~~Best~~ Good Practices for Permissions and Data Sharing

Brian Vanderwende
CISL Consulting Services

November 7, 2024

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Session Agenda

- ➔ Overview of POSIX file permissions
- ➔ NCAR defaults and why you may wish to change them
- ➔ Internal data sharing using ACLs
- ➔ Sharing data with external collaborators

What does it mean, to be a user?

All users on the system have a username, a primary/default group, and potentially a collection of secondary groups:

```
$ id -un   # Print the user's name (or use whoami)
vanderwb
$ id -gn   # Print the user's primary group
csgteam
$ id -Gn   # Print all of the user's groups (or use groups)
csgteam ncar csg ncldev sssg0001 scsg0001 ...
```

*All users will be members of the **ncar** group, one group per active allocation, and perhaps more.*

The POSIX permissions model

```
$ ls -l -d $HOME
drwxr-xr-t 116 vanderwb ncar
```

Above: all can read and execute my home directory, but only I can add and (re)move items within

We will discuss the "t" in a bit!

- All file system items (files + directories) have three *scopes:*
  - **user, group, others**
- Each scope has three possible permissions:
  - **r**ead, **w**rite, e**x**ecute (rwx)
  - These apply differently to files and directories:

	read	write	execute
File	look at contents	modify contents	run as command
Directory	see names of items within	create, move, delete items if execute is set	permits access to items within*

** In practice, directories will almost always need r-x permissions for useful access*

Symbolic vs octal/numeric notation

So far we've used symbolic notation (rwx) to show permissions, but you will also encounter permissions represented using 3-4 octal digits

Numeric permissions

+4 - if **r**ead is allowed

+2 - if **w**rite is allowed

+1 - if e**x**ecute is allowed

```
$ ls -l -d $HOME
drwxr-xr-t 116 vanderwb ncar
```

user	=	4 + 2 + 1	=	**7**
group	=	4 + 1	=	**5**
others	=	4 + 1	=	**5**

setgid and the *sticky bit*

There are two additional directory permissions that can be useful:

- **setgid (S/s)**, when set on a directory, causes any items created in that directory to inherit the group of the directory, rather than the creator's primary group

- **the sticky bit (T/t)**, when set on a directory, ensures that files within can only be deleted by the directory owner, file owner, or root

```
$ ls -l -d /hotel/california
drwxr-sr-t 116 vanderwb eagles
```

Modifying POSIX permissions

Use **chmod** to set or change permissions on a file or directory:

```
$ chmod 664 <file> # all can read; owner & group can write
```

If you use symbolic notation, you can modify a subset of classes. You can also apply changes recursively:

```
$ chmod -R ug+rX <directory>
```

Here we add user and group read permissions to the directory and its contents, and execute permissions to directories, subdirectories, and files that already have some execute permissions only (upper-case X).

➔ *You can also use **a** for all, which is the same as **ugo**.*

Setting a default permissions mask with *umask*

Use **umask** to define the new file creation mask. A permissions *mask* acts as a filter, restricting what permissions a program can assign to a new file or directory. This setting does not restrict **chmod**.

```
$ umask 027
```

These are restrictions, not permissions. What does this mean?

user → **0 = no restrictions (any permissions allowed)**

group → **2 = write is prohibited (read-execute allowed)**

others → **7 = all modes are prohibited**

umask *settings only apply to the current shell session!*

The desire to get things done efficiently can lead to unwise practices…

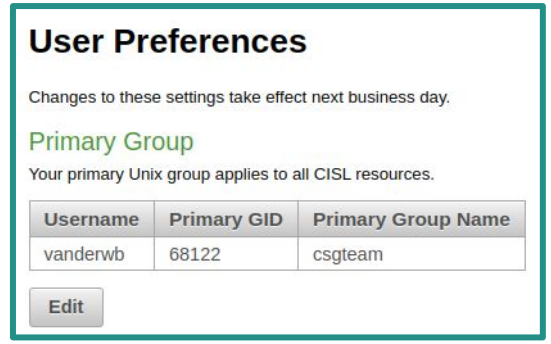
Reviewing NCAR HPC POSIX storage spaces

Storage Space	Type	Quota	Default Permission	Default Group	Intended Use
/glade/u/home/$USER	Snapshot	50 GB	755	ncar	Program settings, critical files, *secrets*
/glade/work/$USER		2 TB	755	ncar	Applications, code repos, small datasets
/glade/derecho/scratch/$USER	Purged	30 TB	755	ncar	Run directories, staged input, outputs
/glade/campaign/[lab]	Allocated	Varies	755 / 750	varies	Stable datasets; collaborative work
/glade/campaign/univ/[project]	Allocated	Varies	770	[project]	Stable datasets; collaborative work

All spaces listed above are globally accessible, and all use spinning disk storage, except for home, which uses flash storage.

NCAR's permissive environment: the default *ncar* group

The default primary group - **ncar** - includes everyone. This makes group-write effectively all-write!



Suggestion 1: change to a project-specific primary group
https://sam.ucar.edu

Suggestion 2: modify file and directory group membership
(*only the file owner can change its group*)

```
$ chgrp -R mygroup /glade/work/$USER/project_dir
```

NCAR's permissive environment: open home directories

User directories are set to **755** by default. This may make sense for *work* and *scratch*, but *home* directories often contain:

- Passwords
- Keys
- Logs with sensitive info

CISL is considering changing the default $HOME permissions in the near future

Suggestion 3: set **700** or **750** permissions on your $HOME. If you allow group-access, **chgrp** the directory to something besides **ncar**

Aside: a use-case for *execute-only* directories

During this presentation, it was asked whether permissions could be used to allow entry into a subdirectory without allowing entry in to the parent directory.

Yes - with execute-only permissions on the parent directory!

```
$ chmod -R 711 /glade/u/home/$USER
$ chmod -R 755 /glade/u/home/$USER/.conda
```

*The above setup will not allow anyone else to see anything in the user's home directory **except** the .conda subdirectory and its contents.*

Use permissions to protect data

Any files with group- or others-write permission is at risk:

- We've seen users run **rm -r /glade/scratch**! 😬😬
- Security weak points could allow malicious deletion of files
  - E.g., GitHub self-hosted runners acting on unvetted pull requests from public repositories

Suggestion 4: only grant write permissions when there is a specific need, and be as selective as possible to whom you give access

The limitations of sharing data with standard permissions

Scenario 1: You wish to share a sensitive dataset with a single collaborator

Scenario 2: You wish to share a project directory with another group in addition to the main project group

Using basic POSIX permissions, you would need to open read-execute access to all users:

```
$ chmod -R o+rX /path/to/project/dataset
```

But there's a better way: ***file access control lists (ACLs)****!*

Precise data sharing with file ACLs

Interact with file ACLs using two commands: **getfacl** and **setfacl**

File ACLs allow you to augment the standard permissions with “named” entries, which grant permissions to specific users or groups

Default entries specify what access each entity will have for new files created within this directory

```
$ getfacl /glade/work/vanderwb/group_data/
# file: glade/work/vanderwb/group_data/
# owner: vanderwb
# group: csgteam
user::rwx
group::rwx
group:ssg:r-x
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:ssg:r-x
default:mask::rwx
default:other::---
```

We’ve granted **rx** to our colleagues in the **ssg** group

Examples of setting file ACLs with *setfacl*

- Grant a collaborator with username **friend** write permissions on a file

```
$ setfacl -m u:friend:rw /path/to/file
```

- Allow members of group **nws** to view current directory contents and any future contents

```
$ setfacl -R -m g:nws:rX /path/to/directory
$ setfacl -R -d -m g:nws:rX /path/to/directory
```

- Remove all extended ACLs on a directory, including *default* settings

```
$ setfacl -b /path/to/file
```

Keeping ACLs while using common tools

One downside of ACLs to keep in mind is that many common file manipulation tools will not preserve them by default! For example:

Command	ACL support flag	Usage
`cp`	`--preserve=mode`	`cp --preserve=mode file archive/file`
`rsync`	`-A/--acls`	`rsync -a --acls output/ archive`
`tar*`	`--acls`	`tar --acls -cf archive.tar file1 file2`

** For **`tar`**, you will need to use the **`--acls`** flag during both tarball creation and data extraction*

Use good data sharing practices

Suggestion 5: don't forget about extended ACLs; in many situations they should be your go-to method for expanding access

ACL tip 1: the **ls** command will tell you if files/directories have extended permissions applied to them

```
$ ls -d -l /glade/campaign/cisl/csg
drwxrwsr-t+ 18 root csg 4096 Jul 31 08:33 /glade/campaign/cisl/csg
```

ACL tip 2: you can also use named entries to restrict access. This is less common, but there are special cases (e.g., restricting a shared account)

Debugging permissions with the *namei* command

Scenario: someone has shared a file with you and you cannot open it, but the permissions look okay to them.

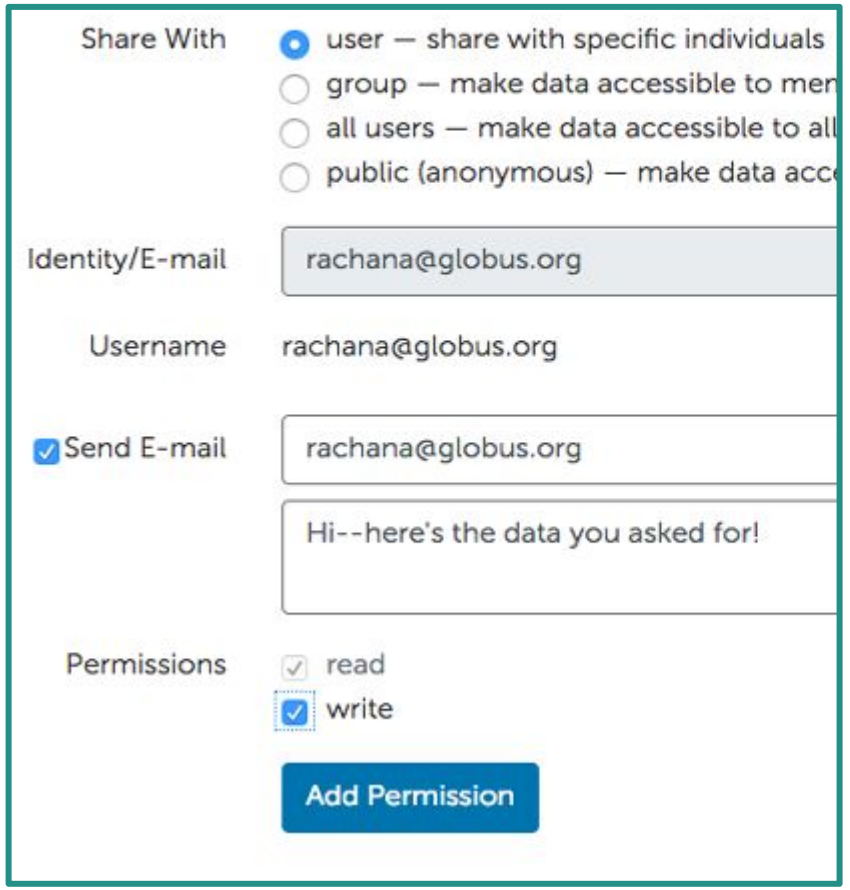
```
$ namei -l /glade/campaign/cisl/csg/pnichols/archive.tgz
f: /glade/campaign/cisl/csg/pnichols/archive.tgz
drwxr-xr-x root  root /
drwxr-xr-x root  root glade
drwxr-xr-x root  root campaign
drwxrwsr-x root  root cisl
drwxrwsr-t root  csg  csg
drwx--S--- 28932 csg  pnichols
                      archive.tgz - Permission denied
```

A quick word about external data sharing permissions

Use Globus *guest collections* to share data with external collaborators.

Permissions set on data in a guest collection are **layered on top of POSIX permissions**.

- *You can't grant access you don't have*
- *User permissions you set on the system dictate the maximum access to guests*



Happy computing!

Getting help from CISL and other users

Help Desk: 303-497-2400
Support tickets: https://rchelp.ucar.edu
1-on-1 Meeting: Virtual Consulting Portal
Community: NCAR HPC User Group Slack

Specific questions from today or feedback:
Contact Brian: vanderwb@ucar.edu