



Chmod/Chown WSL Improvements



Craig

January 12th, 2018

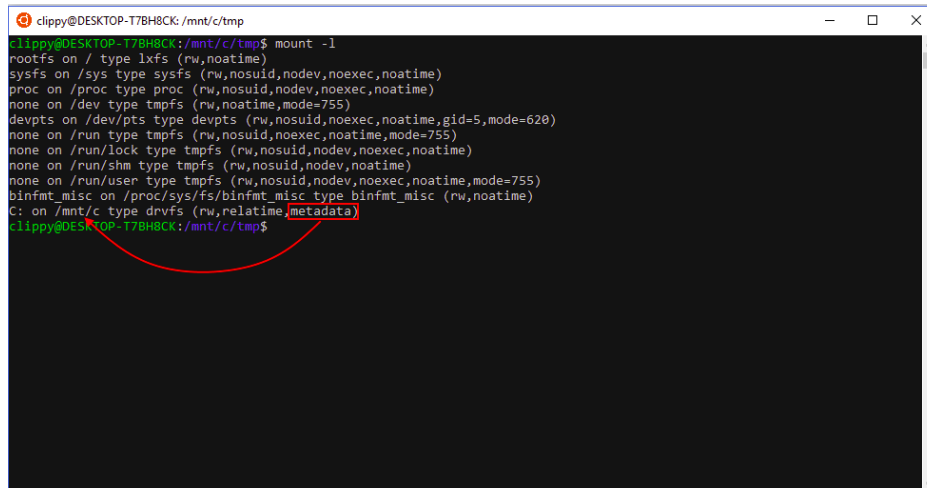


We've added new file system features to WSL in [Insider Build 17063](#). You can now set the owner and group of files using `chmod/chown` and modify read/write/execute permissions in WSL. You can also create special files like fifos, unix sockets, and device files. We're introducing new mounting options with DrvFs for projecting permissions onto files alongside providing new Linux metadata on files and folders.

There's one step you must take before you can enjoy these new features: you must unmount `drvfs` and remount it with the 'metadata' flag. To do this:

```
sudo umount /mnt/c && sudo mount -t drvfs C: /mnt/c -o metadata
```

You can verify that it mounted correctly by running "mount -l" to see the output below:



```
clippy@DESKTOP-T7BH8CK: /mnt/c/tmp
clippy@DESKTOP-T7BH8CK:/mnt/c/tmp$ mount -l
rootfs on / type lxfs (rw,noatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,noatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,noatime)
none on /dev type tmpfs (rw,noatime,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,noatime,gid=5,mode=620)
none on /run type tmpfs (rw,nosuid,noexec,noatime,mode=755)
none on /run/lock type tmpfs (rw,nosuid,nodev,noexec,noatime)
none on /run/shm type tmpfs (rw,nosuid,nodev,noatime)
none on /run/user type tmpfs (rw,nosuid,nodev,noexec,noatime,mode=755)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noatime)
C: on /mnt/c type drvfs (rw,relatime,metadata)
clippy@DESKTOP-T7BH8CK:/mnt/c/tmp$
```

What is DrvFs?

DrvFs is a filesystem plugin to WSL that was designed to support interop between WSL and the Windows filesystem. DrvFs enables WSL to mount drives with supported file systems under `/mnt`, such as `/mnt/c`, `/mnt/d`, etc.

You can learn more about DrvFs and the WSL filesystem in a previous blog post from June 2016, [WSL File System Support](#), which covers the Linux filesystem as it pertains to WSL and compares it to the Windows filesystem.

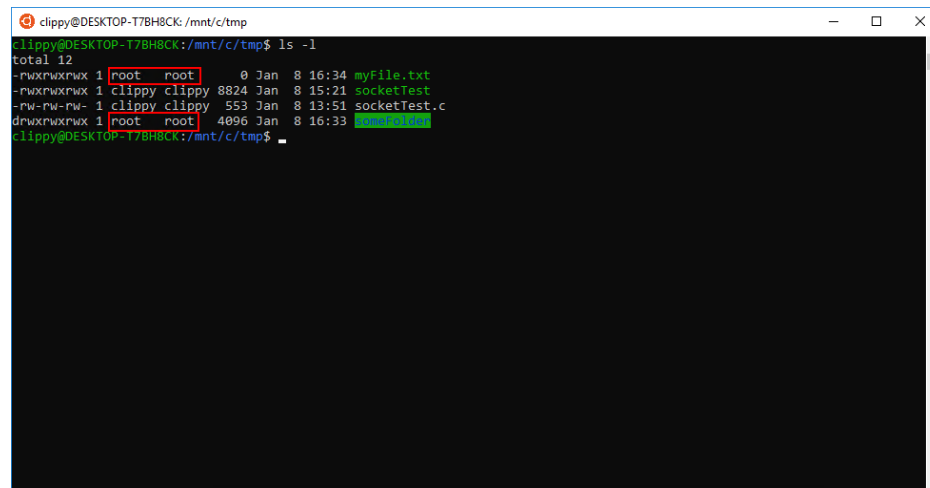
Let's take a look at the two new features in detail. And remember, some of the functionality discussed below already exists on the Linux filesystem side—bringing this over to the Windows side is what's new.

Support for Additional Metadata

Linux permissions are added as additional metadata to the file. This means a file can have Linux *and* Windows read/write/execute permission bits.

How did permissions work in the past?

Prior to Build 17063, all files/folders list “root” as the owner and belonged to the group “root”. The permission bits on each file/folder was derived from Windows permissions—no write bit checked for Windows meant no write bit set in WSL.



```
clippy@DESKTOP-T7BH8CK: /mnt/c/tmp
clippy@DESKTOP-T7BH8CK:/mnt/c/tmp$ ls -l
total 12
-rwxrwxrwx 1 root root  0 Jan  8 16:34 myFile.txt
-rwxrwxrwx 1 clippy clippy 8824 Jan  8 15:21 socketTest
-rw-rw-rw- 1 clippy clippy 553 Jan  8 13:51 socketTest.c
drwxrwxrwx 1 root root 4096 Jan  8 16:33 socketTest
clippy@DESKTOP-T7BH8CK:/mnt/c/tmp$
```

Additionally, attempting to chmod or chown on a file/folder resulted in a no-op (they wouldn't do anything!)

How do permissions work now?

For files that don't have metadata, we apply the same approach as what is described in pre-17063 builds. But now, chmod/chown can assign metadata to the file or folder. Newly created files in WSL will be created with metadata by default and will respect the mount options you've set (discussed later) or the permissions you pass when executing a mkdir/open.

Once the file or folder has metadata, Windows and Linux permissions will not remain in lock-step with each other.

Important Caveats

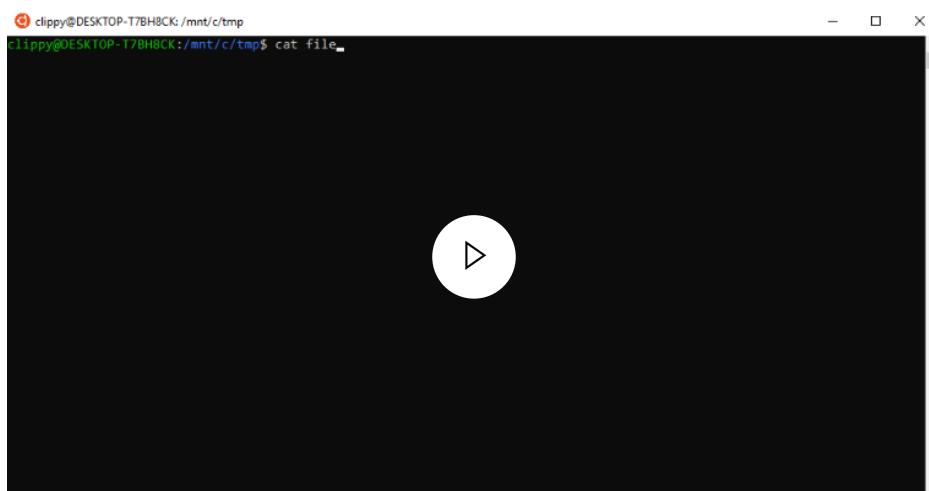
There are a few things to make sure you're aware of when tinkering with the new metadata:

1. Editing a file using a Windows editor may remove the file's Linux metadata. **In this case, the file will revert to its default permissions**
2. Removing *all* write bits on a file in WSL *will* make Windows mark the file as read-only.
3. If you have multiple WSL distros installed or multiple Windows users with WSL installed, they will all use the same metadata on the same files. The uid's of each WSL user account might differ. This something to consider when setting permissions.

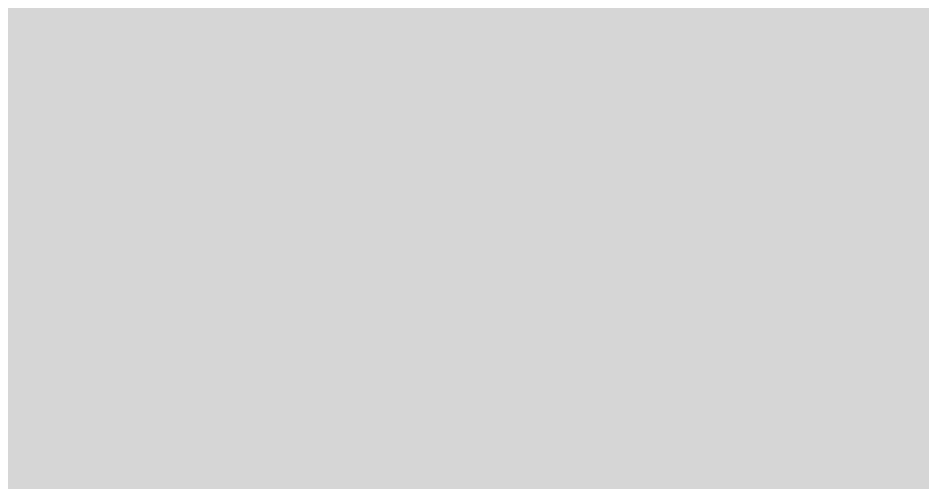
For example, you can disable write permissions on a file in Windows and chmod the file to show write permissions are enabled in WSL. Or you can have read permissions enabled under Windows and remove read permissions in WSL. You can see this concept illustrated below.



In the example below, I remove a file's write permissions in WSL which disables the ability to write changes to the file from WSL. But I can still open the file and write changes from Windows because I have the write permission set in Windows.

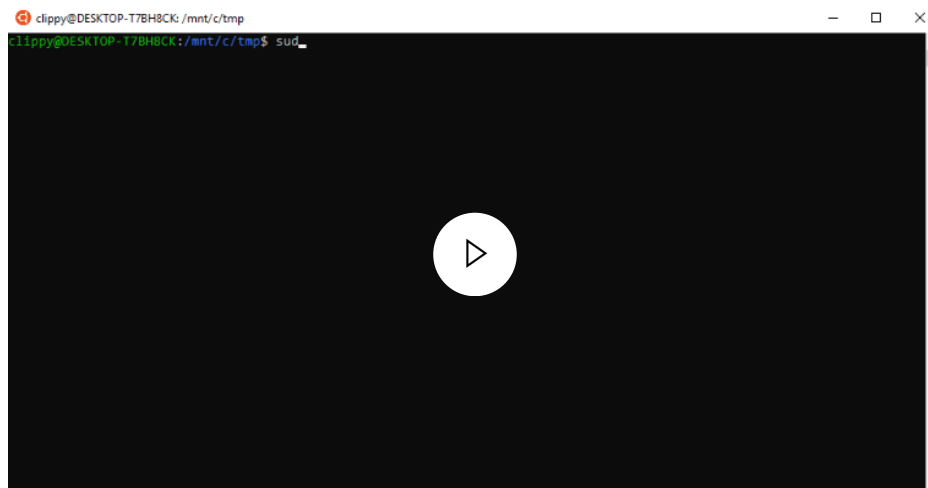


Inversely, if I have a file in WSL which states its Linux permissions allows writing—but I've disabled write permissions in Windows—I still won't be able to perform actions. **Windows permissions for a file or folder will trump the permissions set under WSL.** You can see this concept illustrated below.



Special Files

You can create special files like fifos, unix sockets, and device files.



Mount Options

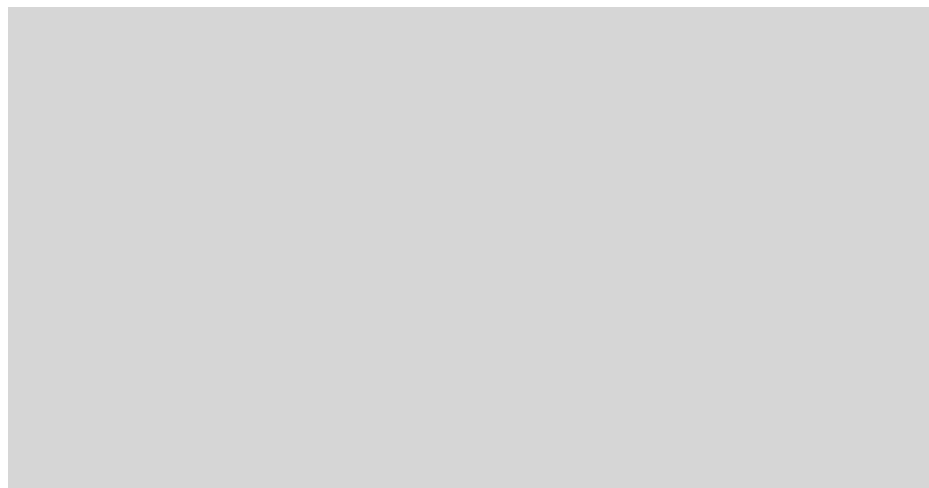
We've added new mount options to DrvFs to control permissions for files without metadata. You can combine with the metadata option to specify default permissions for files without metadata. The new mount options include:

- *uid*: the user ID used for the owner of all files
- *gid*: the group ID used for the owner of all files
- **umask*: *an octal mask of permissions to exclude for all files and directories.
- *fmask*: an octal mask of permissions to exclude for all regular files.
- *dmask*: an octal mask of permissions to exclude for all directories.

A sample command using the mount options could look like this:

```
sudo mount -t drvfs C: /mnt/c -o  
metadata,uid=1000,gid=1000,umask=22,fmask=111
```

After executing the mount command, you will see your mount (in this case, C:) listed with all the parameters you passed in when querying for a list of mounted devices.



If you're unfamiliar with octal masks, consult the man pages for `chmod` to learn more (in the WSL console, enter "man chmod").

Mount Options Example

In my WSL instance, I have created a group called "metadatagroup" and a user named "msbob". They have gid and uid values equal to 1001. I also have a file and folder without metadata in WSL. If I mount DrvFs with this command:

```
sudo mount -t drvfs C: /mnt/c -o  
metadata,uid=1001,gid=1001,umask=22,fmask=111
```

Take a peek at the permissions on the file and directory that did not previously have metadata. It was appropriately assigned msbob and metadatagroup for the owner and group bits, as expected.



Additionally, because we passed in `umask=22` and `fmask=111`, our files and directories have had their read/write/execute permissions update accordingly. The `umask` used in this example will disable execution rights for files by default, which allows you to be more explicit in setting execution permissions on a file-by-file basis.

By default, WSL will set the uid and gid to the default user with drives that are auto-mounted during instance start. If you mount manually, you will have to set these explicitly (the default user that gets created when WSL is first installed has a `uid=1000` and `gid=1000`).

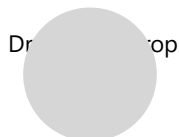
What do you think about the latest changes to DrvFs? Drop us a comment below or tweet at us!

Cheers,

Craig Wilhite ([@CraigWilhite](#))

Posted in [Windows Subsystem for Linux \(WSL\)](#)

Tagged [chmod](#)



Craig Wilhite

WSL

Program Manager, WSL, Containers, and Hyper-V

Read next

Follow 

[Tar and Curl Come to Windows!](#)

One of the most frequent asks we hear across the entire Windows command-line spectrum is "I need curl!" and/or "I need tar". If you're one of these people - HAPPY NEW ...

 [Rich Turner](#) January 18, 2018

 [3 comments](#)

[OpenSSH in Windows 10!](#)

SSH is one of the most important tools in the *NIX world, through which users communicate with shells, applications, and services running on remote machines, devices, VM'...

 [Rich Turner](#) January 22, 2018

 [0 comment](#)

8 comments

Comments are closed. [Login to edit/delete your existing comments](#)



Wil Wilder Apaza Bustamante May 14, 2019 11:39 pm



Thank you for your text Craig. I probably have to read it a few more times before I really understand it completely. Still this was the missing piece of the puzzle that would allow me to use wsl in my daily development. Git wasn't working, checkouts, resets did not fail, but didn't accomplish anything either. Remounting my drive with metadata made it work.



Frank Pigeon May 17, 2019 6:26 am



🏆this blog post is worth its weight in gold. Now I am able to download repos from git on the windows side...may sell my mpb now.



Besmir Zanaj May 23, 2019 8:26 pm



I wonder why this behaviour is not the default one?



Brendan Holmes June 20, 2019 2:21 am



Anyone got metadata working when mounting SMB (samba) mapped drives? The only mounts that I've managed to get "metadata" successfully showing in their options when I do a mount -l are local NTFS drives. In fstab I'm using:

```
V: /mnt/tmp drvfs metadata,rw,noatime,uid=1000,gid=1000,umask=22,fmask=11 0 0
```

where V: is a mapped samba drive. mount -l shows this is mounted without metadata:

```
V: on /mnt/tmp type drvfs (rw,noatime,uid=1000,gid=1000,umask=22,fmask=11,case=off)
```

I've also tried drives mapped to NFS shares using Microsoft Client for NFS to no avail. Can someone confirm whether its possible to use metadata (and hence chmod\chown) with network drives?



Ryan Prosser November 13, 2019 5:15 pm



I too am trying to use chown to change the owner of a file
Network drive mounts would simplify accessing the source data

I suspect an issue I'm facing is how to identify the new own from a Windows AD domain?
Is mapping required from /etc/passwd to the identity source – so I can assign owner username as someone NOT the current windows logon user, running the WSL shell ?

Any input in this area would be appreciated



Karina Condeixa April 20, 2020 11:49 pm



Thank you for your post, Craig.
I am facing a problem when I write

```
sudo umount /mnt/c
```

It returns

```
umount: /mnt/c: target is busy.
```