

```
user@centos8: ~/temp
File Edit View Search Terminal Help
[user@centos8 ~]$ mkdir temp
[user@centos8 ~]$ cd temp/
[user@centos8 temp]$ touch file
[user@centos8 temp]$ stat file
  File: file
  Size: 0                Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 34699313  Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/   user)   Gid: ( 1000/   user)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2021-01-10 22:08:10.455754106 +0400
Modify: 2021-01-10 22:08:10.455754106 +0400
Change: 2021-01-10 22:08:10.455754106 +0400
Birth: -
```

В теме “О файловых системах” мы познакомились с таким понятием, как инода – в ней хранится всякая информация о файле. С помощью утилиты stat мы можем увидеть часть этой информации и сейчас нас интересует 4 строка, где указаны стандартные UNIX права на этот файл - mkdir temp; cd temp; touch file; stat file.

владелец файла – user - u

группа с правами на файл – group - g

остальные – others - o

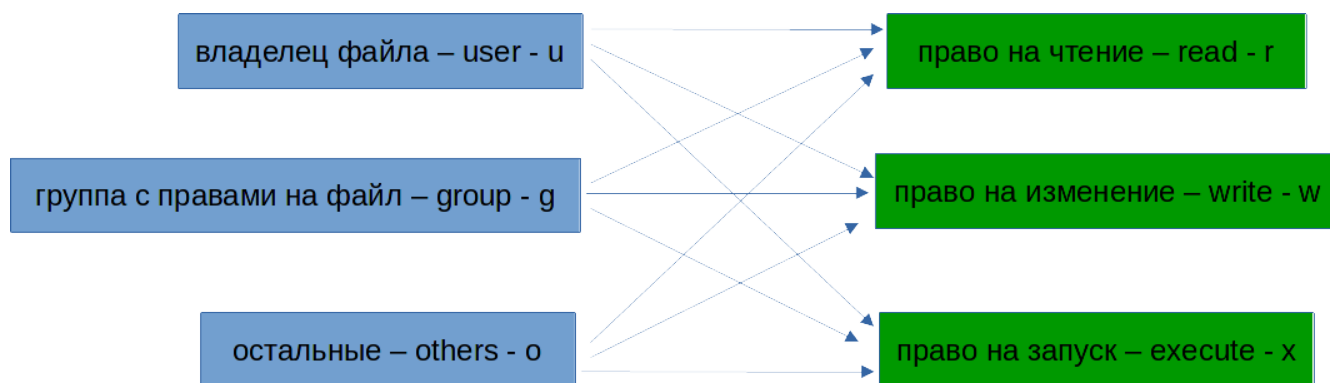
Доступы пользователей к файлам можно разделить на 3 категории: либо пользователь является владельцем этого файла и для него есть обозначение u - user; либо пользователь относится к группе, для которой выделены какие-то права на этот файл – обозначение для них – g - group; либо он относится к остальным – все остальные пользователи, которые не подходят под первую и вторую категорию – для них обозначение o - others. Конечно, есть ещё root – он может делать всё что угодно.

право на чтение – read - r

право на изменение – write - w

право на запуск – execute - x

Что касается самих прав, то они делятся на три типа: право на чтение файла – обозначается как r — read; право на изменение файла – обозначается как w — write; право на запуск файла - обозначается как x – execute. Право на запуск обычно относится к программам и скриптам.



Теперь, указывая для владельца, группы и остальных права на файл, мы получаем такое значение - `gwxgwxgwx`. Первая тройка `gwx` это для владельца, вторая для группы, третья для остальных. Такое обозначение говорит, что мы разрешаем и владельцу, и группе и остальным читать файл, изменять его и запускать.

u	r	w	x
g	r	w	-
o	-	-	-

Если же мы хотим владельцу разрешить всё, группе разрешить только читать и запускать, а остальным ничего, получаем — `gwxr-x---`, где отсутствующие права заменены на дефисы.

```

user@centos8:~/temp
File Edit View Search Terminal Help
[user@centos8 temp]$ ls -l ~
total 20
drwxr-xr-x. 2 user user  6 Oct  2 11:58 Desktop
drwxrwxr-x. 2 user user  6 Nov 23 20:09 dir1
drwxrwxr-x. 2 user user  6 Nov 23 20:09 dir3
drwxr-xr-x. 2 user user  6 Oct  2 11:58 Documents
drwxr-xr-x. 2 user user  6 Oct  2 11:58 Downloads
-rw-rw-r--. 1 user user 2561 Nov 23 20:50 errors
-rw-rw-r--. 1 user user  15 Jan  1 19:45 file
-rw-rw-r--. 1 user user  580 Nov 23 20:43 filelist
drwxr-xr-x. 2 user user  6 Oct  2 11:58 Music
-rw-rw-r--. 1 user user 3490 Nov 23 20:58 output
drwxr-xr-x. 2 user user 4096 Jan 10 22:08 Pictures
  
```

Кроме `stat`, можно использовать `ls -l` или его алиас `ll -ll ~`, где в первом столбце также отображаются права на файлы, в 3 отображается владелец, а в четвёртом группа.

```
user@centos8:~/temp
File Edit View Search Terminal Help
[user@centos8 temp]$ ll file
-rw-rw-r--. 1 user user 0 Jan 10 22:08 file
[user@centos8 temp]$ sudo chown user4 file
[sudo] password for user:
[user@centos8 temp]$ ll file
-rw-rw-r--. 1 user4 user 0 Jan 10 22:08 file
[user@centos8 temp]$
```

Теперь, как это всё менять. Начнём с владельца. Чтобы поменять владельца нужна команды `chown` – change owner. Но даже если мы владелец этого файла, без прав суперпользователя мы это сделать не сможем. Потому что если какой-то пользователь создаст файл, а потом укажет, что владельцем файла является другой пользователь, то он сможет подставить другого пользователя. Или, например, использовать выделенное для второго пользователя место в своих целях, когда каждому пользователю выделено сколько-то места на диске. Поэтому, чтобы поменять владельца, нужно использовать `sudo` — `ll file; sudo chown user4 file, ll file`.

```
user@centos8:~/temp
File Edit View Search Terminal Help
[user@centos8 temp]$ mkdir dir
[user@centos8 temp]$ touch dir/file{1..10}
[user@centos8 temp]$ sudo chown -R user4 dir
[sudo] password for user:
[user@centos8 temp]$ ll dir/
total 0
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file1
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file10
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file2
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file3
-rw-rw-r--. 1 user4 user 0 Jan 10 23:18 file4
```

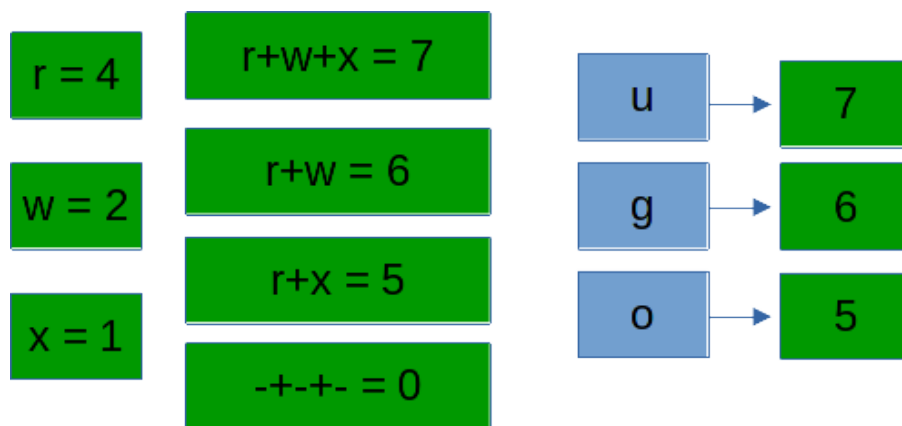
Если мы хотим это сделать для всех файлов в директории и во всех поддиректориях, вы уже знаете, для этого нужно делать рекурсивно - `mkdir dir; touch dir/file{1..10}; sudo chown -R user4 dir; ll dir`.

```
user@centos8:~/temp
File Edit View Search Terminal Help
[user@centos8 temp]$ touch file{1..3}
[user@centos8 temp]$ chown -v :group1 file1
changed ownership of 'file1' from user:user to :group1
[user@centos8 temp]$ sudo chown -v user4:group1 file2
changed ownership of 'file2' from user:user to user4:group1
[user@centos8 temp]$ chgrp group1 file3
[user@centos8 temp]$
```

Группу также можно поменять с помощью `chown - touch file{1..3}; chown -v :group1 file1`. Здесь ключ `-v` - verbose - только для вывода информации. Можно сменить разом владельца и группу — `sudo chown -v user:user file2`. А вообще, чтобы менять группу, есть отдельная команда - `chgrp group1 file3`. Как видите, чтобы менять группу, не нужны права root-а, конечно, если вы владелец файла.

```
user@centos8:~/temp/dir
File Edit View Search Terminal Help
[user@centos8 dir]$ chmod -v u+rw file1
mode of 'file1' retained as 0664 (rw-rw-r--)
[user@centos8 dir]$ chmod -v g-w file2
mode of 'file2' changed from 0664 (rw-rw-r--) to 0644 (rw-r--r--)
[user@centos8 dir]$ chmod -v o+x file3
mode of 'file3' changed from 0664 (rw-rw-r--) to 0665 (rw-rw-r-x)
[user@centos8 dir]$ chmod -v go-r file4
mode of 'file4' changed from 0664 (rw-rw-r--) to 0620 (rw--w----)
[user@centos8 dir]$ chmod -v ugo=r file5
mode of 'file5' changed from 0664 (rw-rw-r--) to 0444 (r--r--r--)
```

Теперь касательно самих прав. Для начала вернём все файлы нашему пользователю, они нам пригодятся - `sudo chown -R user dir; cd dir`. Для смены прав используется команда `chmod`. Допустим, если мы хотим владельцу разрешить читать и изменять файл, пишем `chmod -v u+rw file1`. Хотим группе запретить изменять файл, пишем `chmod -v g-w file2`. Хотим всем разрешить запускать файл, пишем `chmod +x file3`. Группе и остальным запретить читать файл – `chmod -v go-r file4`. `+` добавляет, `-` убавляет. Также можно использовать `=`, чтобы выставить определённые права, допустим, `chmod -v ugo=r file5` – тогда останутся только права на чтение. `chmod` тоже работает рекурсивно с ключом `-R`. В целом по буквам понятно и это несложный способ, но он больше про изменение – кому-то что-то добавить, у кого-то что-то отнять. И если права для всех отличаются – придётся вводить две-три команды, чтобы выставить нужные права.



Часто легче использовать цифровой способ. У каждого права есть своя цифра – у read это 4, у write это 2, у execute это 1. И используя сумму этих чисел можно задать право разом. Допустим, $rw\bar{x}$ – это $4+2+1 = 7$. $rw\bar{x}rw\bar{x}rw\bar{x}$ – это 777. $rw\bar{x}r-x---$ – это 750. $rw\bar{x}rw-r--$ – это 754. Так и пишем – `chmod 765 file6`.

```

user@centos8:~/temp/dir
File Edit View Search Terminal Help
[user@centos8 dir]$ mkdir dir{1..3}
[user@centos8 dir]$ touch dir{1..3}/file
[user@centos8 dir]$ chmod 400 dir1
[user@centos8 dir]$ chmod 600 dir2
[user@centos8 dir]$ chmod 700 dir3
[user@centos8 dir]$ ll dir*
dir1:
ls: cannot access 'dir1/file': Permission denied
total 0
-????????? ? ? ? ? ? ? file

dir2:
ls: cannot access 'dir2/file': Permission denied
total 0
-????????? ? ? ? ? ? ? file

dir3:
total 0
-rw-rw-r--. 1 user user 0 Jan 10 23:57 file

```

Если же говорить о директориях, вспомните, что директория как лист, в котором указаны имена файлов и их иноды – т.е. жёсткие ссылки - `mkdir dir{1..3}; touch dir{1..3}/file; chmod 400 dir1; chmod 600 dir2; chmod 700 dir3; ll dir*`. `read` позволяет видеть только имена файлов – список файлов и директорий — как в `dir1`.

```
user@centos8:~/temp/dir/dir3
File Edit View Search Terminal Help
[user@centos8 dir]$ touch dir2/file1
touch: cannot touch 'dir2/file1': Permission denied
[user@centos8 dir]$ touch dir3/file1
[user@centos8 dir]$ cd dir2
bash: cd: dir2: Permission denied
[user@centos8 dir]$ cd dir3
[user@centos8 dir3]$
```

write позволяет изменять её содержимое - добавлять, переименовывать или удалять файлы – опять же, на самом деле это даже не файлы, а жёсткие ссылки - т.е. «записи на листе». Это как добавлять какие-то записи в лист или стирать их, при этом, не важно, есть у вас права на сами эти файлы или нет - touch dir2/file1; touch dir3/file1. Как вы видите, несмотря на то, что write позволяет изменять содержимое директории, это не работает без execute. А execute позволяет заходить в эту директорию с помощью cd - cd dir2; cd dir3 и выполнять операции с содержимым директории. Т.е. без execute write бесполезен.

```
user@centos8:~/temp/dir
File Edit View Search Terminal Help
[user@centos8 dir]$ ll dir2/
ls: cannot access 'dir2/file': Permission denied
total 0
-?????????? ? ? ? ? ? file
```

```
user@centos8:~/temp/dir
GNU nano 2.9.8 dir2/file
[ Path 'dir2' is not accessible ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit ^R Read File ^\ Replace ^U Uncut Text
```

При этом без execute вы не сможете работать с файлами внутри этой директории, даже если у вас есть права на них nano dir2/file – обратите внимание на ошибку, которая говорит, что данный путь недоступен.

```
user@centos8:~/temp/dir
File Edit View Search Terminal Help
[user@centos8 dir]$ sudo mkdir /home/shared
[user@centos8 dir]$ sudo chgrp group1 /home/shared/
[user@centos8 dir]$ ll -d /home/shared
drwxr-xr-x. 2 root group1 6 Jan 11 00:40 /home/shared
[user@centos8 dir]$ sudo lid -g group1
user4(uid=1111)
user(uid=1000)
[user@centos8 dir]$ sudo chmod 770 /home/shared
[user@centos8 dir]$ ll -d /home/shared
drwxrwx---. 2 root group1 6 Jan 11 00:40 /home/shared
[user@centos8 dir]$
```

Как я сказал, права write и execute позволяют изменять содержимое директории, допустим, удалять файлы, даже если у нас нет права на сами файлы. Допустим, создадим общую директорию для группы пользователей - `sudo mkdir /home/shared`, пусть у неё будет владелец root, а группа group1 - `sudo chgrp group1 /home/shared`; `ll -d /home/shared`, в которую входят user и user4 - `sudo lid -g group1`. И выставим права на директорию - `sudo chmod 770 /home/shared`, чтобы у рута и группы были все права, а у остальных никаких.

```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 ~]$ cd /home/shared/
[user@centos8 shared]$ sudo touch file1
[user@centos8 shared]$ sudo -u user4 touch file2
[user@centos8 shared]$ touch file3
[user@centos8 shared]$ ll
total 0
-rw-r--r--. 1 root root 0 Jan 11 00:47 file1
-rw-r--r--. 1 user4 users 0 Jan 11 00:47 file2
-rw-rw-r--. 1 user user 0 Jan 11 00:47 file3
[user@centos8 shared]$ touch file1
touch: cannot touch 'file1': Permission denied
[user@centos8 shared]$ rm file2
rm: remove write-protected regular empty file 'file2'? y
[user@centos8 shared]$ ll -d .
drwxrwx---. 2 root group1 32 Jan 11 00:49 .
[user@centos8 shared]$
```

Значит, пользователь root, user и user4 могут создавать тут файлы – `cd /home/shared`; `sudo touch file1`; `sudo -u user4 touch file2`; `touch file3`; `ll`. Как видите, тут 3 файла и write права у меня есть только на файл user-a, а другие я изменять не могу — `touch file1`. При этом, я могу запросто

удалить чужие файлы - `rm file2`, потому что у группы `group1` есть права `write` на эту директорию `ll -d`.

```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ sudo chmod +t /home/shared/
[user@centos8 shared]$ ll -d /home/shared
drwxrwx--T. 2 root group1 32 Jan 11 00:49 /home/shared
[user@centos8 shared]$ sudo -u user4 touch file2
[user@centos8 shared]$ rm file2
rm: remove write-protected regular empty file 'file2'? y
rm: cannot remove 'file2': Operation not permitted
[user@centos8 shared]$ rm file3
[user@centos8 shared]$ ll -d /tmp/
drwxrwxrwt. 16 root root 4096 Jan 11 00:56 /tmp/
[user@centos8 shared]$
```

Вы, возможно, подумали – это ж как-то неправильно, нехорошо. Кто-то может случайно или специально удалить чужие файлы. Поэтому есть специальный атрибут – дополнительное право, называемое `sticky bit`, которое защищает файлы внутри директории. Поставить его можно используя буквенное обозначение - `sudo chmod +t /home/shared`, либо используя цифровое обозначение – у стикибита цифра 1 и она ставится перед правами - `sudo chmod 1770 /home/shared`. После того, как вы поставите `sticky bit` на директорию, у неё появится буква `T` после прав - `ll -d /home/shared`, а также `ls` покажет эту директорию по другому. Так вот, теперь вернём файл пользователя `user4` - `sudo -u user4 touch file2` и попытаемся удалить ещё раз - `rm file2`. Как видите, теперь у меня недостаточно прав. Но, как владелец, я могу удалять свои файлы - `rm file3`. А владелец директории, не важно, `root` он или нет, может удалять все файлы. В системе есть директория `/tmp` - `ll -d /tmp`, различные программы в процессе работы могут создавать здесь временные файлы. И чтобы другие процессы не удалили эти файлы здесь стоит `sticky bit`.


```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ ll /usr/bin/passwd
-rwsr-xr-x. 1 root root 33600 Apr  7  2020 /usr/bin/passwd
[user@centos8 shared]$ passwd
Changing password for user user.
Current password: [ ]

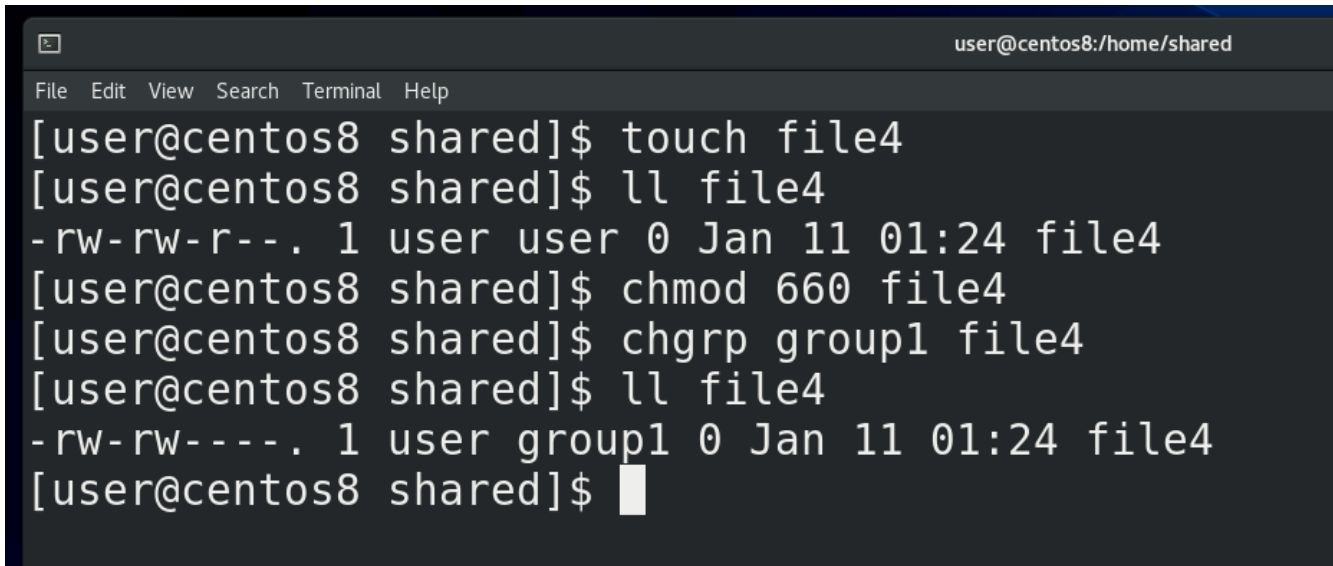
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ ps -ef | grep passwd
root          6140      2836  0 01:02 pts/0      00:00:00 passwd
user          6179      6143  0 01:02 pts/1      00:00:00 grep --col
[user@centos8 shared]$
```

Есть ещё пара специальных атрибутов – `setuid` и `setgid`. Вкратце – они позволяют запускать файл от имени владельца или группы соответственно. Прекрасный пример – программа `passwd` - `ll /usr/bin/passwd`. Как видите, вместо `x` у владельца стоит `s` – это означает что здесь есть атрибут `setuid`, плюс `ls` ярким красным выделяет этот файл. Мы знаем, что `passwd` меняет пароль пользователя. Пользователь запускает программу `passwd`, вводит пароль, программа хэширует пароль и записывает в `/etc/shadow`. Но ведь у обычного пользователя нет прав редактировать файл `/etc/shadow`. И процесс `passwd`, запущенный от имени обычного пользователя, просто не смог бы редактировать этот файл – а значит пароль не поменялся бы. Поэтому здесь стоит `setuid` – когда мы запускаем программу `passwd`, процесс запускается не от имени нашего пользователя - `passwd`; `ps -ef | grep passwd`, а от имени владельца этого файла - `root`-а. А вот процесс, запущенный от `рута`, может редактировать всё что угодно. Если бы владельцем файла являлся `user4`, то программа запускалась бы от имени пользователя `user4`. Вот, собственно, для этого и есть `setuid`.

```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ sudo chmod u+s file1
[sudo] password for user:
[user@centos8 shared]$ sudo chmod g+s file2
[user@centos8 shared]$ sudo chmod 4770 file2
[user@centos8 shared]$
```

`setgid` делает примерно тоже самое, но уже от имени группы этого файла. Задаётся `setuid`, как и `sticky bit` – с помощью букв или чисел – `u+s` для `setuid` и `g+s` для `setgid` - `chmod u+s file`; `chmod g+s file`. Ну и цифрами `setuid` это 4, `setgid` это 2 – всё как в правах. Допустим, чтобы поставить `setuid` и `setgid` - `chmod 6770 file`.

setuid, как и sudo, позволяет повысить привилегии пользователю, а значит это потенциальная брешь в безопасности. Используя программы, где стоит этот атрибут – можно попытаться стать рутом, как я показывал это с sudo. Поэтому использовать setuid вообще нежелательно и он используется только в крайних случаях, как например с passwd.

A terminal window titled 'user@centos8:/home/shared' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows a sequence of commands and their outputs: 1. 'touch file4' is executed. 2. 'll file4' shows the file's permissions as '-rw-rw-r--', owner 'user', group 'user', and timestamp '0 Jan 11 01:24'. 3. 'chmod 660 file4' is executed. 4. 'chgrp group1 file4' is executed. 5. 'll file4' shows the updated permissions as '-rw-rw----', owner 'user', group 'group1', and timestamp '0 Jan 11 01:24'. The prompt '[user@centos8 shared]\$' is shown at the end of the last command.

```
[user@centos8 shared]$ touch file4
[user@centos8 shared]$ ll file4
-rw-rw-r-- . 1 user user 0 Jan 11 01:24 file4
[user@centos8 shared]$ chmod 660 file4
[user@centos8 shared]$ chgrp group1 file4
[user@centos8 shared]$ ll file4
-rw-rw---- . 1 user group1 0 Jan 11 01:24 file4
[user@centos8 shared]$
```

Также, у setgid есть одно особое применение. Если использовать setgid на директорию, то все файлы, создаваемые в этой директории, будут создаваться от имени группы, владеющей этой директорией. Обычно, когда вы создаёте какой-то файл, то владельцем являетесь вы, а группой – ваша основная группа. Это можно увидеть в той shared директории. Смотрите, user создал файл - touch file4; ll file4, и у него группа user, потому что это его user private group, о котором мы говорили в прошлый раз. Пользователь user4 не входит в группу и не является пользователем user, а значит он для этого файла относится к others. Я не хочу, чтобы все пользователи в системе видели этот файл, поэтому меняю права на 660 - chmod 660 file4, но хочу, чтобы у пользователей в группе group1 был доступ к этому файлу - chgrp group1 file4; ll file4. А так как пользователь user4 тоже в этой группе, он сможет редактировать этот файл.

```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ touch file5
[user@centos8 shared]$ ll file5
-rw-rw-r--. 1 user user 0 Jan 11 01:28 file5
[user@centos8 shared]$ sudo chmod g+s /home/shared/
[sudo] password for user:
[user@centos8 shared]$ ll -d /home/shared/
drwxrws--T. 2 root group1 58 Jan 11 01:28 /home/shared/
[user@centos8 shared]$ touch file6
[user@centos8 shared]$ ll file6
-rw-rw-r--. 1 user group1 0 Jan 11 01:30 file6
[user@centos8 shared]$
```

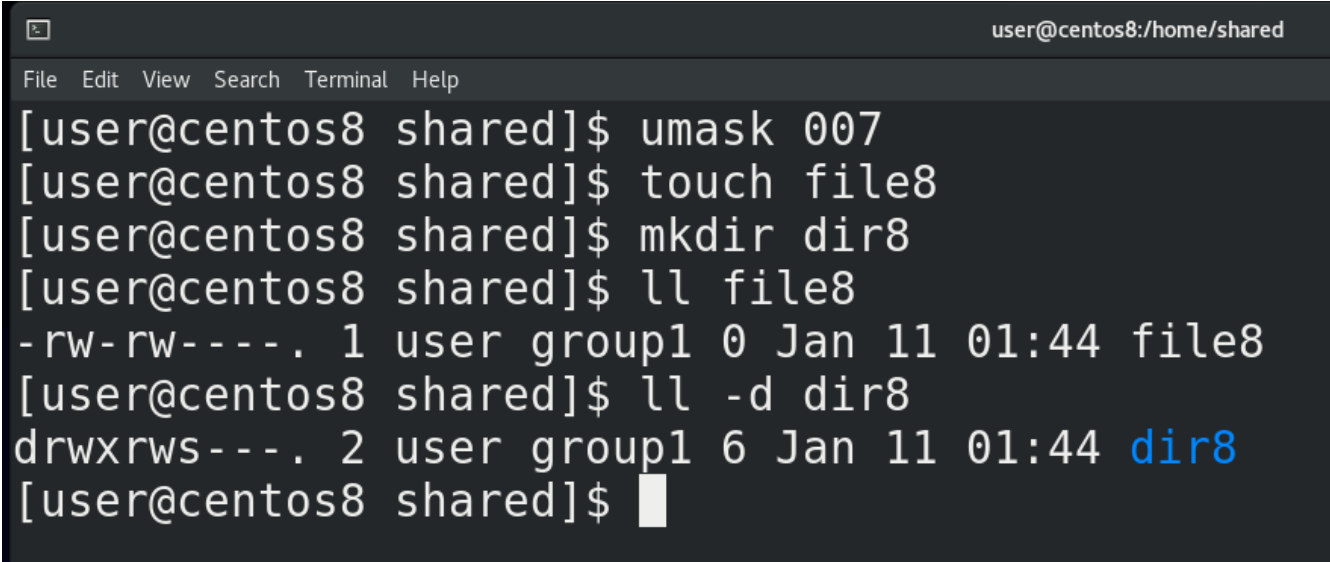
Но если я создам ещё один файл - `touch file5`; `ll file5` – мне придётся опять менять группу для нового файла. Учитывая, что это общая директория, она специально создана для того, чтобы тут несколько пользователей из одной группы работали с файлами, было бы легче, если бы все новые файлы создавались с группой `group1`. И вот для этого можно использовать `setgid` - `sudo chmod g+s /home/shared`; `ll -d /home/shared`. Как видите, для группы теперь стоит `setgid`. И теперь когда я создаю новый файл - `touch file6`; `ll file6` - он автоматически создаётся с этой группой, а не моей основной группой. Все новые файлы в этой директории, независимо от пользователя, будут принадлежать группе `group1`, благодаря чему пользователям не придётся постоянно менять группу файла, чтобы все могли работать с этими файлами.

```
File Edit View Search Terminal Help
[user@centos8 shared]$ umask
0002
[user@centos8 shared]$ umask -S
u=rwx,g=rwx,o=rx
[user@centos8 shared]$
```

Ещё один момент – как вы, возможно, заметили, все файлы, которые я создаю, имеют одни и те же стандартные права – 664. Дефолтные права на новые файлы задаются утилитой `umask`. Если просто запустить `umask`, можно увидеть 0002. Можно ещё запустить `umask -S`, чтобы было понятнее. И так, первая цифра – для sticky bit, `setuid` и `setgid`, остальное для прав. Идея такая – берём максимальное значение – это 777 для директорий и 666 для файлов - и

отнимаем те дефолтные права, которые мы хотели бы. Допустим, если мы хотим, чтобы у всех новых файлов были права 664, мы от 666 отнимаем 664 – получаем 002. Вот у нас 002 и стоит. Ну и если от 777 отнять 002 получим 775.

Вы скажете – для файлов же максимальные права тоже 777. Но вот просто нельзя создавать новые файлы с правами execute, это большая угроза безопасности. Поэтому для файлов дефолтные максимальные права это 666. А с директориями без execute нормально не поработаешь, поэтому для них 777. Если я хочу, чтобы файлы создавались с правами 660, то я от 666 отнимаю 660 – получаю 006. Но если от 777 отнять 006 получится 771, 1 в конце выглядит бессмысленно, поэтому лучше использовать 007 – тогда права для директорий будут 770, а для файлов 660.



```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ umask 007
[user@centos8 shared]$ touch file8
[user@centos8 shared]$ mkdir dir8
[user@centos8 shared]$ ll file8
-rw-rw- - - . 1 user group1 0 Jan 11 01:44 file8
[user@centos8 shared]$ ll -d dir8
drwxrws- - - . 2 user group1 6 Jan 11 01:44 dir8
[user@centos8 shared]$
```

И так, как мне это применить? Я могу это сделать в текущей сессии – написать `umask 007`, и убедиться, создав новый файл и директорию - `touch file8; mkdir dir8; ll file8; ll -d dir8`. Как видите, теперь у новых файлов для остальных пользователей нет никаких прав. Если же мы хотим, чтобы этот `umask` действовал для нашего пользователя всегда, мы добавляем строчку «`umask 007`» в `~/.bash_profile` или `~/.bashrc`. Если помните, `login shell` у нас все равно считывает `~/.bashrc`, а вот если задействуется `non login shell`, то он не прочтёт `bash_profile`, поэтому в некоторых случаях лучше писать в `~/.bashrc`. Ну и если мы говорим про всех пользователей, то используйте файлы `/etc/profile` и `/etc/bashrc`.

Стандартные права делят пользователей на владельца, группу и остальных, что в большинстве случаев достаточно, но иногда всё же нет. Что, если мы хотим дать права на файл всем в группе, кроме двух её участников? Ради этого придётся создавать отдельную группу без этих двух участников. Или, допустим, нужно дать права на 3 группы, а не одну, при этом, дать какому-то пользователю больше прав, какому-то меньше. Для этого можно использовать список контроля доступа – `acl`. В общем-то речь про две команды – `getfacl` и `setfacl`.

```
[user@centos8 shared]$ getfacl file8
# file: file8
# owner: user
# group: group1
user::rw-
group::rw-
other::---

[user@centos8 shared]$ setfacl -m u:user4:r-- file8
[user@centos8 shared]$ setfacl -m g:wheel:rwX file8
[user@centos8 shared]$ ll file8
-rw-rwx---+ 1 user group1 0 Jan 11 01:44 file8
[user@centos8 shared]$ getfacl file8
# file: file8
# owner: user
# group: group1
user::rw-
user:user4:r--
group::rw-
group:wheel:rwX
mask::rwX
other::---

[user@centos8 shared]$ setfacl -b file8
```

getfacl file8 показывает текущие права на файл. Как видим, владельцу user и группе group1 можно читать и изменять этот файл. User4 тоже в группе group1, но я хочу запретить ему изменять этот файл – для этого я использую утилиту setfacl - setfacl -m u:user4:r-- file8. Команда говорит, что нужно модифицировать(-m) права для пользователя user4 и выставить их такими-то. Давайте ещё позволим группе wheel иметь полный доступ на файл - setfacl -m g:wheel:rwX file8. После выставления acl ls показывает рядом с правами значок +, а с помощью getfacl file8 узнаем текущие права на файл. Теперь user4 не может редактировать этот файл - sudo -u user4 nano file8. Чтобы удалить дополнительные права используем ключ -b setfacl -b file8; getfacl file8. Возможно, тему ACL я еще затрону отдельно, но вкратце этого достаточно.

```
user@centos8:/home/shared
File Edit View Search Terminal Help
[user@centos8 shared]$ ll /etc/shadow
----- . 1 root root 1542 Jan 10 19:58 /etc/shadow
[user@centos8 shared]$
```

Ну и напоследок, есть интересный пример с файлом `/etc/shadow` - `ll /etc/shadow`. Как вы видите, на этот файл никаких прав нет, даже у его владельца – `root`-а. Но при этом мы знаем, что при смене пароля с `passwd` новый пароль прописывается в этом файле, да и если открыть этот файл с `папо` – то мы можем читать и изменять этот файл.

```
user@centos8:/home/shared
File Edit View Search Terminal Help
CAP_DAC_OVERRIDE
Bypass file read, write, and execute permission checks. (DAC is
an abbreviation of "discretionary access control".)
```

Суть в том, что все разрешения работы с файлом проверяет ядро операционной системы – оно проверяет, соответствует ли `uid` пользователя, обращающегося к файлу, с `uid`-ом владельца файла на файловой системе, есть ли пользователь в группе и т.п. И при некоторых условиях – когда к файлу обращается залогиненный `root`, в том числе при выполнении команды `passwd`, ядро просто пропускает проверку и сразу даёт доступ к файлу - `man capabilities`, `DAC_OVERRIDE`. А вот для каких-нибудь сервисов, которые работают от имени рута, но запущены, допустим, при включении системы, а не вручную, этот файл недоступен для чтения. А при работе `root` может просто игнорировать все права на файлы.

Подводя итоги, мы с вами разобрали стандартные права – `read`, `write` и `execute`, команды `chown`, `chgrp` и `chmod` для смены прав и владельцев файлов, атрибуты `sticky bit`, `setuid` и `setgid`, права по умолчанию – `umask`, дополнительные права – `acl`. Администраторы всегда что-то делают с правами, тема хоть и простая, но может иметь много всяких нюансов, которые можно встретить при работе. Также для лучшего понимания советую почитать [статью](#) по ссылке.