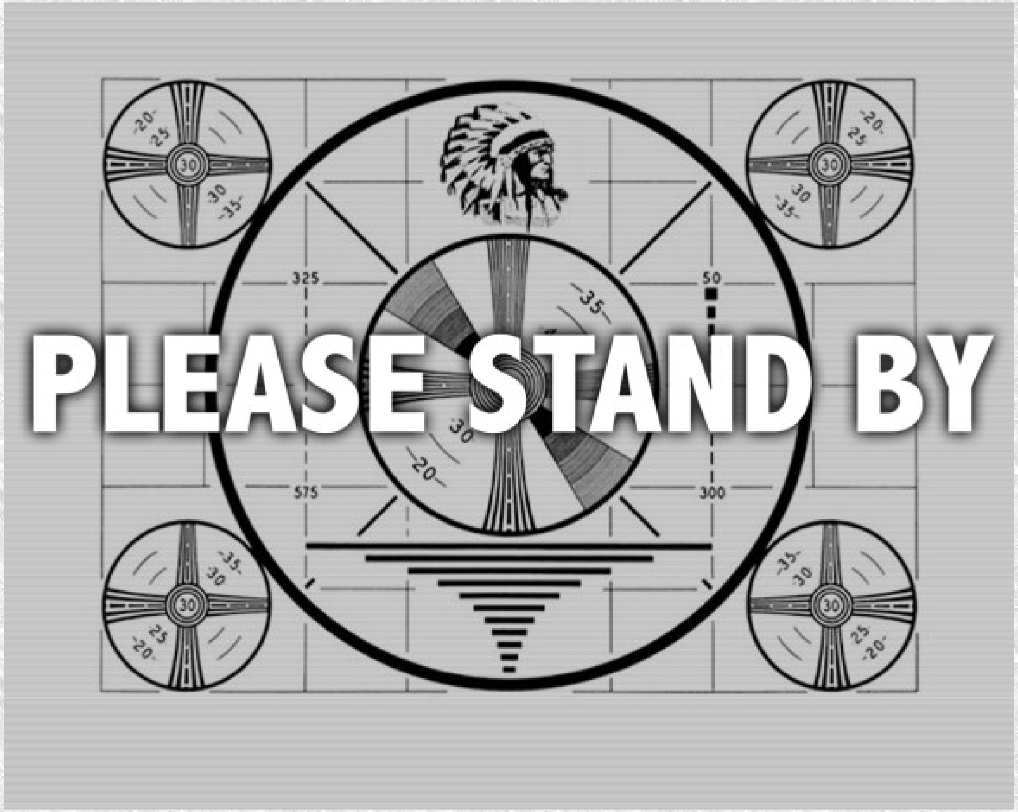
Stand By



Welcome

A basic tutorial about UNIX/Linux

Permissions

...of files, directories, and more if we have time.

by Stan Reichardt

stanr@sluug.org

Maybe

- If time permits there will be brief coverage of directory permissions, special permissions, file attributes and Access Control Lists (ACL).
-

Files

A file is not merely its contents, a name, and a file type.

A file also has an owner (a user ID), a group (a group ID), permissions (what the owner can do with the file, what people in the group can do, and what everyone else can do), various timestamps, and other information.

-- quote extracted from info page on ??????

Looking for file permissions

-
  ```
  user@example:~$ whatis permissions
  permissions: nothing appropriate.
  ```

-
  ```
  user@example:~$ man permission
  No manual entry for permission
  ```

-
  ```
  user@example:~$ man permissions
  No manual entry for permissions
  ```

-
  ```
  user@example:~$ whatis file
  file (1)             - determine file type
  ```

-
  ```
  user@example:~$ man file (doesn’t talk about file permissions)
  ```

-
  ```
  user@example:~$ man files
  No manual entry for files
  ```

user@example:~$ apropos permission

- user@example:~$ apropos permission
- access (2)   - check user's permissions for a file
- chmod (2)   - change permissions of a file
- eaccess (3)   - check effective user's permissions for a file
- euidaccess (3)   - check effective user's permissions for a file
- faccessat (2)   - check user's permissions for a file
- faked (1)   - daemon that remembers fake ownership/permissions of files manipulated by fakeroot processes.
- faked-sysv (1)   - daemon that remembers fake ownership/permissions of files manipulated by fakeroot processes.
- faked-tcp (1)   - daemon that remembers fake ownership/permissions of files manipulated by fakeroot processes.
- fchmod (2)   - change permissions of a file
- fchmodat (2)   - change permissions of a file
- flatpak-permission-list (1) - List permissions
- flatpak-permission-remove (1) - List permissions
- flatpak-permission-reset (1) - Reset permissions
- flatpak-permission-show (1) - List permissions
- ioperm (2)   - set port input/output permissions
- WWW::RobotRules (3pm) - database of robots.txt-derived permissions

pinfo

- I prefere to use **pinfo** instead of the usual **info** command.
- user@example:~$ sudo apt-get install pinfo ## on Debian systems
- user@example:~$ whatis info   ## might require installation

  info (5)   - readable online documentation

  info (1)   - read Info documents
- user@example:~$ whatis pinfo   ## might require installation

  pinfo (1)   - curses based lynx-style info browser

Remember This!

- Some things to remember:
  - The usual **man pages** are of little to no help learning about permissions.
  - If a **man page** does not exist, there may be an **info page**.
  - The **info pages** may provide information beyond expected of **man pages**.
  - The **pinfo** command has more features than the **info** command.
  - The **pinfo** command navigation is similar to **vi/vim** navigation
- ★ The one most important thing to remember from this tutorial:
  - Using the **info pages** gives you the best description of permissions.
  - user@example:~$ pinfo File permissions

pinfo file permissions

Each file has a set of "file mode bits" that control the kinds of access that users have to that file. They can be represented either in symbolic form or as an octal number.

-- quote extracted from info page on File permissions

user@example:~$ aptitude show info

```
user@example:~$ aptitude show info

Package: info            Version: 6.5.0.dfsg.1-2            State: installed            Automatically installed: no

Multi-Arch: foreign   Priority: standard            Section: doc

Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>

Architecture: amd64                 Uncompressed Size: 599 k          Depends: libc6 (>= 2.15), libtinfo5 (>= 6), install-info

Suggests: texinfo-doc-nonfree                 Conflicts: info:i386          Replaces: texinfo (< 4.7-2), texinfo:i386 (< 4.7-2)

Provides: info-browser, info-browser:i386, info:i386 (= 6.5.0.dfsg.1-2)              Provided by: info:i386 (6.5.0.dfsg.1-2)

Description: Standalone GNU Info documentation browser

The Info file format is an easily-parsable representation for online documents. This program allows you to
view Info documents, like the ones stored in /usr/share/info.

 Much of the software in Debian comes with its online documentation in the form of Info files, so it is most

likely you will want to install it.
```

user@example:~$ aptitude show pinfo

```
user@example:~$ aptitude show pinfo

Package: pinfo                    Version: 0.6.9-5.2     State: installed        Automatically installed: no

Priority: optional       Section: universe/doc

Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>     Architecture: amd64
Uncompressed Size: 237 k

Depends: libc6 (>= 2.4), libncursesw5 (>= 6), libreadline7 (>= 6.0), libtinfo5 (>= 6), install-info

Suggests: mutt | mail-reader, w3m | www-browser, cups-bsd | lpr                    Conflicts: pinfo:i386
Provides: info-browser

Description: An alternative info-file viewer

pinfo is an viewer for Info documents, which is based on ncurses. The key-commands are in the style of lynx.

Homepage: http://pinfo.alioth.debian.org/
```

man 5 password

- The USERID and GROUPID in the **/etc/passwd** file are used to identify file permissions of each file.
- user@example:~$ man 5 passwd ## to see password file format
- ***NOTE: We have to juggle a bit here within the pinfo command.***
- user@example:~$ pinfo shadow ## jump to passwd(1)
- user@example:~$ pinfo shadow ## jump to passwd(5)

pinfo shadow

DEMONSTRATION:

Here, I have to jiggle this a bit, as a work-around is needed because of some oddities in the way pinfo doesn’t seem to work as I exepect it to work. (bug?)

- user@example:~$ pinfo shadow ## jump to group

user@example:~$ stat permissions.txt

- user@example:~$ stat permissions.txt

- stat (1)		- display file or file system status
- stat (2)		- get file status

```
File: permissions.txt
Size: 812          Blocks: 8        IO Block: 4096   regular file
Device: 806h/2054d     Inode: 49294519    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    stan)   Gid: ( 1000/    stan)
Access: 2020-02-07 21:57:49.054739940 -0600
Modify: 2020-02-07 21:57:07.919102399 -0600
Change: 2020-02-07 21:57:07.975101906 -0600
Birth: -
```

user@example:~$ pinfo file permissions

- user@example:~$ pinfo permissions

```
Przemek's Info Viewer v0.6.9
Error: could not open info file, trying manual
Error: No manual page found
```

- user@example:~$ pinfo file permissions ## used this phrase
- user@example:~$ pinfo file   ## don’t need word “permissions”

Three Permission Sets

The Linux filesystem gives us three types of permissions.

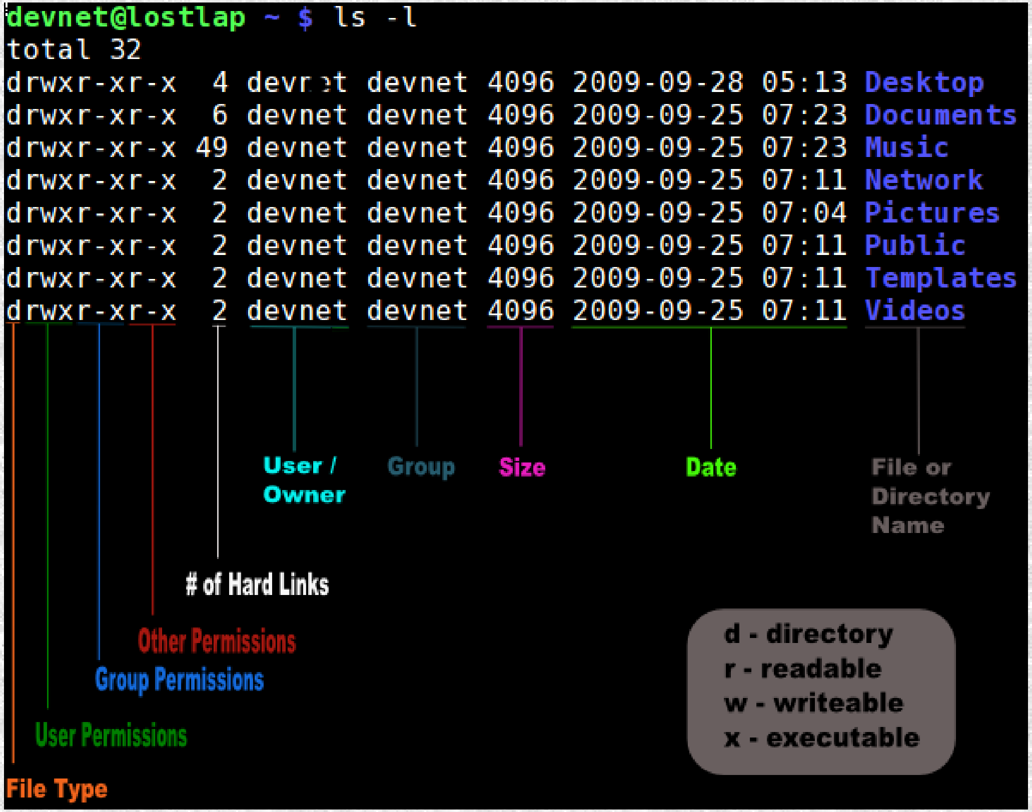
Here is a simplified review:

User (or user owner)

Group (or owner group)

Other (everyone else)

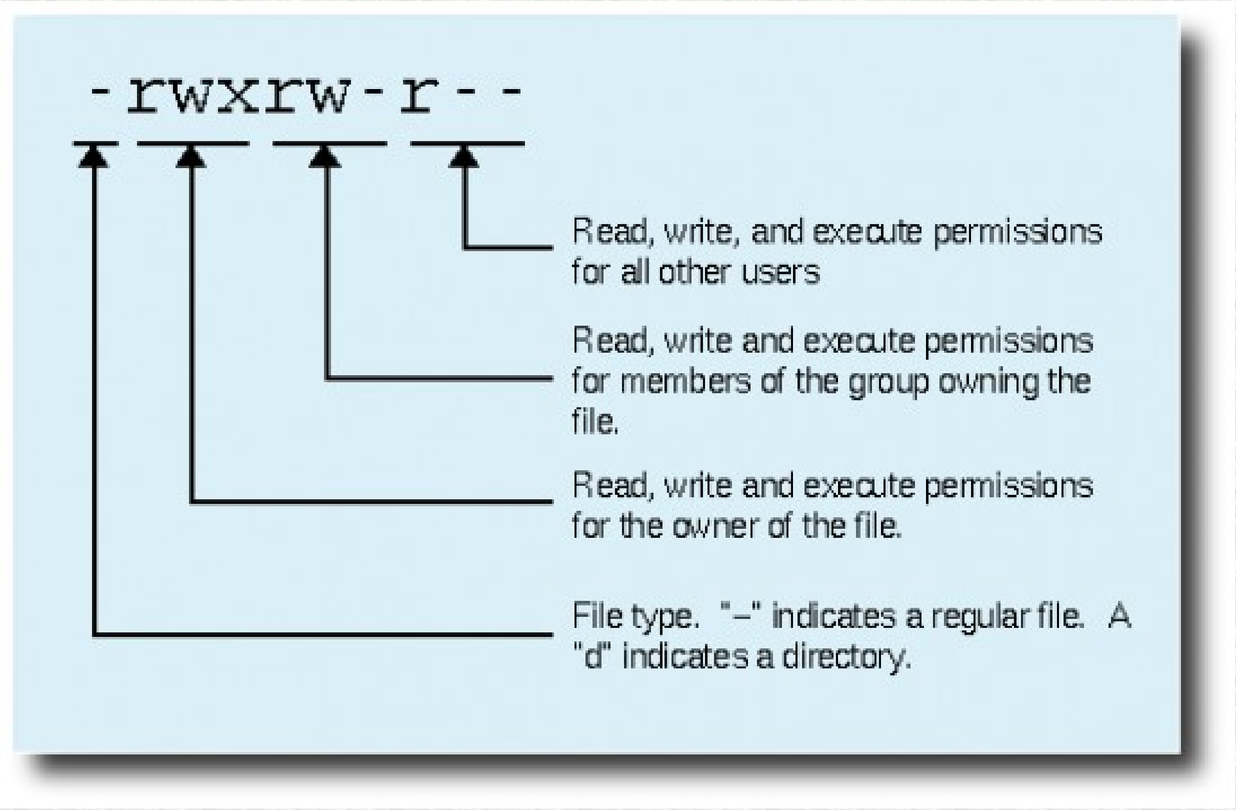
user@example:~$ ls -l



```
devnet@lostlap ~ $ ls -l
total 32
drwxr-xr-x  4 devnet devnet 4096 2009-09-28 05:13 Desktop
drwxr-xr-x  6 devnet devnet 4096 2009-09-25 07:23 Documents
drwxr-xr-x 49 devnet devnet 4096 2009-09-25 07:23 Music
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Network
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:04 Pictures
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Public
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Templates
drwxr-xr-x  2 devnet devnet 4096 2009-09-25 07:11 Videos
```

d - directory
r - readable
w - writeable
x - executable

Permissions ~ Breakdown



```
-rwxrw-r--
```

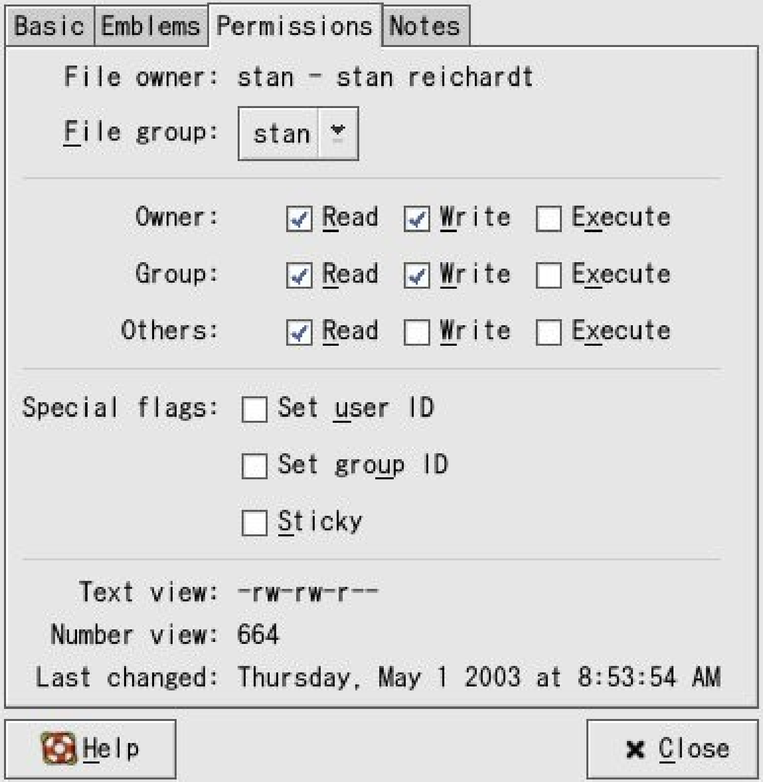
Read, write, and execute permissions for all other users

Read, write and execute permissions for members of the group owning the file.

Read, write and execute permissions for the owner of the file.

File type. "–" indicates a regular file. A "d" indicates a directory.

Controlling with a Grapical Form within a Desktop Environment



File owner: stan - stan reichardt

File group: stan

Owner: ☑ Read ☑ Write □ Execute

Group: ☑ Read ☑ Write □ Execute

Others: ☑ Read □ Write □ Execute

Special flags: □ Set user ID

□ Set group ID

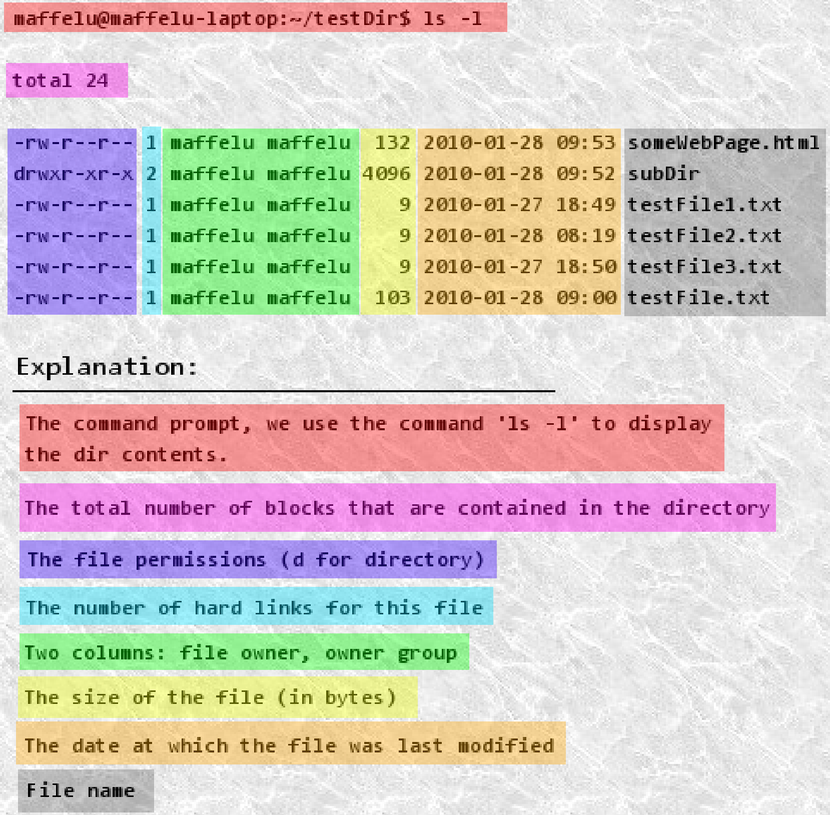
□ Sticky

Text view: -rw-rw-r--

Number view: 664

Last changed: Thursday, May 1 2003 at 8:53:54 AM

More information



```
maffelu@maffelu-laptop:~/testDir$ ls -l

total 24

-rw-r--r-- 1 maffelu maffelu  132 2010-01-28 09:53 someWebPage.html
drwxr-xr-x 2 maffelu maffelu 4096 2010-01-28 09:52 subDir
-rw-r--r-- 1 maffelu maffelu    9 2010-01-27 18:49 testFile1.txt
-rw-r--r-- 1 maffelu maffelu    9 2010-01-28 08:19 testFile2.txt
-rw-r--r-- 1 maffelu maffelu    9 2010-01-27 18:50 testFile3.txt
-rw-r--r-- 1 maffelu maffelu  103 2010-01-28 09:00 testFile.txt
```

Explanation:

The command prompt, we use the command 'ls -l' to display the dir contents.

The total number of blocks that are contained in the directory

The file permissions (d for directory)

The number of hard links for this file

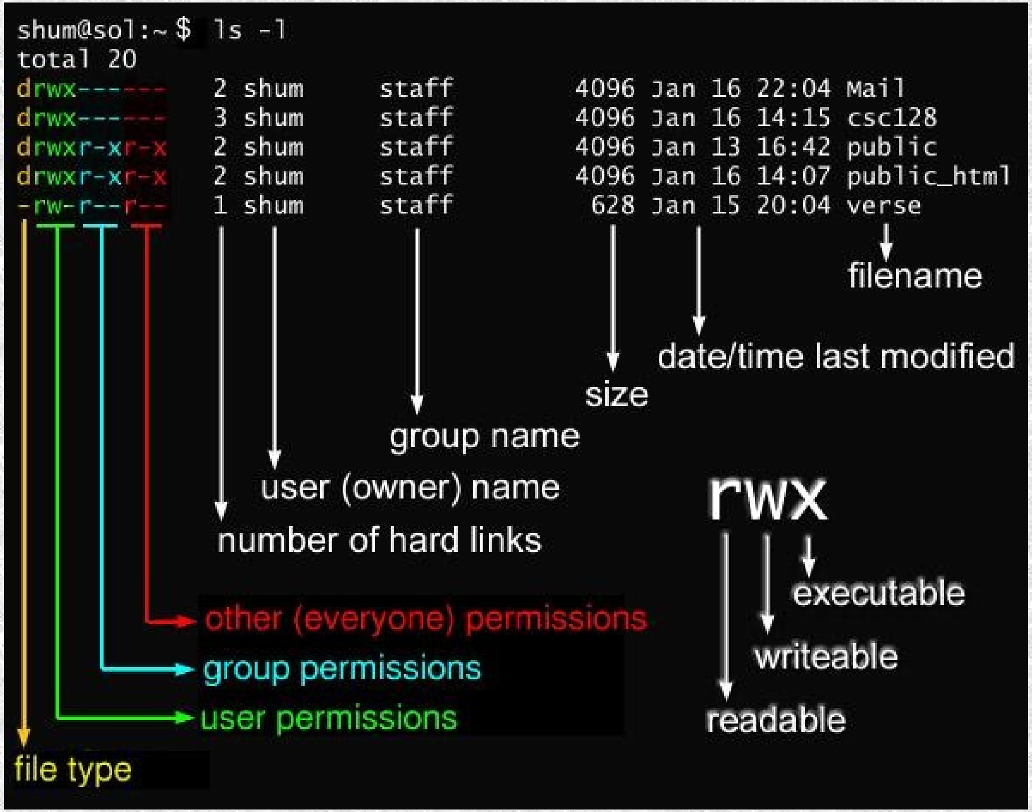
Two columns: file owner, owner group

The size of the file (in bytes)

The date at which the file was last modified

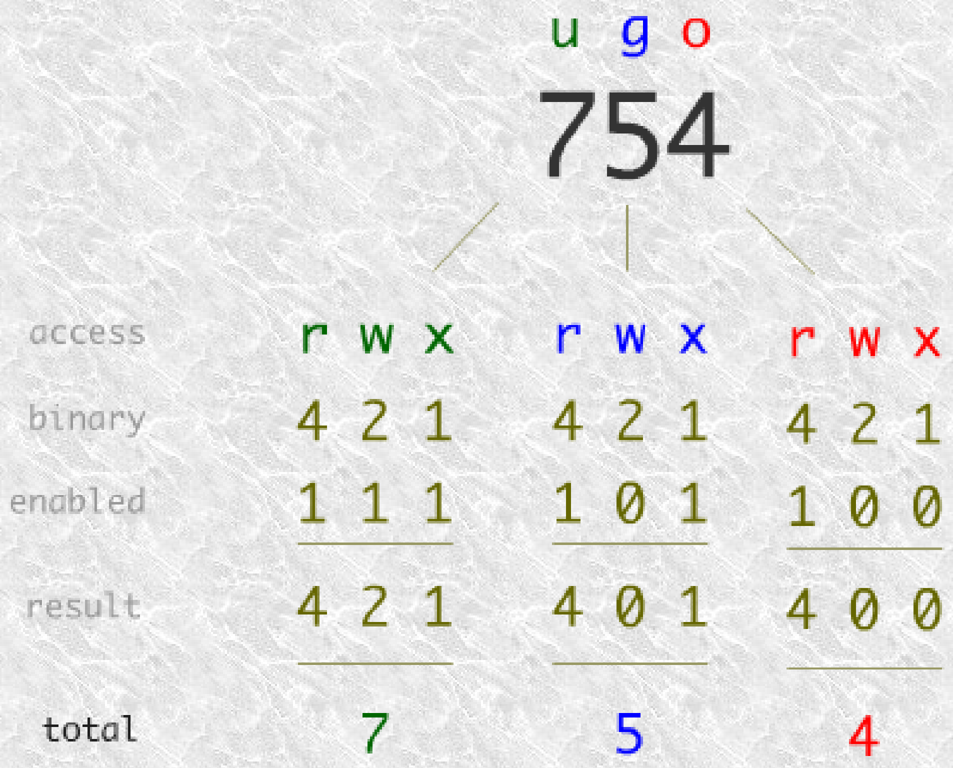
File name

user@example:~$ ls -l



```
shum@sol:~$ ls -l
total 20
drwx------   2 shum     staff         4096 Jan 16 22:04 Mail
drwx------   3 shum     staff         4096 Jan 16 14:15 csc128
drwxr-xr-x   2 shum     staff         4096 Jan 13 16:42 public
drwxr-xr-x   2 shum     staff         4096 Jan 16 14:07 public_html
-rw-r--r--   1 shum     staff          628 Jan 15 20:04 verse
```

Octal representation

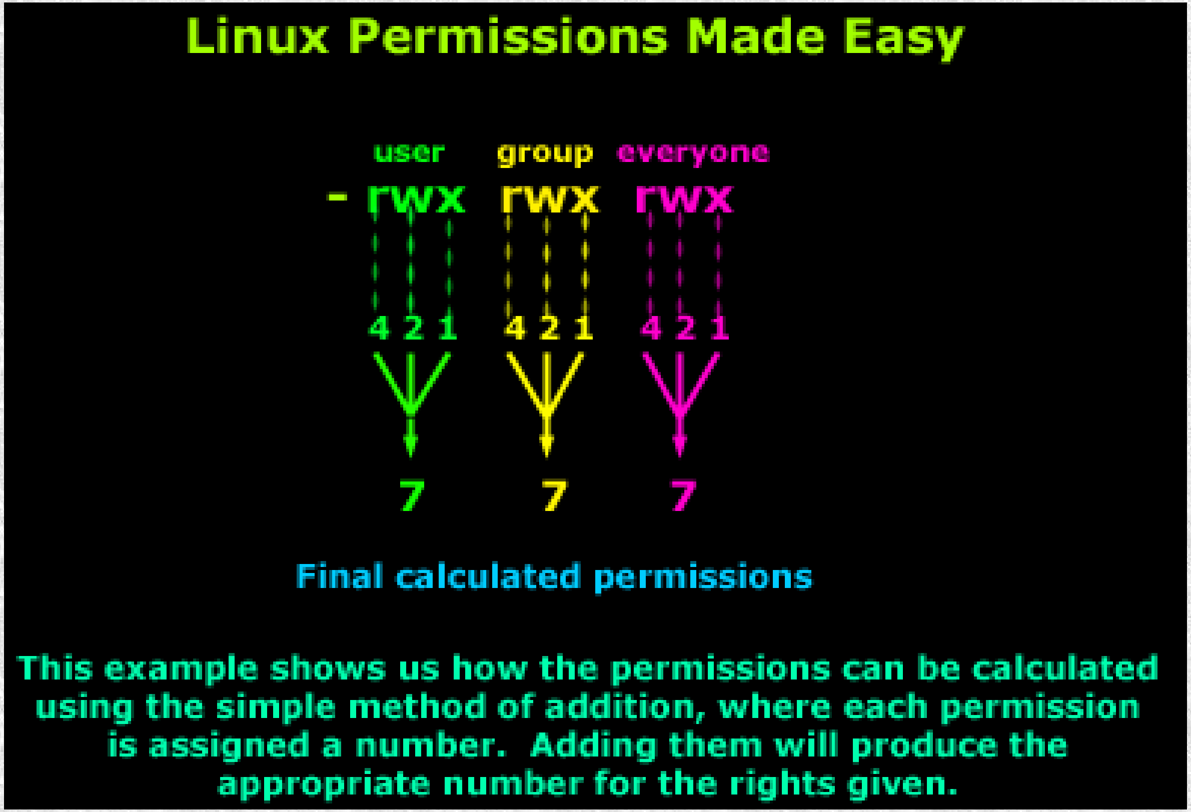


u g o

754

access	r w x	r w x	r w x
binary	4 2 1	4 2 1	4 2 1
enabled	1 1 1	1 0 1	1 0 0
result	4 2 1	4 0 1	4 0 0
total	7	5	4

Calculating permissions



Linux Permissions Made Easy

This example shows us how the permissions can be calculated using the simple method of addition, where each permission is assigned a number. Adding them will produce the appropriate number for the rights given.

Directory permissions are different

- Remember that a directory is a list of files (and maybe directories).
- When applying permissions to directories on Unix/Linux, the permission bits have different meanings than on regular files.
- There are three kinds of permissions that a user can have for a file:

1. permission to read the file.

    For directories, this means permission to list the contents of the directory.

2. permission to write (to change) the file.

    For directories, this means permission to create and remove files in the directory.

3. permission to execute the file (run it as a program).

    For directories, this means permission to access files in the directory.

Directory permissions

- The read bit (**r**) allows the affected user to list the files within the directory.
- The write bit (**w**) allows the affected user to create, rename, or delete files within the directory, and modify the directory's attributes
- The execute bit (**x**) allows the affected user to enter the directory, and access files and directories inside
- The sticky bit (**T**, or **t** if the execute bit is set for others) states that files and directories within that directory may only be deleted or renamed by their owner (or root).

Special components

In addition to the three sets of three permissions ..., the file mode bits have three special components, which affect only executable files (programs) and, on most systems, directories.

1. **SUID**, the set-user-ID-bit

Set the process's effective user ID to that of the file upon execution. On some systems sets owner to same owner as the directory, no matter who creates it.

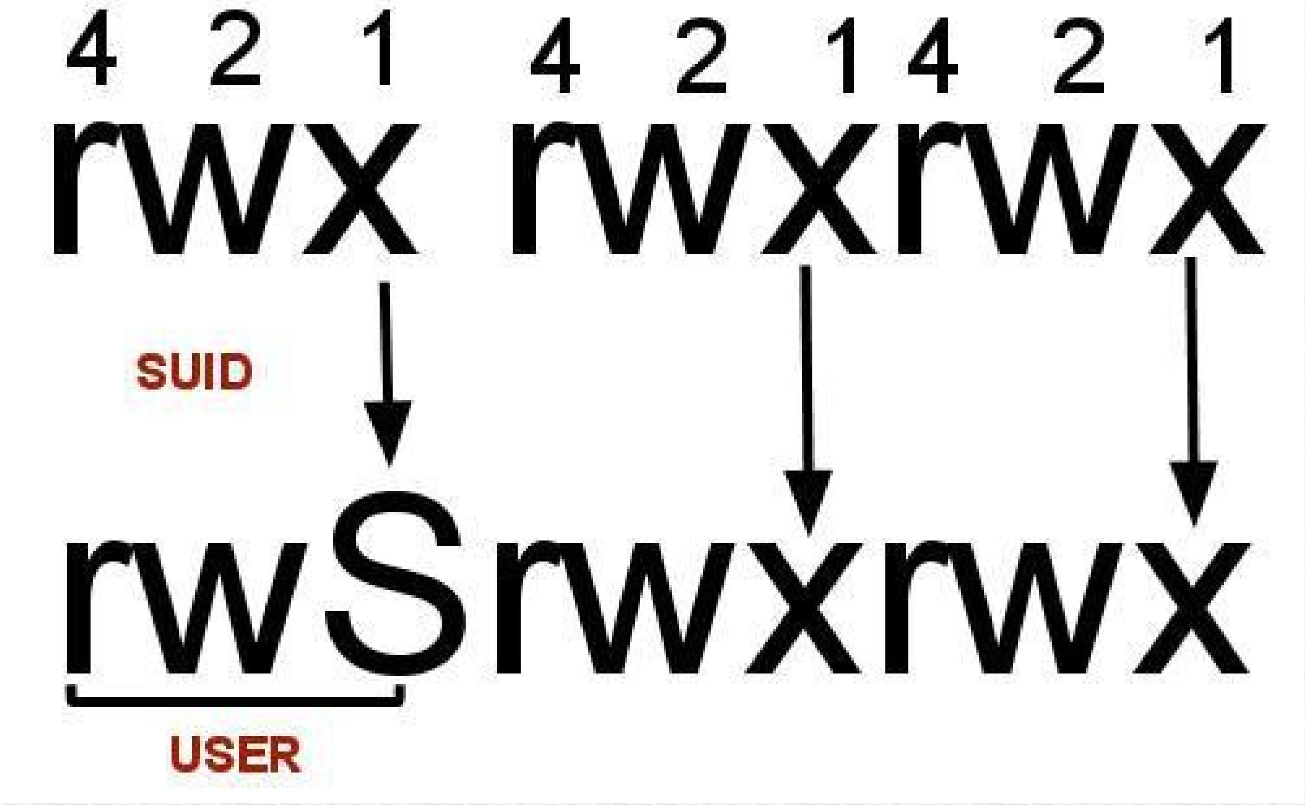
2. **SGID**, the set-group-ID-bit

Set the process's effective group ID to that of the file upon execution. On some systems sets group to same group as the directory, no matter who creates it.

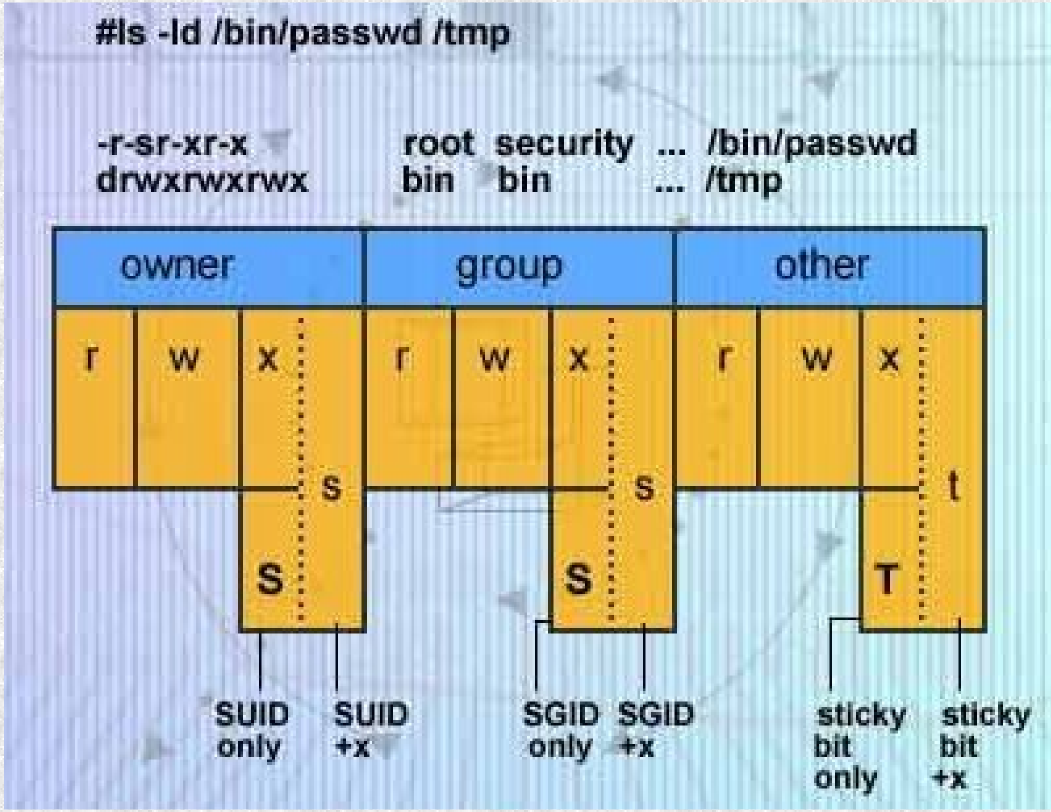
3. **sticky bit**,

Prevents unprivileged users from removing or renaming a file in a directory unless they own the file or directory. Older systems used this to keep a program or text image in memory.

Special permissions



SUID or SGID or sticky bit



```
#ls -ld /bin/passwd /tmp

-r-sr-xr-x     root  security  ... /bin/passwd
drwxrwxrwx     bin   bin       ... /tmp
```

Setting sticky bit and...

```
user@example:~$ touch junk.txt        ## make a dummy file
user@example:~$ ls -l junk.txt
-rw-rw-r-- 1 stan stan 0 Feb 12 16:41 junk.txt
user@example:~$ chmod 1664 junk.txt
user@example:~$ ls -l junk.txt
-rw-rw-r-T 1 stan stan 0 Feb 12 16:41 junk.txt
user@example:~$ chmod 2664 junk.txt
user@example:~$ ls -l junk.txt
-rw-rwSr-- 1 stan stan 0 Feb 12 16:41 junk.txt
```

Setting SUID and SGID...

```
user@example:~$ chmod 3664 junk.txt
user@example:~$ ls -l junk.txt
-rw-rwSr-T 1 stan stan 0 Feb 12 16:41 junk.txt
user@example:~$ chmod 4664 junk.txt
user@example:~$ ls -l junk.txt
-rwSrw-r-- 1 stan stan 0 Feb 12 16:41 junk.txt
user@example:~$ chmod 5664 junk.txt
user@example:~$ ls -l junk.txt
-rwSrw-r-T 1 stan stan 0 Feb 12 16:41 junk.txt
user@example:~$ chmod 6664 junk.txt
user@example:~$ ls -l junk.txt
-rwSrwSr-- 1 stan stan 0 Feb 12 16:41 junk.txt
```

Access Control Lists

- ...there may be file attributres specific to the file system, e.g., access control lists (ACLs), whether a file is compressed, whether a file can be modified (immutability), and whether a file can be dumped. These are usually set using programs specific to the file system. For example:
- ext2
- On GNU and GNU/Linux the file attributes specifc to the ext2 file system are set using '**chattr**'.
- FFS
- On FreeBSD the file flags specific to the FFS file system are set using 'chflags'.
- Even if a file's mode bits allow an operation on that file, that operation may still fail, because:
- , the file-system-specific attributes or flags do not permit it; or
- , the file system is mounted as read-only.
- For example, if the immutable attribute is set on a file, it cannot be modified, regardles of the fact that you may have just run 'chmod a+w FILE'.

Access Control Lists (ACLs)

These **ACLs** are additional ways to control file and directory accesses than the normal file permission scheme.

- user@example:~$ whatis chattr

chattr (1)   - change file attributes on a Linux file system

- user@example:~$ whatis lsattr

lsattr (1)   - list file attributes on a Linux second extended file system

user@example:~$ pinfo chattr

There are many more attributes besides permissions.

```
user@example:~$ pinfo chattr
```

The format of a symbolic mode is +-=[aAcCdDeijPsStTu].

The operator '+' causes the selected attributes to be added to the existing attributes of the files; '-' causes them to be removed; and '=' causes them to be the only attributes that the files have.

The letters 'aAcCdDeijPsStTu' select the new attributes for the files: append only (a), no atime updates (A), compressed (c), no copy on write (C), no dump (d), synchronous directory updates (D), extent format (e), immutable (i), data journalling (j), project hierarchy (P), secure deletion (s), synchronous updates (S), no tail-merging (t), top of directory hierarchy (T), and undeletable (u).

Remember This!

- Some things to remember:
  - The usual **man pages** are of little to no help learning about permissions.
  - If a **man page** does not exist, there may be an **info page**.
  - The **info pages** may provide information beyond expected of **man pages**.
  - The **pinfo** command has more features than the **info** command.
  - The **pinfo** command navigation is similar to **vi/vim** navigation
- ★ The one most important thing to remember from this tutorial:
  - Using the **info pages** gives you the best description of permissions.
  - user@example:~$ pinfo File permissions

References

- EzeeLinux.com ~ 2019-09-23
  - Linux_Terminal_Basics_4_Users,_Groups,_Aliases_and_Function.webm
- https://www.redhat.com/sysadmin/linux-access-control-lists

Questions

What are your questions?

by Stan Reichardt
stanr@sluug.org