(editor),505(development)
[ian@echidna ~]$ groups
ian development editor
[ian@echidna ~]$ id tom
uid=1012(tom) gid=1012(tom) groups=1012(tom),505(development)
[ian@echidna ~]$ groups tom
tom : tom development
[ian@echidna ~]$ su tom
Password:
[tom@echidna ian]$ groups
tom development
[tom@echidna ian]$ groups ian
ian : ian editor development
```

Принадлежность файлов и права доступа к ним

Каждый пользователь имеет свой идентификатор и является членом одной основной группы, а каждый файл в Linux имеет одного владельца и одну группу, связанные с ним.

Обычные файлы

Чтобы определить пользователя и группу, которым принадлежит файл, используйте команду `ls -l`.

Листинг 3. Определение владельца файла

```
[ian@echidna ~]$ ls -l /bin/bash .bashrc helloworld.C
-rw-r--r--. 1 ian  ian            124 Mar 31  2010 .bashrc
-rwxr-xr-x. 1 root root        943360 May 21  2010 /bin/bash
-rw-rw-r--. 1 ian  development    116 Nov 30 10:21 helloworld.C
```

В этом примере владельцем файла .bashrc является пользователь ian, и этот файл принадлежит группе ian, которая является основной группой этого пользователя.

Аналогично, владельцем файла /bin/bash является пользователь root, и файл принадлежит группе root. Однако владельцем файла helloworld.C является пользователь ian, но этот файл принадлежит группе development. Имена групп и пользователей принадлежат различным пространствам имен, поэтому указанное имя может являться как именем пользователя, так и именем группы. В действительности многие дистрибутивы по умолчанию создают для каждого нового пользователя назначаемую ему группу с таким же названием.

В модели безопасности Linux для каждого объекта файловой системы существует три типа разрешений: чтение (r), запись (w) и выполнение (x). Разрешение на запись также позволяет изменять или удалять объект. Кроме того, эти разрешения указываются отдельно для владельца файла, членов группы файла и для всех остальных.

Обратите внимание на то, что в столбце листинга 3 содержится строка из 11 символов. Одиннадцатый символ был введен не так давно, и я расскажу о нем чуть позже. Первый символ указывает на тип объекта (в этом примере символ "-" означает обычный файл), а остальные девять символов образуют три группы по три символа в каждой. Первая группа содержит разрешения на чтение, запись и выполнение для владельца файла. Символ "-" означает, что соответствующее разрешение не предоставлено. Таким образом, пользователь ian может читать и записывать в файл .bashrc, но не выполнять его, тогда как пользователь root может читать, записывать **и** выполнять файл /bin/bash. Вторая группа содержит разрешения на чтение, запись и выполнение для группы файла. Члены группы development могут читать и записывать в файл helloworld.C (который принадлежит пользователю ian), а все остальные могут только читать его. Аналогично, члены группы root и все остальные могут читать или выполнять файл /bin/bash.

Директории

Для директорий используются те же самые флаги разрешений, что и для обычных файлов, но интерпретируются они по-другому. Пользователи, имеющие разрешение на чтение директории, могут просматривать ее содержимое. Пользователи, имеющие разрешение на запись в директорию, могут создавать и удалять из нее файлы. Пользователи, имеющие разрешение на выполнение, могут входить в эту директорию и во все ее поддиректории. При отсутствии разрешений на выполнение объекты файловой системы, находящиеся в директории, недоступны. При отсутствии разрешений на чтение объекты файловой системы, находящиеся внутри директории, недоступны для просмотра при выводе содержимого директории, но к ним можно получить доступ, зная полный путь. Это демонстрирует пример, приведенный в листинге 4 .

Листинг 4. Разрешения и директории

```
[ian@echidna ~]$ ls -l /home
total 32
drwxr-x---. 38 editor   editor      12288 Nov 30 10:49 editor
drwxr-x---.  4 greg     development  4096 Nov 30 12:44 greg
drwx------. 21 gretchen gretchen     4096 Nov 30 11:26 gretchen
drwxr-xr-x. 41 ian      ian          4096 Nov 30 10:51 ian
drwx------. 21 ianadmin ianadmin     4096 May 28  2010 ianadmin
d-wx--x--x. 21 tom      tom          4096 Nov 30 11:30 tom
[ian@echidna ~]$ ls -a ~greg/.ba*
/home/greg/.bash_history  /home/greg/.bash_profile
/home/greg/.bash_logout   /home/greg/.bashrc
[ian@echidna ~]$ ls -a ~gretchen
ls: cannot open directory /home/gretchen: Permission denied
[ian@echidna ~]$ ls -a ~tom
ls: cannot open directory /home/tom: Permission denied
[ian@echidna ~]$ head -n 3 ~tom/.bashrc
# .bashrc

# Source global definitions
```

Первый символ в подробном выводе содержимого директории описывает тип объекта (`d` означает директорию). Пользователям группы development предоставлены разрешения на чтение и выполнение домашней директории пользователя greg, поэтому пользователи tom и ian могут просматривать ее содержимое. Пользователи группы gretchen и все остальные пользователи не имеют разрешений на чтение или выполнение домашней директории gretchen, поэтому пользователь ian не может получить доступ к ней. Для домашней директории пользователя tom предоставлены разрешения на выполнение, но не на чтение, поэтому пользователь ian не может просмотреть ее содержимое, но может получить доступ к объектам внутри этой директории, точно зная о том, что они существуют.

Другие объекты файловой системы

Вывод команды `ls -l` может содержать объекты файловой системы, отличные от файлов и директорий, что можно определить по первому символу листинга. Более подробно я расскажу них чуть позже, а пока просто познакомьтесь со всеми возможными типами объектов.

Таблица 1. Типы объектов файловой системы

Код	Тип объекта
-	Обычный файл
d	Директория
l	Символическая ссылка
c	Специальное символьное устройство
b	Специальное блочное устройство
p	Буфер FIFO
s	Сокет

Одиннадцатый символ

Одиннадцатый символ в выводе команды `ls` является нововведением, поэтому в некоторых дистрибутивах он пока не применяется. В других случаях одиннадцатым символом может быть пробел, поэтому вы можете не заметить его. Этот символ указывает на использование альтернативного метода доступа к файлу. Если символ, следующий за битами режима файла, является пробелом, альтернативный метод доступа не используется. Если же этот символ является печатным знаком, то используется альтернативный метод доступа. Таким методом может являться, например, список контроля доступов. Символ '.' (точка) команды GNU `ls` означает, что для файла используется только контекст безопасности SELinux. Файлы, для которых используются любые другие комбинации альтернативных методов доступа, отмечены знаком '+' (плюс).

Изменений разрешений

Добавление разрешений

Предположим, вы создали сценарий командной оболочки "Hello world". Как правило, вновь создаваемые сценарии не являются исполняемыми. Чтобы добавить для сценария разрешение на выполнение, используйте команду `chmod` с опцией `+x`, как показано в листинге 5.

Листинг 5. Создание исполняемого сценария командной оболочки

```
[ian@echidna ~]$ echo 'echo "Hello world!"'>hello.sh
[ian@echidna ~]$ ls -l hello.sh
-rw-rw-r--. 1 ian ian 20 Nov 30 13:05 hello.sh
[ian@echidna ~]$ ./hello.sh
bash: ./hello.sh: Permission denied
[ian@echidna ~]$ chmod +x hello.sh
[ian@echidna ~]$ ./hello.sh
Hello world!
[ian@echidna ~]$ ls -l hello.sh
-rwxrwxr-x. 1 ian ian 20 Nov 30 13:05 hello.sh
```

Подобным образом можно использовать опцию `+r` для установки разрешений на чтение и опцию `+w` – для установки разрешений на запись. Фактически, можно использовать любые комбинации опций `r`, `w` и `x`. Например, команда `chmod +rwx` устанавливает для файла разрешения на чтение, запись и выполнение. Запуск `chmod` в такой форме добавляет разрешения, которые еще не были установлены.

Выборочное изменение

Вы могли заметить, что в последнем примере разрешение на выполнение было установлено для владельца, группы **и** всех остальных. Для выборочной установки разрешений можно использовать префиксы `u` (установка разрешений для пользователей), `g` (установка разрешений для групп) и `o` (установка разрешений для всех остальных). Префикс `a` устанавливает разрешения для всех пользователей (что эквивалентно отсутствию какого-либо префикса вообще). В листинге 6 показано, как добавить разрешения на запись и выполнение пользователю и группе другой копии файла сценария.

Листинг 6. Выборочное добавление разрешений

```
[ian@echidna ~]$ echo 'echo "Hello world!"'>hello2.sh
[ian@echidna ~]$ chmod ug+xw hello2.sh
[ian@echidna ~]$ ls -l hello2.sh
-rwxrwxr--. 1 ian ian 20 Nov 30 13:08 hello2.sh
```

Удаление разрешений

Иногда требуется не добавить разрешения, а удалить их. Для этого просто замените `+` на `-`, и все указанные разрешения будут удалены. В листинге 7 показано, как удалить все разрешения для других пользователей с двух файлов сценариев командной оболочки.

Листинг 7. Удаление разрешений

```
[ian@echidna ~]$ ls -l hello*.sh
-rwxrwxr--. 1 ian ian 20 Nov 30 13:08 hello2.sh
-rwxrwxr-x. 1 ian ian 20 Nov 30 13:05 hello.sh
[ian@echidna ~]$ chmod o-xrw hello*.sh
[ian@echidna ~]$ ls -l hello*.sh
-rwxrwx---. 1 ian ian 20 Nov 30 13:08 hello2.sh
-rwxrwx---. 1 ian ian 20 Nov 30 13:05 hello.sh
```

Заметьте, что можно изменять разрешения сразу для нескольких файлов. Как и для других команд, с которыми вы познакомились в руководствах по теме 103, для выполнения рекурсивных действий с файлами и директориями вы даже можете использовать опцию `-R` (или `--recursive`).

Установка разрешений

Теперь, когда вы умеете добавлять и удалять разрешения, пора объяснить, как можно установить только определенный набор разрешений. Для этого вместо знаков `+` или `–` используется знак `=`. Чтобы запретить другим пользователям получать доступ к двум файлам сценариев из последних примеров, вместо команды для удаления разрешений можно использовать команду `chmod o= hello*`.

Если вы хотите установить различные разрешения для пользователя, группы и всех остальных, вы можете разделить различные выражения запятыми, например, `ug=rwx,o=rx`; можно также указывать разрешения в числовом формате, как это будет описано далее.

Представление разрешений в восьмеричном формате

До сих пор для установки разрешений мы использовали символы (ugoa и rxw). В каждой группе существует три типа разрешений. Вместо символов для установки разрешений можно использовать восьмеричные числа. При таком способе установки разрешений может потребоваться использовать до четырех восьмеричных чисел. Первое число мы будем рассматривать при обсуждении атрибутов. Второе число определяет разрешения для пользователя, третье число –разрешения для группы и, наконец, четвертое число определяет разрешения для всех остальных. Каждое из этих трех чисел формируется путем сложения необходимых разрешений: чтение (4), запись (2) и выполнение (1). В примере,

приведенном в листинге 5, сценарий hello.sh был создан с разрешениями -rw-r--r--, что соответствует значению 644 в восьмеричной форме. Установка разрешений на выполнение для всех пользователей и групп без исключения изменит это значение на 755.

Использование числового формата очень удобно тогда, когда вы хотите установить все разрешения сразу, не указывая их для каждой группы. В таблице 2 перечислены все возможные разрешения в числовом формате.

Таблица 2. Numeric permissions

Символическое представление	Восьмеричное значение
`rwx`	`7`
`rw-`	`6`
`r-x`	`5`
`r--`	`4`
`-wx`	`3`
`-w-`	`2`
`--x`	`1`
`---`	`0`

Режимы доступа

Когда вы входите в систему, запускается новый процесс командного интерпретатора, использующий ваши идентификаторы пользователя и группы. Эти идентификаторы управляют вашим доступом ко всем файлам системы. Обычно это означает, что вы не можете получить доступ к чужим, а также к системным файлам. На самом деле мы, будучи пользователями, всецело зависим от других программ, выполняющих различные действия от нашего имени. Поскольку запускаемые вами программы наследуют *ваш* идентификатор, они не могут получить доступ к объектам файловой системы, доступ к которым не был вам предоставлен.

Важным примером является файл /etc/passwd, который не может быть напрямую изменен обычными пользователями, поскольку разрешение на запись в этот файл имеет только пользователь root; тем не менее, обычные пользователи должны иметь возможность так или иначе изменять файл /etc/passwd, когда им требуется сменить свой пароль. Итак, если пользователь не может изменять этот файл, то как это сделать?

Режимы доступа suid и sgid

В модели разрешений Linux имеется два специальных режима доступа, которые называются suid (set user id – установить идентификатор пользователя) и sgid (set group id – установить идентификатор группы). Когда для исполняемой программы установлен режим доступа suid, она запускается с правами владельца файла, а не с правами пользователя, фактически запустившего ее. Аналогично, если для программы установлен режим доступа sgid,

она запускается с правами пользователя, являющегося членом группы файла, а с правами группы запустившего ее пользователя. Эти режимы можно устанавливать как по отдельности, так и вместе.

В листинге 8 показано, что владельцем исполняемого файла `passwd` является пользователь root.

Листинг 8. Режим доступа suid к файлу /usr/bin/passwd

```
[ian@echidna ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 34368 Apr  6  2010 /usr/bin/passwd
```

Обратите внимание на то, что вместо символа `x` в группе разрешений пользователя стоит символ `s`, означающий, что для этой конкретной программы установлен быт suid и бит выполнения. Поэтому программа `passwd` запускается так, как если бы ее запустил пользователь root с полными правами суперпользователя, а не с правами того пользователя, который фактически запустил ее. И поскольку программа `passwd` выполняется с правами доступа пользователя `root`, она может изменять файл /etc/passwd.

В подробном выводе содержимого директории биты suid и sgid находятся на месте бита `x` для пользователя и группы. Если файл является исполняемым, биты suid и sgid (если они установлены) отображаются как `s` (символ в нижнем регистре), в противном случае – как `S` (в верхнем регистре).

Несмотря на то, что режимы доступа suid и sgid удобны, и во многих ситуациях даже необходимы, неправильное использование этих режимов может привести к появлению брешей в защите системы. По возможности следует использовать как можно меньше программ с режимом доступа suid. Команда `passwd` – это одна из немногих команд, для которой **необходимо** использовать режим suid.

Установка битов suid и sgid

Биты suid и sgid устанавливаются и снимаются с использованием символа `s`; например, `u +s` устанавливает режим доступа suid, а `g-s` снимает режим sgid. В восьмеричном формате режиму suid соответствует значение 4 в первой цифре (старший разряд), а режиму sgid – значение 2.

Директории и режим доступа sgid

Когда для директории установлен режим sgid, все созданные в ней файлы и поддиректории будут наследовать идентификатор группы этой директории. Это чрезвычайно полезно для деревьев каталогов, которые используются группой пользователей, работающих над одним проектом. В листинге 9 показано, как пользователь greg может настроить директорию так, чтобы все пользователи группы development могли использовать ее, а также приведен пример того, как пользователь gretchen может создать файл в директории. После создания файла gretchen.txt все члены группы смогут изменять его, поэтому с помощью команды `chmod g-w` gretchen убирает права на запись в этот файл для группы.

Листинг 9. Режим доступа sgid и директории

```
[greg@echidna ~]$ mkdir lpi101
[greg@echidna ~]$ chmod g+ws lpi101
[greg@echidna ~]$ ls -ld lpi101
drwxrwsr-x. 2 greg development 4096 Nov 30 13:30 lpi101/
[greg@echidna ~]$ su - gretchen
Password:
[gretchen@echidna ~]$ touch ~greg/lpi101/gretchen.txt
[gretchen@echidna ~]$ ls -l ~greg/lpi101/gretchen.txt
-rw-rw-r--. 1 gretchen development 0 Nov 30 14:12 /home/greg/lpi101/gretchen.txt
[gretchen@echidna ~]$ chmod g-w ~greg/lpi101/gretchen.txt
[gretchen@echidna ~]$ ls -l ~greg/lpi101/gretchen.txt
-rw-r--r--. 1 gretchen development 0 Nov 30 14:12 /home/greg/lpi101/gretchen.txt
```

Теперь любой член группы development может создавать файлы в директории lpi101 пользователя greg. Как видно из листинга 10, другие члены группы не могут изменять файл gretchen.txt. Тем не менее, они имеют разрешение на запись в директорию и поэтому могут удалить этот файл.

Листинг 10. Режим доступа sgid и владение файлом

```
[gretchen@echidna ~]$ su - tom
Password:
[tom@echidna ~]$ echo "something" >> ~greg/lpi101/gretchen.txt
-bash: /home/greg/lpi101/gretchen.txt: Permission denied
[tom@echidna ~]$ rm ~greg/lpi101/gretchen.txt
rm: remove write-protected regular empty file `/home/greg/lpi101/gretchen.txt'? y
[tom@echidna ~]$ ls -l ~greg/lpi101/
total 0
```

Бит закрепления в памяти

Вы только что увидели, как любой пользователь, имеющий разрешения на запись в директорию, может удалять находящиеся в ней файлы. Это может быть приемлемо для проекта рабочей группы, но не желательно для файлового пространства, находящемся в глобальном общем доступе, например, для директории /tmp. К счастью, существует решение этой проблемы.

Существует еще один, последний бит режима доступа, который называется битом *закрепления в памяти* (sticky bit). В символьном виде он обозначается символом `t`, а в числовом виде – значением 1 в восьмеричной цифре старшего разряда. В подробном выводе содержимого директории бит закрепления в памяти находятся на месте бита выполнения для всех остальных пользователей (последний символ); значение верхнего и нижнего регистра аналогично значениям для suid и sgid. Если бит закрепления в памяти установлен для директории, то удалять файлы или ссылки, находящиеся в этой директории, может только владелец или суперпользователь (root). В листинге 11 показано, как пользователь greg может установить бит закрепления в памяти для своей директории lpi101, а также то, что этот бит установлен для директории /tmp.

Листинг 11. Каталоги с закреплением в памяти

```
[greg@echidna ~]$ chmod +t lpi101
[greg@echidna ~]$ ls -ld lpi101 /tmp
drwxrwsr-t.  2 greg development  4096 Nov 30 14:16 lpi101
drwxrwxrwt. 24 root root        12288 Nov 30 14:22 /tmp
```

Исторически в системах UNIX ® бит закрепления в памяти устанавливался для файлов и означал, что исполняемые файлы следует хранить в пространстве подкачки, чтобы исключить их повторную загрузку. В современных ядрах Linux бит закрепления в памяти, установленный для файлов, игнорируется.

Сводка режимов доступа

В таблице 3 приведены символьное и восьмеричное представления для всех трех режимов доступа, о которых мы говорили.

Таблица 3. Режимы доступа

Режим доступа	Символическое представление	Восьмеричное значение
suid	`s` with `u`	`4000`
sgid	`s` with `g`	`2000`
sticky	`t`	`1000`

Сопоставив эту таблицу с той информацией о разрешениях, которая уже вам известна, можно увидеть, что полное восьмеричное представление разрешений и режимов доступа к папке lpi101 пользователя greg (drwxrwsr-t) – это число 3775. Команда `ls` не отображает разрешения в восьмеричном формате, но это можно сделать с помощью команды `find`, как показано в листинге 12.

Листинг 12. Вывод разрешений в символическом и восьмеричном представлениях

```
[greg@echidna ~]$ find . -name lpi101  -printf "%M %m %f\n"
drwxrwsr-t 3775 lpi101
```

Неизменяемые файлы

Режимы доступа и разрешения позволяют осуществлять гибкий контроль за тем, кто и что может делать с файлами и директориями. Тем не менее, они не позволяют избежать, например, непреднамеренного удаления файлов пользователем root. Выходя за рамки темы 104.5, стоит упомянуть о том, что в различных файловых системах существуют дополнительные *атрибуты*, предоставляющие дополнительные возможности. Одним из таких атрибутов является атрибут *immutable* (неизменяемый). Если этот атрибут установлен, то никто, включая пользователя root, не сможет удалить файл до тех пор, пока атрибут не будет снят.

Чтобы узнать, установлен ли для файла или директории флаг immutable (или любой другой атрибут), используйте команду `lsattr`. Чтобы сделать файл неизменяемым, используйте команду `chattr` с флагом `-i`.

В листинге 13 показано, что пользователь root может создать неизменяемый файл, но не может удалить его до тех пор, пока не будет снят флаг immutable.

Листинг 13. Неизменяемые файлы

```
[root@echidna ~]# touch keep.me
[root@echidna ~]# chattr +i keep.me
[root@echidna ~]# lsattr keep.me
----i--------e- keep.me
[root@echidna ~]# rm -f keep.me
rm: cannot remove `keep.me': Operation not permitted
[root@echidna ~]# chattr -i keep.me
[root@echidna ~]# rm -f keep.me
```

Для изменения флага immutable необходимо иметь привилегии пользователя root или, по крайней мере, способность CAP_LINUX_IMMUTABLE. Часто файлы делают неизменяемыми для обеспечения защиты системы или обнаружения попыток вторжения. Для получения дополнительной информации обратитесь к man-странице (`man capabilities`).

Маска создания файлов

Каждый раз при создании нового файла система определяет, какие разрешения должны быть для него установлены. Часто устанавливается режим 0666, который позволяет выполнять чтение и запись файла любому пользователю. Для директорий по умолчанию устанавливается режим 0777. В любом случае установка разрешений управляется значением *umask*, определяющим, какие разрешения пользователь **не** хочет автоматически предоставлять при создании новых файлов или директорий. Система использует значение umask с целью понизить изначально установленные разрешения. Значение параметра umask можно посмотреть с помощью команды `umask`, как показано в листинге 14.

Листинг 14. Вывод значения параметра umask в восьмеричном формате

```
[ian@echidna ~]$ umask
0002
```

Помните о том, что umask определяет, какие разрешения **не** должны быть предоставлены. В операционных системах Linux, в которых пользователям не назначаются персональные группы, обычно значением umask является 0022, которое **снимает** с создаваемых файлов разрешения на запись для групп и остальных пользователей. Если пользователям назначаются персональные группы (как, например, в операционной системе Fedora, использующейся в наших примерах), то обычно значением umask является 0002, которое снимает разрешения на запись для остальных пользователей. Чтобы посмотреть значение umask в символическом представлении, т. е. увидеть, какие разрешения **установлены**, используйте опцию `-S`.

Для установки значения umask, а также для его просмотра, используется команда `umask`. Итак, если вы желаете обеспечить дополнительную защиту и запретить всей группе или другим пользователям получать доступ к создаваемым вами файлам, то можно использовать значение параметра umask, равное 0077, или установить его в символическом виде с помощью команды `umask u=rwx,g=,o=`, как показано в листинге 15.

Листинг 15. Установка umask

```
[ian@echidna ~]$ umask -S
u=rwx,g=rwx,o=rx
[ian@echidna ~]$ umask u=rwx,g=,o=
[ian@echidna ~]$ umask
0077
[ian@echidna ~]$ touch newfile
[ian@echidna ~]$ ls -l newfile
-rw-------. 1 ian ian 0 Nov 30 15:40 newfile
```

Установка владельца файла и группы файла

Группа файла

Для изменения группы файла используйте команду `chgrp`, указав также имя группы и один или несколько имен файлов. Вместо имени группы можно указывать ее номер. Обычный пользователь должен являться владельцем файла, а также быть членом группы, назначаемой файлу. Пользователь root может назначать файлам любые группы. В листинге 16 приведен пример изменения группы файла.

Листинг 16. Изменение группы файла

```
[ian@echidna ~]$ touch file{1,2}
[ian@echidna ~]$ ls -l file*
-rw-rw-r--. 1 ian ian 0 Nov 30 15:54 file1
-rw-rw-r--. 1 ian ian 0 Nov 30 15:54 file2
[ian@echidna ~]$ chgrp development file1
[ian@echidna ~]$ chgrp 505 file2
[ian@echidna ~]$ ls -l file*
-rw-rw-r--. 1 ian development 0 Nov 30 15:54 file1
-rw-rw-r--. 1 ian development 0 Nov 30 15:54 file2
```

Как и многие другие команды, рассматриваемые в этом руководстве, команда `chgrp` имеет опцию `-R`, позволяющую рекурсивно применять изменения ко всем выбранным файлам и поддиректориям.

Группа по умолчанию

При изучении материала раздела Режимы доступа вы узнали, как установка режима sgid для директории влияет на создаваемые в ней файлы, а именно, всем этим файлам назначается группа директории, а не группа создающего их пользователя.

Также мы могли использовать команду `newgrp` для временного изменения вашей основной группы на другую группу, членом которой вы являетесь. При этом создается новая командная оболочка, и когда вы выходите из нее, ваша группа восстанавливается, как показано в листинге 17.

Листинг 17. Использование команды newgrp для временного изменения группы по умолчанию

```
[ian@echidna ~]$ groups
ian development editor
[ian@echidna ~]$ newgrp development
[ian@echidna ~]$ groups
development ian editor
[ian@echidna ~]$ touch file3
[ian@echidna ~]$ ls -l file3
-rw-r--r--. 1 ian development 0 Nov 30 16:00 file3
[ian@echidna ~]$ exit
[ian@echidna ~]$ groups
ian development editor
```

Владелец файла

Пользователь root может изменять владельцев файлов с помощью команды `chown`. В простейшей форме синтаксис этой команды похож на синтаксис команды `chgrp` за исключением того, что вместо имени или идентификатора группы используется имя или цифровой идентификатор пользователя. С помощью команды `chown` можно сразу изменить и группу файла, добавив после имени (или идентификатора) пользователя двоеточие и имя (или идентификатор) группы. Если добавить только двоеточие, то будет использоваться группа пользователя по умолчанию. Как следует ожидать, опция `-R` позволяет рекурсивно применять все изменения. Пример приведен в листинге 18.

Листинг 18. Использование команды chown для изменения владельца файла

```
[ian@echidna ~]$ touch file4
[ian@echidna ~]$ su -
Password:
[root@echidna ~]# ls -l ~ian/file4
-rw-rw-r--. 1 ian ian 0 Nov 30 16:04 /home/ian/file4
[root@echidna ~]# chown greg ~ian/file4
[root@echidna ~]# ls -l ~ian/file4
-rw-rw-r--. 1 greg ian 0 Nov 30 16:04 /home/ian/file4
[root@echidna ~]# chown tom:gretchen ~ian/file4
[root@echidna ~]# ls -l ~ian/file4
-rw-rw-r--. 1 tom gretchen 0 Nov 30 16:04 /home/ian/file4
[root@echidna ~]# chown :tom ~ian/file4
[root@echidna ~]# ls -l ~ian/file4
-rw-rw-r--. 1 tom tom 0 Nov 30 16:04 /home/ian/file4
```

Существует устаревшая форма указания пользователя и группы, в которой вместо двоеточия используется точка. Не рекомендуется использовать эту форму, поскольку если в именах используется точка, это может привести к путанице.

На этом мы заканчиваем рассмотрение файловых разрешений в Linux.

Об авторе

Ян Шилдс

No bio.