Permissions

CS14 - S26

Linux Permissions

In Unix based systems, every file/directory has an **owner**, a **group**, and a set of **permissions**.

These determine who can access a file and what they can do with it.

Users and Groups

Linux accounts are referred to as **users**.

Each user is part of one or more **groups**. Many users can belong to a single group.

- `groups` shows groups that a user belongs to.
- `getent group` shows all groups on the system.

Each file/directory belongs to a single user and a single group. This can be seen using `ls -l`.

* More fine-grained file/directory access can be supported with Access Control Lists, which are outside the scope of this lecture.

e.g., `/local/submissions` belongs to user “ckazer” and group “cs14user”

```
cs14user@cs14box00:~$ ls -l /local
total 4
drwxrwxr-x 4 ckazer cs14user 4096 Feb 11 18:13 submissions
cs14user@cs14box00:~$
```

File Access Modes/Permissions

In general, there are three things you can do with a file/directory in Linux.

- (R)ead - Access the contents of a file, e.g. print them to stdout
- (W)rite - Modify the contents of a file, e.g. in a text editor
- e(X)ecute - Treating the file as a script or binary executable, attempt to run it

These are collectively referred to as “access modes” or “permissions”, and each action can be set as allowed or disallowed.

Users, Groups, Others Permissions

For a file/directory, each access mode can be set for each the

- (u)ser the file/directory belongs to
- (g)roup the file/directory belongs to
- (o)thers, that is everyone else on the system

These permissions are listed in the order `ugo` by `` `ls -l` ``

e.g.,

- `foo` is a file.
  - The user has rw permissions
  - The group has rw permissions
  - others have r permissions
- `submissions` is a directory.
  - The user has rwx permissions
  - The group has rwx permissions
  - others have rx permissions

```
cs14user@cs14box00:~$ ls -l /local
total 4
-rw-rw-r-- 1 cs14user cs14user    0 Mar 16 17:46 foo
drwxrwxr-x 3 ckazer   cs14user 4096 Mar 16 17:23 submissions
cs14user@cs14box00:~$
```

Exercise: Check Permissions

1. On the CS Network, who owns `/usr/swat/bin`? What are its access permissions?
2. On The CS Network, who owns `~/.ssh`? What are its access permissions?
3. On your cs14vm, who owns `/local/submissions/exercises`? What are its access permissions?
4. For each of the above, why do you think the permissions are set the way that they are?

Changing Ownership

- A file's user owner can be changed with `chown`
- A file's group can be changed with `chgrp`

To use these commands, you must either be the user who owns the file, or be a superuser (e.g. `sudo chown`).

Changing Permissions

`chmod` is used to change file/directory permissions. You can change permissions using **symbolic** mode or **numeric** mode.

In symbolic mode you add and remove permissions using the ownership categories +/-/= the specific mode. E.g.,

- `chmod u+x foo` adds execute permissions for the user for `foo`
- `chmod go-rw` removes read and write permissions for the group and others for `foo`
- `chmod a=rx,u+w foo` sets read and execute permissions for all (user, group, and others) and write permissions for the user for `foo`

Changing Permissions, cont.

rwx permissions are represented internally as a single bit for each permission. So, an rwx triplet can be represented using a single octal number.

Read bit	Write bit	Execute bit	Octal
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
…	…	…	…
1	1	1	7

chmod also supports setting permissions by specifying the octal value for each category. E.g.,

- `chmod 600 foo` sets rw permissions for the user and no permissions for group or other
- `chmod 755 foo` sets rwx permissions for the user and rx permissions for group and others.

Some additional notes

- `chmod` can be invoked recursively with the `-R` flag before the mode arguments. Note that it is uppercase!
- You can set default permissions for new files by adding a call to `umask` to your `.bashrc`. See `man 1 umask` for details.
- For others to copy/execute files in a directory that they don't own, they must also have execute permissions for the directory containing the file.
    - E.g. the directory `/home/ckazer/public` has r-x permissions set for others so that everyone is allowed to copy files from there.

Exercise: Make a public directory

In your home directory, make a directory named `public`, add a file named `foo` to it, and fill the file with some text. Set the permissions so that everyone can read from `public` and can read `foo`. Check with your neighbors to see that they can access `foo` in your `/home/<username>/public` directory.

Once you are done, you can remove `public`.