Права доступа к файлам и директориям

Модель прав доступа в Linux

Один пользователь, одна группа

В этом разделе мы рассмотрим права доступа в Linux и модель владения (ownership). Мы уже видели, что каждый файл принадлежит одному пользователю и одной группе. Это сама суть модели прав доступа в Linux. Вы можете узнать, какому пользователю и группе принадлежит файл в выводе команды ls -l.

```
$ ls l /bin/bash
 rwxrxrx 1 root wheel 430540 Dec 23 18:27 /bin/bash
```

В данном примере исполнимый файл /bin/bash принадлежит пользователю root и группе wheel. Модель прав доступа позволяет задать три независимых уровня прав на каждый объект файловой системы — для владельца, для группы и для всех остальных пользователей.

Понимание «ls -l»

Давайте рассмотрим вывод команды ls -l. Взглянем на первую колонку листинга:

```
$ ls l /bin/bash
rwxrxrx 1 root wheel 430540 Dec 23 18:27 /bin/bash
```

Первое поле -rwxr-xr-x содержит символическое представление прав на данный файл. Первый знак (-) в этом поле определяет тип файла, в данном случае это обычный файл. Другие возможные значения:

'd' директория
'l' символическая ссылка
'c' устройство символьного ввода-вывода
'b' устройство блочного ввода-вывода
'p' FIFO
's' сокет

Три тройки

$ ls l /bin/bash
rwxrxrx 1 root wheel 430540 Dec 23 18:27 /bin/bash

Остальная часть поля состоит из трех троек символов. Первая тройка представляет права владельца файла, вторая представляет права группы файла и третья права всех остальных пользователей.

"rwx"
"rx"
"rx"

Выше r означает, что чтение (просмотр данных содержащихся в файле) разрешено, w означает запись (изменение, а также удаление данных) разрешено и x означает исполнение (запуск программы разрешен).
Собрав все воедино мы видим, что кому угодно разрешено читать содержимое и исполнять этот файл, но только владельцу (root) разрешено как либо модифицировать этот файл. Так что если нормальным пользователям разрешено копировать содержимое этого файла, то только root может изменять или удалять его.

Кто я?

Перед тем, как мы узнаем как изменить владельца или группу которой принадлежит файл, давайте сперва рассмотрим, как узнать вашего текущего пользователя и группу к которой вы принадлежите. Если вы не использовали команду su недавно, ваш текущий пользователь это тот, которым вы вошли в систему. Если вы часто используете su, вы можете не помнить пользователя под которым вы работаете в данный момент. Чтобы узнать под каким пользователем вы работаете, наберите whoami:

```
# whoami
root
# su drobbins
$ whoami
drobbins
```

В каких группах я состою?

Чтобы увидеть к каким группам вы принадлежите используйте команду groups:

```
$ groups
 drobbins wheel audio
```

Из этого примера видно, что я состою в группах drobbins, wheel, и audio. Если вы хотите посмотреть, в каких группах состоит другой пользователь, то передайте его имя в качестве аргумента.

```
$ groups root daemon
root : root bin daemon sys adm disk wheel floppy dialout tape video
daemon : daemon bin adm
```

Изменение пользователя и группы владельца

Чтобы изменить владельца или группу файла (или другого объекта) используется команды chown или chgrp соответственно. Сначала нужно передать имя группы или владельца, а потом список файлов.

```
# chown root /etc/passwd
# chgrp wheel /etc/passwd
```

Вы также можете изменить пользователя и группу одновременно используя команду chown в другой форме:

chown root:wheel /etc/passwd

Рекурсивное изменение прав

Команды chown и chgrp могут быть использованы с параметром -R, что позволяет рекурсивно изменить владельца или группу у всех объектов в

данной директории и ниже. Пример:

chown R drobbins /home/drobbins

Знакомство с chmod

chown и chgrp используются для изменения владельца и группы объекта файловой системы, но кроме них существует и другая программа, называемая chmod, которая используется для изменения прав доступа на чтение, запись и исполнение, которые мы видим в выводе команды ls -l. chmod использует два и более аргументов: метод, описывающий как именно необходимо изменить права доступа с последующим именем файла или списком файлов, к которым необходимо применить эти изменения:

$ chmod +x scriptfile.sh

В примере выше в качестве метода указано +x. Как можно догадаться, метод +x указывает chmod, что файл необходимо сделать исполняемым для пользователя, группы и для всех остальных. Если мы решим отнять все права на исполнение файла, то сделаем вот так:

$ chmod x scriptfile.sh

Разделение между пользователем, группой и всеми остальными

До сих пор, наши примеры команды chmod влияли на права доступа всех трех наборов прав доступа — пользователя, группы и всех остальных пользователей. Часто бывает удобно изменить только один или два набора за раз. Чтобы сделать это, просто используйте специальный символ для обозначения набора прав доступа, который вам необходимо изменить, со знаком + или — перед ним. Используйте u для пользователя, g для группы и o для остальных пользователей.

$ chmod gow scriptfile.sh

Мы только что удалили право на запись для группы и всех остальных пользователей, но оставили права владельца нетронутыми.

Сброс разрешений

Помимо переключения бит, отвечающих за права доступа, в состояние вкл/выкл, мы можем задать конкретные значения для всех сразу. Используя оператор равенства мы можем указать chmod, что хотим задать только указанные права доступа:

```
$ chmod =rx scriptfile.sh
```

Этой командой мы установили все биты чтения и исполнения и сбросили все биты записи. Если вы хотите задать значения конкретной тройки бит, то можете сделать это, указав ее символьное наименование перед оператором равенства:

```
$ chmod u=rx scriptfile.sh
```

Числовые режимы

До сих пор, мы использовали то что называется символическим способом указания прав доступа для команды chmod. Однако есть еще один достаточно распространенный способ указания прав: использование четырехзначных восьмеричных чисел. Этот синтаксис, называется числовым синтаксисом прав доступа, где каждая цифра представляет тройку разрешений.

Например, в 1777, 777 устанавливают флаги о которых мы говорим в этом разделе, для владельца, группы, и остальных пользователей. 1 используется для указания специального бита прав доступа, который мы рассмотрим позже (смотрите «Неуловимая первая цифра» в конце раздела). Эта таблица показывает как транслируются права доступа на числовые значения.

Режим	Число
rwx	7
rw-	6
rx	5
r--	4
-wx	3
-w-	2
--x	1
---	0

umask

Числовой синтаксис прав доступа

Числовой синтаксис прав доступа особенно полезен когда требуется указать все разрешения для файла, как показано в следующем примере:

```
$ chmod 0755 scriptfile.sh
$ ls l scriptfile.sh
rwxrxrx 1 drobbins drobbins 0 Jan 9 17:44 scriptfile.sh
```

В этом примере мы назначили права доступа 0755, что равносильно комбинации прав -rwxr-xr-x.

umask

Когда процесс создает новый файл, он указывает, какие права доступа нужно задать для данного файла. Зачастую запрашиваются права 0666 (чтение и запись всеми), что дает больше разрешений, чем необходимо в большинстве случаев. К счастью, каждый раз, когда в Linux создается новый файл, система обращается к параметру, называемому umask. Система использует значение umask чтобы понизить изначально задаваемые разрешения на что-то более разумное и безопасное. Вы можете просмотреть текущие настройки umask набрав umask в командной строке:

```
$ umask
0022
```

В Linux-системах значением по умолчанию для umask является 0022, что позволяет другим читать ваши новые файлы (если они могут до них добраться), но не изменять их. Чтобы автоматически обеспечивать больший уровень защищенности для создаваемых файлов, можно изменить настройки umask:

$ umask 0077

Такое значение umask приведет к тому, что группа и прочие не будут иметь совершенно никаких прав доступа для всех, вновь созданных файлов.

Итак, как работает umask?

В отличие от «обычного» назначения прав доступа к файлу, umask задает какие права доступа должны быть отключены. Снова посмотрим на таблицу соответствия значений чисел и методов:

Режим	Число
rwx	7
rw	6
rx	5
r	4
wx	3
w	2
x	1
	0

Воспользовавшись этой таблицей мы видим, что последние три знака в 0077 обозначают ---rwxrwx. Теперь вспомните, что umask показывает системе, какие права доступа отключить. Совместив первое и второе становится видно, что все права для группы и остальных пользователей будут отключены, в то время как права владельца останутся нетронутыми.

Знакомство с suid и sgid

В момент вашего входа в систему запускается новый процесс оболочки. Вы уже знаете об этом, но можете не знать о том, что этот новый процесс оболочки (обычно это bash) работает от имени вашего пользователя. И таким образом программа bash может обращаться ко всем файлам и директориям, владельцем которых вы являетесь. В действительности мы, как пользователи, полностью зависим от программ, выполняющих операции от нашего имени. И поскольку программы, которые вы запускаете, наследуют ваш пользовательский идентификатор, они не могут обращаться объектам файловой системы, к которым вам не предоставлен доступ. К примеру, обычные пользователи не могут напрямую изменять содержимое файла passwd потому что флаг записи отключен для всех пользователей кроме root:

```
$ ls l /etc/passwd
 rwrr 1 root wheel 1355 Nov 1 21:16 /etc/passwd
```

Однако, обычным пользователям тоже нужно иметь возможность хотя бы опосредованно менять содержимое /etc/passwd когда им понадобится сменить пароль. Но если пользователь не может изменить этот файл, как это сделать?

suid

К счастью, в модели прав доступа Linux имеются два специальных бита, называемых suid и sgid. Когда для запускаемой программы установлен бит suid, она будет работать от имени владельца исполняемого файла, а не от имени того, кто запустил программу. Теперь можем вернуться к вопросу с /etc/passwd. Если посмотрим на исполняемый файл passwd, увидим, что его владельцем является пользователь root:

```
$ ls l /usr/bin/passwd
rwsrxrx 1 root wheel 17588 Sep 24 00:53 /usr/bin/passwd
```

Обратите внимание, что вместо x в триплете прав доступа владельца стоит s. Это означает что для этой конкретной программы установлены биты suid и права на запуск. По этой причине при запуске программы passwd она будет работать от имени пользователя root (со всеми правами доступа суперпользователя), а не пользователя, запустившего её. И поскольку passwd работает с правами суперпользователя, она способна редактировать
/etc/passwd без каких либо сложностей.

Предупреждения о suid/sgid

Мы увидели как работает suid, sgid работает похожим образом. Она позволяет программе наследовать права доступа группы, а не текущего пользователя.

Важно!

Немного разрозненная, но в то же время очень важная информация о suid и sgid. Во-первых, биты suid и sgid занимают те же поля в выводе команды ls -l. Если бит x тоже задан, соответствующие биты будут показаны как s (в нижнем регистре). Однако, если бит x не задан то он будет отображаться как S (в верхнем регистре).

Еще одно важное замечание: suid и sgid бывают удобны во многих ситуациях, но неправильное их использование может привести к появлению уязвимостей в защите системы. Лучше всего иметь как можно меньшее количество suid программ. Команда passwd — одна из немногих, которая должна быть suid.

Изменение suid и sgid

Способ установки и удаления битов suid и sgid чрезвычайно прост. Вот так мы задаем бит suid:

```
# chmod u+s /usr/bin/myapp
```

А в следующем примере мы снимаем флаг sgid с директории. Вы увидите, как бит sgid работает с директориями немного ниже:

```
# chmod gs /home/drobbins
```

Права и директории

До текущего момента мы рассматривали права доступа с точки зрения обычных файлов. Когда речь заходит о директориях, появляются некоторые отличия. Директории используют те же флаги прав доступа, но их интерпретация имеет немного другой смысл.

Если для директории задан флаг чтения, то вы можете просматривать список содержимого директории; флаг записи означает, что вы можете создавать файлы в директории; и флаг исполнения означает, что вы можете войти в директорию и обращаться ко всем поддиректориям внутри. Без флага исполнения у вас не будет доступа к объектам файловой системы внутри директории. Без флага чтения объекты файловой системы внутри директории нельзя просмотреть, но к объектам внутри директории все еще можно обратиться, если вы знаете полный путь к объекту на диске.

Директории и флаг sgid

В случае же, если для директории установлен бит sgid, все объекты файловой системы, создаваемые внутри, наследуют группу директории. Эта возможность бывает кстати, когда вам необходимо создать дерево директорий и все они должны принадлежать одной группе. Это можно сделать вот так:

```
# mkdir /home/groupspace
# chgrp mygroup /home/groupspace
# chmod g+s /home/groupspace
```

Директории и флаг sgid

В случае же, если для директории установлен бит sgid, все объекты файловой системы, создаваемые внутри, наследуют группу директории. Эта возможность бывает кстати, когда вам необходимо создать дерево директорий и все они должны принадлежать одной группе. Это можно сделать вот так:

```
# mkdir /home/groupspace
# chgrp mygroup /home/groupspace
# chmod g+s /home/groupspace
```

Теперь любые пользователи группы mygroup могут создавать файлы и директории внутри /home/groupspace и им также будет автоматически задана принадлежность группе mygroup. В зависимости от настроек umask для данного пользователя новые объекты файловой системы могут быть или не быть читаемыми, изменяемыми или исполняемыми другими пользователями группы mygroup.

Директории и удаление

По умолчанию директории в Linux ведут себя не самым удобным во многих ситуациях образом. Обычно кто угодно может переименовать или удалить файл внутри директории если у них есть права на запись в этой директории. Для директорий, которыми владеют отдельные пользователи, такое поведение обычно не вызывает проблем.

Однако для директорий, которыми пользуется большое количество пользователей, в особенности /tmp и /var/tmp, это может вызвать целую кучу проблем. Все потому, что кто угодно может писать в эти директории, кто угодно может удалять и переименовывать чьи угодно файлы — даже если они им не принадлежат! Очевидно, довольно сложно использовать /tmp даже для временного хранения чего угодно, когда любой пользователь в любой момент может напечатать rm -rf /tmp/* и уничтожить файлы всех остальных.

Хорошая новость в том, что в Linux существует так называемый sticky бит. Когда для /tmp установлен sticky бит (командой chmod +t), единственные, кто могут удалить или переименовать файлы в /tmp — это либо владельцы этих файлов либо суперпользователь.

Неуловимый первый знак

В завершение этого раздела мы наконец обратим внимание на первый знак, используемый в численном синтаксисе.

Он используется для задания битов sticky, suid и sgid:

suid	sgid	sticky	режим
on	on	on	7
on	on	off	6
on	off	on	5
on	off	off	4
off	on	on	3
off	on	off	2
off	off	on	1
off	off	off	0

chmod 1775 /home/groupfiles

Управление аккаунтами в Linux

/etc/passwd

В этом разделе мы познакомимся с механизмом управления аккаунтами в Linux и начнем с файла /etc/passwd, в котором определены все пользователи, которые существуют в системе. Вы можете посмотреть свой файл /etc/passwd, набрав команду less /etc/passwd. Каждой строкой в /etc/passwd определяется аккаунт пользователя.

Пример из /etc/passwd:

drobbins:x:1000:1000:Daniel Robbins:/home/drobbins:/bin/bash

Как видите, в одной строке не так уж много информации. Каждая из них содержит несколько полей, разделённых ":". Первое поле отвечает за имя пользователя (drobbins), второе поле содержит «x». На устаревших Linux-системах второе поле содержало зашифрованных пароль для аутентификации, но фактически, сейчас все Linux-системы хранят эту информацию в другом файле. Третье поле отвечает за числовой пользовательский идентификатор, связанный с конкретным пользователем, а четвертое поле ассоциирует этого пользователя с конкретной группой; скоро мы увидим, где определена группа 1000. Пятое поле содержит текстовое описание аккаунта, в нашем случае это имя пользователя. Шестое поле определяет домашний каталог пользователя, седьмое — устанавливает стартовую оболочку пользователя, которая будет автоматически запускаться когда пользователь входит в систему.

/etc/passwd, советы и хитрости

Вы вероятно заметили, что в системе намного больше пользовательских аккаунтов, которые определены в /etc/passwd, чем тех, которые логинятся в систему на самом деле. Всё это потому, что различные компоненты Linux используют некоторые аккаунты для повышения безопасности. Обычно, такие системные аккаунты имеют идентификатор (uid) меньший 100, и у многих из них в качестве стартовой оболочки установлена /bin/false. Так как эта программа ничего не делает, кроме как выходит и возвращает код ошибки, это эффективно препятствует использованию этих аккаунтов в качестве обычных аккаунтов для логина — т.е. они предназначены только для внутрисистемного пользования.

/etc/shadow

Сами пользовательские аккаунты определены в /etc/passwd.
Системы Linux вдобавок к /etc/passwd содержат его файл-компаньон /etc/shadow. Он, в отличие от /etc/passwd, доступен для чтения только суперпользователю и содержит зашифрованную информацию о паролях.
Взглянем на образец строки из /etc/shadow:

drobbins:$1$12345678901234567890123456789O1:11664:0:1:1:1:1:0

Каждая строка определяет информацию о пароле конкретного аккаунта, поля в ней разделены знаком ":". Первое поле определяет конкретный пользовательский аккаунт, которому соответствует данная «теневая» запись. Во втором поле содержится зашифрованный пароль.
Оставшиеся поля описаны в таблице ниже:

- поле 3 — количество дней с 01.01.1970 до момента, когда пароль был изменен

- поле 4 — количество дней до того, как будет разрешено сменить пароль («0» — «менять в любое время»)

- поле 5 — количество дней до того, как система заставит пользователя сменить пароль ("-1" — «никогда»)

- поле 6 — количество дней до истечения срока действия пароля, когда пользователь получит предупреждение об этом ("-1" — «не предупреждать»)

- поле 7 — количество дней после истечения срока действия пароля, по прошествии которых аккаунт будет автоматически отключен системой ("-1" — «не отключать»)

- поле 8 — количество дней, прошедшее с момента отключения этого аккаунта ("-1" — «этот аккаунт включен»)

- поле 9 — зарезервировано для будущего использования

/etc/group

Теперь взглянем на файл /etc/group, который определяет группы в системе Linux. Вот примерная строка из него:

drobbins:x:1000:

Формат полей файла /etc/group следующий: первое поле определяет имя группы, второе поле — это поле остаточного пароля, которое сейчас просто зарезервировано x, и третье поле определяет числовой идентификатор для конкретной группы. Четвертое поле (которое пусто в примере выше) определяет всех членов группы.
В нашем образце строки из /etc/passwd есть «ссылка» на группу с идентификатором 1000. Мы сможем поместить пользователя drobbins в группу drobbins, даже несмотря на отсутствие имени drobbins в четвертом поле /etc/group.

Примечания о группах

Замечание насчет соответствия пользователей с группами: на некоторых системах каждый новый логин-аккаунт связан с группой, имеющей то же имя (и обычно идентификатор). На других системах все логин-аккаунты будут принадлежать к одной группе пользователей. Какой из этих методов выбрать зависит от вас. Создание соответствующей группы для каждого пользователя имеет преимущество в том, что позволяет им более легко контролировать их собственный доступ просто помещая доверенных друзей в свою личную группу.

Ручное создание пользователей и групп

Теперь, я покажу как создать аккаунты для пользователя и группы.

Лучший путь узнать как это сделать это добавить нового пользователя в Ручное создание пользователей и групп систему вручную. Для начала убедитесь что вашей переменной окружения EDITOR соответствует ваш любимый редактор:

```
# echo $EDITOR
vim
```

Если это не так, то вы можете установить переменную EDITOR, набрав что-то, вроде:

```
# export EDITOR=/usr/bin/emacs
# vipw
```

Теперь ваш редактор должен быть запущен с уже загруженным /etc/passwd экране. Изменяя системные файлы passwd и group обязательно используйте команды vipw и vigr. Они имеют повышенные меры предосторожности, оберегая ваши файлы от участи быть испорченными.

Редактирование /etc/passwd

Итак, у вас уже есть готовый файл /etc/passwd, добавьте теперь следующую строку:

testuser:x:3000:3000:LPI tutorial test user:/home/testuser:/bin/false

Мы только что добавили пользователя «testuser» с идентификатором 3000. Мы определили его в группу с таким же идентификатором, которую еще не создали. Но мы можем добавить его к уже имеющейся группе пользователей, если нужно. У этого пользователя установлен комментарий, гласящий «LPI tutorial test user», домашний каталог установлен как "/home/testuser", а командная оболочка — как "/bin/false", в целях безопасности. Если бы мы создавали не тестовый аккаунт, мы бы установили командную оболочку как "/bin/bash".

Редактирование /etc/shadow

Сейчас нам нужно добавить запись в /etc/shadow для этого пользователя. Для этого наберите vipw -s. Вас как всегда встретит ваш любимый редактор в котором уже открыт файл /etc/shadow. Теперь скопируйте строку существующего пользовательского аккаунта (того, у которого есть пароль и запись которого длиннее стандартных записей системных аккаунтов)

drobbins:$1$12345678901234567890123456789O1:11664:0:1:1:1:1:0

Замените имя пользователя в скопированной строке на имя вашего пользователя и убедитесь что все поля (особенно старый пароль) установлены как вам надо:

testuser:$1$12345678901234567890123456789O1:11664:0:1:1:1:1:0

Установка пароля

Вы вернетесь к командной строке. Теперь, самое время задать пароль для вашего нового пользователя.

```
# passwd testuser
Enter new UNIX password: (enter a password for testuser)
Retype new UNIX password: (enter testuser's new password again)
```

Редактирование /etc/group

Теперь /etc/passwd и /etc/shadow готовы и самое время как следует настроить /etc/group. Для этого, наберите:

```
# vigr
```

Перед вами появится ваш /etc/group файл, готовый для редактирования. Итак, если ранее вы решили добавить созданного пользователя к уже имеющейся группе, то вам не понадобиться создавать новую группу в /etc/groups. Если это не так, вам нужно добавить новую группу для этого пользователя, введите следующую строку:

```
testuser:x:3000:
```