Права доступа к файлам и директориям

Принцип распределения прав доступа

	Файл	Директория
Чтение	Возможность читать содержимое файла	Возможность просматривать список файлов
Запись	Возможность изменять содержимое файла	Возможность создавать и удалять файлы
Исполнение	Возможность запускать файл (имеет смысл лишь в том случае, если файл является программой или сценарием)	Возможность читать метаданные (свойства) файлов. Только при наличии этого атрибута можно сделать директорию текущей, т.е. перейти в неё командой «cd»

Поскольку операционная система Linux с самого начала разрабатывалась как многопользовательская, в ней предусмотрен механизм, позволяющий разграничить полномочия пользователей, работающих в системе. В частности, права доступа позволяют отдельным пользователям и группам иметь «личные» файлы и директории.

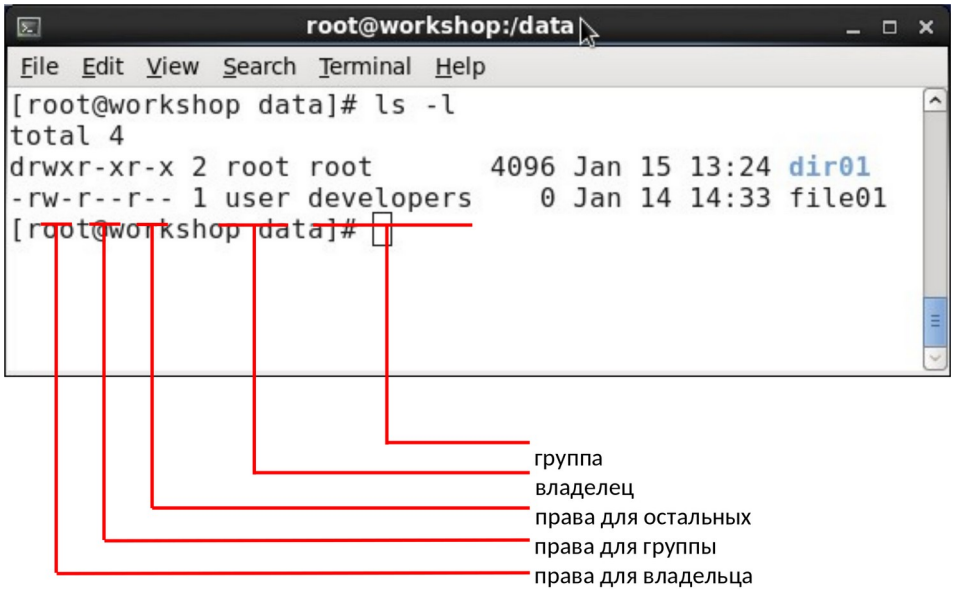
У любого файла в системе есть владелец - один из пользователей. Одновременно с этим, каждый файл принадлежит некоторой группе пользователей.

Права доступа предполагают наличие/отсутствие трёх типов действий: чтение, запись и исполнение. Эти права доступа могут быть предоставлены трём классам пользователей: владельцу файла, группе, которой принадлежит файл, а также всем остальным пользователям, не входящим в эту группу.

Каждый работающий в системе процесс имеет своего владельца. Предположим, некий процесс пытается записать в файл какую-то информацию. Первое, что проверяет в этом случае система - это является ли владелец процесса владельцем файла? И если да, то применяются права для владельца. Если нет, то вторым шагом проверяется, входит ли владелец процесса в группу, связанную с файлом и если да, то применяются права для группы. Если же и членство в группе не подтверждается, то применяются права для всех остальных

Права доступа к файлам и директориям в GNU/Linux могут распределять не только владельцы, но и некоторые системные процессы: например - монтирование файловых систем. Таким образом, права доступа ко вновь подключенным частям файловой системы будут изначально определяются системными правилами.

Просмотр и изменение прав доступа



Посмотреть права доступа, а также информацию о владельце можно с помощью команды ls с параметром «-l» (long format).

Наличие права обозначается соответствующей буквой ("r" означает чтение, "w" - запись и "x" - исполнение), отсутствие права - прочерком. В данном примере владелец файла file01 (пользователь с именем «user») имеет право на чтение и запись, пользователи системы, входящие в группу developers имеют право только на чтение и все остальные пользователи также имеют право только на чтение. Некоторые права имеют смысл только в сочетании с другими. Например, для того, чтобы сценарий мог корректно выполняться, помимо права на исполнение, на него необходимо установить право на чтение, чтобы интерпретатор мог читать содержимое файла.

Сменить владельца файла и связанную с ним группу можно с помощью команды chown:

```
[demo@localhost tmp]$ sudo chown demo file01
[demo@localhost tmp]$ sudo chown :webmasters file02
[demo@localhost tmp]$ sudo chown demo:webmasters file03
```

Для изменения группы может быть использована самостоятельная команда chgrp. Не забывайте, что в момент смены владельца указанный пользователь тут же получает на файл соответствующие ему права.

Для изменения прав доступа используется команда chmod. Она имеет два формата параметров, символьный и числовой. Рассмотрим для начала символьный вариант. Владелец файла обозначается буквой "u" (user), группа - буквой "g" (group) и все остальные - буквой "o" (other). Если нужно применить какое-то действие сразу ко всем трём категориям, то можно использовать букву "a" (all). Права доступа обозначаются своими стандартными символами r,w,x. А действия над ними - символами "+" (добавить право), "-" отобрать право и "=" установить права в соответствии с тем, что стоит справа от знака равенства. Т.е. если к примеру, в команде указано u=rx, то неважно, какие права были ранее. После выполнения команды это будет чтение и исполнение.

Рассмотрим несколько примеров:

```
[demo@localhost ~ ]$ chmod u+x script01.sh
[demo@localhost ~ ]$ chmod u+x,g-w script02.sh
[demo@localhost ~ ]$ chmod o=- script03.sh
```

Первая команда добавляет владельцу файла право на исполнение, вторая добавляет владельцу право на исполнение, а у группы отбирает право на изменение файла, третья команда отбирает у всех остальных все права.

И команда chown, и команда chmod поддерживают опцию "-R" которая позволяет менять права доступа рекурсивно. Т.е. команда "chmod -R g+w /somedir" Установит право на чтение для группы не только на саму директорию но и на все вложенные в неё файлы и поддиректории.

В связи с этим для атрибута, дающего право на исполнение предусмотрен вариант, позволяющий указывать его как в нижнем (x), так и в верхнем (X) регистре. Если атрибут указан в верхнем регистре, то это означает, назначать его только на директории, пропуская файлы. Таким образом, при рекурсивном вызове команды (например, "chmod -R g+X /somedir"), это позволит избежать ситуации, когда все файлы в директории станут исполняемыми. Для атрибутов, дающих право на чтение и на запись, такой вариант не предусмотрен.

Числовой формат прав доступа

Право	Двоичное число	Десятичное число
r	100	4
w	010	2
x	001	1

```
644 = 110 100 100 = rw-r--r--
751 = 111 101 001 = rwxr-x--x
326 = 011 010 110 = -wx-w-rw-
```

Числовой формат выглядит более компактно. Для его использования необходимо запомнить следующее:

- 4 - означает право на чтение;
- 2 - означает право на запись;
- 1 - означает право на исполнение

Почему в качестве атрибутов выбраны цифры 1-2-4, а не 1-2-3? Это связано с их двоичным представлением. Цифры 1,2,4 - восьмеричные. Если посмотреть на их двоичное представление, то мы увидим, что

18=0012
28=0102
48=1002

При необходимости назначения нескольких атрибутов, эти цифры суммируются. Предположим, нам нужно, чтобы владелец файла имел возможность чтения, записи и исполнения, пользователи связанной с файлом группы имели бы право чтения и исполнения и все остальные пользователи должны иметь право только просматривать файл. Получаем следующее значение: 4+2+1=7 (1112)

4+1=5 (1012)
4=4

Таким образом, команда будет выглядеть так:

[demo@localhost ~]$ chmod 754 script04.sh

Если необходимо, то просмотреть права доступа в цифровом формате позволяет команда stat (комбинация "\n" отвечает за перенос строки): [demo@localhost ~]$ stat --printf="%a\n" /etc/resolv.conf

Дополнительные атрибуты

Атрибут	Значение для файлов	Значение для директорий
SUID	Запускать с правами владельца файла	—
SGID	Запускать с правами группы файла	Наследовать группу файла от группы каталога
Sticky	—	Запрет на удаление чужих файлов

Помимо перечисленных атрибутов, встречающихся наиболее часто, есть и другие, а именно: suid(4), sgid(2), sticky bit(1).

Sticky bit в настоящее время применим только для каталогов и обладает следующим эффектом: если на каталог установлен атрибут sticky bit, то удалить файл в этом каталоге может только его владелец. Обозначается данный атрибут буквой «t» и во вновь установленной системе присутствует на директории «/tmp». Установить его можно с помощью обычной команды chmod:

chmod o+t /somedir или
chmod 1777 /somedir

Атрибут suid предназначен для установки на исполняемые бинарные файлы. Он имеет следующий эффект: при запуске такого файла, порождённый процесс будет исполняться не с правами того пользователя, который его запустил (как это происходит в обычных обстоятельствах), а с правами с правами владельца файла. Установка данного атрибута осуществляется командой:

chmod u+s somefile.sh

В операционной системе присутствует некоторое количество исполняемых файлов с установленным атрибутом suid, например, команда passwd. (Убедиться в этом можно, введя команду ls-l /usr/bin/passwd). Для чего команде passwd нужен этот атрибут? Дело в том, что каждый пользователь системы имеет право менять пароль самому себе. Однако, файл, в котором хранятся данные о паролях, защищён даже от просмотра, не говоря уже о записи непривилегированными пользователями. Благодаря наличию у команды passwd атрибута suid, пользователь получает возможность обновлять данные о пароле, т.к. владелец файла - root. Найти все файлы, на которых установлен атрибут suid можно с помощью команды find:

[demo@localhost ~]$ sudo find / -type f -perm -4000 2> /dev/null

У атрибута sgid аналогичный смысл, просто речь идет не о пользователе, а о группе. Если sgid установлен на директорию, то при создании в этой директории файла, он будет получать туже самую группу, которая владеет директорией, а не такую же, как у владельца. Если в директории с установленным sgid создаётся поддиректория, она также автоматически получает атрибут sgid. Задать атрибут sgid можно, например так:

```
[demo@localhost ~]$ sudo mkdir -m 777 /tmp/demo
[demo@localhost ~]$ sudo chown :webmasters /tmp/demo
[demo@localhost ~]$ echo test > /tmp/demo/file01
[demot@localhost ~]$ sudo chmod g+s /tmp/demo
[demo@localhost ~]$ echo test > /tmp/demo/file02
[demo@localhost ~]$ ls -l /tmp/demo total 8
-rw-rw-r--. 1 demo  demo    5 May 18 07:27 file01
-rw-rw-r--. 1 demo  webmasters 5 May 18 07:27 file02
```

Файл file01, созданный до того, как на директорию был установлен атрибут suid имеет владельца и группу "student:student", а файл file02 в качестве группы уже имеет "webmasters"

Стоит отметить, что создавать файлы с установленными атрибутами suid/sgid нужно только в тех случаях, когда это необходимо, понимая при этом, для чего это делается и к каким последствиям приведет. Дело в том, что в случае ошибки с правами доступа такой файл может стать угрозой безопасности.

Права по умолчанию

Для файлов	666	110110110
666	022	000010010
Для каталогов	----	----------
777	644	110100100

Когда мы создаём файлы и директории, то они сразу же получают некоторые права доступа. По какому принципу они задаются?

При создании файла ему назначаются права доступа 666, а при создании директории - 777. Однако, на самом деле права получаются другие. Так происходит потому, что в системе присутствует маска. Посмотреть значение маски можно с помощью команды umask:

```
[demo@localhost ~]$ umask
0002
[demo@localhost ~]$ umask -S u=rwx,g=rwx,o=rx
```

Маска представляет собой набор восьмеричных цифр, битовое представление которых определяет, какие именно права мы хотим исключить из максимально возможного значения. Т.е. если права для создаваемого файла 666, а значение маски 022, то итоговое значение прав доступа будет 644.

Задаётся маска с помощью той же самой команды umask:

```
[demo@localhost tmp]$ umask
0002
[demo@localhost tmp]$ touch test01
[demo@localhost tmp]$ ls -l test*
-rw-rw-r--. 1 demo  demo  0 May 21 03:16 test01
[demo@localhost tmp]$ umask 026
[demo@localhost tmp]$ touch test02
[demo@localhost tmp]$ ls -l test*
-rw-rw-r--. 1 demo  demo  0 May 21 03:16 test01
-rw-r-----.   1 demo  demo  0 May 21 03:16 test02
```

Инициализационные скрипты

/etc/profile

/etc/profile.d/*.sh

/etc/bashrc

~/.bash_profile

~/.bashrc

Однако, стоит помнить о том, что команда umask имеет динамический характер. Это значит, что изменения маски нигде не сохраняются и при следующем запуске оболочки маска опять примет своё исходное значение. Откуда это значение берётся? В системе есть два файла, где определяется значение маски. Это файл /etc/profile и файл /etc/bashrc, в которых присутствует вот такой фрагмент:

```
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then     umask 002 else
    umask 022
fi
```

Согласно этому фрагменту, у непривилегированных пользователей маска будет 002, а у пользователя root и системных пользователей (тех, у кого uid < 200), она будет иметь значение 022.

Почему файла два? Дело в том, что запущенные экземпляры командного интерпретатора можно поделить на две категории: login-shell и nonlogin-shell. Первый запускается, если процесс попадания в систему предполагал аутентификацию. Например, логин, доступ по ssh, выполнение команды "su -". А второй запускается, если аутентификация не предполагалась. Например, пользователь просто запустил команду bash или использовал "su" без дефиса.

Разница между login и nonlogin заключается в том, что в первом случае для инициализации используются файлы /etc/profile и /etc/bashrc, а во втором - только /etc/bashrc, файл profile не применяется.

Последнее примечание, касающееся маски: если вы измените значение umask в файле /etc/bashrc, то изменения коснуться всех пользователей системы. Поменять маску какому-то конкретному пользователю можно, отредактировав файл .bashrc, лежащий в домашней директории этого пользователя. В этом случае изменения не затронут остальных пользователей системы.

Задание 1

Интерпретация прав доступа

Ниже приведены права доступа к файлу, записанные в цифровом формате. Как будет выглядеть символьное представление каждого набора?

644 rw-r--r--

755 _____

000 _____

777 _____

600 _____

Ниже приведены права доступа к файлу, записанные в цифровом формате. Как будет выглядеть символьное представление каждого набора?

644 rw-r--r--
755 rwxr-xr-x
000 ---------
777 rwxrwxrwx
600 rw-------

Задание 2

Предположим, в системе существуют следующие пользователи и группы:

Пользователь	Группа
hulk	hulk, avengers
ironman	ironman, avengers
loki	loki, asgard
thor	thor, asgard

А также директория «storage», содержащая несколько файлов:

```
drwxrwxr-x   ironman avengers   Dec 30 00:37 /storage
   -rw-rw-r--    hulk          hulk          Dec 30 00:37 archive01
   -rw-r--rw-    hulk          avengers   Dec 30 00:37 archive02
   -rw-rw-r--    ironman   avengers   Dec 30 00:37 archive03
   -rw-r-----     ironman   avengers   Dec 30 00:37 archive04
```

Попробуйте ответить на следующие вопросы (Да/Нет):

1. Пользователь hulk может изменять содержимое файла archive01
2. Пользователь ironman может просматривать содержимое файла archive02 но не может изменять его содержимое
3. Пользователь ironman может удалить файлы archive01 и archive02
4. Пользователь loki может изменять содержимое файла archive02
5. Пользователь hulk может изменять содержимое файла archive03
6. Пользователь ironman может просматривать и изменять содержимое файла archive04
7. Пользователь hulk может просматривать но не может изменять содержимое файла archive04
8. Пользователи loki и thor могут только просматривать файл archive04

Задание 3

Настройка прав доступа

1) Добавьте в систему группу "developers"
2) Создайте пользователя chip, входящего в группу developers
3) Создайте пользователя dale, который не входит в группу developers
4) Создайте директорию «/tmp/project», удовлетворяющую следующим условиям: пользователи, входящие в группу "developers" должны иметь возможность создавать файлы в директории, все прочие пользователи должны иметь возможность только просматривать файлы в директории.
5) Проверьте результат, попытавшись создать файл в директории «/tmp/project» сначала от лица пользователя chip, а потом от лица пользователя dale.

Добавьте в систему группу "developers"

sudo groupadd developers

Создайте пользователя chip, входящего в группу developers

sudo useradd -G developers chip

Создайте пользователя dale, который не входит в группу developers

sudo useradd dale

Создайте директорию «/tmp/project», удовлетворяющую следующим условиям: пользователи, входящие в группу "developers" должны иметь возможность создавать файлы в директории, все прочие пользователи должны иметь возможность только просматривать файлы в директории.

sudo mkdir /tmp/project sudo chgrp developers /tmp/project sudo chmod g+w,o=rx /tmp/project

Проверьте результат, попытавшись создать файл в директории «/tmp/project» сначала от лица пользователя chip, а потом от лица пользователя dale.

sudo -i -u chip echo "test" > /tmp/project/test.chip exit sudo -i -u dale
echo "test" > /tmp/project/test.dale exit

su

su [-] [имя_пользователя]

Как мы с вами уже убедились, наша с вами операционная система - это система с очень чётким разграничением полномочий пользователей. Это касается не только доступа к файлам и директориям, но и, разумеется, вопросов, связанных с запуском различных программ. В обычных обстоятельствах непривилегированные пользователи лишены возможности выполнять любые команды, вносящие изменения в конфигурацию операционной системы. Каким образом можно выполнить команду, подразумевающую наличие административных привилегий? Первое, что может прийти в голову - это просто авторизоваться в системе под учётной записью root. Это действительно простой, но КРАЙНЕ НЕ РЕКОМЕНДУЕМЫЙ способ. Более того, в ряде дистрибутивов возможность входа в систему пользователем root отключена по-умолчанию, чтобы ни у кого не возникло соблазна так поступить. Поэтому, для того, чтобы выполнить такую команду, используют механизмы, называемые su или (что чаще) sudo. Рассмотрим их работу подробнее:

Команда su (switch user) позволяет запустить шелл с привилегиями другого пользователя. Формат команды:

su [-] [имя_пользователя]

Обратите внимание, что дефис в данном случае это не признак опции, а самостоятельный элемент команды. Имя пользователя не является обязательным параметром, если оно опущено, то подразумевается, что мы запускаем шелл, работающий с привилегиями администратора. После того, как все команды, которые требуется выполнить от лица другого пользователя, будут завершены, шелл необходимо закрыть с помощью команды exit или комбинации Ctrl-D. Наличие дефиса говорит о том, что необходимо переключиться к учётной записи пользователя вместе с его окружением (будет применён профиль, изменятся значения некоторых переменных окружения, т.е. ситуация будет выглядеть ближе к тому, как если бы вы изначально вошли этим пользователем в систему). Без дефиса изменяется только UID.

sudo

sudo команда

/etc/sudoers:
кто где=(от имени кого) что

Использование su для выполнения административных задач имеет несколько неудобств: во-первых, всем, кому потребуется её применение, необходимо сообщить пароль пользователя root. Во-вторых, запуск оболочки с повышенными привилегиями даёт возможность выполнить любую потенциально опасную команду.

Рассмотрим следующую ситуацию: допустим, в компании есть привлечённый специалист, занимающийся обслуживанием вебсервера. Время от времени ему требуется перезапускать процесс вебсервера. Перезапуск такого процесса - это процедура, требующая административных привилегий. Получается, что придется либо сообщить этому сотруднику пароль пользователя root (что неприемлемо), либо каждый раз он будет вынужден обращаться к нам, как к администратору системы, чтобы осуществить перезапуск процесса. Для описанного случая идеально подходит другой инструмент - команда sudo. Она позволяет расширить привилегии пользователя, дав ему возможность использовать конкретные команды или группы команд без необходимости вводить пароль пользователя root.

Данные о том, кому и что разрешено делать в системе, хранятся в файле /etc/sudoers. Также можно создавать собственные конфигурационные файлы в директории /etc/sudoers.d

Хотя конфигурация представляет собой обычный текстовый файл, использование обычных редакторов для внесения изменений не рекомендуется. Для этих целей есть отдельный инструмент - команда visudo. На самом деле это обвязка для редактора vim, которая при сохранении файла sudoers проверяет его синтаксическую корректность.

Формат записей, отвечающих за распределение полномочий, можно описать следующим образом:

кто где=(от имени кого) что

- "кто" - имя пользователя или группы. Если подразумевается группа, то перед её именем ставится знак "%", например: %webmaster
- "где" - имя хоста или IP-адрес узла, на котором допускается запускать команду. Если этот параметр не принципиален, то можно использовать зарезервированное слово "ALL".
- "от имени кого" - от имени кого будет выполнена команда. Если параметр не указывается, то подразумевается пользователь root.
- "что" - список команд, которые допустимо выполнять пользователю или группе

Таким образом, если мы хотим группе webmasters дать возможность перезапускать вебсервер, мы должны добавить в файл sudoers следующую строку:

```
%webmasters  ALL=/bin/systemctl  restart  httpd.service
```

Задание 5

Использование команды su для делегирования полномочий

Являются ли эквивалентными следующие команды а и б:

а) su -

б) su - root

Проделайте следующие шаги:

1) Войдите в систему под своей учётной записью (либо пользователем student)
2) выполните команду whoami, которая используется для того, чтобы определить, под какой учётной записью осуществляется работа. Запишите результат
3) выполните команду su -
4) снова выполните команду whoami. Запишите результат
5) что необходимо сделать, чтобы вернуться к предыдущей учётной записи?

Задание 6

1) Добавьте в систему нового локального пользователя testuser с паролем testuser

sudo useradd testuser sudo passwd testuser

2) С помощью команды su переключитесь на учётную запись testuser

su - testuser

3) Попробуйте выполнить команду "/sbin/fdisk /dev/sda"

/sbin/fdisk /dev/sda

4) Команда не должна запуститься. Так ли это?

5) Попробуйте выполнить команду "/sbin/fdisk /dev/sda" с помощью sudo

sudo /sbin/fdisk /dev/sda

6) Команда также не должна запуститься. Так ли это?

7) Обратите внимание на вывод команды sudo

8) Вернитесь к своей учётной записи exit

9) С помощью команды sudo добавьте в файл sudoers строчку вида:

testuser ALL=/sbin/fdisk

1) Снова переключитесь на учётную запись testuser

su - testuser

2) Еще раз попробуйте выполнить команду "/sbin/fdisk /dev/sda"

/sbin/fdisk /dev/sda

3) Запустилась ли она теперь? _____

4) Почему?

_____

5) Снова попробуйте выполнить команду "/sbin/fdisk /dev/sda" с помощью sudo

sudo /sbin/fdisk /dev/sda

6) Запустилась ли она теперь? _____

7) Завершите работу команды fdisk, нажав клавишу "q"

8) Вернитесь к своей учётной записи exit

9) Ознакомьтесь с содержимым файла /var/log/secure

sudo tail -n 15 /var/log/secure