**9. Привилегии**

Операционные системы, следующие традициям Unix, отличаются от систем, следующих традициям MS-DOS, тем, что являются не только *многозадачными*, но и *многопользовательскими*.

Что это означает на самом деле? Это означает, что компьютером могут одновременно пользоваться несколько человек. Несмотря на то что обычно компьютер имеет всего одну клавиатуру и монитор, это обстоятельство не мешает совместному пользованию. Например, если компьютер подключен к локальной сети или к Интернету, удаленные пользователи смогут зайти на него через ssh (secure shell — безопасная командная оболочка) и выполнять операции. Фактически удаленные пользователи могут запускать приложения с графическим интерфейсом и получать изображение на удаленном дисплее. X Window System поддерживает такую возможность изначально.

Поддержка многопользовательского режима работы — не недавнее «изобретение» Linux, а возможность, глубоко внедренная в архитектуру операционной системы. Учитывая окружение, в котором создавалась система Unix, это имело определенный смысл. В те времена, когда компьютеры еще не были «персональными», они были большими и дорогими. Типичная компьютерная система университета, например, состояла из большого центрального компьютера в одном здании и терминалов, разбросанных по всему университетскому городку и соединенных с большим центральным компьютером. Компьютер мог одновременно обслуживать множество пользователей.

Чтобы подобная возможность имела практическую ценность, необходим способ определенной «изоляции» пользователей друг от друга. В конце концов, действия рядового пользователя не должны приводить к аварийному завершению работы компьютера, и ни один пользователь не должен иметь возможность вносить изменения в файлы, принадлежащие другому пользователю.

В данной главе мы рассмотрим эту важную сторону безопасности системы и познакомимся со следующими командами:

• id — выводит информацию об идентичности пользователя.

• chmod — изменяет режим доступа к файлу.

• umask — определяет разрешения доступа к файлам по умолчанию.

• su — запускает командную оболочку от имени другого пользователя.

• sudo — выполняет команду от имени другого пользователя.

• chown — изменяет владельца файла.

• chgrp — изменяет группу файла.

• passwd — изменяет пароль пользователя.

**Владельцы, члены группы и все остальные**

Знакомясь с системой в главе 4, вы уже сталкивались со следующей проблемой при исследовании файлов, таких как */etc/shadow*:

[me@linuxbox ~]$ **file /etc/shadow**

/etc/shadow: Обычный файл, нет прав на чтение

[me@linuxbox ~]$ **less /etc/shadow**

/etc/shadow: Отказано в доступе

Причина этого сообщения об ошибке заключается в том, что обычные пользователи не имеют права читать этот файл.

В модели безопасности Unix пользователь может *владеть* файлами и каталогами. Если пользователь владеет файлом или каталогом, он может управлять доступом к нему. Пользователи могут также принадлежать *группе*, состоящей из одного или нескольких пользователей, и получить права доступа к файлам и каталогам для членов группы, которые определяются владельцами. Кроме прав доступа для группы, владелец может также определить некоторые права доступа для всех остальных, их в терминологии Unix называют *мир* (world). Получить информацию о своей идентичности можно с помощью команды id:

[me@linuxbox ~]$ **id**

uid=500(me) gid=500(me) groups=500(me)

Давайте рассмотрим этот вывод. Когда создается учетная запись пользователя, ей присваивается число, которое называют *идентификатором пользователя* (user ID), или *uid*. Это число, исключительно ради удобства человека, отображается как имя пользователя. Пользователю назначается *идентификатор основной группы* (primary group ID), или *gid*, и дополнительно пользователь может включаться в состав других групп. Предыдущий пример взят из системы Fedora. В других системах, таких как Ubuntu, вывод команды может немного отличаться.

[me@linuxbox ~]$ **id**

uid=1000(me) gid=1000(me)

groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),  
46(plugdev),108(lpadmin),114(admin),1000(me)

Как видите, числа uid и gid отличаются. Это объясняется тем, что в Fedora нумерация учетных записей обычных пользователей начинается с 500, тогда как в Ubuntu — с 1000. Кроме того, пользователь в Ubuntu принадлежит множеству других групп. Это связано с особенностями управления привилегиями доступа к системным устройствам и службам в Ubuntu.

А где же вся эта информация хранится? Как и многое другое в Linux, она хранится в паре текстовых файлов. Учетные записи пользователей хранятся в файле */etc/passwd*, а информация о группах — в файле */etc/group*. Когда создаются новые учетные записи и группы, эти файлы изменяются вместе с файлом */etc/shadow*, где хранится информация о пароле пользователя. Для каждой учетной записи в файле */etc/passwd* определяется имя пользователя (для входа), числовой идентификатор пользователя (uid), числовой идентификатор основной группы (gid), действительное имя пользователя, путь к домашнему каталогу и командная оболочка входа (login shell). Заглянув внутрь */etc/passwd* и */etc/group*, можно заметить, что помимо учетных записей обычных пользователей здесь также хранятся учетные записи суперпользователя (uid 0) и различных других системных пользователей.

В главе 10, где рассказывается о процессах, вы узнаете, что некоторые из этих других «пользователей» в действительности существуют не просто так.

Несмотря на то что во многих Unix-подобных системах обычных пользователей включают в общую группу, такую как *users*, в современных дистрибутивах Linux принято создавать для каждого пользователя свою, уникальную группу с одним членом и именем, совпадающим с именем пользователя. Это упрощает распределение определенных типов привилегий.

**Чтение, запись и выполнение**

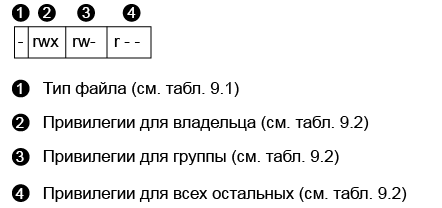
Права доступа к файлам и каталогам определяются в терминах права на чтение, права на запись и права на выполнение. Если взглянуть на вывод команды ls, можно увидеть некоторые подсказки о том, как эти права реализованы:

[me@linuxbox ~]$ **> foo.txt**

[me@linuxbox ~]$ **ls -l foo.txt**

-rw-rw-r-- 1 me    me   0 2012-03-06 14:52 foo.txt

Первые 10 символов в выводе — это *атрибуты файла* (рис. 9.1). Первый из этих символов определяет *тип файла*. В табл. 9.1 перечислены типы файлов, которые чаще всего встречаются на практике (существуют также другие, реже используемые типы файлов). Остальные девять символов в атрибутах файла называются *режимом доступа к файлу* и представляют права на чтение, запись и выполнение для владельца файла, группы — владельца файла и всех остальных.



**Рис. 9.1.** Атрибуты файла

Установленные атрибуты режима r, w и x оказывают определенное влияние на файлы и каталоги, как показано в табл. 9.2.

**Таблица 9.1. Типы файлов**

|  |  |
| --- | --- |
| **Атрибут** | **Тип файла** |
| - | Обычный файл |
| d | Каталог |
| l | Символическая ссылка. Обратите внимание, что для символических ссылок все остальные атрибуты имеют значение rwxrwxrwx и не отражают действительные права доступа. Фактические права доступа к файлу определяются атрибутами самого файла, на который указывает символическая ссылка |
| c | *Специальный файл символьного устройства*. Файлы этого типа соответствуют устройствам, таким как терминал или модем, которые обрабатывают данные как потоки байтов |
| b | *Специальный файл блочного устройства*. Файлы этого типа соответствуют устройствам, таким как привод жесткого диска или CD-ROM, которые обрабатывают данные блоками |

**Таблица 9.2. Атрибуты прав доступа**

| **Атрибут** | **Файлы** | **Каталоги** |
| --- | --- | --- |
| r | Разрешается открывать и читать содержимое файла | Разрешается читать содержимое каталога, если вместе с этим атрибутом установлен атрибут права на выполнение |
| w | Разрешается записывать в файл или усекать его; однако этот атрибут не дает права переименовывать и удалять файлы. Возможность переименования и удаления файлов определяется атрибутами вмещающего каталога | Разрешается создавать, удалять и переименовывать файлы внутри каталога, если вместе с этим атрибутом установлен атрибут права на выполнение |
| x | Разрешается интерпретировать файл как программу и выполнять ее. Файлы, содержащие программы на языках сценариев, дополнительно должны быть доступны для чтения, иначе они не будут выполняться | Разрешается входить в каталог, то есть выполнять команду cd для перехода в него |

В табл. 9.3 приводится несколько примеров установки атрибутов файлов.

**Таблица 9.3. Примеры установки атрибутов прав доступа к файлам**

|  |  |
| --- | --- |
| **Атрибуты файлов** | **Значение** |
| -rwx------ | Обычный файл, доступный владельцу для чтения, записи и выполнения. Никто другой не имеет прав доступа к файлу |
| -rw------- | Обычный файл, доступный владельцу для чтения и записи. Никто другой не имеет прав доступа к файлу |
| -rw-r--r-- | Обычный файл, доступный владельцу для чтения и записи. Члены группы имеют право читать файл. Все остальные имеют право читать файл |
| -rwxr-xr-x | Обычный файл, доступный владельцу для чтения, записи и выполнения. Все остальные имеют право читать и выполнять файл |
| -rw-rw---- | Обычный файл, доступный для чтения и записи только владельцу и членам группы |
| Lrwxrwxrwx | Символическая ссылка. Все символические ссылки имеют недействительные значения атрибутов. Фактические права доступа к файлу определяются атрибутами самого файла, на который указывает символическая ссылка |
| drwxrwx--- | Каталог. Владелец и члены группы могут входить в каталог, создавать, переименовывать и удалять файлы внутри каталога |
| drwxr-x--- | Каталог. Владелец может входить в каталог, создавать, переименовывать и удалять файлы внутри каталога. Члены группы могут входить в каталог, но не могут создавать, переименовывать и удалять файлы внутри каталога |

**chmod — изменение режима доступа к файлу**

Для изменения режима (прав) доступа к файлу или каталогу используется команда chmod. Имейте в виду, что права доступа к файлу или каталогу может изменить только владелец. Команда chmod поддерживает два разных способа изменения режима: с использованием восьмеричных чисел и символического представления. Сначала рассмотрим использование восьмеричных чисел.

**А почему именно восьмеричное?**

*Восьмеричная* (по основанию 8) и родственная ей *шестнадцатеричная* (по основанию 16) системы счисления часто используются для представления чисел в компьютерах. Мы, люди, рождаемся с десятью пальцами на руках (по крайней мере большинство из нас), поэтому для счета используем систему счисления с основанием 10. Компьютеры, напротив, рождаются с одним пальцем и потому используют для вычисления *двоичную* систему счисления (по основанию 2). Их числа состоят всего из двух цифр, нуля и единицы. Поэтому в двоичной системе счет выглядит так: 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011...

В восьмеричной системе используются цифры от нуля до семи: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21...

В шестнадцатеричной системе используются цифры от нуля до девяти плюс буквы от A до F: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13...

В двоичной системе счисления еще можно увидеть смысл (поскольку компьютеры имеют лишь один палец), но в чем польза восьмеричной и шестнадцатеричной систем счисления? Они были придуманы для удобства человека. Очень часто небольшие порции данных представляются в компьютерах битовыми шаблонами. Примером может служить представление цвета в формате RGB. В большинстве дисплеев компьютеров цвет каждого пикселя определяется тремя цветовыми составляющими: 8 бит для красного цвета, 8 бит для зеленого и 8 бит для синего. Красивый сине-голубой цвет можно представить в виде 24-разрядного числа: 010000110110111111001101.

Хотели бы вы видеть и читать такие числа весь день? Я так не думаю. Именно в таких случаях на выручку приходят другие системы счисления. Каждая цифра в шестнадцатеричной системе счисления представляет четыре двоичные цифры. В восьмеричной системе каждой цифре соответствуют три двоичные цифры. То есть 24-разрядное значение сине-голубого цвета можно сжать до 6-значного шестнадцатеричного числа: 436FCD. Поскольку цифры в шестнадцатеричных числах «выстраиваются в ряд» с битами в двоичных числах, можно заметить, что красный компонент нашего цвета имеет значение 43, зеленый — 6F и синий — CD.

В наше время шестнадцатеричная форма записи получила большее распространение, чем восьмеричная, но, как будет показано ниже, восьмеричная форма записи все еще оказывается весьма полезной для представления групп из трех двоичных битов.

**Восьмеричное представление**

При использовании восьмеричной формы записи шаблон желаемых привилегий определяется восьмеричными числами. Так как каждая цифра в восьмеричном числе определяется тремя двоичными разрядами, она точно отображается в схему хранения режима доступа к файлу. В табл. 9.4 поясняется, что мы имеем в виду.

**Таблица 9.4. Режимы доступа к файлу в двоичном и восьмеричном представлениях**

|  |  |  |
| --- | --- | --- |
| **Восьмеричное** | **Двоичное** | **Режим доступа** |
| 0 | 000 | --- |
| 1 | 001 | --x |
| 2 | 010 | -w- |
| 3 | 011 | -wx |
| 4 | 100 | r-- |
| 5 | 101 | r-x |
| 6 | 110 | rw- |
| 7 | 111 | rwx |

C помощью трех восьмеричных цифр мы можем определить режим доступа к файлу для владельца, для группы и для остального мира.

[me@linuxbox ~]$ **> foo.txt**

[me@linuxbox ~]$ **ls -l foo.txt**

-rw-rw-r-- 1 me    me   0 2012-03-06 14:52 foo.txt

[me@linuxbox ~]$ **chmod 600 foo.txt**

[me@linuxbox ~]$ **ls -l foo.txt**

-rw------- 1 me    me   0 2012-03-06 14:52 foo.txt

Передав аргумент 600, мы установили права для владельца, позволяющие ему читать данные из файла и записывать их в файл, и при этом отобрали все права у группы и остального мира. Несмотря на кажущееся неудобство необходимости запоминания соответствий между восьмеричными и двоичными представлениями, вам, скорее всего, придется использовать лишь несколько наиболее популярных шаблонов: 7 (rwx), 6 (rw-), 5 (r-x), 4 (r--) и 0 (---).

**Символическое представление**

Команда chmod поддерживает также символическую форму определения режимов доступа к файлу. Символическая форма записи делится на три части: для кого устанавливаются разрешения, какие операции с разрешениями будут выполняться и на какие разрешения эти операции будут влиять. Чтобы указать, для кого устанавливаются разрешения, используется комбинация символов *u*, *g*, *o* и *a*, как показано в табл. 9.5.

**Таблица 9.5. Символическая форма записи аргументов команды chmod**

|  |  |
| --- | --- |
| **Символ** | **Значение** |
| u | Сокращенно от *user* (пользователь), означает владельца файла или каталога |
| g | Группа |
| o | Сокращенно от *other* (другие, остальные), означает весь остальной мир |
| a | Сокращенно от *all* (все); комбинация из всех трех символов: *u*, *g* и *o* |

Если не указан ни один символ, предполагается a (*all* — все). Операцией может быть знак +, соответствующий добавлению заданных разрешений, знак -, соответствующий отъему заданных разрешений, или знак =, указывающий, что только заданные разрешения должны быть установлены, а все остальные отобраны.

Разрешения определяются символами r, w и x. В табл. 9.6 перечислены некоторые примеры символической формы записи.

**Таблица 9.6. Примеры символической формы записи прав доступа к файлам**

|  |  |
| --- | --- |
| **Атрибуты файлов** | **Значение** |
| u+x | Добавляет право на выполнение, но только для владельца |
| u-x | Отнимает право на выполнение у владельца |
| +x | Добавляет право на выполнение для владельца, группы и остального мира. Эквивалент записи a+x |
| o-rw | Отнимает право на чтение и запись у всех, кроме владельца и группы |
| go=rw | Устанавливает право на чтение и запись для всех, кроме владельца. Если прежде файл имел разрешение на выполнение для группы и всего мира, это право отнимается |
| u+x,go=rx | Добавляет право на выполнение для владельца и устанавливает право на чтение и выполнение для группы и всего мира. При выполнении сразу нескольких операций с привилегиями они должны разделяться запятой |

Кто-то предпочитает пользоваться восьмеричной формой записи, кому-то больше нравится символическая. Символическая форма записи удобна тем, что позволяет установить единственный атрибут, не влияя на остальные.

Дополнительную информацию и полный список параметров команды chmod можно найти на странице справочного руководства (man). А теперь несколько слов о параметре --recursive: он воздействует и на файлы, и на каталоги, поэтому он не так полезен, как можно было бы предположить, потому что редко требуется устанавливать одинаковые разрешения для файлов и каталогов.

**Установка режима доступа к файлу с помощью графического интерфейса**

Теперь, ознакомившись с тем, как устанавливаются разрешения для файлов и каталогов, вы лучше поймете диалоги установки разрешений в графическом интерфейсе. В Nautilus (GNOME) и Konqueror (KDE) можно щелкнуть правой кнопкой мыши на файле или на каталоге и вывести диалог со свойствами. На рис. 9.2 изображен такой диалог из KDE 3.5.

**Рис. 9.2.** Диалог со свойствами файла из KDE 3.5

Здесь вы видите, какие разрешения установлены для владельца, группы и остального мира. Если в KDE щелкнуть на кнопке **Advanced Permissions** (Дополнительные разрешения), появится другой диалог, в котором можно будет установить атрибуты режима по отдельности. Еще один маленький шаг человека в большом мире под названием Командная строка!

**umask — определение разрешений доступа к файлам по умолчанию**

Команда umask определяет разрешения по умолчанию, которые устанавливаются для файла при его создании. В ней с помощью восьмеричной формы записи определяется битовая маска для сбрасываемых атрибутов режима доступа.

Взгляните:

[me@linuxbox ~]$ **rm -f foo.txt**

[me@linuxbox ~]$ **umask**

0002

[me@linuxbox ~]$ **> foo.txt**

[me@linuxbox ~]$ **ls -l foo.txt**

-rw-rw-r-- 1 me    me   0 2012-03-06 14:53 foo.txt

Сначала мы удалили существующий файл *foo.txt*, чтобы, так сказать, начать с чистого листа. Далее мы выполнили команду umask без аргумента, чтобы увидеть текущее значение маски. Она вернула нам значение 0002 (часто также используется значение 0022) — восьмеричное представление действующей маски. Затем мы создали новый файл *foo.txt* и вывели для него разрешения.

Как видите, владелец и группа получили права на чтение и запись, тогда как все остальные — только право на чтение. Весь мир не получил права на запись из-за значения маски. Давайте повторим пример, но на этот раз определим свою маску:

[me@linuxbox ~]$ **rm foo.txt**

[me@linuxbox ~]$ **umask 0000**

[me@linuxbox ~]$ **> foo.txt**

[me@linuxbox ~]$ **ls -l foo.txt**

-rw-rw-rw- 1 me    me   0 2012-03-06 14:58 foo.txt

После установки маски в значение 0000 (таким способом мы фактически выключили ее) вновь созданный файл получил разрешение на запись для всего мира. Чтобы лучше понять суть происходящего, мы снова должны вернуться к восьмеричным числам. Если развернуть маску в двоичное представление и сравнить ее с двоичным представлением атрибутов, можно понять, что произошло:

|  |  |
| --- | --- |
| Исходный режим доступа к файлу | --- rw- rw- rw- |
| Маска | 000 000 000 010 |
| Результат | --- rw- rw- r-- |

Забудем пока про начальные нули (мы вернемся к ним чуть позже) и обратим внимание, что атрибут, соответствующий той позиции, где в маске стоит 1, был сброшен, — в данном случае право на запись для всего мира. Теперь понятно, что делает маска. В любой позиции, где в маске появляется 1, соответствующий атрибут сбрасывается. Если посмотреть на значение маски 0022, легко увидеть, что оно делает:

|  |  |
| --- | --- |
| Исходный режим доступа к файлу | --- rw- rw- rw- |
| Маска | 000 000 010 010 |
| Результат | --- rw- r-- r-- |

И снова атрибуты, соответствующие позициям, где в маске стоит 1, были сброшены. Поэкспериментируйте с другими значениями (попробуйте несколько 7), чтобы лучше усвоить, как действует маска. Закончив эксперименты, не забудьте все вернуть в исходное состояние:

[me@linuxbox ~]$ **rm foo.txt; umask 0002**

**некоторые специальные разрешения**

Обычно разрешения в восьмеричном представлении мы видим как трехзначные числа, но технически более правильно выражать их четырехзначными числами. Почему? Потому что в дополнение к разрешениям на чтение, запись и выполнение существует еще несколько редко используемых разрешений.

Первый атрибут — бит *setuid* (восьмеричное значение 4000). Если это разрешение применяется к выполняемому файлу, в качестве *эффективного идентификатора пользователя* для процесса устанавливается не идентификатор реального пользователя (пользователя, фактически запустившего программу), а идентификатор владельца программы. Чаще этот бит устанавливается для программ, владельцем которых является суперпользователь. Когда обычный пользователь запускает программу с установленным битом *setuid* и принадлежащую пользователю *root*, программа выполняется с эффективными привилегиями суперпользователя. Это дает возможность программам обращаться к файлам и каталогам, недоступным для обычного пользователя. Очевидно, что из-за возникающих проблем безопасности число таких программ в системе должно быть сведено к минимуму.

Второй редко используемый атрибут — бит *setgid* (восьмеричное значение 2000). По аналогии с битом *setuid* он устанавливает *эффективный идентификатор группы* для процесса, выбирая вместо идентификатора группы реального пользователя группу владельца файла. Если установить бит *setgid* для каталога, вновь создаваемые файлы в этом каталоге будут принадлежать группе владельца каталога, а не группе владельца файла, создавшего его. Это разрешение может пригодиться для установки на каталоги, содержимое которых должно быть доступно всем членам основной группы владельца каталога, независимо от принадлежности к основной группе владельца файла.

Третий атрибут называется битом *sticky* (восьмеричное значение 1000). Это пережиток, оставшийся от первых версий Unix, которые предоставляли возможность пометить выполняемый файл как «невытесняемый». Linux игнорирует бит *sticky* у файлов, но если установить его для каталога, он не позволит пользователю удалять или переименовывать файлы, если только пользователь не является владельцем каталога, владельцем файла или суперпользователем. Это разрешение часто применяется для управления доступом к общим каталогам, таким как */tmp*.

Ниже приводится несколько примеров использования chmod с символической формой определения этих специальных разрешений. Первый пример — установка бита *setuid* на файл программы:

chmod u+s program

Далее — установка бита *setgid* на каталог:

chmod g+s dir

Наконец, установка бита *sticky* на каталог:

chmod +t dir

Специальные разрешения мы видим в выводе команды ls. Ниже приводится несколько примеров. Первый — программа с битом *setuid*:

-rwsr-xr-x

Теперь — каталог с атрибутом *setgid*:

drwxrwsr-x

Наконец, каталог с битом *sticky*:

drwxrwxrwt

Вам редко придется изменять маску, потому что значение по умолчанию, устанавливаемое дистрибутивом, прекрасно подходит для большинства нужд. Однако в некоторых ситуациях, требующих повышенной безопасности, маску может понадобиться изменить.

**Изменение идентичности**

Время от времени возникает необходимость приобрести идентичность другого пользователя. Чаще всего требуется получить привилегии суперпользователя, чтобы выполнить некоторые административные задачи, но точно так же можно «превратить» в другого обычного пользователя, чтобы, к примеру, проверить настройки учетной записи. Существует три способа приобрести альтернативную идентичность:

• выйти из системы и войти вновь с учетными данными другого пользователя;

• воспользоваться командой su;

• воспользоваться командой sudo.

Мы пропустим первый способ, потому что уже знаем, как им воспользоваться, и он не так удобен, как два других. В рамках сеанса работы с командной оболочкой команда su позволяет приобрести идентичность другого пользователя и либо начать новый сеанс командной оболочки с идентификатором этого пользователя, либо запустить одиночную команду от его имени. Команда sudo позволяет администратору записать настройки в конфигурационный файл с именем */etc/sudoers* и определить конкретные команды, которые сможет выполнять тот или иной пользователь под приобретенной идентичностью. Выбор между su и sudo в значительной степени определяется используемым дистрибутивом Linux. Большинство дистрибутивов включают обе команды, но в настройках предпочтение отдается той или иной. Начнем с команды su.

**su — запуск командной оболочки с подстановкой идентификаторов пользователя и группы**

Команда su используется для запуска нового сеанса работы с командной оболочкой от имени другого пользователя. Команда имеет следующий синтаксис:

su [-[l]] [*пользователь*]

Если указан параметр -l, запущенная командная оболочка станет оболочкой входа для указанного пользователя. Это означает, что будет загружено окружение пользователя и текущим рабочим каталогом станет домашний каталог пользователя. Часто это именно то, что требуется. Если пользователь не указан, подразумевается суперпользователь. Обратите внимание, что (довольно необычно) параметр -l можно сократить до -, и эта особенность часто используется на практике. Запустить командную оболочку от имени суперпользователя можно следующим образом:

[me@linuxbox ~]$ **su -**

Password:

[root@linuxbox ~]#

После ввода команды будет запрошен пароль суперпользователя. После ввода правильного пароля появится новое приглашение к вводу, показывающее, что данная командная оболочка обладает привилегиями суперпользователя (символ # в конце вместо символа $) и текущим рабочим каталогом теперь стал домашний каталог суперпользователя (обычно */root*). После запуска новой оболочки можно выполнять команды с привилегиями суперпользователя. Завершим работу, введя команду exit, чтобы вернуться в предыдущую командную оболочку:

[root@linuxbox ~]# **exit**

[me@linuxbox ~]$

С помощью su можно так же просто выполнить единственную команду, не запуская новый интерактивный сеанс:

su -c '*команда*'

При использовании этой формы команде su передается единственная командная строка для выполнения. Не забудьте заключить команду в кавычки, чтобы предотвратить дополнительную ее интерпретацию механизмами подстановки текущей командной оболочки:

[me@linuxbox ~]$ **su -c 'ls -l /root/\*'**

Пароль:

-rw------- 1 root root     754 2011-08-11 03:19 /root/anaconda-ks.cfg

/root/Mail:

итого 0

[me@linuxbox ~]$

**sudo — выполнение команды от имени другого пользователя**

Команда sudo во многом подобна команде su, но имеет некоторые важные дополнительные особенности. Администратор может определить порядок использования sudo обычными пользователями, ограничив возможность запуска команд от имени другого пользователя (обычно суперпользователя). В частности, пользователю может быть разрешен доступ к одним командам и запрещен к другим. Еще одно важное отличие состоит в том, что sudo не требует ввода пароля суперпользователя. Для аутентификации в команде sudo пользователь должен ввести свой пароль. Например, допустим, что настройки sudo позволяют выполнить некоторую мифическую программу резервного копирования с именем backup\_script, требующую привилегий суперпользователя.

С помощью sudo ее можно запустить так:

[me@linuxbox ~]$ **sudo backup\_script**

Пароль:

System Backup Starting...

После ввода команды вам будет предложено ввести пароль (ваш, а не суперпользователя), и по завершении аутентификации указанная команда будет выполнена. Одно важное отличие между su и sudo — последняя не запускает новую командную оболочку и не загружает окружение другого пользователя. Это означает, что команды не требуется экранировать как-то иначе, чем при запуске той же команды без использования sudo. Имейте в виду, что такое ее поведение можно переопределить с помощью различных параметров. Подробности ищите на странице справочного руководства (man) для sudo.

**ubuntu и sudo**

Обычные пользователи иногда сталкиваются с необходимостью выполнить некоторую операцию, требующую привилегий суперпользователя. К числу таких операций относится установка и обновление программного обеспечения, правка системных конфигурационных файлов и доступ к устройствам. В мире Windows эта проблема часто решается передачей пользователям административных привилегий, что позволяет им решать подобные задачи. Однако программы, запускаемые такими пользователями, получают те же привилегии. В большинстве случаев это именно то, что нужно, но это также дает возможность беспрепятственной работы вредоносному программному обеспечению, такому как вирусы.

В мире Unix, вследствие многопользовательской природы этой операционной системы, всегда проводилась четкая грань между обычными пользователями и администраторами. Идеология Unix заключается в том, чтобы предоставлять привилегии суперпользователя, только когда они действительно необходимы. Для этого часто используются команды su и sudo.

Еще несколько лет тому назад большинство дистрибутивов Linux использовали с этой целью команду su. Команда su не требует настройки, как команда sudo, а наличие учетной записи *root* — давняя традиция в Unix. Вместе это порождает проблему. Пользователи могут испытывать соблазн действовать от имени root без всякой необходимости. Фактически некоторые пользователи вообще работают в своих системах, регистрируясь исключительно как root, чтобы избежать появления раздражающих сообщений «permission denied» (доступ запрещен). Такой подход ухудшает защищенность Linux, низводя ее до уровня Windows. Не самое лучшее решение.

Создатели Ubuntu предприняли иной подход. По умолчанию Ubuntu запрещает регистрироваться в системе с учетной записью root (не позволяя устанавливать пароль для этой учетной записи), а для получения привилегий суперпользователя предлагает использовать sudo. Начальная учетная запись пользователя обладает полным доступом к привилегиям суперпользователя через sudo и может наделять аналогичными привилегиями другие, вновь создаваемые учетные записи.

Чтобы увидеть, какие привилегии дает команда sudo, вызовите ее с параметром -l:

[me@linuxbox ~]$ **sudo -l**

User me may run the following commands on this host:

    (ALL) ALL

**chown — изменение владельца и группы файла**

Команда chown используется для изменения владельца и группы файла или каталога. Для использования этой команды необходимы привилегии суперпользователя. Команда chown имеет следующий синтаксис:

chown [*владелец*][:[*группа*]] *файл*...

chown может изменить владельца и/или группу файла в зависимости от первого аргумента. В табл. 9.7 приводится несколько примеров команды.

**Таблица 9.7. Примеры аргументов команды chown**

|  |  |
| --- | --- |
| **Аргумент** | **Результаты** |
| bob | Изменит принадлежность файла, назначив владельцем пользователя bob |
| bob:users | Изменит принадлежность файла, назначив владельцем пользователя bob и группу users |
| :admins | Изменит принадлежность файла, назначив группу admins |
| bob: | Изменит принадлежность файла, назначив владельцем пользователя bob и группу этого пользователя |

Представьте, что существуют два пользователя: *janet*, имеющий доступ к привилегиям суперпользователя, и *tony*, лишенный таких привилегий. Пользователю *janet* нужно скопировать файл из своего домашнего каталога в домашний каталог пользователя *tony*. Поскольку пользователь *janet* хочет, чтобы пользователь *tony* смог редактировать файл, *janet* должен изменить владельца скопированного файла, назначив владельцем *tony*:

[janet@linuxbox ~]$ **sudo cp myfile.txt ~tony**

Password:

[janet@linuxbox ~]$ **sudo ls -l ~tony/myfile.txt**

-rw-r--r-- 1 root   root  8031 2012-03-20 14:30 /home/tony/myfile.txt

[janet@linuxbox ~]$ **sudo chown tony: ~tony/myfile.txt**

[janet@linuxbox ~]$ **sudo ls -l ~tony/myfile.txt**

-rw-r--r-- 1 tony   tony  8031 2012-03-20 14:30 /home/tony/myfile.txt

Здесь видно, как пользователь *janet* копирует файл из своего каталога в домашний каталог пользователя *tony*. Далее *janet* заменяет владельца файла *root* (результат использования sudo) на *tony*. Добавив двоеточие в конец первого аргумента, *janet* одновременно изменяет группу, которой принадлежит файл, на основную группу пользователя *tony*, которая, так уж получилось, носит то же имя *tony*.

Заметили ли вы, что после первого использования команда sudo не предложила пользователю *janet* вновь ввести пароль? Это объясняется тем, что в большинстве конфигураций sudo продолжает «доверять» пользователю в течение нескольких минут (пока не истечет время ее действия).

**chgrp — изменение группы файла**

В старых версиях Unix команда chown изменяла только владельца файла, но не группу. Чтобы изменить группу, предоставлялась другая команда, chgrp. Она действует практически так же, как chown, но имеет больше ограничений.

**Использование привилегий**

Теперь, когда мы разобрались, как действует механизм привилегий, самое время научиться пользоваться ими. Далее демонстрируется решение типичной задачи — настройка общего каталога. Представьте себе двух пользователей, *bill* и *karen*. Оба имеют коллекции музыкальных произведений и хотели бы настроить общий каталог, где могли бы хранить файлы в формате Ogg Vorbis или MP3. Пользователь *bill* имеет доступ к привилегиям суперпользователя через sudo.

Первое, что нужно сделать, — это создать группу, куда будут входить оба пользователя, *bill* и *karen*. С помощью графического инструмента GNOME для управления пользователями *bill* создает группу с именем *music* и добавляет в нее пользователей *bill* и *karen*, как показано на рис. 9.3.

**Рис. 9.3.** Создание новой группы в GNOME

Далее *bill* создает каталог для музыкальных файлов:

[bill@linuxbox ~]$ **sudo mkdir /usr/local/share/Music**

Пароль:

Поскольку *bill* манипулирует файлами за пределами своего домашнего каталога, ему необходимы привилегии суперпользователя. После создания каталог получает следующие права доступа и владельца:

[bill@linuxbox ~]$ **ls -ld /usr/local/share/Music**

drwxr-xr-x 2 root root 4096 2012-03-21 18:05 /usr/local/share/Music

Как видите, каталогом владеет *root*, который имеет права доступа 755. Чтобы сделать каталог общим, *bill* должен изменить группу каталога и права доступа для группы:

[bill@linuxbox ~]$ **sudo chown :music /usr/local/share/Music**

[bill@linuxbox ~]$ **sudo chmod 775 /usr/local/share/Music**

[bill@linuxbox ~]$ **ls -ld /usr/local/share/Music**

drwxrwxr-x 2 root music 4096 2012-03-21 18:05 /usr/local/share/Music

И что все это означает? А означает это следующее: владельцем каталога */usr/local/share/Music* является *root*, и члены группы *music* получают права на запись и чтение в этом каталоге. Группа *music* включает пользователей *bill* и *karen*; то есть *bill* и *karen* могут создавать файлы в каталоге */usr/local/share/Music*. Другие пользователи могут просматривать содержимое каталога, но не могут создавать файлы в нем.

Но остается нерешенной еще одна проблема. С текущими разрешениями файлы и каталоги внутри каталога *Music* будут создаваться с обычными разрешениями для пользователей *bill* и *karen*:

[bill@linuxbox ~]$ **> /usr/local/share/Music/test\_file**

[bill@linuxbox ~]$ **ls -l /usr/local/share/Music**

-rw-r--r-- 1 bill   bill   0 2012-03-24 20:03 test\_file

В действительности здесь наблюдаются две проблемы. Во-первых, маска umask в этой системе имеет значение 0022, что не позволяет членам группы записывать в файлы, принадлежащие другим членам группы. Это не проблема, если общий каталог хранит только файлы, но так как в данном каталоге предполагается хранить музыкальные произведения, а музыкальные произведения обычно принято организовывать в иерархии по исполнителям и альбомам, членам группы может понадобиться создавать файлы в каталогах, принадлежащих другим членам. Нам нужно изменить маску umask для пользователей *bill* и *karen* на 0002.

Во-вторых, каждый файл и каталог, созданный одним членом группы, будет принадлежать основной группе пользователя, а не группе *music*. Исправить этот недостаток можно установкой бита *setgid* на каталог:

[bill@linuxbox ~]$ **sudo chmod g+s /usr/local/share/Music**

[bill@linuxbox ~]$ **ls -ld /usr/local/share/Music**

drwxrwsr-x 2 root music 4096 2012-03-24 20:03 /usr/local/share/Music

Теперь можно проверить, устранили ли проблему вновь добавленные разрешения. *bill* устанавливает маску umask в значение 0002, удаляет предыдущий проверочный файл и создает новый проверочный файл и каталог:

[bill@linuxbox ~]$ **umask 0002**

[bill@linuxbox ~]$ **rm /usr/local/share/Music/test\_file**

[bill@linuxbox ~]$ **> /usr/local/share/Music/test\_file**

[bill@linuxbox ~]$ **mkdir /usr/local/share/Music/test\_dir**

[bill@linuxbox ~]$ **ls -l /usr/local/share/Music**

drwxrwsr-x 2 bill   music 4096 2012-03-24 20:24 test\_dir

-rw-rw-r-- 1 bill   music 0 2012-03-24 20:22 test\_file

[bill@linuxbox ~]$

И файл и каталог теперь созданы с правильными правами доступа, позволяющими всем членам группы *music* создавать файлы и каталоги внутри каталога *Music*.

Осталась только проблема с маской umask. Дело в том, что установленная маска действует лишь до конца сеанса и сбрасывается по его завершении. В главе 11 мы узнаем, как сохранить действие измененной маски umask между сеансами.

**Изменение своего пароля**

Последняя тема этой главы: изменение собственного пароля (и паролей других пользователей при наличии привилегий суперпользователя). Для установки и изменения пароля используется команда passwd. Она имеет следующий синтаксис:

passwd [*пользователь*]

Чтобы изменить свой пароль, просто введите команду passwd. Вам будет предложено ввести старый, а затем новый пароль:

[me@linuxbox ~]$ **passwd**

Смена пароля для me.

(текущий) пароль UNIX:

Введите новый пароль UNIX:

Команда пытается вынудить пользователей вводить «сильные» пароли. Это означает, что она будет отвергать слишком короткие пароли, слишком похожие на предыдущие пароли, пароли, являющиеся словарными словами или легко угадываемые:

[me@linuxbox ~]$ **passwd**

Смена пароля для me.

(текущий) пароль UNIX: :

Введите новый пароль UNIX:

BAD PASSWORD: is too similar to the old one

Введите новый пароль UNIX:

Выберите пароль большей длины

Введите новый пароль UNIX :

BAD PASSWORD: it is based on a dictionary word

При наличии привилегий суперпользователя можно передать команде passwd аргумент с именем пользователя, чтобы установить пароль для этого пользователя. Суперпользователю доступна также возможность блокировки учетных записей, установки времени действия пароля и многое другое. За подробностями обращайтесь к странице справочного руководства (man) для команды passwd.