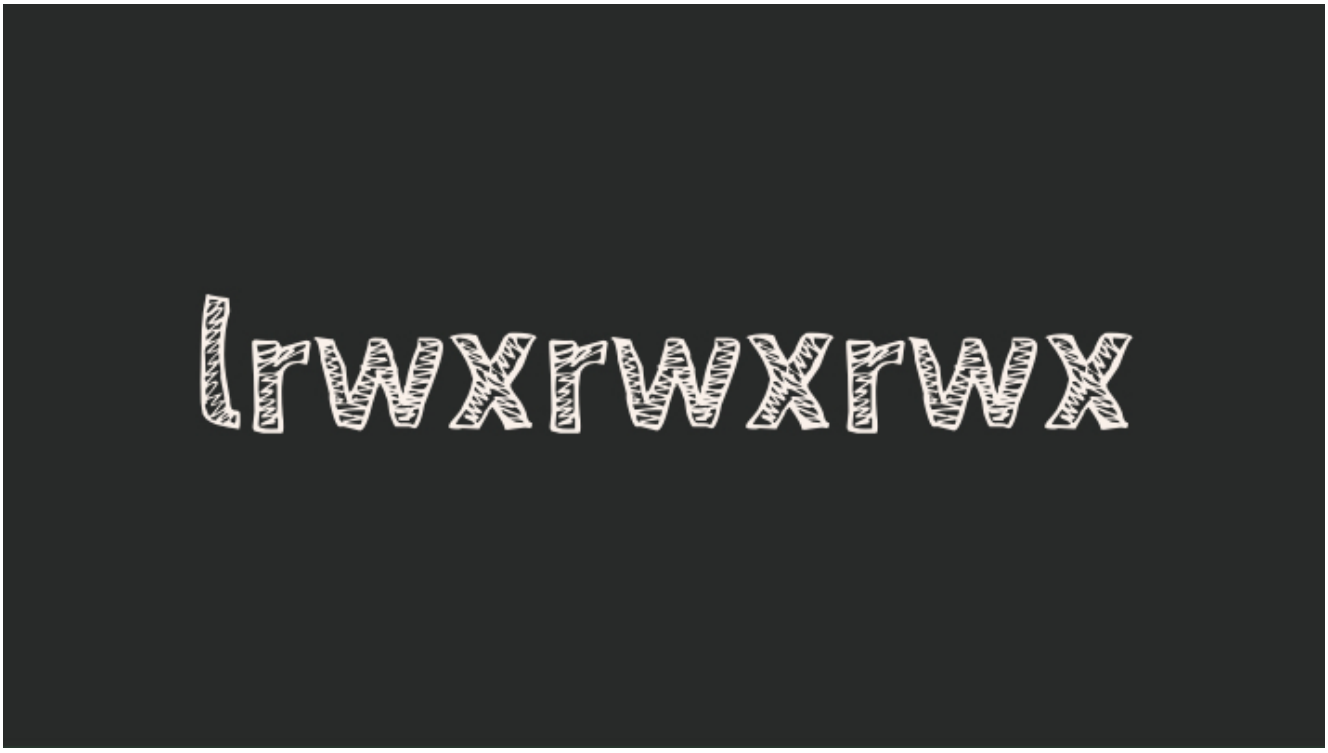
lrwxrwxrwx



lrwxrwxrwx permissions

If you come across a rather cryptic word "**lrwxrwxrwx**" when listing files and directories, here's how you can interpret it.

As you know, file permissions in Unix are traditionally provided using 3 levels:

- **user (file owner) permissions** – specifically, permissions for the user currently setup as the file owner
- **group permissions** – since each file belongs to a particular group, this permission level confirms the access other group members will enjoy
- **other** (everyone else on the system) permissions

lrwxrwxrwx permissions

lrwxrwxrwx follows a permissions structure:

tUUUGGGOOO, where **t** is the file type indicator, **UUU** are the three characters specifying user (file owner) permissions, **GGG** are the group permissions and **OOO** are the others permissions.

So in the **lrwxrwxrwx** case, l stands for symbolic link – a special kind of pointer allowing you to have multiple filenames pointing to the same Unix file.

rwxrwxrwx is a repeated set of permissions, **rwx** meaning the maximum permissions allowable within basic settings.

Meaning of rwx

rwx permissions mean the following access is permitted:

- **r** – read
- **w** – write
- **x** – execute (or change directory)

Interestingly, lrwxrwxrwx is a permission that's rather uncommon: usually symlinks get a different (less forgiving) file permissions. Since symlinks are just pointers to other files, it doesn't matter much if you provide w (write) permissions or not – they would not allow you to control write access to the destination file.

Example: we use the touch command to create a simple file called **"file".** We specifically remove write permissions and this means we can't write anything into the file as you can see:

```
greys@maverick:~ $ touch file
greys@maverick:~ $ ls -al file
-rw-r--r-- 1 greys staff 0 3 Oct 23:36 file
greys@maverick:~ $ chmod u-w file
```

```
greys@maverick:~ $ ls -al file
-r--r--r-- 1 greys staff 0 3 Oct 23:36 file
greys@maverick:~ $ echo test > file
-bash: file: Permission denied
```

If we create a symlink **file2** pointing to **file**, it will actually show first group of permission block (user permissions) to be **rwx**, so it may see you have write access to the file it's pointing to:

```
greys@maverick:~ $ ln -s file file2
greys@maverick:~ $ ls -al file*
-r--r--r-- 1 greys staff 0 3 Oct 23:36 file
lrwxr-xr-x 1 greys staff 4 3 Oct 23:37 file2 -> file
```

But if we try to write the same word "**test**" into **file2** symlink, we'll still get an error cause it's pointing to the file which only has read permissions.

Finally, if we allow write permissions on the **file** again, we can write into **file2** symlink and it will work just fine this time:

```
greys@maverick:~ $ chmod u+w file
greys@maverick:~ $ ls -al file*
-rw-r--r-- 1 greys staff 0 3 Oct 23:36 file
lrwxr-xr-x 1 greys staff 4 3 Oct 23:37 file2 -> file
greys@maverick:~ $ echo test > file2
greys@maverick:~ $ cat file
test
greys@maverick:~ $ cat file2
test
```

See also

- **How to Use ln command for making links and symlinks**
- Advanced Unix Commands
- Unix Commands
- chmod vs chown