17. Altering Permissions

commands table

command	application

01. What actually matters in this section

- `important`: chmod command basics, sudo command
- `useful`: chown command
- `nice to have`: chmod octal notation, sudo command, working with groups

02. The chmod command

- `chmod` is used to change permissions of a file or directory
- To use the command, we need to tell it:
    1. who we are changing permissions for
    2. what change are we making? Adding? Removing?

3. Which permissions are we sitting?

- syntax: `chmod mode file`
- When specifying permissions with `chmod`, we use a special syntax to write permission statements.

  first, we need to specify the “who” using the following values:

  1. `u`: user or owner of the file
  2. `g`: group
  3. `o`: others
  4. `a`: all of the above
- we use `+` to add the permission, `-` to remove the permission, and `=` to set permission and remove others
- example:
  1. `chmod g+w file.txt` will enable write permissions to the file
  2. `chmod a-w file.txt` will remove write permissions for all
  3. `chmod u+x file.txt` will add executable permissions for the owner
  4. `chmod a=r file.txt` will set permissions to read-only for all and remove other permissions.
  5. `chmod u+wx file.txt` will set permissions to write and execute for users or owners

03. Using Octal Notation with Chmod

- `chmod` supports octal way to represent the permission patterns.

permission patterns and their respective octal notation

octal	binary	file mode
0	000	- - -
1	001	- - x
2	010	- w -
3	011	- w x
4	100	r - -
5	101	r - x
6	110	r w -
7	111	r w x

- example:
    1. `chmod 755 file.txt`: will set permissions to set `rwx` for users or owners, `rx` for groups, and `rx` for others.
    2. `chmod 700 file.txt`: will set permissions to set `rwx` for users or owners, `---` for groups, `---` for others
    3. `chmod 644 file.txt`: will set permissions to set `rw-` for users or owners, `r--` for groups, `r--` for others

04. The su command - substitute user

- It stands for substitute user
- `su` command can be used to start shell as another user.
- to leave the session, type `exit`
- example:
    1. `su - <username>`: to login as passed username

05. The super special Root directory

- super user can run all commands and access any file on the machine, regardless of the file's actual owner.
- Root user has tons of power and could easily damage or even destroy the system by running the wrong commands
- Root user is locked by default on ubuntu
- all files we can see owner as `root` is super user.

06. Using the Sudo Command

- even if the root user is locked by default, we can still run specific commands as the root user by using the `sudo` command
- `sudo -l` will show the list of commands permitted for particular user.
- only administrator user have privellage to execute as super user

07. Change ownership with chown

- `chown` command is used to change the owner and /or owner of the specific file or directory
- syntax: `chown USER[:GROUP] FILE(s)`
- example:
    1. `chown bojack file.txt`: to make `bojack` owner of the `file.txt`
    2. `chown :groupname file.txt`: to make `groupname` *(which is a group)* group owner of the `file.txt`
    3. `chown bojack:groupname file.txt`: to make `bojack` and `groupname` owner and group owner respectively of `file.txt`
- only super user can execute the command to change the ownership of the files

08. Working with groups demo

- we can create a group using `addgroup <groupname>`
- example:
    1. `addgroup moviegroup`: will create a group called `moviegroup`
- one has to have sudo previlage to run the command