Managing File Permissions

Preparation

In your home directory, create a file called **tux.jpg** using the **touch** command:

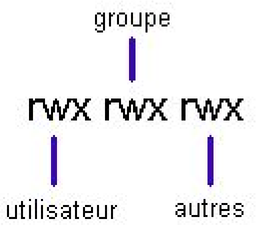
```
$ touch tux.jpg [Enter]
```

```
[trainee@centos ~]$ pwd
/home/trainee
[trainee@centos ~]$ touch tux.jpg
[trainee@centos ~]$ ls -l | grep tux.jpg
-rw-rw-r--. 1 trainee trainee     0 Oct 27 09:29 tux.jpg
```

<note important> The file **tux.jpg** is a text file. Linux does not use file extensions to determine file types. </note>

Basic Unix File Permissions

Basic Unix/Linux file permissions are expressed as follows:



;#; ;#;

where r = read, w = write and x = executable

Each **inode** contains the UID of the owner of a file. When the file is opened, the system compares the UID of the user opening the file with the UID stored in the inode. If they match, the user gets granted the permissions in the user section of the permissions **mask**. If they do not, the system compares the GID of the user opening the file with the GID stored in the inode. If they match the user gets granted the permissions in the **group** section of the mask. If neither the UID or the GID match, the user gets granted the permissions in the **other** section of the permission mask.

Permissions for directories are slightly different:

r	The user can list the contents of the directory.
w	The user can create or delete objects within the directory.
x	The user can position himself within the directory.

Changing Permissions with chmod

Symbolic Mode

Permissions can be changed by using the **chmod** command. The syntax of that command is as follows:

chmod [-R] ugoa +-= rwxXst *file or directory*

where:

u	user
g	group
o	other
a	all
+	add a permission
-	delete a permission
=	set the permissions as indicated
r	read
w	write
x	execute
X	execute - only if the target is a directory or if the file is already executable for one of the u, g or o categories.

s	SUID/SGID bit
t	sticky bit

For example:

```
$ chmod o+w tux.jpg [Enter]
```

will give write access to **other**:

```
[trainee@centos ~]$ chmod o+w tux.jpg
[trainee@centos ~]$ ls -l | grep tux.jpg
-rw-rw-rw-. 1 trainee trainee     0 Oct 27 09:29 tux.jpg
```

whilst:

```
$ chmod ug-w tux.jpg [Enter]
```

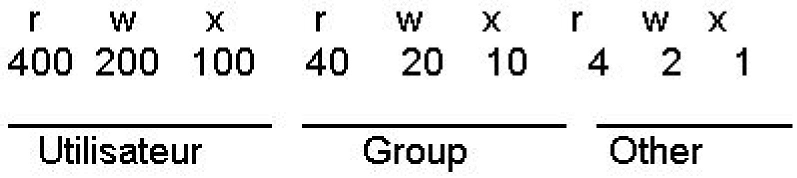
will remove write access for the **user** and the **group**:

```
[trainee@centos ~]$ chmod ug-w tux.jpg
[trainee@centos ~]$ ls -l | grep tux.jpg
-r--r--rw-. 1 trainee trainee     0 Oct 27 09:29 tux.jpg
```

<note tip> Only the owner of the file or **root** are able to change the permissions. </note>

Octal Mode

The **chmod** commande can also use the **Octal Mode** (8 base). The octal values of the permissions are as follows:



;#; ;#;

<note important> Full permissions are therefore **777**. </note>

In this case, the syntax of the chmod command is as follows:

chmod [-R] octal mode *file or directory*

For example, the following command:

```
$ chmod 644 tux.jpg [Enter]
```

corresponds to the following permissions: rw- r– r–

```
[trainee@centos ~]$ chmod 644 tux.jpg
[trainee@centos ~]$ ls -l | grep tux.jpg
-rw-r--r--. 1 trainee trainee     0 Oct 27 09:29 tux.jpg
```

The default permissions assigned to an object by the system differ depending on the type of object:

Directories	rwx rwx rwx	777
Files	rw– rw– rw-	666

Command Line Switches

The command line switches for the chmod command are :

```
[trainee@centos ~]$ chmod --help
Usage: chmod [OPTION]... MODE[,MODE]... FILE...
```

```
  or:  chmod [OPTION]... OCTAL-MODE FILE...
  or:  chmod [OPTION]... --reference=RFILE FILE...
Change the mode of each FILE to MODE.

  -c, --changes           like verbose but report only when a change is made
      --no-preserve-root  do not treat `/' specially (the default)
      --preserve-root     fail to operate recursively on `/'
  -f, --silent, --quiet   suppress most error messages
  -v, --verbose           output a diagnostic for every file processed
      --reference=RFILE   use RFILE's mode instead of MODE values
  -R, --recursive         change files and directories recursively
      --help     display this help and exit
      --version  output version information and exit

Each MODE is of the form `[ugoa]*([-+=]([rwxXst]*|[ugo]))+'.

Report chmod bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'chmod invocation'
```

The umask command

A user can change these default values by modifying his/her **umask** value:

```
[trainee@centos ~]$ umask
0002
```

The value of the umask us deducted from the default permissions when the object is created:

Default permissions for a file	rw- rw- rw-	666
umask value	— -w- -w-	022

Effective permissions	rw- r– r–	644

Consider the following example:

```
[trainee@centos ~]$ umask 044
[trainee@centos ~]$ touch tux1.jpg
[trainee@centos ~]$ ls -l | grep tux1.jpg
-rw--w--w-. 1 trainee trainee     0 Oct 27 09:32 tux1.jpg
[trainee@centos ~]$ umask 0002
[trainee@centos ~]$ umask
0002
```

Commande Line Switches

The command line switches for the umask command are :

```
[trainee@centos ~]$ help umask
umask: umask [-p] [-S] [mode]
    Display or set file mode mask.
    Sets the user file-creation mask to MODE.  If MODE is omitted, prints
    the current value of the mask.
    If MODE begins with a digit, it is interpreted as an octal number;
    otherwise it is a symbolic mode string like that accepted by chmod(1).
    Options:
      -p    if MODE is omitted, output in a form that may be reused as input
      -S    makes the output symbolic; otherwise an octal number is output
    Exit Status:
    Returns success unless MODE is invalid or an invalid option is given.
```

Changing the Owner or the Group with chown and chgrp

Changing the owner of an object can only be done by **root**.

In the following example, tux.jpg belongs to the **trainee** user. **root** can modify the owner by using the following command:

```
# chown root tux.jpg [Enter]
```

```
[trainee@centos ~]$ su -
Password: fenestros
[root@centos ~]# cd /home/trainee
[root@centos trainee]# chown root tux.jpg
[root@centos trainee]# ls -l | grep tux.jpg
-rw-r--r--. 1 root    trainee     0 Oct 27 09:29 tux.jpg
```

A similar operation is used to change the group:

```
# chgrp root tux.jpg [Enter]
```

for example :

```
[root@centos trainee]# chgrp root tux.jpg
[root@centos trainee]# ls -l | grep tux.jpg
-rw-r--r--. 1 root    root        0 Oct 27 09:29 tux.jpg
```

Commande Line Switches

The command line switches for the chown command are :

```
[root@centos ~]# chown --help
Usage: chown [OPTION]... [OWNER][:[GROUP]] FILE...
  or:  chown [OPTION]... --reference=RFILE FILE...
Change the owner and/or group of each FILE to OWNER and/or GROUP.
With --reference, change the owner and group of each FILE to those of RFILE.

```

```
  -c, --changes          like verbose but report only when a change is made
      --dereference      affect the referent of each symbolic link (this is
                         the default), rather than the symbolic link itself
  -h, --no-dereference   affect each symbolic link instead of any referenced
                         file (useful only on systems that can change the
                         ownership of a symlink)
      --from=CURRENT_OWNER:CURRENT_GROUP
                         change the owner and/or group of each file only if
                         its current owner and/or group match those specified
                         here.  Either may be omitted, in which case a match
                         is not required for the omitted attribute.
      --no-preserve-root  do not treat `/' specially (the default)
      --preserve-root    fail to operate recursively on `/'
  -f, --silent, --quiet  suppress most error messages
      --reference=RFILE  use RFILE's owner and group rather than
                         specifying OWNER:GROUP values
  -R, --recursive        operate on files and directories recursively
  -v, --verbose          output a diagnostic for every file processed

The following options modify how a hierarchy is traversed when the -R
option is also specified.  If more than one is specified, only the final
one takes effect.

  -H                     if a command line argument is a symbolic link
                         to a directory, traverse it
  -L                     traverse every symbolic link to a directory
                         encountered
  -P                     do not traverse any symbolic links (default)

      --help     display this help and exit
      --version  output version information and exit

Owner is unchanged if missing.  Group is unchanged if missing, but changed
to login group if implied by a `:' following a symbolic OWNER.
```

```
OWNER and GROUP may be numeric as well as symbolic.

Examples:
  chown root /u        Change the owner of /u to "root".
  chown root:staff /u  Likewise, but also change its group to "staff".
  chown -hR root /u    Change the owner of /u and subfiles to "root".

Report chown bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'chown invocation'
```

The command line switches for the chgrp command are :

```
[root@centos ~]# chgrp --help
Usage: chgrp [OPTION]... GROUP FILE...
  or:  chgrp [OPTION]... --reference=RFILE FILE...
Change the group of each FILE to GROUP.
With --reference, change the group of each FILE to that of RFILE.

  -c, --changes          like verbose but report only when a change is made
      --dereference      affect the referent of each symbolic link (this is
                         the default), rather than the symbolic link itself
  -h, --no-dereference   affect each symbolic link instead of any referenced
                         file (useful only on systems that can change the
                         ownership of a symlink)
      --no-preserve-root  do not treat `/' specially (the default)
      --preserve-root    fail to operate recursively on `/'
  -f, --silent, --quiet  suppress most error messages
      --reference=RFILE  use RFILE's group rather than specifying a
                         GROUP value
  -R, --recursive        operate on files and directories recursively
  -v, --verbose          output a diagnostic for every file processed
```

```
The following options modify how a hierarchy is traversed when the -R
option is also specified.  If more than one is specified, only the final
one takes effect.

  -H                     if a command line argument is a symbolic link
                         to a directory, traverse it
  -L                     traverse every symbolic link to a directory
                         encountered
  -P                     do not traverse any symbolic links (default)

      --help     display this help and exit
      --version  output version information and exit

Examples:
  chgrp staff /u      Change the group of /u to "staff".
  chgrp -hR staff /u  Change the group of /u and subfiles to "staff".

Report chgrp bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'chgrp invocation'
```

Advanced Unix Permissions

SUID/SGID bit

The following command prints to standard output information concerning the **/etc/passwd** file and the binary **/usr/bin/passwd**. The latter can be used by any user to change his/her password. By doing so, the user writes to the /etc/passwd file. However, a casual glance at the permissions of the /etc/passwd file indicates that only root can write to that file.

```
[root@centos trainee]# ls -l /etc/passwd /usr/bin/passwd
```

```
-rw-r--r--. 1 root root  1581 Oct 25 10:36 /etc/passwd
-rwsr-xr-x. 1 root root 25980 Feb 22  2012 /usr/bin/passwd
```

To remedy this apparent contradiction, Linux uses two advanced permissions:

- Set UserID bit (SUID bit)
- Set GroupID bit (SGID bit)

When a SUID bit is placed on a binary, the user that executes that binary is given the UID of the owner of that binary for the duration of it'd execution, in this case the UID of **root**. The SUID bit is represented by the letter **s** in the user part of the permissions mask.

The same can also be applied to the group by placing the SGID bit, represented by the letter **s** in the group part of the permissions mask

To assign the advanced permissions it is possible to use the Symbolic Mode:

- chmod u+s file/directory
- chmod g+s file/directory

Or the Octal Mode where each advanced permission is assigned a value:

- SUID = 4000
- SGID = 2000

Inheritance Flag

The SGID bit can also be placed on a directory. In this case, the files and directories created within the directory are given the group of the parent directory. This advanced permission is called the **Inheritance Flag**.

For example:

```
[root@centos trainee]# cd /tmp
[root@centos tmp]# mkdir inherit
[root@centos tmp]# chown root:trainee inherit
[root@centos tmp]# chmod g+s inherit
```

```
[root@centos tmp]# touch inherit/test.txt
[root@centos inherit]# mkdir inherit/testrep
[root@centos tmp]# cd inherit; ls -l
[root@centos tmp]# cd inherit; ls -l
total 4
drwxr-sr-x. 2 root trainee 4096 Oct 27 09:37 testrep
-rw-r--r--. 1 root trainee    0 Oct 27 09:35 test.txt
```

<note important> Note that the Inheritance Flag has been automatically assigned to the **testrep** directory. </note>

Sticky bit

The last advanced permission is calle the **sticky** bit. The sticky bit is assigned to directories where everyone has full file permissions such as the **/tmp** directory. By assigning the sticky bit, only the owner of an object can delete it. The sticky bit is assigned by using one of the two following methods :

```
# chmod o+t /directory
```

or

```
# chmod 1777 /directory
```

For example:

```
# mkdir /tmp/public_directory; cd /tmp; chmod o+t public_directory [Enter]
```

or

```
# mkdir /tmp/public_directory; cd /tmp; chmod 1777 public_directory [Enter]
```

will create the public_directory with the following permissions:

```
[root@centos inherit]# mkdir /tmp/public_directory; cd /tmp; chmod o+t public_directory
```

```
[root@centos tmp]# ls -l | grep public_directory
drwxr-xr-t. 2 root    root     4096 Oct 27 09:38 public_directory
```

ACLs

An extension to the permissions under Linux are the ACLs.

Navigate to the existing /tmp/inherit directory and list the ACLs for the test.txt file:

```
[root@centos ~]# cd /tmp/inherit
[root@centos inherit]# getfacl test.txt
# file: test.txt
# owner: root
# group: trainee
user::rw-
group::r--
other::r--
```

To set ACLs on a file, you need to use the **setfacl** command:

```
# setfacl --set u::rwx,g::rx,o::-,u:trainee:rw test.txt [Enter]
```

Once again use the getfacl command to view the ACLs :

```
# getfacl test.txt [Entrée]
```

YOu will obtain a result similar to the following example:

```
[root@centos inherit]# setfacl --set u::rwx,g::rx,o::-,u:trainee:rw test.txt
[root@centos inherit]# getfacl test.txt
# file: test.txt
# owner: root
# group: trainee
user::rwx
user:trainee:rw-
group::r-x
mask::rwx
other::---
```

<note> The owner of the file has **rwx** permissions. User trainee has **rw-** permissions. </note>

<note warning>

```
         mask    A mask ACL entry specifies the maximum access which can be
                 granted by any ACL entry except the user entry for the file owner
                 and the other entry (entry tag type ACL_MASK).
```

</note>

ACLs on directories are managed slightly differently. Placing ACLs on the directory testrep takes the following form :

```
# setfacl --set d:u::r,d:g::-,d:o::- testrep [Enter]
```

The use of the letter **d** here means you are setting **default** ACLs.

Now create a file called **test1.txt** in the **testrep** directory:

```
# touch /tmp/inherit/testrep/test1.txt [Entrée]
```

Once again use the getfacl command to see the ACLs:

```
# getfacl -R /tmp/inherit/testrep [Enter]
```

You will obtain a result similar to the following example:

```
[root@centos inherit]# setfacl --set d:u::r,d:g::-,d:o::- testrep
```

```
[root@centos inherit]# touch /tmp/inherit/testrep/test1.txt
[root@centos inherit]# getfacl -R /tmp/inherit/testrep
getfacl: Removing leading '/' from absolute path names
# file: tmp/inherit/testrep
# owner: root
# group: trainee
# flags: -s-
user::rwx
group::r-x
other::r-x
default:user::r--
default:group::---
default:other::---

# file: tmp/inherit/testrep/test1.txt
# owner: root
# group: trainee
user::r--
group::---
other::---
```

<note important> The ACLs positioned on the file test1.txt are the ACLs positioned by default on the parent directory. </note>

Lastly the standard archiving commands under Linux do not understand ACLs. As a result, the ACLs need to be backed-up to a file using the following command:

```
# getfacl -R --skip-base . > backup.acl [Enter]
```

Restoring ACLs is acheived by using the following command:

```
# setfacl --restore=backup.acl [Enter]
```

<note important> In order to be able to **use** the ACLs, the partition concerned must be mounted with the **acl** option. </note>

Commande Line Switches

The command line switches for the setfacl command are :

```
[root@centos inherit]# setfacl --help
setfacl 2.2.49 -- set file access control lists
Usage: setfacl [-bkndRLP] { -m|-M|-x|-X ... } file ...
  -m, --modify=acl        modify the current ACL(s) of file(s)
  -M, --modify-file=file  read ACL entries to modify from file
  -x, --remove=acl        remove entries from the ACL(s) of file(s)
  -X, --remove-file=file  read ACL entries to remove from file
  -b, --remove-all        remove all extended ACL entries
  -k, --remove-default    remove the default ACL
      --set=acl           set the ACL of file(s), replacing the current ACL
      --set-file=file     read ACL entries to set from file
      --mask              do recalculate the effective rights mask
  -n, --no-mask           don't recalculate the effective rights mask
  -d, --default           operations apply to the default ACL
  -R, --recursive         recurse into subdirectories
  -L, --logical           logical walk, follow symbolic links
  -P, --physical          physical walk, do not follow symbolic links
      --restore=file      restore ACLs (inverse of `getfacl -R')
      --test              test mode (ACLs are not modified)
  -v, --version           print version and exit
  -h, --help              this help text
```

The command line switches for the getfacl command are :

```
[root@centos inherit]# getfacl --help
getfacl 2.2.49 -- get file access control lists
Usage: getfacl [-aceEsRLPtpndvh] file ...
  -a,  --access           display the file access control list only
  -d, --default           display the default access control list only
  -c, --omit-header       do not display the comment header
  -e, --all-effective     print all effective rights
  -E, --no-effective      print no effective rights
  -s, --skip-base         skip files that only have the base entries
  -R, --recursive         recurse into subdirectories
  -L, --logical           logical walk, follow symbolic links
  -P, --physical          physical walk, do not follow symbolic links
  -t, --tabular           use tabular output format
  -n, --numeric           print numeric user/group identifiers
  -p, --absolute-names    don't strip leading '/' in pathnames
  -v, --version           print version and exit
  -h, --help              this help text
```

Ext2/Ext3/Ext4 Attributes

File attributes are an addition to the classic file permissions in Ext2/Ext3 and ReiserFS file systems.

The principal attributes are :

Attribute	Description
a	The file cannot be deleted and only the addition of data to the file is permitted. This attribute is often used for log files.
i	The file can neither be deleted, modified or moved. In addition, a link cannot be placed on the file.
s	The file will be physically destroyed when deleted.
D	Synchronous directory.
S	Synchronous file.
A	The date and time of the last file access are not updated in the inode.

<note tip> Synchronous implies that the modifications are immediately written to disk. </note>

The two commands associated with attributes are:

Command	Description
chattr	Modify the attributes.
lsattr	View attributes.

To clarify the use of the two commands, create the following directory: **/tmp/attributes/dir**.

```
[root@centos inherit]# cd /tmp; mkdir -p attributes/dir
```

Next create the files **/tmp/attributes/file1** and **/tmp/attributes/dir/file2** :

```
[root@centos tmp]# touch attributes/file1
[root@centos tmp]# touch attributes/dir/file2
```

Now modify the attributes recursively :

```
[root@centos tmp]# chattr +i -R attributes/
```

View the attributes using the **lsattr** command :

```
[root@centos tmp]# lsattr -R attributes
----i--------e- attributes/dir

attributes/dir:
----i--------e- attributes/dir/file2

----i--------e- attributes/file1
```

If you now try and move **file1** to **/tmp/attributes/dir/**, you will get the following error message:

```
[root@centos tmp]# mv /tmp/attributes/file1 /tmp/attributes/dir/file1
mv: cannot move `/tmp/attributes/file1' to `/tmp/attributes/dir/file1': Permission denied
```

~~DISCUSSION:off~~