Linux 103

Workshop

What’s In It For Me

(WIIFM)

- Linux is a basic requirement
- Our application is running on Linux

Linux 101

Recap

- Filesystem
- Path
- Shortcut characters
- High Level Structure
- Command Line
- I/O Redirections

Commands

Recap

- pwd
- ls
- cd
- mkdir
- touch
- echo
- cat
- mv
- rm
- cp
- man

Linux 102

- Files
- File Types
- Text Files
- File Globbing
- File Links
- Finding Files
- Pipeline
- Here Document
- File Permissions

Commands

Recap

- which
- file
- stat
- clear
- ln
- truncate
- find
- wc

Agenda

- Users
- Groups
- Root user
- Standard Linux Permissions
- Permission modes
- Permission methods

Users

- Multiple Users system
- Attributes:
  - user name (case sensitive)
  - User ID (UID) - numeric
  - password (to login)
- Can belong to multiple groups
- One and only one of the groups is the primary group
- Config file: /etc/passwd

Groups

- Multiple Groups system
- Attributes:
  - group name (case sensitive)
  - Group ID (GID) - numeric
- Multiple users can belong to a group
- A group can not belong to a group
- Config file: /etc/group

Command - id

Display user and group information

```
vagrant@ubuntu-xenial:~$ id
vagrant@ubuntu-xenial:~$ id ubuntu
```

root user

- /root - home directory
- root - primary group
- Administrative privileges

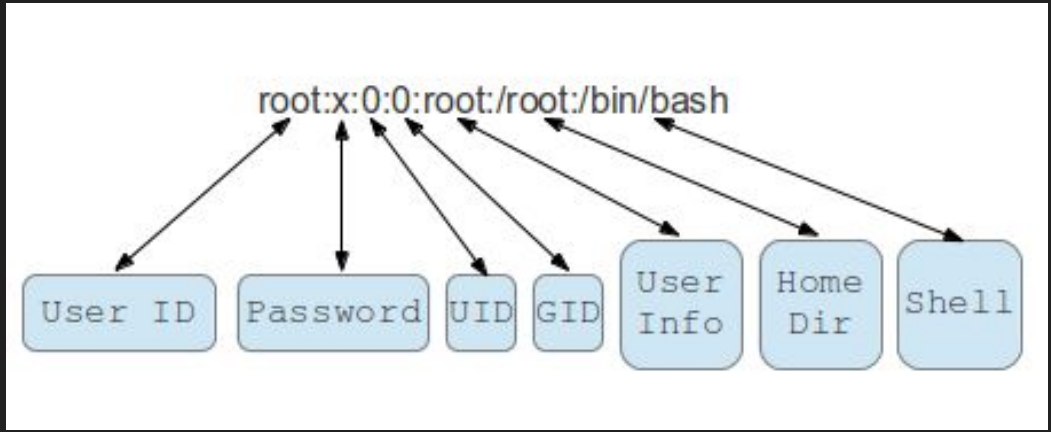
Command - less

Display file content one page at a time

```
vagrant@ubuntu-xenial:~$ less /etc/passwd
```

/etc/passwd

- One line for each user
- root user, usually, in the first line
- 7 delimited columns (:)

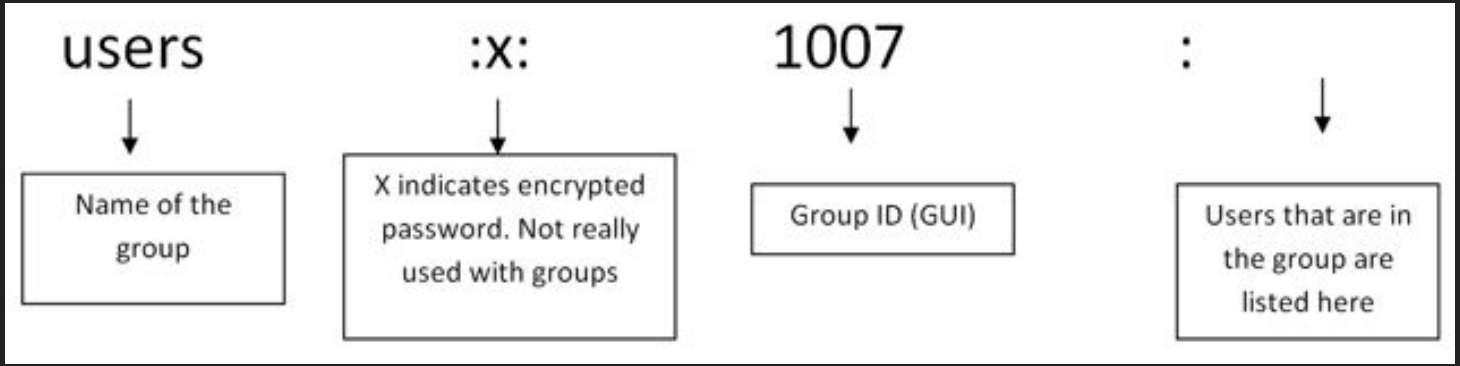


Command - sudo

Super user do

```
vagrant@ubuntu-xenial:~$ less /etc/shadow
vagrant@ubuntu-xenial:~$ sudo less /etc/shadow
```

/etc/group


users
:x:
1007
:
Name of the group
X indicates encrypted password. Not really used with groups
Group ID (GUI)
Users that are in the group are listed here

Standard Linux Permissions

- Users can belong to multiple groups
- Files belong to one user owner
- Files belong to one group owner
- Permissions can be set for the user, group, or other
- Users can read, write or execute files
- User can list, create new files and traverse directories
- Permissions support privilege elevation
- Permissions support group owner inheritance
- Support default file permissions

Shortfalls of Standard Linux Permissions

- Files and directories can only belong to one user
- Files and directories can only belong to one group
- Permissions set for others are not concise
- Inheritance only supports group ownership and not permissions
- There is no easy way to backup and restore permissions
- There is no easy way to temporarily restrict permissions

Permission modes

- read
- write
- execute

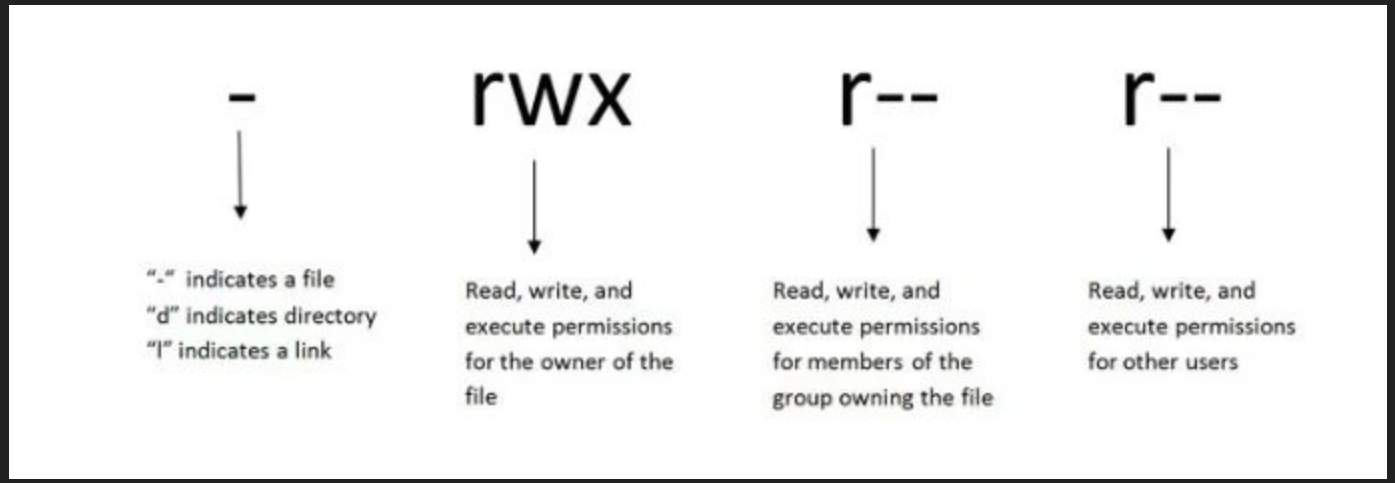
File Permissions

- read - read contents of file
- write - write/modify contents of file
- execute - load file into memory and run on the CPU

Directory Permissions

- read - read the metadata of files in the directory
- write - create files in the directory
- execute - enter or traverse the directory

File Permissions



Exercise

- Who is the owner (user and group) of the file /etc/passwd?
- Create a file in your home directory. Examine its ownership and file permissions.

Command: chown

Change Owner

Syntax: chown [options] <user>:<group> <file>

Common options:

-R - recursive

```
vagrant@ubuntu-xenial:~$ touch file1.txt file2.txt file3.txt
vagrant@ubuntu-xenial:~$ ls -l file*.txt
vagrant@ubuntu-xenial:~$ sudo chown ubuntu:ubuntu file1.txt
vagrant@ubuntu-xenial:~$ sudo chown ubuntu file2.txt
vagrant@ubuntu-xenial:~$ sudo chown :ubuntu file3.txt
vagrant@ubuntu-xenial:~$ ls -l file*.txt
```

Exercise

1. List file permissions of the following files: /etc/passwd, /etc/shadow
2. Why reading /etc/shadow requires using sudo and /etc/passwd not?

Permissions Methods

- Numeric
- Symbolic

Numeric

d	r read	w write	x exec	r read	- write	x exec	r read	- write	- exec
File type	Owner permissions			Group permissions			User permissions		
(directory)	4	2	1	4	2	1	4	2	1
	7			5			4		

Command: chmod

Change File Permissions

Syntax: chmod [options] <permissions> <file>

```
vagrant@ubuntu-xenial:~$ mkdir perm;cd perm
vagrant@ubuntu-xenial:~$ touch file.txt
vagrant@ubuntu-xenial:~$ ls -l file.txt
vagrant@ubuntu-xenial:~$ chmod 750 file.txt
vagrant@ubuntu-xenial:~$ ls -l
```

Exercise

- What is the numerical representation of the file1.txt file
- What is the numerical representation of full permissions?
- What is the numerical representation of “-rwxr-x---” ?
- Change the file permission using the numerical value discovered of file file1.txt

Symbolic

```
vagrant@ubuntu-xenial:~$ ls -l file.txt
vagrant@ubuntu-xenial:~$ chmod u=rwx,g=rx,o= file.txt
vagrant@ubuntu-xenial:~$ ls -l
vagrant@ubuntu-xenial:~$ chmod g+w file.txt
vagrant@ubuntu-xenial:~$ ls -l
vagrant@ubuntu-xenial:~$ chmod g-w file.txt
```

Symbol	Meaning
u	Short for “user” but means the file or directory owner.
g	Group owner.
o	Short for “others,” but means world.
a	Short for “all.” The combination of “u”, “g”, and “o”.

Notation	Meaning
u+x	Add execute permission for the owner.
u-x	Remove execute permission from the owner.
+x	Add execute permission for the owner, group, and world. Equivalent to a+x.
o-rw	Remove the read and write permission from anyone besides the owner and group owner.
go=rw	Set the group owner and anyone besides the owner to have read and write permission. If either the group owner or world previously had execute permissions, they are removed.
u+x,go=rx	Add execute permission for the owner and set the permissions for the group and others to read and execute. Multiple specifications may be separated by commas.

Numeric vs. Symbolic

Numeric applies the full set of permissions. One uses the desired permissions to apply.

Symbolic can apply partial permissions. For example: removing the execute bit.