# Getting Started with Linux Permissions

Mike Jager
Network Startup Resource Center
mike.jager@synack.co.nz

# The Format of a Command

```
command [options] parameters
```

"Traditionally, UNIX command-line options consist of a dash, followed by one or more lowercase letters. The GNU utilities added a double-dash, followed by a complete word or compound word."

Two very typical examples are:

```
-h
--help
```

and

```
-v
--version
```

## Goals

**Understand the following:**

- The Linux / Unix security model
- How a program is allowed to run
- Where user and group information is stored
- Details of file permissions

## Users and Groups

Linux understands Users and Groups

A user can belong to several groups

A file can belong to only one user and one group at a time

A particular user, the superuser *"root"* has extra privileges (uid = "0" in /etc/passwd)

Only root can change the ownership of a file

# Users and Groups

User information in `/etc/passwd`

Password info is in `/etc/shadow`

Group information is in `/etc/group`

`/etc/passwd` and `/etc/group` divide data fields using ":"

/etc/passwd:

```
joeuser:x:1000:1000:Joe User,,,:/home/joeuser:/bin/bash
```

/etc/group:

```
joeuser:x:1000:
```

# A Program Runs...

A program may be run by a user, when the system starts or by another process.

Before the program can execute the kernel inspects several things:

- Is the file containing the program accessible to the user or group of the process that wants to run it?
- Does the file containing the program permit execution by that user or group (or anybody)?
- In most cases, while executing, a program inherits the privileges of the user/process who started it.

## A Program In Detail

When we type:

```
ls -l /usr/bin/top
```

We'll see:

```
-rwxr-xr-x 1 root root 68524 2011-12-19 07:18 /usr/bin/top
```

What does all this mean?

# A Program In Detail

```
-r-xr-xr-x  1   root     root     68524   2011-12-19 07:18 /usr/bin/top

---------- --- -------  -------  -------- ------------    -------------
    |       |     |        |         |         |                 |
    |       |     |        |         |         |             File Name
    |       |     |        |         |         |
    |       |     |        |         |         +---  Modification Time/Date
    |       |     |        |         |
    |       |     |        |         +-------------   Size (in bytes
    |       |     |        |
    |       |     |        +-----------------------        Group
    |       |     |
    |       |     +--------------------------------        Owner
    |       |
    |       +--------------------------------------  “link count”
    |
    +----------------------------------------------   File Permissions
```

**Group**
    The name of the group that has permissions in addition to the file's owner.
**Owner**
    The name of the user who owns the file.
**File Permissions**
    The first character is the type of file. A "-" indicates a regular (ordinary) file. A "d” indicate a directory. Second set of 3 characters represent the read, write, and execution rights of the file's owner. Next 3 represent the rights of the file's group, and the final 3 represent the rights granted to everybody else.

(Example modified from **http://www.linuxcommand.org/lts0030.php**)

## Access Rights

Files are owned by a *user* and a *group* (ownership)

Files have permissions for the user, the group, and *other*

"*other*" permission is often referred to as "world"

The permissions are *Read, Write* and *Execute* (r, w, x)

The user who owns a file is always allowed to change its permissions

## Some Special Cases

When looking at the output from "`ls -l`" in the first column you might see:

```
d = directory
- = regular file
l = symbolic link
s = Unix domain socket
p = named pipe
c = character device file
b = block device file
```

# Some Special Cases

In Owner, Group and other columns you might see:

|  |  |
|---|---|
| `s = setuid` | [when in Owner column] |
| `s = setgid` | [when in Group column] |
| `t = sticky bit` | [when at end] |

## Some References

**http://www.tuxfiles.org/linuxhelp/filepermissions.html**

**http://www.cs.uregina.ca/Links/class-info/330/Linux/linux.html**

**http://www.onlamp.com/pub/a/bsd/2000/09/06/FreeBSD_Basics.html**

# File Permissions

There are two ways to set permissions when using the `chmod` command:

Symbolic mode:

*testfile* has permissions of `-r--r--r--`

```
                                  u  g  o*

$ chmod g+x testfile     ==> -r--r-xr--

$ chmod u+wx testfile    ==> -rwxr-xr--

$ chmod ug-x testfile    ==> -rw--r--r–

u=user, g=group, o=other (world)
```

# File Permissions

Absolute mode:

We use octal (base eight) values represented like this:

| Letter | Permission | Value |
|---|---|---|
| r | read | 4 |
| w | write | 2 |
| x | execute | 1 |
| - | none | 0 |

For each column, User, Group or Other you can set values from 0 to 7. Here is what each means:

| | | | |
|---|---|---|---|
| 0= **---** | 1= **--x** | 2= **-w-** | 3= **-wx** |
| 4= **r--** | 5= **r-x** | 6= **rw-** | 7= **rwx** |

## Inherited permissions

Two critical points:

1. The permissions of a directory affect whether someone can see its contents or add or remove files in it.
2. The permissions on a file determine what a user can do to the data in the file.

Example:

If you don't have write permission for a directory, then you can't delete a file in the directory. If you have write access to the file you can update the data in the file.

# Inherited Permissions

Numeric mode cont:

Example index.html file with typical permission values:

```
$ chmod 755 index.html
$ ls -l index.html
-rwxr-xr-x  1 root  wheel  0 May 24 06:20 index.html

$ chmod 644 index.html
$ ls -l index.html
-rw-r--r--  1 root  wheel  0 May 24 06:20 index.html
```

## Conclusion

To reinforce these concepts let's do some exercises.

In addition, a very nice reference on using the `chmod` command is:

*An Introduction to Unix Permissions* -- Part Two

By Dru Lavigne (note, this is for FreeBSD)

http://www.onlamp.com/pub/a/bsd/2000/09/13/FreeBSD_Basics.html