System Administration & Security

COMP 175 | Fall 2021 | University of the Pacific | Jeff Shafer

Linux Fundamental Skills: File Permissions, Searching, and Redirection

Overview

Recap

- AWS
  - EC2 instances
  - Security Groups
  - VPC networks
  - Billing & alerts
- Linux Fundamentals
  - SSH
  - Directories & Navigation
  - User Accounts
  - Password Principles

This Week

- Lecture
  - File Permissions
  - File Searching
  - I/O Redirection
- Lecture
  - HTTPS
  - DNS
- **Lab 6 – Web Server (Part 3)**

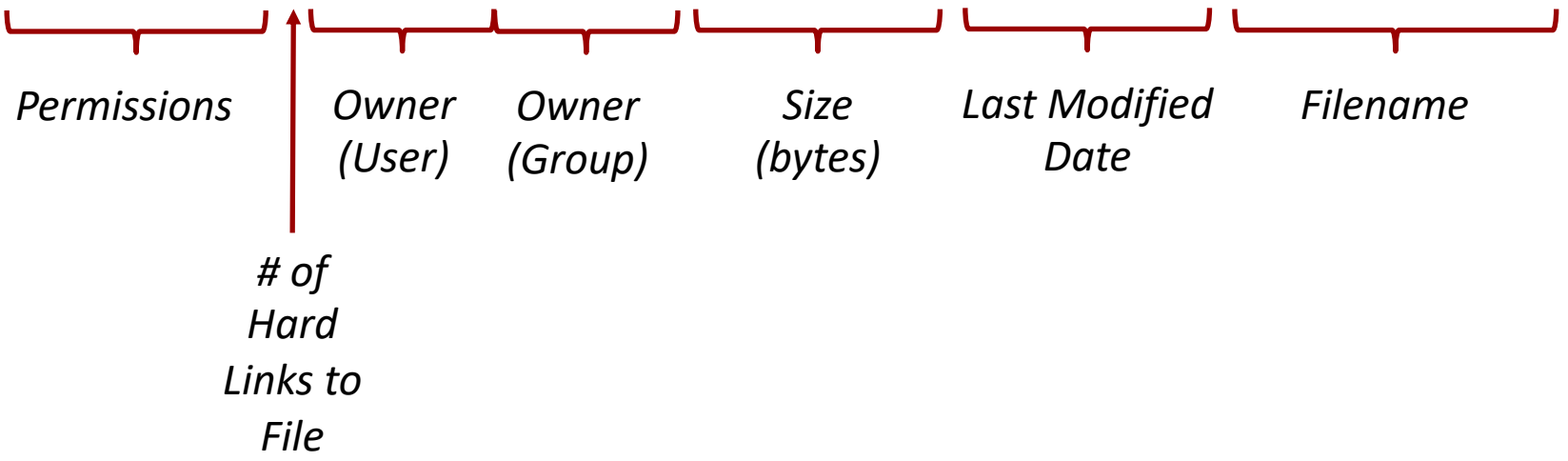
File Ownership & Permissions

File Ownership

Example directory listing

```
$ ls -l     # -l = Long listing
```

```
-rw------- 1 steve  staff    4478979  6 Feb  2011 private.txt
-rw-rw-r-- 1 katie  staff    4478979  6 Feb  2011 press.txt
-rwxr-xr-x 1 eliza  staff    4478979  6 Feb  2011 upload.bin
```



File Ownership

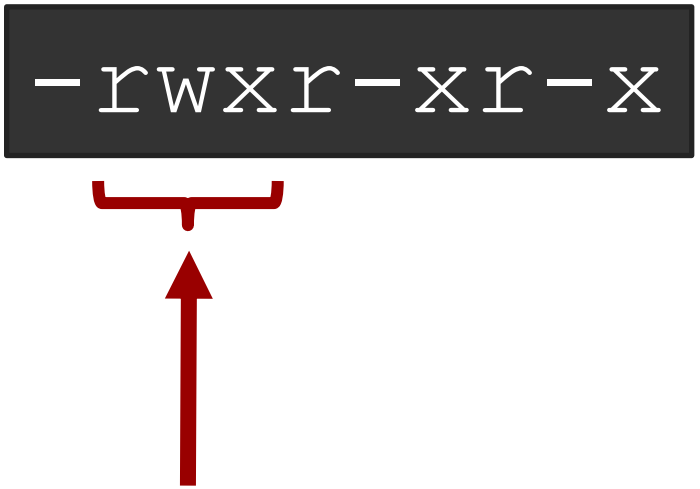
- A file is owned by both a user and a group
  - A group might be just the single user, or the group might be multiple users
  - Groups are defined in `/etc/groups`

File Permissions

```
-rwxr-xr-x
```

- First Character: File Type
  - `-` - **Regular file**
  - `b` - Block special file
  - `c` - Character special file
  - `d` - **Directory**
  - `l` - **Symbolic link**
  - `n` - Network file
  - `p` - FIFO
  - `s` - Socket

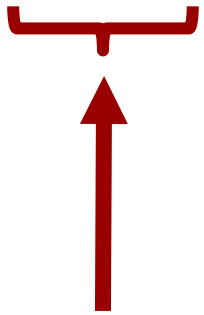
File Permissions



- Next 3 Characters:
  - Permissions for the **user**

File Permissions

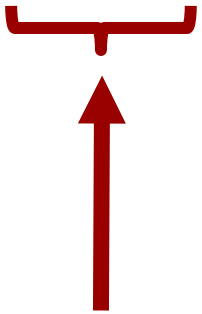
```
-rwxr-xr-x
```



- Next 3 Characters:
  - Permissions for the **group**

File Permissions

```
-rwxr-xr-x
```



- Next 3 Characters:
  - Permissions for **other**
  - *Any other users not part of the group*

- All three sections (user, group, other) answer the same question
  - Read – Can I access the file contents?
  - Write – Can I modify the file contents?
  - Execute – Can I run the file contents? (binary or shell script)

Permission Notation

Symbolic

--- no permission
--x execute
-w- write
-wx write & execute
r-- read
r-x read & execute
rw- read & write
rwx read write & execute

Numeric

0 --- no permission
1 --x execute
2 -w- write
3 -wx write & execute
4 r-- read
5 r-x read & execute
6 rw- read & write
7 rwx read write & execute

File Permissions

```
-rwxr-xr-x eliza staff...upload.bin
```

- Regular file
- User “eliza”
  - Can read, write, and execute the file `upload.bin`
- Group “staff”
  - Can read and execute the file `upload.bin`
- Others
  - Can read and execute the file `upload.bin`

Changing File Permissions

- "Change Mode" – `chmod`
  - Syntax: user(u) /group(g) /other(o)/all(a) +/- permission to modify (r)(w)(x)

Add **W**rite permissions to the group for file "test.txt"

```
$ chmod g+w test.txt
```

Remove e**X**ecute permissions from other users for file "myProgram"

```
$ chmod o-x myProgram
```

Changing File Permissions

Make “newfile” permissions match those of “oldfile”

```
$ chmod --reference=oldfile newfile
```

Recursively add read/write group permissions to all files in a directory

```
$ chmod -R g+rw directory1/
```

Changing File Permissions

- "Change Mode" – `chmod`
  - Syntax: user(u) /group(g) /other(o)/all(a) +/- permission to modify (r)(w)(x)

Add **R**ead permissions to the group and other users for file "test2.txt"

```
$ chmod g+r,o+r test2.txt
```

Add e**X**ecute permission to all users for file "test3.bin"

```
$ chmod a+x test3.bin
```

Changing File Permissions

- Numeric value passed to `chmod` (if used) is always in **octal** form (digit-by-digit)

```
777 (octal)
== Binary:       0b 111 111 111
== Permissions:  rwxrwxrwx
```

Setuid

- **Setuid** allows a user to execute that file or program with the permission of the owner of that file
  - Used to *elevate* privileges of the current user
  - **Trusted apps only!**

```
$ ls -ls /usr/bin/sudo
148 -rwsr-xr-x 1 root root  ...    /usr/bin/sudo
```


s = Setuid bit

- Sudo is invoked as *your* user. But the binary launches and runs as the *root* user
  - First thing it does is check to see if you are permitted to use it!

Searching for Files

Utility: Locate

- **`locate`** will search for files matching a specific name or pattern (regular expression)
- Fast! Because it searches a database that is updated daily *(and may be out of date)*
  - Script is in `/etc/cron.daily/mlocate`

```
$ locate error.log
/var/log/nginx/error.log
/var/log/nginx/error.log.1
/var/log/nginx/error.log.10.gz

$ sudo updatedb     # Force database update
```

Utility: Find

- **`find`** will search for files matching a specific name, size, owner, age, … (*myriad options!*)
- Slow! Search the filesystem on demand, no cache/index available
- Runs as *your user* – Can only see files you have access to

```
$ find /var -name error*
# Search in /var where name starts with "error"

$ find /home -iname *lab*
# Search in /home for filenames containing "lab"
# Case insensitive
```

Utility: Find

- **`find`** will search for files matching a specific name, size, owner, age, … (*myriad options!*)

```
$ find /home -type f -perm /a=x
# Search /home for all files that are executable

$ find / -type f -perm 0777 -print -exec chmod 644 {} \;
# Find Files with 777 Permissions and chmod to 644
```

https://www.tecmint.com/35-practical-examples-of-linux-find-command/

Utility: Grep

- **grep** will search the contents of (text) files, including recursive searching of entire directories

```
$ grep -i "error" /var/log/syslog
# Case insensitive search for "error" in syslog

$ grep -R -i "permitrootlogin" /etc
# Case insensitive recursive search for string
# anywhere in the /etc directory tree
```

Utility: Which

- **`which`** will tell you the location of a binary that runs for a given command
  - Helpful if you have multiple versions installed!
    - One installed by the package manager
    - One installed yourself from source
  - Helpful if you can run the program as one user but not another
    - `$PATH` issues?

```
$ which nano
/bin/nano
```

I/O Redirection

I/O on Unix

- Three data streams that should be familiar to programmers
  - Standard Input (stdin) – Stream **0**
    - Keyboard
  - Standard Output (stdout) – Stream **1**
    - Terminal
  - Standard Error (stderr) – Stream **2**
    - Terminal

I/O Redirection

Overwrite File

- **<**
  - Take *standard input* from file
- **>**
  - Save *standard output* to file
- **2>**
  - Save *standard error* to file

Append to File

- **>>**
  - Append *standard output* to file
- **2>>**
  - Append *standard error* to file

I/O Redirection

Save process list to a file (via output redirection)

```
$ ps aux > running_processes.txt
```

Append the process list to a file (via output redirection)

```
$ ps aux >> running_processes.txt
```

Can combine input and output redirection

```
$ command < input-file > output-file
```

Can save multiple streams (stdout, stderr) into one file

```
$ command > output-file 2>&1
```

Special location for worthless data to be discarded (/dev/null)

```
$ command > /dev/null
```

Pipes

Send output from one program directly to another
(bypassing console and bypassing a file)

```
$ command1 | command2
```

Get a long-form directory listing, but display it one page at a time

```
$ ls -l | less
```

Project 1

Project 1

- Labs *to date* have been scripted by instructor
  - One correct path from start to finish
  - We all enter (basically) the same commands
  - We all get the same result
- **Time to Level Up!**
- *Just like in programming, system administration has a fair amount of "Keep Googling until you find the right ~~code~~ command"*

Project 1

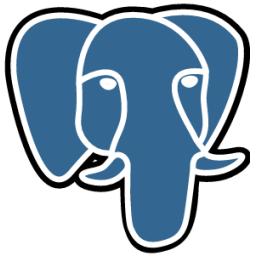
- Project 1
  - You pick a “server application”
  - You install it
  - You configure it
  - You secure it
  - You demonstrate that it works
  - You document your process
  - You present it to your peers

Example Applications

Memcached

MongoDB

MySQL



PostGreSQL

WordPress
(Web CMS)

Joomla
(Web CMS)

Drupal
(Web CMS)

NextCloud
(self-hosted
Dropbox)

Seafile
(self-hosted
Dropbox)

The sky's the limit!

Project 1 – Proposals

- Submit a **project proposal** (no longer than 1 page) that contains the following information:
  - What **server application** are you proposing to use?
  - What does this application **do**?
  - What references do you have that explain **how to install it** on a server?
  - How will you **demonstrate** that you have successfully installed it?
  - What references do you have that explain **how to properly secure** this application?
  - How will you **demonstrate** that you have successfully secured it?
  - What will your project **cost** in terms of AWS credits? What is the basis of this cost estimate?

Project 1 – Proposals

- **The instructor reserves the right to reject project proposals for applications that are "too easy" or are "too similar" to those from other students**
- Examples
  - No XAMPP "server in a box" scripts that auto-install everything with a single command
  - No quick-start AWS wizards that auto-deploy everything
  - No managed AWS services (e.g. RDS)

Project 1 – Deliverables

- Proposal
  - Due Oct 5th
- Installation Report
  - Due Oct 19th
- Presentation Video (~10 minutes)
  - Due Oct 26th
- Peer Reviews of Videos (3)
  - Due Nov 2nd

Wrap-Up

- Questions?
- Concerns?

- **This Week**
  - **Lab 5** – Web Server (Part 2)
  - **Lab 6** – Web Server (Part 3)
- **Upcoming**
  - **Project 1 Proposal** (Oct 5th)