Using access control lists

Access control lists (ACLs) are tools for managing permissions within a file system by giving users and groups read, write, and/or execute permissions on files or directories *outside of* the traditional UNIX permissions. The UNIX permissions for managing files on GLADE remain in effect, but ACLs can be used to facilitate short-term file sharing among users who cannot be put in the same UNIX group.

In the Cheyenne/GLADE environment, the most common use cases are:

- Sharing files among users in different NCAR labs or universities.
- Sharing files with short-term visitors, interns, students, or others during a short project period.

Following are examples of how to create an ACL that allows other individuals and groups to work with your files, and how to propagate permissions to new files and directories. *To create and manage ACLs on the Campaign Storage file system, log in to Casper or the data-access nodes rather than Cheyenne.*

Page contents

Create and view an access control list

Use the **setfacl** command with the **--modify** option to give an individual user access to a file or directory that you own.

In this example, user shiquan is sharing a file with user bjsmith. Listing the file before and after creating the ACL shows how the file permissions changed.

```
ls -l testfile.txt
-rw------- 1 shiquan ncar 18 Mar 4 09:00 testfile.txt
setfacl --modify user:bjsmith:rwx testfile.txt
ls -l testfile.txt
-rw-rwx---+ 1 shiquan ncar 18 Mar 4 09:00 testfile.txt
```

The **+** as the last character in the permissions string indicates that an ACL exists for the directory.

About execute flags: X vs. x

When setting permissions, the execute flag can be set to upper-case **X**, which differs from the lower-case **x** setting. The **X** permission allows execution only if the target is a directory or if the execute permission has already been set for the user or group. It is useful in the case of handling directory trees recursively.

Run **getfacl** as shown here to view any ACLs applied to a file or directory.

```
getfacl testfile.txt
```

The output will look something like this, illustrating in this case that one user's permissions (**rwx**) are different from other users' permissions (**rw-**).

```
# file: testfile.txt
# owner: shiquan
# group: ncar
user::rw-
user:bjsmith:rwx
group::---
mask::rwx
other::---
```

Give access to a group

To give an existing *group* access to a file or directory, use **setfacl** as shown here. In this example, the name of the group being given read/write/execute permissions is **csg**.

```
setfacl --modify group:csg:rwx testfile.txt
```

Run **getfacl** to see the resulting ACL.

```
getfacl testfile.txt
# file: testfile.txt
# owner: shiquan
# group: ncar
user::rw-
user:bjsmith:rwx
group::---
group:csg:rwx
mask::rwx
other::---
```

Use default ACLs to propagate permissions

ACLs can be set to propagate permissions to files and subdirectories as they are created. This is done using *default* ACLs.

In this example, a new directory is accessible only to the user who created it.

```
drwx------  2 shiquan ncar 4096 Mar  6 10:12 my_shared_directory/
```

To enable another user to read and navigate into the directory, follow this example.

```
setfacl --modify user:bjsmith:r-x my_shared_directory
```

To set the directory's default ACL so that any new files or subdirectories automatically have those same permissions, use the **setfacl** command as shown here.

```
setfacl --modify default:user:bjsmith:r-x my_shared_directory
ls -ld my_shared_directory
drwxr-x---+ 2 shiquan ncar 4096 Mar  6 10:12 my_shared_directory
```

The **+** as the last character in the permissions string indicates that an ACL exists for the directory.

Run **getfacl** to see the resulting ACL and the *default* ACLs.

```
getfacl my_shared_directory
# file: my_shared_directory
# owner: shiquan
# group: ncar
user::rwx
user:bjsmith:r-x
group::---
mask::r-x
other::---
default:user::rwx
default:user:bjsmith:r-x
default:group::---
default:mask::r-x
default:other::---
```

Default permissions for a specified user

ACLs for a specified user or group are *independent of default ACLs*.

The next example illustrates how to modify a default ACL to set default permissions for a specified user.

```
setfacl --default --modify user:bjsmith:rwx my_shared_directory
```

The directory now has a default ACL that will make any new file in the directory accessible to the designated user with the specified permissions (**rwx**). Here is an example of a new file created in that directory.

```
-rw-rw----+ 1 shiquan ncar 23 Mar  6 10:19 my_shared_directory/newfile
```

Most users in the group get **rw-** permission. However, as a result of the default ACL behavior established above, user bjsmith's permissions are different (**rwx**). The following **getfacl** output with the comment "#effective:rw-" shows the difference.

```
getfacl my_shared_directory/newfile
# file: my_shared_directory/newfile
# owner: shiquan
# group: ncar
user::rw-
user:bjsmith:rwx #effective:rw-
group::---
mask::rw-
other::---
```

Remove an access control list

To remove all ACLs from a file, run **setfacl --remove-all** followed by the filename.

```
setfacl --remove-all testfile.txt
```

To remove selected permissions previously set for a user, run **setfacl -x** as shown here.

```
setfacl -x user:bjsmith testfile.txt
```

Advanced use of ACLs

Users often have different default umask settings that can conflict with a file or directory ACL. The following example sets an ACL for a directory and all of its files and subdirectories, and it also sets the default ACL for any future files and subdirectories, ensuring the directory is protected from unknown umask settings.

```
setfacl  -R -m o:r-x  -dm o:r-x  /glade/scratch/bjsmith
```

Note that there are two clauses in this example.

1. The first (**-R -m o:r-x**) recursively sets permissions for "others" to read and execute (r-x) on all existing files and subdirectories under /glade /scratch/bjsmith.
2. The second (**-dm o:r-x**) sets the *default* ACL for "others" to read and execute (r-x) on all future files and subdirectories.

Execute flags: X vs. x

When setting permissions, the execute flag can be set to upper-case **X**, which differs from the lower-case **x** setting. The **X** permission allows execution only if the target is a directory or if the execute permission has already been set for the user or group. It is useful in the case of handling directory trees recursively.

More information

See the man pages for the commands for more information.

```
man setfacl
man getfacl
```