Работа с правами доступа и атрибутами файлов

В Linux система прав доступа определяет, кто и в каком объеме имеет доступ к файлам и директориям. Права доступа к файлам обозначаются с использованием букв r (чтение), w (запись) и x (выполнение). Эти права разделяются между тремя классами пользователей: владельцем файла, группой файла и остальными пользователями.

1. Основы системы прав доступа:

1. **Чтение (r):**
   - *Файл:* Пользователь или группа имеют право читать содержимое файла.
   - *Директория:* Пользователь или группа имеют право просматривать список файлов в директории.
2. **Запись (w):**
   - *Файл:* Пользователь или группа имеют право изменять содержимое файла, включая создание и удаление файлов.
   - *Директория:* Пользователь или группа имеют право создавать, удалять и изменять файлы в директории.
3. **Выполнение (x):**
   - *Файл:* Пользователь или группа имеют право выполнить файл, если он является исполняемым.
   - *Директория:* Пользователь или группа имеют право входить в директорию.

Три варианта записи прав пользователя

двоичная	восьмеричная	символьная	права на файл	права на директорию
000	0	---	нет	нет
001	1	--x	выполнение	чтение файлов и их свойств
010	2	-w-	запись	нет
011	3	-wx	запись и выполнение	всё, кроме чтения списка файлов
100	4	r--	чтение	чтение имён файлов
101	5	r-x	чтение и выполнение	доступ на чтение
110	6	rw-	чтение и запись	чтение имён файлов
111	7	rwx	все права	все права

Представление прав доступа:

Права доступа к файлам представляются в виде строки из 10 символов. Первый символ обозначает тип файла (обычный файл, директория и т. д.), а следующие три группы по три символа каждая обозначают права доступа для владельца, группы и остальных пользователей.

Пример строки прав доступа: **-rwxr-xr--**

- **-** - обычный файл.
- **rwx** - права доступа для владельца (чтение, запись, выполнение).
- **r-x** - права доступа для группы (чтение, выполнение).
- **r--** - права доступа для остальных пользователей (только чтение).

Владелец - это пользователь, который создал файл или директорию.
Группа - это набор пользователей, объединенных для облегчения управления правами доступа. Владелец файла может принадлежать к определенной группе, и группа может иметь свои собственные права доступа к файлу.

Команды для работы с правами доступа:

1. **chmod:**
   - Изменение прав доступа к файлам и директориям.
   - Пример: **chmod +x file** (добавить право выполнения).
2. **chown:**
   - Изменение владельца файла или директории.
   - Пример: **chown user:group file** (изменить владельца и группу).
3. **chgrp:**
   - Изменение группы файла или директории.
   - Пример: **chgrp group file** (изменить группу).

Примеры использования:

1. **Просмотр прав доступа к файлам и директориям:**
   `ls -l`
2. **Изменение прав доступа к файлу:**
   `chmod +rwx filename`
3. **Изменение владельца файла:**
   `chown newowner filename`
4. **Изменение группы файла:**
   `chgrp newgroup filename`

2. Определение текущих прав доступа к файлам и директориям с использованием команды ls -l.

Команда **ls -l** в Linux позволяет отобразить детальную информацию о файлах и директориях в текущем рабочем каталоге. Включая права доступа, владельца, группу, размер, дату последнего изменения и другие атрибуты. Вот как можно определить текущие права доступа с использованием этой команды:

```
ls -l
```

Результат будет примерно следующим:

```
-rw-r--r-- 1 user group 1024 Dec 8 10:00 example.txt drwxr-xr-x 2 user group 4096 Dec 8 10:00 example_directory
```

В приведенном выше примере:

- **rw-r--r--** - это строка, представляющая права доступа к файлу **example.txt**. Первый символ (**-** в данном случае) указывает на тип файла (обычный файл).
- **1** - количество жестких ссылок на файл.
- **user** - владелец файла.
- **group** - группа файла.
- **1024** - размер файла в байтах.
- **Dec 8 10:00** - дата последнего изменения файла.
- **example.txt** - имя файла.

Преобразуем строку **rw-r--r--**:

- Первые три символа (**rw-**) представляют права доступа для владельца (чтение и запись).

- Следующие три символа (**r--**) представляют права доступа для группы (только чтение).
- Последние три символа (**r--**) представляют права доступа для остальных пользователей (только чтение).

В примере с директорией **example_directory**, **drwxr-xr-x** также указывает на права доступа, но буква **d** в начале означает, что это директория.

3. Изменение прав доступа

Изменение прав доступа с использованием символьного представления:

1. **Добавление прав:**
   - Для добавления прав доступа используются символы **+**. Например, чтобы добавить право выполнения для всех пользователей:

     ```
     chmod +x filename
     ```
2. **Удаление прав:**
   - Для удаления прав доступа используются символы **-**. Например, чтобы удалить право записи для группы:

     ```
     chmod g-w filename
     ```
3. **Установка конкретных прав:**
   - Чтобы установить конкретные права, используйте символ **=**. Например, чтобы установить только права чтения для владельца:

     ```
     chmod u=r filename
     ```

Изменение прав доступа с использованием числового представления:

1. **Числовое представление прав:**
   - Каждая комбинация прав представляется числом от 0 до 7:
     - **r** (чтение) = 4
     - **w** (запись) = 2
     - **x** (выполнение) = 1
   - Пример: права **rwx** = 4 + 2 + 1 = 7.
2. **Изменение прав с использованием чисел:**
   - Например, чтобы установить права **rw-r--r--** для владельца, используйте:

     ```
     chmod 644 filename
     ```
3. **Комбинирование чисел:**
   - Первая цифра для владельца, вторая для группы, третья для остальных. Например, чтобы установить **rwxr-xr--**, используйте:

     ```
     chmod 754 filename
     ```

Рекурсивное изменение прав для директорий:

Если вам нужно изменить права для всех файлов внутри директории и ее поддиректорий, используйте опцию **-R** (рекурсивно):

```
chmod -R 755 directory
```

Это изменит права для всех файлов и поддиректорий внутри указанной директории.

Примеры:

1. **Добавление права записи для группы и остальных:**
   `chmod g+w,o+w filename`
2. **Установка прав rwx для владельца, rx для группы и остальных:**
   `chmod 711 filename`
3. **Добавление права записи для всех:**
   `chmod a+w filename`
4. **Установка прав rwxr-xr-x для всех:**
   `chmod 755 filename`

4. Атрибуты SUID, SGID и Sticky Bit.

Атрибуты SUID (Set User ID), SGID (Set Group ID) и Sticky Bit - это специальные атрибуты, которые можно установить на исполняемых файлах и директориях в системах Linux. Эти атрибуты позволяют изменять стандартное поведение файлов в отношении прав доступа и безопасности. Вот их краткое описание:

1. **SUID (Set User ID):**
   - Когда установлен атрибут SUID на исполняемом файле, он выполняется с привилегиями владельца файла, а не тем, кто запускает файл.
   - Это может быть полезным, например, при выполнении программ, которые требуют привилегий пользователя root.
   - Пример: **`chmod +s file`** или **`chmod 4755 file`**.
2. **SGID (Set Group ID):**
   - Когда установлен атрибут SGID на исполняемом файле, он выполняется с привилегиями группы файла, а не группы пользователя, который его запустил.
   - Это может быть полезным, например, для обеспечения доступа к файлам и директориям в рамках определенной группы.
   - Пример: **`chmod +s file`** или **`chmod 2755 file`**.
3. **Sticky Bit:**
   - Когда установлен Sticky Bit на директории, только владелец файла имеет право удалять или изменять файлы в этой директории, даже если у других пользователей есть права записи в этой директории.
   - Обычно используется для директории **`/tmp`**, чтобы предотвратить удаление файлов другими пользователями.
   - Пример: **`chmod +t directory`** или **`chmod 1757 directory`**.

Примеры использования:

1. **SUID:**
   - Установка SUID для выполнимого файла:
     `chmod +s executable_file`
   - Проверка установки SUID:
     `ls -l`

Если атрибут SUID установлен, вы увидите букву "s" вместо буквы "x" в разряде выполнения для владельца файла. Например:
-rwsr-xr-x 1 user group 1024 Dec 8 10:00 executable_file

2. **SGID:**
   - Установка SGID для выполнимого файла:
     bashCopy code
   - chmod g+s executable_file Проверка установки SGID:

```
ls -l executable_file
```

После выполнения этой команды, если вы посмотрите на вывод команды ls -l для файла, вы увидите букву "s" в разряде выполнения для группы файла. Например:
-rwxr-sr-x 1 user group 1024 Dec 8 10:00 executable_file

3. **Sticky Bit:**
   - Установка Sticky Bit для директории:
     ```
     chmod +t directory
     ```
   - Проверка установки Sticky Bit:
     ```
     ls -ld directory
     ```

Если Sticky Bit установлен, вы увидите букву "t" в выводе. Например:
drwxrwxrwt 2 user group 4096 Dec 8 10:00 directory

Примечание:

- Обратите внимание, что использование SUID и SGID может создавать потенциальные уязвимости безопасности, поэтому они должны использоваться осторожно и только при необходимости.
- Использование Sticky Bit на директориях может быть полезным для общедоступных директорий, таких как **/tmp**, чтобы предотвратить удаление файлов другими пользователями.

5. Использование umask

Команда umask задаёт маску прав для новых файлов и каталогов. При создании любого файла операционная система запрашивает маску прав и рассчитывает маску на основе неё. По умолчанию стоит маска ***0002***, Первая цифра ни на что не влияет и является пережитком синтаксиса языка C. Дальше цифры аналогичны правам доступа в Linux: первая - владелец, вторая - группа и третья - все остальные. Эта маска используется для расчета прав файла. Если не вдаваться в подробности, то рассчитывается всё довольно просто, от максимальных прав отнимается маска и получаются права для файла. Фактически, получается, что маска содержит права, которые не будут установлены для файла. Поэтому права по умолчанию для файла будут **666 - 002 = 664**, а для каталога - **777 - 002 = 775**.

Каждую цифру маски 002 можно перевести в двоичную систему. Последняя ***2*** описывает категорию other и в двоичной системе выглядит как ***010***. Биты читаются слева направо и описывают права rwx. В данном примере ***1*** означает запрет на запись, а нули разрешают чтение и выполнение. Если будет стоять битовая маска ***100***, то получится ***4*** в восьмеричной системе, то это будет означать запрет на чтение.

Важное замечание, что с помощью маски не получится разрешить выполнение файлов. Флаг x с помощью маски можно установить только для каталогов. Поскольку права файла рассчитываются на основе прав 666, в которых выполнение уже отключено rw-rw-rw, то маска тут уже ничего сделать не может. Зато для каталогов всё работает, потому что используются права 777. Для наглядности маску по умолчанию можно представить в виде таблицы:

Категория прав	User			Group			Other		
Буквенное обозначение	r	w	x	r	w	x	r	-	x
Битовая маска	0	0	0	0	0	0	0	1	0
Маска в восьмеричном виде	0			0			2		

Синтаксис и опции umask

Команда umask, как было сказано ранее, определяет битовую маску, которая будет применена к новым файлам. У команды довольно простой синтаксис и есть только несколько опций:

```
$ umask опции маска_в_восьмеричном_виде
```

Помимо маски в восьмеричном виде есть и способ задания прав по умолчанию схожий с синтаксисом команды chmod:

```
$ umask опции u=права,g=права,o=права
```

Опции утилиты:
-p - вывести команду umask, которая при выполнении задаст текущую маску в восьмеричном виде;
-S - вывести права по умолчанию для папки в формате u=rwx, g=rwx, o=rwx рассчитанные по текущей маске.
Посмотреть текущее значение маски можно двумя способами. Если команде передать опцию -p, то она выведет команду для установки текущей маски:

```
svv@server22:~$ umask -p
umask 0002
```

Параметр **-S** выводит текущие разрешения в формате **u=rwx, g=rwx, o=rx**, где x (выполнение) относится только к каталогам:

```
svv@server22:~$ umask -S
u=rwx,g=rwx,o=rx
```

6. Команда **chgrp**

Команда **chgrp** в Linux используется для изменения группы, к которой принадлежит файл или директория. Она позволяет вам изменять группу файла, не меняя его владельца (пользователя).

Синтаксис команды **chgrp** следующий:

```
$ chgrp [параметры] новая_группа имя_файла
```

Примеры использования:

1. **Изменение группы файла:**
   `chgrp newgroup file.txt`
   Эта команда изменит группу файла **file.txt** на **newgroup**.
2. **Изменение группы директории и её содержимого (рекурсивно):**
   `chgrp -R newgroup directory`
   Опция **-R** применяет изменения рекурсивно ко всем файлам и поддиректориям внутри указанной директории.
3. **Изменение группы нескольких файлов:**
   `chgrp group1 file1.txt file2.txt`
   Эта команда изменит группу файлов **file1.txt** и **file2.txt** на **group1**.
4. **Изменение группы с использованием числового идентификатора группы (GID):**
   `chgrp 1000 file.txt`
   В этом примере **1000** - это числовой идентификатор группы (GID), и файлу **file.txt** будет назначена группа с этим идентификатором.
5. **Изменение группы с использованием переменной окружения:**
   `chgrp $USER file.txt`
   В этом примере **USER** - это переменная окружения, предоставляющая имя текущего пользователя. Файлу **file.txt** будет назначена группа, соответствующая текущему пользователю.