Users, Groups, and Permissions

# Goal

**Understand the following:**

* Managing Users and Groups
* 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

To know more about the user, use: id somnog

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: ali:x:1000:1000:Ali Abdi,,,:/home/ali:/bin/bash /etc/group: ali:x:1000:

Adding a new user to the systemin

sudo adduser ahmed

To verify, run:

cat /etc/passwd

To access the system with the new user:

su – ahmed

To modify user, use usermod command. To know more about usermod:

info usermod

You can add a new comment to the user ali

sudo usermod –c “Ali Alas” ahmed

To add ali to a new supplementary group:

sudo usermod –aG somnog ali

To Lock user ali:

sudo usermod –L ali

To unlock:

sudo usermod –U ali

To remove user:

sudo userdel ahmed

Add –r to remove user and its personal files.

To add a new group:

sudo groupadd sales

Add a new group with specific GID:

sudo groupadd –g 5000 hr

To remove a group:

sudo groupdel sales

**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?

## -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 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**

**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 cont.**

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** |

**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.

**ps** Get the status of one or more processes. PPID-parent process ID ; PID-process ID

Eg: ps ax |more to see all processes including daemons

Eg : ps –ef | grep <process> **pstree** Display the tree of running processes.

**pgrep** looks through the currently running processes and lists the process IDs which matches the selection criteria to stdout. All the criteria have to match.

**top** The top program provides a dynamic realtime view of a running system. It can display system summary information as well as a list of tasks currently being managed by the Linux kernel.

**bg** Starts a suspended process in the background

**fg** Starts a suspended process in the foreground.

**kill** Ex: "kill 34" - Effect: Kill or stop the process with the process ID number 34.

**killall** Kill processes by name. Can check for and restart processes.

**pid** Find the process ID of a running program

Apt

The apt command is a powerful command-line tool, which works with Ubuntu's Advanced Packaging Tool (APT) performing such functions as installation of new software packages, upgrade of existing software packages, updating of the package list index, and even upgrading the entire Ubuntu system.

Some examples of popular uses for the apt utility:

**Install a Package**: Installation of packages using the apt tool is quite simple.

For example, to install the network scanner nmap, type the following:

sudo apt install nmap

**Remove a Package**: Removal of a package (or packages) is also straightforward. sudo apt remove nmap

**Update the Package Index**: The APT package index is essentially a database of available packages from the repositories defined in the /etc/apt/sources.list file and in the /etc/apt/sources.list.d directory.

sudo apt update

**Remove a Package**: Removal of a package (or packages) is also straightforward. sudo apt remove nmap

**Upgrade Packages**: Over time, updated versions of packages currently installed on your computer may become available from the package repositories (for example security updates).

sudo apt upgrade