File Permissions

Linux System Administration
Fall 2022

UNIX File Modes

- A unix file’s mode is a collection of numbers that represents permissions.
  - Fundamental controls user’s access to system.
  - r - Read controls data that can be read
  - w - Write controls what data can be written
  - x - Execute controls what commands can be run

Unix file modes

Permission	Symbolic	Binary	Octal
everything	rwx	111	7
read and write	rw-	110	6
read and execute	r-x	101	5
read only	r--	100	4
write and execute	-wx	011	3
write only	-w-	010	2
execute only	--x	001	1
none	---	000	0

Access Modes

- File
  - **r** can read file
  - **w** can modify file
  - **x** can execute file
    - run file as command
- Directories
  - r can see file names
  - w can add or remove files
  - x allow access to files in the directory
    - access to inodes

File mode

- Special:User:Group:Other
  - Special - SetUID,SetGID,Sticky
  - User - File owner access
  - Group - Group access
  - Other - Access by all others

Special Bits

- SetUID
  - Run executable with UID of owner
- SetGID
  - Executable: Run with GID of owner
  - Directory: New files inherit directory group
- Stricky bit
  - Legacy: Keep executable in swap on exit
  - Directory: Only root or owner can delete or rename files
  - Files: Usually ignored

r and x

- Binary executables only need x
  - Executed directly by the kernel
- Shell scripts need r and x
  - Interpreter needs to read the file
- Directories
  - r required to list files
  - x required to access files

File mode examples

```
-rw-------  1 mscott mscott /home/mscott/.ssh/id_rsa
-rw-r--r--  1 root   root   /etc/issue
-rwxr-xr-x  1 root   root   /usr/bin/zip
-rwsr-xr-x  1 root   root   /usr/bin/passwd
-rwxr-sr-x  1 root   tty    /usr/bin/wall
drwxrws---  1 root   sales  /home/sales
drwxrwxrwt 25 root   root   /tmp
```

Manipulating owner & permissions

- **ls -l** show permissions
- **stat** show detailed file permissions
- **chmod** sets permissions (-R for recursive)
- **chown** change user[:group] (-R for recursive)
- **chgrp** change group

chmod symbolic

- [ugoa] [+-=] [rwxst]
  - **u**ser, **g**roup, **o**ther, **a**ll
  - **+** add, **-** remove, **=** set
  - **r**ead, **w**rite, e**x**ecute, **s**et*id, s**t**icky
- Examples
  - u+w add write permission for user
  - a-x remove execute permission for all
  - ug=rw,o=r set user & group to rw, others to r

chmod numeric

644	rw-r--r--	u=rw,go=r
755	rwxr-xr-x	u=rwx,go=rx
0755	rwxr-xr-x	u=rwx,go=rx
4755	rwsr-xr-x	u=rwxs,go=rx
2775	rwxrwsr-x	u=rwx,g=rwxs,o=rx
1777	rwxrwxrwt	ug=rwx,o=rwxt

umask

- Sets file creation mode mask
  - In bash this is a shell builtin command
  - Set on login
- umask sets permissions to mask
  - If the bit is set in umask, it is 0 in the file mode
  - Result is `perm & ~umask`
- umask is often 0002 (o-w) or 0022 (go-w)

Access Control Lists (ACL)

- Set permissions for individual users
  - `setfacl –m mscott:rwx /home/accounting`
- Show ACL
  - `getfacl /home/accounting`
  - A trailing + in ls -l indicates presence of ACL
- Not all filesystems support ACLs
  - ACL support can be set by mount

setcap/getcap

- **capabilities** are privileges normally reserved for root that can be selectively granted to executables
  - `setcap cap_net_raw=ep` *`command`*
    command can access raw sockets without being root
  - `getcap command`
    show capabilities associated with command
- More selective than `setuid`

• Example capabilities

`man capabilities` show all capabilities

- `CAP_CHOWN`
  - Change file ownershop
- `CAP_KILL`
  - Send any signal
- `CAP_NET_RAW`
  - Access to raw sockets
- `CAP_SYS_RAWIO`
  - Acces to hardware
- `CAP_SYS_TIME`
  - Access to real time clock
- `CAP_SYSLOG`
  - Access to logging