

# Permissions: who can control what

## Unix/IP Preparation Course


May 23, 2010  
Kigali, Rwanda



# Goal

---

## **Understand the following:**

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

# Users and Groups

---

Unix 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 cont.

---

User information in `/etc/passwd`

Password info is in `/etc/master.passwd`

Group information is in `/etc/group`


`/etc/passwd` and `/etc/group` divide data fields using “:”

`/etc/passwd:`

```
ausert*:1000:1000:A. User:/home/user:/usr/local/bin/bash
```

`/etc/group:`

```
users*:99:ausert
```




# 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:

```
-r-xr-xr-x 1 root wheel 46424 Nov 21 28 2009 /usr/bin/top
```

What does all this mean?





# 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 cont

---

In the 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](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>U</u>	<u>G</u>	<u>O</u> *
\$ <code>chmod g+x testfile</code>	<code>==&gt;</code>	<code>-r--r-xr--</code>		
\$ <code>chmod u+wx testfile</code>	<code>==&gt;</code>	<code>-rwxr-xr--</code>		
\$ <code>chmod ug-x testfile</code>	<code>==&gt;</code>	<code>-rw--r--r--</code>		

U=user, G=group, O=other (world)



# File permissions cont.

---

## Absolute mode:

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

<u>Letter</u>	<u>Permission</u>	<u>Value</u>
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



# File permissions cont.

---

## 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
```



# 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.



# 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

[http://www.onlamp.com/pub/a/bsd/2000/09/13/FreeBSD\\_Basics.html](http://www.onlamp.com/pub/a/bsd/2000/09/13/FreeBSD_Basics.html)

