Linux System Administration and IP Services
TERNET 2012

Exercises: Permissions

Notes

* Commands preceded with "$" imply that you should execute the command as
  a general user - not as root.
* Commands preceded with "#" imply that you should be working as root with
  "sudo"
* Commands with more specific command lines (e.g. "RTR-GW>" or "mysql>")
  imply that you are executing commands on remote equipment, or within
  another program.

REFERENCE

If you look at files in a directory using "ls -al" you will see the
permissions for each file and directories. Here is an example:

```
drwxrwxr-x    3 hervey   hervey       4096 Feb 25 09:49 directory
-rwxr--r--   12 hervey   hervey       4096 Feb 16 05:02 file
```

The left column is important. You can view it like this:

```
Type User    Group Other Links  owner  group  size   date   hour  name
d    rwx     rwx   r-x   3      hervey hervey 4096   Feb 25 09:49 directory
-    rwx     r     r     12     hervey hervey 4096   Feb 16 05:02 file
```

So, the directory has r (read), w (write), x (execute) access for the
User and Group. For Other it has r (read) and x (execute) access. The
file has read/write/execute access for User and read only access for
everyone else (Group and Other).

To change permissions you use the "chmod" command. chmod uses a base
eight (octal) system to configure permissions. Or, you can use an
alternate form to specify permissions by column (User/Group/Other) at a
time.

Permissions have values like this:

```
Letter  Permission   Value

R       read         4
W       write        2
X       execute      1
-       none         0
```

Thus you can give permissions to a file using the sum of the values for
each permission you wish to give for each column. Here is an example:

```
Letter  Permission                  Value

---     none                        0
```

--x	execute	1
-w-	write only (rarely used)	2
-wx	write and execute (rare)	3
r--	read only	4
r-x	read and execute	5
rw-	read and write	6
rwx	read, write, and execute	7

This is just one column. Since we have three areas of permissions (User, Group, Other), it looks like this will all 3 sets:

Permissions	Numeric equivalent	Description
-rw-------	600	User has read & execute permission.
-rw-r--r--	644	User has read & execute. Group and Other have read permission.
-rw-rw-rw-	666	Everyone (User, Group, Other) have read & write permission (dangerous?)
-rwx------	700	User has read, write, & execute permission.
-rwxr-xr-x	755	User has read, write, & execute permission. Rest of the world (Other) has read & execute permission (typical for web pages or 644).
-rwxrwxrwx	777	Everyone has full access (read, write, execute).
-rwx--x--x	711	User has read, write, execute permission. Group and world have execute permission.
drwx------	700	User only has access to this directory. Directories require execute permission to access.
drwxr-xr-x	755	User has full access to directory. Everyone else can see the directory.
drwx--x--x	711	Everyone can list files in the directory, but Group and Other need to know a filename to do this.

1.) CHANGING FILE PERMISSIONS

If you are logged in as the root user on your machine please do the following:

```
# exit
```

To become a normal user, like sysadm. Your prompt should change to include a “$” sign.

```
$
```

Once logged in we'll create a file and set permissions on it in various ways.

```
$ cd
$ echo “test file” > working.txt
$ chmod 444 working.txt
```

What does that look like?

```
$ ls -lah working.txt
```

In spite of the fact that the file does not have write permission
for the owner, the owner can still change the file's permissions so
that they can make it possible to write to it.

```
$ chmod 644 working.txt
```

Or, you can do this by using this form of chmod:

```
$ chmod u+w working.txt
```

Note: when you type these command you should be able to use the tab key for
command completion once you've typed the "w" in the file name "working.txt" -
This will save you quite a bit of time. It's highly recommended! :-)

To remove the read permission for the user on a file you would do

```
$ chmod u-r working.txt
```

Or, you can do something like:

```
$ chmod 344 working.txt
```

You probably noticed that you can use the "-" (minus) sign to remove
permissions from a file. Try reading your file:

```
$ cat working.txt
```

What happened? Uh oh! You can't read your file. Please make the file readable
by you !

```
$ chmod ??? working.txt
```

Ask your instructor for help if you don't know what to put in for
“???”. Or, look at your reference at the start of these exercises to
figure this out.

2. PROGRAM EXECUTION, PRIVILEGES & SUDO

As a general user you can see that there is a file called “/etc/shadow”:

```
$ ls /etc/shadow
```

But, you cannot see its contents:

```
$ less /etc/shadow
```

What permissions does this file have? Use the examples above to figure this out. Fill
in the blanks below once you know the permissions. We've filled in one item to get you
stated:

-___R_____

As a general user, however, you can see the /etc/shadow file if you do the following:

```
$ sudo less /etc/shadow
```

What is sudo? Read about it:

```
$ man sudo
```

3. CREATE A NEW GROUP

```
$ sudo groupadd team1
```

Prove that it really exists:

```
$ grep team1 /etc/group
```

Now let's place our sysadm user in this new group:

```
$ whoami
```

Just to be sure we really are the "sysadm" user right now:

```
$ groups
```

You can see that sysadm is a member of the groups:

```
sysadm adm cdrom plugdev lpadmin sambashare admin
```

Let's add our user to the team1 group - the '-a' is important!

```
$ sudo usermod -a -G team1 sysadm
```

You won't be able to use your new group until you have logged in and out from your account, or have simulated this process by doing this:

```
$ su - sysadm

(type your own password)
```

Now try typing:

```
$ groups
```

You should see something like this:

```
sysadm adm cdrom plugdev lpadmin sambashare admin team1
```

sysadm is now a member of the team1 group.

Using groups like this can be useful for working in teams on a project, giving access to web directories, etc.

4. GIVE GROUP ACCESS TO A FILE

Do the following:

```
$ cd
$ echo “This is our group test file” > group.txt
$ chgrp team1 group.txt
```

What permissions does the file have now?

```
$ ls -l group.txt
```

You should see something like:

```
-rw-r--r-- 1 sysadm team1 28 2012-04-16 01:32 group.txt
```

How would you give members of the group team1 read/write access to this file? Before you look below try solving this on your own.

We'll use the numeric chmod functionality.

```
$ chmod 664 group.txt
```

Alternatively you could have typed:

```
$ chmod g+w group.txt
```

Look at the file's permissions:

```
$ ls -l group.txt
```

You should see something like:

```
-rw-rw-r-- 1 sysadm team1 28 2012-04-16 01:32 group.txt
```

By the way… Did you remember to just type the "g" in the filename "group.txt" and then use the tab key to save time in the exercises above? If not, try using tab in upcoming exercises. It's really worth it!

5. MAKE A FILE EXECUTABLE

Do this exercise as the sysadm user.

```
$ cd
$ touch hello
```

Now add a single line to the file that reads:

```
echo 'Hello, world!'
```

```
$ echo "echo 'Hello, world'" > hello
```

NOTE: We'll use file editors for operations like this after our next session.

Let's try to run this file:

```
$ ./hello
```

You'll probably see something like:

```
bash: ./hello: Permission denied
```

This implies that the file is not executable. We need to set the file's permission to be executable by our sysadm user. How would you do this?

```
$ chmod 755 hello
```

would work. Now try running the file:

```
$ ./hello
```

You should see ...

```
Hello, world!
```

... on your screen.

Congratulations: you've just written your first script!

Now set your hello file to be readable by everyone, NOT executable by the sysadm user, and executable by the Group and by Other. Can you figure out how to do this on your own?

Look at the file's permissions to get started:

```
$ ls -l hello
```

```
-rwxr-xr-x 1 sysadm sysadm 20 2012-04-16 01:38 hello
```

You want the permission to be:

```
-rw-r-xr-x 1 sysadm sysadm 20 2012-04-16 01:38 hello
```

There are several ways you can do this with the chmod command.

Once you have set the permissions like this, what happens if you now type?

```
$ ./hello
```

Why does this happen? If you execute the file as a different user it will still work! Does this seem odd? (Hint: think “left to right”)

You can get the file to execute, for example, by typing:

```
$ sudo ./hello
```

Now set the file back so that the sysadm can execute it. Verify that this works.

CONCLUSION

What's the “./” about?

In our example above when you typed “hello” the file “hello” is in your home directory. Your home directory is not in your default path as configured for the bash shell. Thus, bash will not find the hello file, even though it's in the same directory where you are typing the command. By using “./” before the filename we tell bash to explicitly look in the same directory for the file to execute.