Lecture- 5

Users, Groups and permissions

Users

- Every user is assigned a unique user ID number (UID)
    - UID 0 identifies root
- Users' names and UIDs are stored in /etc/passwd
- Users are assigned a home directory and a program that is when they log in (usually a shell)
- Users cannot read, write or execute each others' files without permission

Groups

- Users are assigned to groups
- Each group is assigned a unique group ID number (gid)
- GIDs are stored in /etc/group
- Each user in given their own private group
    - Can be added to other groups for additional access
- All users in a group can share files that belong to the group

Groups:

Sometimes user need to collaborate. This file can be accomplished by having users assigned to groups and setting appropriate group permission for files or directories.
Every users is a members. Gids. The group names and GIDs are stored in the */etc/group file*

User private Group Scheme:

By default a user belong to a group that is named the same as their username. That is user Digby is a member of group Digby and, by default is the only member of that group. This system can be abandoned by system administrators when they set up accounts and so this may not be the case at your location.

Primary Group:

A user's primary group is defined in the /etc/passwd file and secondary groups are defined in the /etc/group file the primary group is important because files created by the user will inherit that group affiliation. The primary group can temporarily be changed by running ***newgrp groupname***, where group name is one of the user's secondary groups. The user can return to their original group by typing **exit.**

Linux File Security:

- Every file is owned by a UID and a GID
- Every process runs as a UID and one or more GIDs
    - Usually determined by who run process
- Three access categories:
    - Processes running with the same UID as the file (user)
    - Processes running with the same GID as the file (group)
    - All other processes (other)

Permission Precedence:

- **If UID matches, user permission apply**
- **Otherwise, if GID matches, group permission apply**
- **If neither match, other permission apply**

Permission Types:

- Four symbols are used when displaying permissions
    - r: permission to read a file or list a directory contents
    - w: permission to write to a file or create and remove files from a directory
    - X: permission to execute a program or change into directory and do a long listing of the directory
    - -: no permission (in place of the *r, w, or x)*

Examining Permissions:

- File permission may be viewed using ls –l

$ ls –l /bin/login
-rwxr –xr –x l root root 19080 apr l 18:26 /bin/login

- File type and permission represented by a 10-character string

Interpreting Permissions:

-rwxr –x --- l andersen trusted 2948 oct 11 14:07 myscript

- Read, write and execute for the owner, andersen
- Read and execute for members of the *trusted* group
- No access for all others

Interpreting permission

The file shown at the top of this slide can be read, written and executed (rwx) by the user Anderson, read and executed (r – x) by member of the group trusted and has grants no permissions (---) to anyone else. Some more examples are shown below.

Given the following file permission:

```
-rwxr-xr--      1  fred    fred    26807   mar   8   22:55   penguin
-rw-r -- r --   1  mary    admin   1601    mar   5   22:36   redhat
-rw-rw-r--      1 root     staff    8671    mar   8   22:59   tuxedo
```

And the following file permission:

User	Primary group	Secondary Group
fred	fred	staff
mary	mary	staff.admin

The following will be true:

- Penguin can be read, written and executed by fred, but only red by mary;
- Redhat can be read and written by mary, but only read by fred :
- Tuxedo can be read and written by both Mary and Fred.

Changing File Ownership:

- Only root can change a file's Owner
- Only root or the owner can change a file's group
- Ownership is changed with chown:
    - Chown [-R] user_name file\directory
- Group-Ownership is changed with chgrp:
    - chgrp [-r] group_name file\directory

File ownership can be changed with the **chown** command. For example. To grant ownership of this file foofile to student, the following command could be used:

[root@statin ~] # **chown student foofile**

Chown can be used with the **–R** option to recursively change ownership of an entire directory tree. The following command would grant ownership of foodir and all files and subdirectories within it to student

[root@station ~] # **chown _r student foodir**

Only root can change the ownership of a file. Group ownership, however can be set by root or this files owner. Root can grant ownership to any group. While non-root users can grant ownership only to groups they belong to. Changing the group ownership of a file is done with **the chgrp** command. The syntax is identical to that of chown. Including the use of –r to affect entire directory trees.

Changing permissions – symbolic Method:

- To change access modes:
  - **Chmod [-R] mode file**
- Where *mode* is:
  - U.g or o for user, group and other
  - + or – for grant or deny
  - r,w or x for read, write and execute
- Examples:
  - Ugo+r: grant access to all
  - o-wx: deny write and execute to others

For example:
[root@station ~] # chmod u+r filename
[root@station ~] # chmod g+rw filename
[root@station ~] # chmod o+x filename

Changing Permission - Numeric Method:

- Uses a three-digit mode number
  - First digit specified owner's permissions
  - Second digit specified group's permissions
  - Third digit represented other's permissions
- Permissions are calculated by adding:
  - 4(for read)
  - 2 (write)
  - 1 (for execute)
- Example:
  - chmode 640 myfile

Numerical mode setting

To change all the permission on a file at once, it is often easier and quick to use the numeric method.
In this method, the first argument to the chmod command is a three digit number representing the permission that are set. In this method, the read permission has value of four, the write permission has a value of two, and the execute permission has a value of one. Add together the permission you wish to set for user, group, and others.

Examples:

Chmod 664 *file* grants read/write to the owner and groups, read-only to others.
Chmod 660 *file* grants read/write to the owner and groups; no permission for others
Chmod 600 *file*: grants read/write to the owner; no permission set for the group and others
Chmod 444 *file*: grants read only to all

With directory permission when you set the read permission. You almost always want to set execute permission.
Example:
Chmod 755 dir. Grants full permission for the owner read and execute to group and other.
Chmod 770 dir. Grants full permission to the owner and group, no permission to others
Chmod 700 dir. Grants full permission to the owner. No permission to the group or others
Chmod 555 dir. Grants read and execute permission set to all.

Changing Permissions – Nautilus:

- Nautilus can be used to set the permissions and group membership of files and directories.
    - In a Nautilus window, right click on a file
    - Select properties from the context menu
    - Select the **permission** tab