Version : **2024.01**

Last update : 2024/11/27 10:10

RH12408 - Managing File Permissions

Module content

- **RH12408 - Managing File Permissions**
  - Contents
  - Presentation
  - Preparation
  - LAB #1 - Simple Unix File Permissions
    - 1.1 - Changing File Permissions
      - The chmod Command
        - Symbolic Mode
        - Octal mode
      - The umask command
    - 1.2 - Changing the owner or group
      - The chown command
      - The chgrp command
  - LAB #2 - Advanced Unix File Permissions
    - 2.1 - SUID/SGID bit
    - 2.2 - Inheritance Flag
    - 2.3 - Sticky bit
  - LAB #3 - Extending Linux Permissions using ACLs and Attributes
    - 3.1 - ACLs
    - 3.2 - Attributes

Presentation

In its basic design, Linux uses a **DAC** security approach:

Security Type	Name	Description
DAC	*Discretional Access Control*	Accessing file objects is a function of the identity of the accessing user (user,group). A user can allow other users to access his/her objects.

Preparation

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

```
[root@redhat9 ~]# exit
logout

[trainee@redhat9 ~]$ pwd
/home/trainee

[trainee@redhat9 ~]$ touch tux.jpg

[trainee@redhat9 ~]$ ls -l | grep tux.jpg
-rw-r--r--. 1 trainee trainee 0 Sep 27 12:42 tux.jpg
```

Important : The file **tux.jpg** is a text file. Linux does not use file extensions to determine file types.

LAB #1 - Simple Unix File Permissions

File permissions in Linux are communicated as follows:

User/Owner	Group	Others
rwx	rwx	rwx

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

Each inode stores the UID of the user to whom the file belongs and the GID. When the file is opened, the system compares the user UID with the UID stored in the inode (Reference User). If these two numbers are identical, the user obtains the permissions of the file owner. If the numbers differ, the system checks whether the user is in the group referenced in the inode. If so, the user will have the permissions specified for the group. If no conditions are met, the user is given the permissions of 'others'.

The 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.

1.1 - Changing permissions

The chmod Command

Symbolic Mode

To modify file access permissions, use the chmod command, whose syntax is as follows:

chmod [-R] category operator permissions file_name

or

chmod [-R] ugoa +-= rwxXst file_name

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 the following command will give others write access to the file tux.jpg :

```
[trainee@redhat9 ~]$ chmod o+w tux.jpg

[trainee@redhat9 ~]$ ls -l | grep tux.jpg
-rw-r--rw-. 1 trainee trainee 0 Sep 27 12:42 tux.jpg
```

while the following command will remove write access permissions for the user and group:

```
[trainee@redhat9 ~]$ chmod ug-w tux.jpg

[trainee@redhat9 ~]$ ls -l | grep tux.jpg
-r--r--rw-. 1 trainee trainee 0 Sep 27 12:42 tux.jpg
```

Octal mode

The chmod command can also be used with an octal representation. The octal values for access permissions are:

User/Owner			Group			Other		
r	w	x	r	w	x	r	w	x
400	200	100	40	20	10	4	2	1

Important : Full permissions are therefore **777**

The chmod command takes the following form:

```
chmod [ -R ] Octal_value filename
```

For example, the following command corresponds to the following permissions: rw- r– r–:

```
[trainee@redhat9 ~]$ chmod 644 tux.jpg

[trainee@redhat9 ~]$ ls -l | grep tux.jpg
-rw-r--r--. 1 trainee trainee 0 Sep 27 12:42 tux.jpg
```

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

Directories	rwx rwx rwx	777
Normal file	rw- rw- rw-	666

Command Line Switches

The options for this command are:

```
[trainee@redhat9 ~]$ 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.
With --reference, change the mode of each FILE to that of RFILE.

  -c, --changes like verbose but report only when a change is made
  -f, --silent, --quiet suppress most error messages
  -v, --verbose output a diagnostic for every file processed
      --no-preserve-root do not treat '/' specially (the default)
      --preserve-root fail to operate recursively on '/'
      --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]))+|[-+=][0-7]+'.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/chmod>
or available locally via: info '(coreutils) chmod invocation'
```

The umask Command

Users can change their default permission mask when creating objects using the umask command :

```
[trainee@redhat9 ~]$ umask
0022
[trainee@redhat9 ~]$ su -
Password: fenestros
[root@redhat9 ~]# umask
```

```
0022
[root@redhat9 ~]# exit
logout
[trainee@redhat9 ~]$
```

For example:

```
$ umask 002 [Enter]
```

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

Maximum mask when creating a file	rw- rw- rw-	666
Permissions to be removed	— -w-	002
Result	rw- rw- r–	664

Consider the following example:

```
[trainee@redhat9 ~]$ umask 044

[trainee@redhat9 ~]$ touch tux1.jpg

[trainee@redhat9 ~]$ ls -l | grep tux1.jpg
-rw--w--w-. 1 trainee trainee 0 Sep 27 12:48 tux1.jpg

[trainee@redhat9 ~]$ umask 022

[trainee@redhat9 ~]$ umask
0022
```

Command Line Switches

The options for this command are:

```
[trainee@redhat9 ~]$ 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.
```

.2 - Change owner or group

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

The chown Command

In the case of the file tux.jpg belonging to trainee, root can change the owner from trainee to root with the following command:

```
[trainee@redhat9 ~]$ su -
Password: fenestros

[root@redhat9 ~]# cd /home/trainee

[root@redhat9 trainee]# chown root tux.jpg

[root@redhat9 trainee]# ls -l | grep tux.jpg
```

```
-rw-r--r--. 1 root trainee 0 Sep 27 12:42 tux.jpg
```

Command Line Switches

The options for this command are:

```
[root@redhat9 trainee]# 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
  -f, --silent, --quiet suppress most error messages
  -v, --verbose output a diagnostic for every file processed
      --dereference affect the referent of each symbolic link (this is
                        the default), rather than the symbolic link itself
  -h, --no-dereference affect symbolic links 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 '/'
      --reference=RFILE use RFILE's owner and group rather than
                        specifying OWNER:GROUP values
  -R, --recursive operate on files and directories recursively

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’.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/chown>
or available locally via: info ‘(coreutils) chown invocation’
```

The chgrp Command

The same applies to the group :

```
[root@redhat9 trainee]# chgrp root tux.jpg

[root@redhat9 trainee]# ls -l | grep tux.jpg
-rw-r--r--. 1 root root 0 Sep 27 12:42 tux.jpg
```

Important: The permission to delete a file depends on the permissions of the directory in which the file is stored, not the permissions of the file itself.

Command Line Switches

The options for this command are:

```
[root@redhat9 trainee]# 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
  -f, --silent, --quiet suppress most error messages
  -v, --verbose output a diagnostic for every file processed
      --dereference affect the referent of each symbolic link (this is
                        the default), rather than the symbolic link itself
  -h, --no-dereference affect symbolic links 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 '/' (the default)
      --reference=RFILE use RFILE's group rather than specifying a
                        GROUP value
  -R, --recursive operate on files and directories recursively

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’.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/chgrp>
or available locally via: info ‘(coreutils) chgrp invocation’
```

LAB #2 - Advanced Unix Permissions

2.1 - 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, note that the permissions of the /etc/passwd file indicate that only root can write to that file:

```
[root@redhat9 trainee]# ls -l /etc/passwd /usr/bin/passwd
-rw-r--r--. 1 root root 2162 Sep 26 14:57 /etc/passwd
-rwsr-xr-x. 1 root root 32648 Aug 10 2021 /usr/bin/passwd
```

To remedy this apparent contradiction, Linux has two advanced access file 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 its execution.

In the case of a password change, each user who launches the /usr/bin/passwd program is temporarily assigned the user number of the owner of the /usr/bin/passwd program, i.e. root. This advanced file permission is indicated by the letter s instead of the letter x in the user/owner part of the mask.

The same function exists for the group, using the SGID bit.

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

- chmod u+s file_name
- chmod g+s filename

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

- SUID = 4000
- SGID = 2000

To identify executables with the SGID or SUID bit, use the following command:

```
[root@redhat9 trainee]# find / -type f \( -perm -4000 -o -perm -2000 \) -exec ls {} \;
find: ‘/proc/9995/task/9995/fdinfo/6’: No such file or directory
find: ‘/proc/9995/fdinfo/5’: No such file or directory
/usr/bin/fusermount3
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/fusermount
/usr/bin/mount
/usr/bin/umount
/usr/bin/su
```

```
/usr/bin/write
/usr/bin/pkexec
/usr/bin/crontab
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/locate
/usr/bin/chsh
/usr/bin/vmware-user-suid-wrapper
/usr/bin/at
/usr/bin/chfn
/usr/bin/screen
/usr/sbin/grub2-set-bootflag
/usr/sbin/pam_timestamp_check
/usr/sbin/unix_chkpwd
/usr/sbin/userhelper
/usr/sbin/lockdev
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/libexec/utempter
/usr/libexec/openssh/ssh-keysign
/usr/libexec/dbus-1/dbus-daemon-launch-helper
/usr/libexec/sssd/krb5_child
/usr/libexec/sssd/ldap_child
/usr/libexec/sssd/proxy_child
/usr/libexec/sssd/selinux_child
/usr/libexec/Xorg.wrap
/usr/libexec/cockpit-session
```

2.2 - Inheritance Flag

The SGID bit can also be assigned to a directory. In this way, files and directories created inside will have the group of the parent directory as their group. This advanced file permission is therefore called the **Inheritance Flag**.

For example:

```
[root@redhat9 trainee]# cd /tmp

[root@redhat9 tmp]# mkdir inherit

[root@redhat9 tmp]# chown root:trainee inherit

[root@redhat9 tmp]# chmod g+s inherit

[root@redhat9 tmp]# touch inherit/test.txt

[root@redhat9 tmp]# mkdir inherit/testdir

[root@redhat9 tmp]# cd inherit; ls -l
total 0
drwxr-sr-x. 2 root trainee 6 Sep 27 12:55 testdir
-rw-r--r--. 1 root trainee 0 Sep 27 12:54 test.txt

[root@redhat9 inherit]#
```

Important: Note that despite the fact that root created the two objects, they are not associated with the **root** group but with the **trainee** group, i.e. the group of the parent directory (inherit). Also note that the system has set the inheritance flag on the **testdir** subdirectory.

2.3 - Sticky bit

There is one last case which is called the sticky bit. The sticky bit is used for directories where everyone has full permissions. In this case, anyone can delete files in the directory. By adding the sticky bit, only the owner of the file can delete it.

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

or

```
# chmod 1777 /directory
```

For example:

```
[root@redhat9 inherit]# mkdir /tmp/public_directory; cd /tmp; chmod o+t public_directory

[root@redhat9 tmp]# ls -l | grep public_directory
drwxr-xr-t. 2 root root 6 Sep 27 12:56 public_directory
```

LAB #3 - Extending Linux Permissions using ACLs and File Attributes

3.1 - ACLs

An extension to the permissions under Linux are the ACLs.

To list the ACL's on a file, use the **getfacl** file:

```
[root@redhat9 tmp]# getfacl /home/trainee/tux.jpg
getfacl: Removing leading '/' from absolute path names
# file: home/trainee/tux.jpg
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

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

```
[root@redhat9 tmp]# setfacl --set u::rwx,g::rx,o::-,u:trainee:rw /home/trainee/tux.jpg
[root@redhat9 tmp]# getfacl /home/trainee/tux.jpg
getfacl: Removing leading '/' from absolute path names
# file: home/trainee/tux.jpg
# owner: root
# group: root
user::rwx
user:trainee:rw-
group::r-x
mask::rwx
other::---
```

Important - 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).

Create the directory /home/trainee/dir1 :

```
[root@redhat9 tmp]# mkdir /home/trainee/dir1
```

ACLs on directories are managed slightly differently. Placing ACLs on the directory dir1 takes the following form :

```
[root@redhat9 tmp]# setfacl --set d:u::r,d:g::-,d:o::- /home/trainee/dir1
```

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

Now create a file called **file1** in the **dir1** directory:

```
[root@redhat9 tmp]# touch /home/trainee/dir1/file1
```

Once again use the getfacl command to see the ACLs:

```
[root@redhat9 tmp]# getfacl /home/trainee/dir1
getfacl: Removing leading '/' from absolute path names
# file: home/trainee/dir1
# owner: root
# group: root
user::rwx
group::r-x
other::r-x
default:user::r--
default:group::---
default:other::---

[root@redhat9 tmp]# getfacl /home/trainee/dir1/file1
getfacl: Removing leading '/' from absolute path names
# file: home/trainee/dir1/file1
# owner: root
# group: root
user::r--
group::---
other::---
```

The ACLs positioned on the file **file1** are the ACLs positioned by default on the parent directory.

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:

```
[root@redhat9 tmp]# cd /home/trainee/dir1
[root@redhat9 dir1]# getfacl -R --skip-base . > backup.acl
[root@redhat9 dir1]# cat backup.acl
# file: .
# owner: root
# group: root
user::rwx
group::r-x
```

```
other::r-x
default:user::r--
default:group::---
default:other::---
```

Restoring ACLs is acheived by using the following command:

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

Commande Line Switches

The command line switches for the getfacl command are :

```
[root@redhat9 dir1]# getfacl --help
getfacl 2.2.53 -- 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
```

The command line switches for the setfacl command are :

```
[root@redhat9 dir1]# setfacl --help
setfacl 2.2.53 -- 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
```

3.2 - Attributes

File attributes are an addition to the classic file permissions in Ext2/Ext3/Ext4 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.

Attribute	Description
D	Synchronous directory.
S	Synchronous file.
A	The date and time of the last file access are not updated in the inode.

Important - Synchronous implies that the modifications are immediately written to disk.

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 directory **/root/attributs/dir**:

```
[root@redhat9 dir1]# cd /root
[root@redhat9 ~]# mkdir -p attributs/dir
```

Create the files **file** et **dir/file1** :

```
[root@redhat9 ~]# touch attributs/file
[root@redhat9 ~]# touch attributs/dir/file1
```

Now modify the attributes recursively:

```
[root@redhat9 ~]# chattr +i -R attributs/
```

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

```
[root@redhat9 ~]# lsattr -R attributs
----i----------- attributs/dir
```

```
attributs/dir:
----i---------- attributs/dir/file1

----i---------- attributs/file
```

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

```
[root@redhat9 ~]# cd attributs; mv /root/attributs/file /root/attributs/dir/file
mv: cannot move '/root/attributs/file' to '/root/attributs/dir/file': Operation not permitted
```
