8.9.4 Permission Facts

This lesson covers the following topics:

- Inode modes
- Permissions
- Managing permissions

Inode Modes

Every file and directory in the Linux file system has an *inode* (index node) that stores information about the file or directory, including when it was last modified, size, data block location, permissions, and ownership. The portion of the inode that stores permission information is called the *mode*. The mode has three sections:

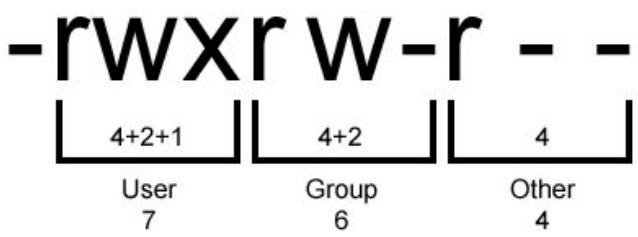
- User permissions (owner)
- Group permissions (group owner)
- Other permissions (everyone else on the Linux system who is not an owner or a member of the owning group)

There are three types of permissions contained in the mode, each of which is described in the table below:

Permission	Letter Abbreviation	Octal Value	Allowed Actions on Files	Allowed Actions on Directories
Read	r	4	Open and read the file	List directory contents if the execute permission is also present
Write	w	2	Edit the file and save the changes	Add, delete, and rename files if the execute permission is also present
Execute	x	1	Execute the file, if it's a program file or a shell script (must be used in conjunction with the read permission)	Enter the directory and access its contents

Permissions

Permissions are identified with either the letter abbreviation (**r**, **w**, or **x**), or the octal value that corresponds to the permission. The following graphic depicts how permissions are referenced:



In the mode:

- A **d** preceding the permissions indicates that the object is a directory.
- A dash (-) preceding the permissions identifies a file (the example above is for a file).
- Permissions are grouped according to user, group, or other permissions.
- If a given permission has not been assigned, a dash (-) takes its place in the mode.
- When using numbers to represent permissions, add the numbers together within each permission group. Then string the numbers together. For example, the mode of the file in the graphic above can be represented by the number 764.
- The root user has all permissions to files and directories regardless of the mode settings.

Managing Permissions

The table below lists the most common commands for managing permissions:

Command	Function	Example
ls -l	View a long listing of files and directories. A long listing displays the permissions assigned to files and directories (among other information)	**drwxr-xr-x** 22 root root 4096 Jun 19 15:01 sales

		(This is a directory with 755 permissions assigned.)
chmod	Change the permissions for the specified file. You can use the following syntax options: ▪ ***entity+permission*** adds a permission for a user, group, or other to a file or directory. ▪ ***entity-permission*** removes a permission for a user, group, or other from a file or directory. ▪ ***entity=permission*** sets the permission equal to the permission specified for a user, group, or other for a file or directory. ▪ **decimal_value** sets the permissions for the file according to the numbers represented for each mode entity. ▪ **-R** sets permissions recursively.	**chmod u+x,g+x,o+x myfile** adds the execute permission to the myfile file for user, group, and other. **chmod g-w,o-w myfile** removes the write permission for group and other from the myfile file. **chmod u=rwx myfile** grants the user read, write, and execute permissions for the myfile file. **chmod 711 myfile** grants the user read, write, and execute permissions (7) while group and other both receive execute permission (1) for the myfile file.
getfacl *file*	For each file, **getfacl** displays the file name, owner, the group, and the Access Control List (ACL). If a directory has a default ACL, **getfacl** also displays the default ACL. Non-directories cannot have default ACLs. getfacl options: ▪ **-a** Display the file access control list. ▪ **-d** Display the default access control list. ▪ **-c** Do not display the comment header (the first three lines of each file's output). ▪ **-e** Print all effective rights comments, even if identical to the rights defined by the ACL entry. ▪ **-E** Do not print effective rights comments. ▪ **-s** Skip files that only have the base ACL entries (owner, group,others. ▪ **-R** List the ACLs of all files and directories recursively. ▪ **-L** Logical walk, follow symbolic links to directories. The defaultbehavior is to follow symbolic link arguments, and skip symbolic links encountered in subdirectories. Only effective in combination with -R. ▪ **-P** Physical walk, do not follow symbolic links to directories. This also skips symbolic link arguments. Only effective in combination with -R. ▪ **-t** Use an alternative tabular output format. The ACL and the default ACL are displayed side by side. Permissions that are ineffective due to the ACL mask entry are displayed capitalized. The entry tag names for the ACL_USER_OBJ and ACL_GROUP_OBJ entries are also displayed in capital letters, which helps in spotting those entries. ▪ **-p** Do not strip leading slash characters ('/'). The default behavior is to strip leading slash characters. ▪ **-n** List numeric user and group IDs ▪ **-v** Print the version of getfacl and exit.	**getfacl myfile** Output: # file: myfile # owner: cpelfrey # group: admin user::rw- group::r-- other::r--
setfacl *file*	This utility sets Access Control Lists (ACLs) of files and directories. setfacl options: ▪ **-m** modifies the ACL of a file or directory. ACL entries for this operation must include permissions. ▪ **-x** remove ACL entries. It is not an error to remove an entry which does not exist. Only ACL entries without the perms field are accepted as parameters, unless the POSIXLY_CORRECT environment variable is defined. ▪ **-b** Remove all extended ACL entries. The base ACL entries of the owner, group and others are retained. ▪ **-k** Remove the Default ACL. If no Default ACL exists, no warnings are issued. ▪ **-n** Do not recalculate the effective rights mask. The default behavior of setfacl is to recalculate the ACL mask entry, unless a mask entry was explicitly given. The mask entry is set to the union of all permissions of the owning group, and all named user and group entries. (These are exactly the entries affected by the mask entry). ▪ **-d** All operations apply to the Default ACL. Regular ACL entries in the input set are promoted to Default ACL entries. Default ACL entries in the input set are discarded. (A warning is issued if that happens). ▪ **-R** Apply operations to all files and directories recursively. This option cannot be mixed with "--restore". ▪ **-L** "Logical walk": follow symbolic links to directories. The default behavior is to follow symbolic link arguments, and skip symbolic links encountered in subdirectories. Only effective in combination with -R. This option cannot be mixed with "--restore".	**setfacl -m u:cplefrey:r myfile** grants the user cplefry read access to the file named myfile. **setfacl -m m::rx myfile** Revoke write access from all groups and all named users (using the effective rights mask) for the file named myfile.

	▪ **-P** "Physical walk": do not follow symbolic links to directories. This also skips symbolic link arguments. Only effective in combination with -R. This option cannot be mixed with "--restore". ▪ **-v** Print the version of setfacl, and exit.	