ACCESS CONTROL ON FILES AND DIRECTORIES

Aim

1. To display the permissions of the files
2. To change a ownership of a file
3. To set the permissions using symbols
4. To set the permissions using binary masks
5. To display permissions of the directories
6. To set ownership permissions using sticky bit and umask

Commands

Permissions

A file or directory may have read, write, and execute permissions. When a file is created, it is automatically given read and write permissions for the owner, enabling you to display and modify the file.

Permission Categories

Three different categories of users can have access to a file or directory: the owner, the group, and all others not belonging to that group. The owner is the user who created the file. A user can grant access to a file to the members of a designated group. Finally, you can also open up access to a file to all other users on the system.

Read, Write, Execute Permissions

Each category has its own set of read, write, and execute permissions. The first set controls the user's own access to his or her files—the owner

access. The second set controls the access of the group to a user's files. The third set controls the access of all other users to the user's files.

1. To display the permissions of the files

The **ls** command with the **-l** option displays detailed information about the file, including the permissions. An empty permission is represented by a dash, **-**. The read permission is represented by **r**, write by **w**, and execute by **x**. There are ten positions. The first character indicates the file type. In a general sense, a directory can be considered a type of file. If the first character is a **dash**, a file is being listed. If the first character is **d**, information about a directory is being displayed. The next nine characters are arranged according to the different user categories. The first set of three characters is the **owner's** set of permissions for the file. The second set of three characters is the **group's** set of permissions for the file. The last set of three characters is the **other users'** set of permissions for the file.

Syntax

ls -l <filename>

Ownership

Files and directories belong to both an owner and a group. A group usually consists of a collection of users, all belonging to the same group. A group can also consist of one user, normally the user who creates the file. Each user on the system, including the root user, is assigned his or her own group of which he or she is the only member, ensuring access only by that user.

Command or Option	Execution
`chmod`	Changes the permission of a file or directory.
Options	
`+`	Adds a permission.
`-`	Removes a permission.
`=`	Assigns entire set of permissions.
`r`	Sets read permission for a file or directory. A file can be displayed or printed. A directory can have the list of its files displayed.
`w`	Sets write permission for a file or directory. A file can be edited or erased. A directory can be removed.
`x`	Sets execute permission for a file or directory. If the file is a shell script, it can be executed as a program. A directory can be changed to and entered.
`u`	Sets permissions for the user who created and owns the file or directory.
`g`	Sets permissions for group access to a file or directory.
`o`	Sets permissions for access to a file or directory by all other users on the system.
`a`	Sets permissions for access by the owner, group, and all other users.
`s`	Sets User ID and Group ID permission; program owned by owner and group.
`t`	Sets sticky bit permission; program remains in memory.
Commands	
`chgrp` *groupname filenames*	Changes the group for a file or files.
`chown` *user-name filenames*	Changes the owner of a file or files.
`ls -l` *filename*	Lists a filename with its permissions displayed.
`ls -ld` *directory*	Lists a directory name with its permissions displayed.
`ls -l`	Lists all files in a directory with its permissions displayed.

2. To change ownership of a file

The **chown** command transfers control over a file to another user. This command takes as its first argument the name of the other user. The username is followed by the files that are transferred. The group of a file can also be changed using the **chgrp** command. **chgrp** takes as its first argument the name of the new group for a file or files. Following the new group name, the files that are to be changed to that group are listed.

Syntax

chown <username> <filename>

chgrp <groupname> <filename>

3. To set permissions using symbols

The symbolic method of setting permissions uses the characters **r, w,** and **x** for **read, write, and execute**, respectively. Any of these permissions can be added or removed. The symbol to add a permission is the plus sign, **+**. The symbol to remove a permission is the minus sign, **-**. Permission symbols also specify each user category. The **owner, group, and others** categories are represented by the **u, g, and o** characters, respectively. The symbol for a category is placed before plus and minus sign preceding the read, write, and execute permissions. If no category symbol is used, all categories are assumed, and the permissions specified are set for the user, group, and others. Another permission character exists, **a**, which represents all the categories. The **a** character is the default. One of the most common permission operations is setting a file's executable permission. This is often done in the case of shell program files. The executable permission indicates a file contains executable instructions and can be directly run by the system.

Syntax

chmod u/g/o/a +rwx <filename>

4. To set permissions using binary masks

The absolute method changes all the permissions at once, instead of specifying one or the other. It uses a binary mask that references all the permissions in each category. The three categories, each with three permissions, conform to an octal binary format. Octal numbers have a base 8 structure. When translated into a binary number, each octal digit becomes three binary digits. A binary number is a set of 1 and 0 digits. Three octal digits in a number translate into three sets of three binary digits, which is nine altogether—and the exact number of permissions for a

file. Each octal digit applies to one of the user categories. The digits match up with the permission categories from left to right, beginning with the owner category. The first octal digit applies to the owner category, the second to the group, and the third to the others category. The actual octal digit you choose determines the read, write, and execute permissions for each category.

Syntax

chmod <numbers> filename

5. To display permissions of the directories

The **ls** command with the **-l** option lists all files in a directory. To list only the information about the directory itself, add a **d** modifier ,**–ld** shows the directory permissions.

Syntax

 # ls -ld <directory name>

6. To set ownership permissions

In addition to the read/write/execute permissions, the ownership permissions can also be set for executable programs. The Set User ID permission allows the original owner of the program to own it always, even while another user is running the program. To add both the User ID and Group ID permissions to a file, you use the **s** option.

Syntax

 # chmod +s <directory name>

Sticky bit permissions

Files in a directory with the sticky bit set can only be deleted or renamed by the root user or the owner of the directory. The sticky bit permission symbol is **t**. The sticky bit shows up as a **t** in the execute position of the other permissions. A program with read and execute permissions with the sticky bit has its permissions displayed as **r-t**.

Syntax

chmod +t <directoryname>

umask

Whenever a file or directory is created, it is given default permissions. The current defaults can be displayed or changes with the umask command. To display the current default permissions, use the umask command with no arguments. The **-S** option uses the symbolic format.

Syntax

$ umask

$ umask –S

$ umask <binary mask>