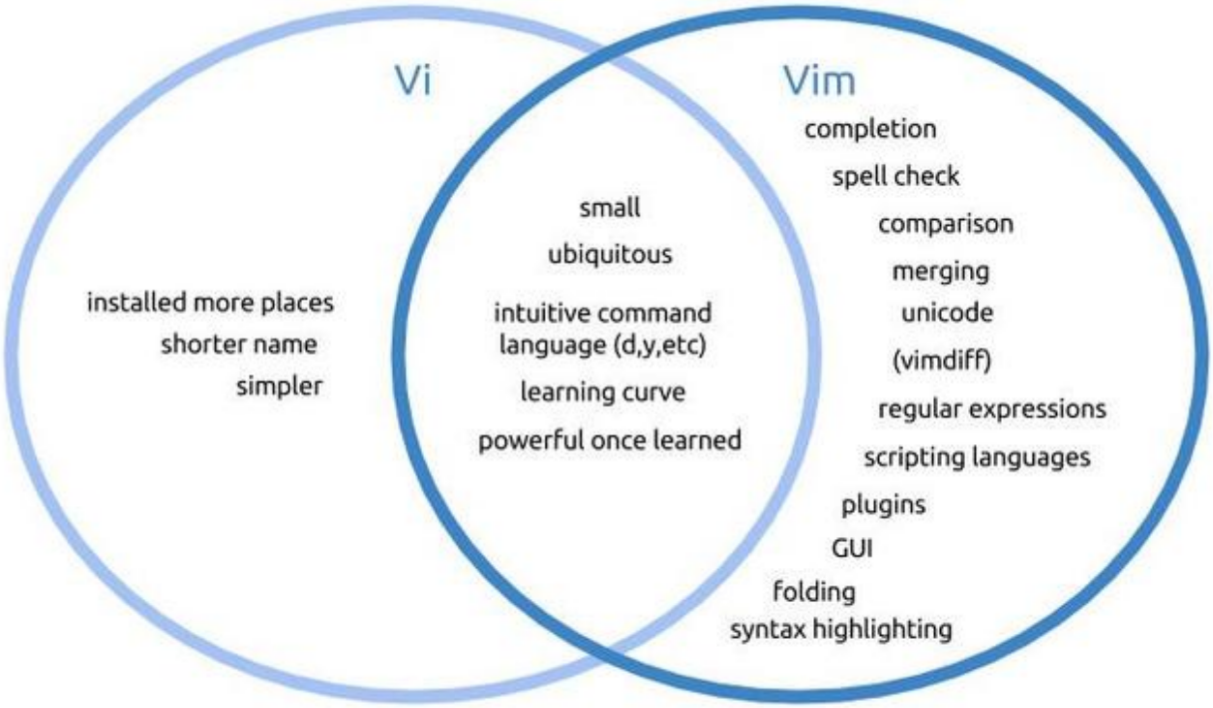
Vim

What is WIM

- The **vi command-line** text editor is included in all POSIX compliant OS.
- The **vi command** is now linked to the **vim command**.
- Even when you issue the **vi command**, you're actually starting the **vim editor**.
    - To install vim
        - ```
          sudo apt install vim
          ```
    - If **vi** is installed in all Linux distros, Why am i learning **VIM**?
        - vim had more features
        - vim is also light weight



Vim modes:

- **Insert mode:** used for writing text
- **Normal mode:** used for manipulating text
- **Command mode:** used for entering vim commands
- **Ex-mode:** Similar to the command-line mode but optimized for batch processing.
    - vim starts in normal mode
    - From normal mode **press i** to enter insert mode. The world **--INSERT--** will appear.
    - To switch back to normal mode **press esc**.
    - In the lack of the **esc key press ctrl + c**.

Insert text:
          ```

- you can create a file and open **vim** at the same time by typing **vim** and a file name.
    - Example:
        - ```
          vim notes.txt
          ```
    - In insert mode, you can use:
        - **The arrow keys** to move around,
        - **Enter Key** to continue in the next line,
        - **Backspace** for deleting.

Saving and quitting vim

- To save a text file you need to enter **normal mode** using: and the use the **w** key.
    - **:w** will save the file
    - **:w new.txt** will save the file as new.txt
    - **:wq** will save the file and quit
    - **:wqa!** will save the file and close all files open in the buffer

Editing a file with wim

- You can tell vim that you want to edit another file by using the e command
- `:e new.txt` -> will open `new.txt` and allow you to edit
- You can use auto completion here
- Ctrl + g will show the file that you are currently editing in the status line
- You can also use :f in command mode to see the file that you are currently working on.

Navigating a file

- In normal mode use the keys:
    - H = left
    - J = down
    - K = up
    - L = right
- You can prefix the number of times by adding the number after the letter 10H will move 10 character to the left.

Moving around Words, sentences, and paragraphs
          ```

- To move between words use w e
  - `w -> moves word by word to the beginning of each word`
  - `e -> moves word by word to the end of each word`
- You can prefix the number of words you want to move
- 10e will move 10 words
- To move between sentences use ()
  - (-> previous sentences
  -) -> next sentence
- To move between paragraphs use {}
  - { -> previous paragraph
  - } -> next paragraph
- *A paragraph beginnings after an empty line and ends in an empty line.*

Searching words in wim

- Use / and the word you are looking for to search forward
  - `/hello`
- letter n will repeat the search for the next word
- ? To search backward
  - `?hello`
- * will search for the next occurrence of the word under the cursor
- # will search backward for the previous occurrence of the word under the cursor

Screen movement

- G **uppercase g** Moves to end of the file
- gg **2 lowercase g** moves to the beginning of a file
- **ctrl + f** moves a page forward at a time
- **ctrl + b** moves a page backward at a time

Moving to Lines

- To move to a specific line use : plus the line number
  - :8 will move you to line 8
  - Additionally use 8G
- $ will move to the end of the line
- 0 will move to the beginning of the line
- `vim sample.txt +100` will open sample.txt and move to line 100
- + executes any vim command from the shell prompt

Delete text and copy and paste

- dw = delete current word
- u = undo
- dd = delete line under the cursor
- d + /word = delete until the word given
- yw = copy the current word
- p = for paste after the cursor
- P = for paste before the cursor
- yy = copies a whole line
- x = for cut

Useful to Know

- Read files
  - Shift o enters a new empty line
  - :r file name = insert the text of the file given into the file being edited
- Create a vim custom file
  - In your home directory create a file named .vimrc and add the commands to that file
  - http://learnvimscriptthehardway.stevelosh.com/
- Run an external command
  - :!+command
- To run a command and paste in a file
  - :r !+command

Managing Data

Basic Terminology

- **Backup:** Copies files and directories to an archive
- **System Backup**
- **Archive**
- List of important directories to include in system backup:
  - /etc
  - /home
  - /opt
  - /root
  - /var

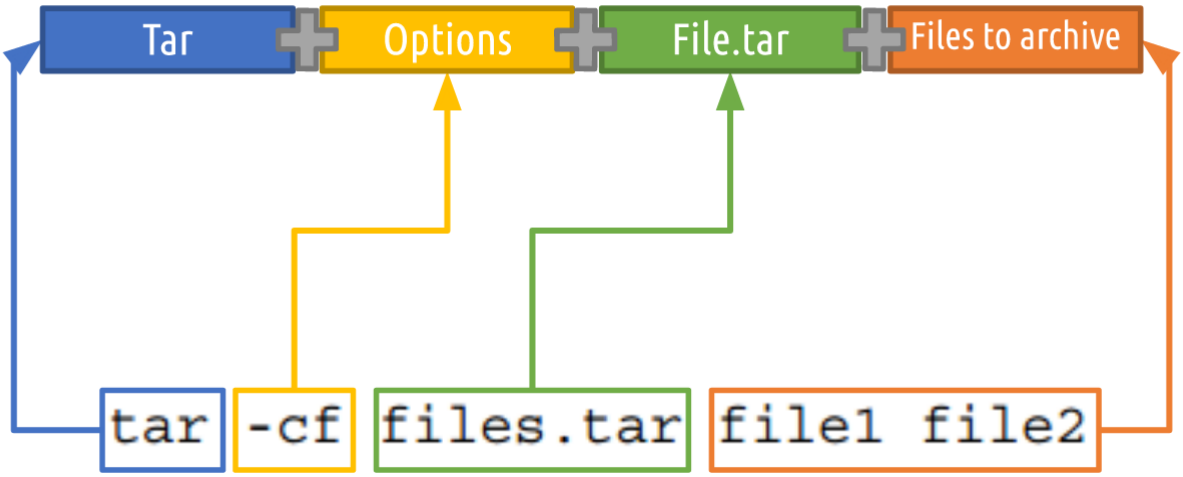
Archiving utilities

- **Tar (tape archive):** create archives by combining files and directories into a single file.
- **CPIO:** Create an archive, restores files from an archive, copies a directory hierarchy.
  - Create (copy-out) mode places multiple files into a single archive file
  - Extract (copy-in) mode restores files from an archive
  - Pass-through (copy-pass) mode copies a directory hierarchy.
- **Ar:** creates, modifies, and extracts form archives.

The tar program

- To create an archive
  - > tar + option + archive name + files to add to archive
    - The **option-f** is always required.
    - Files inside an archive are called members.
- To extract an archive:
  - > tar + options + file to extract

Tar command example explained



The CPIO program

The CPIO program

Cpio requires a list of files to archive. The option to create an archive is -o

- `ls | cpio -ov > archive.cpio`

To extract an a archive to cpio use the -i option with <

- `cpio -iv < archive.cpio`

Archive specific files

- `find . -iname *.sh | cpio -ov > scriptsArchive.cpio`

Create a tar archive with cpio

- `ls | cpio -ov -H tar -F sample.tar`

Extract *.tar Archive File using cpio

- `cpio -idv -F sample.tar`

View the content of *.tar Archive File

- `cpio -it -F sample.tar`

The ar utility

The GNU **ar** program creates, modifies, and extracts from archives.

- Archive files with **ar**
  - ```
    ar r test.a *.txt
    ```
- List content of an archive
  - ```
    ar t test.a
    ```

- Add a new member to an archive
    - ```
      ar r test.a test3.txt
      ```
- Delete a member from archive
    - ```
      ar d test.a test3.txt
      ```

File Compression

- Gzip (GNU Zip) has better compression ratios than compress does and can uncompress files that were compressed using the compress command.
- The **`gzip`**, **`bzip2`**, and **`xz`** commands are used for compression.
- When you compress a file with any of these tools the result is a file with a similar name but with the correspondent file extension.
- **Example:**
    - `file.txt ----> file.txt.gz`
    - `file.txt ----> file.txt.bz2`
    - `file.txt ----> file.txt.xz`

File Compression | GZIP, BZIP2, XZ

- Gzip, bzip2, and xz compress files in place meaning the original file is deleted after compression.
- **`bzip2`** offers better compression ratios in comparison to gzip.
- **`xz`** produces better compression ratios than gzip and bzip2.

Important:

Do not confuse gzip with zip. Zip is used to pack and unpack zip archives containing several files compressed into a single file that has been imported from or is being exported to a Windows system.

File Compression | zip, 7zip, and rar

- Zip is an archiving and compression utility.
- To use zip: **`zip + archiveName.zip + files to include in archive`**
- Example: **`zip allmyfiles.zip file1 file2 file3`**
- To unarchive use: **`unzip archive.zip`**

- 7-Zip is an open source, cross-platform, and fully-featured file archiver with a high compression ratio.
- To use 7zip on linux you need the package: **p7zip-full**
- The general formula to use 7z is: **`7z + option + fileName.7z + file(s) to archive`**
- See next slide for examples

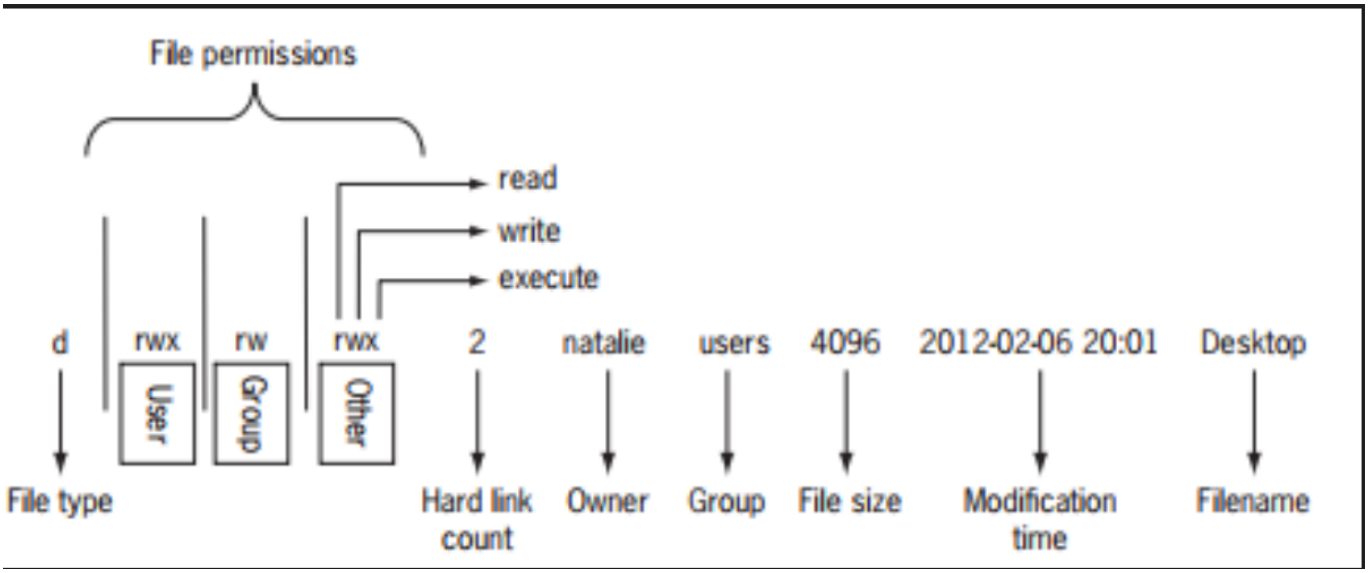
- RAR is a proprietary archive file format developed by Eugene Roshal. The command unrar allows Linux users to extract rar archives. The command rar allows you to create rar archives
- To use unrar: **`unrar + option + filename.rar`**
  - Example: **`unrar x games.rar`**
- To use rar: **`rar + option + archivename.rar + files to archive`**
  - Example: `rar a archivename.rar file2 file2`

Linux File Permissions

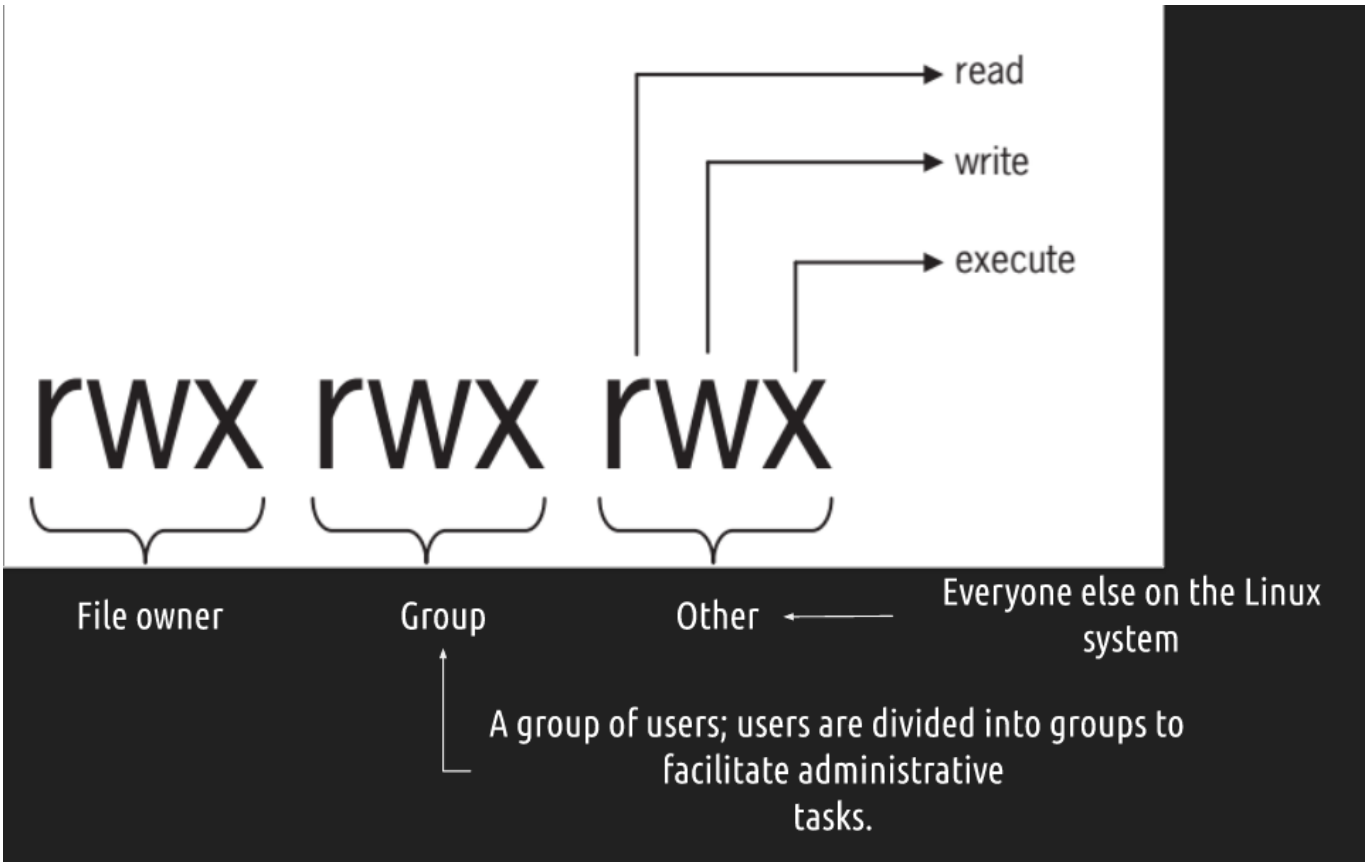
Linux File Permissions | File Ownership

- A file can be owned only by one user and one group.
- The **/etc/passwd** file contains a list of all the users in Linux.

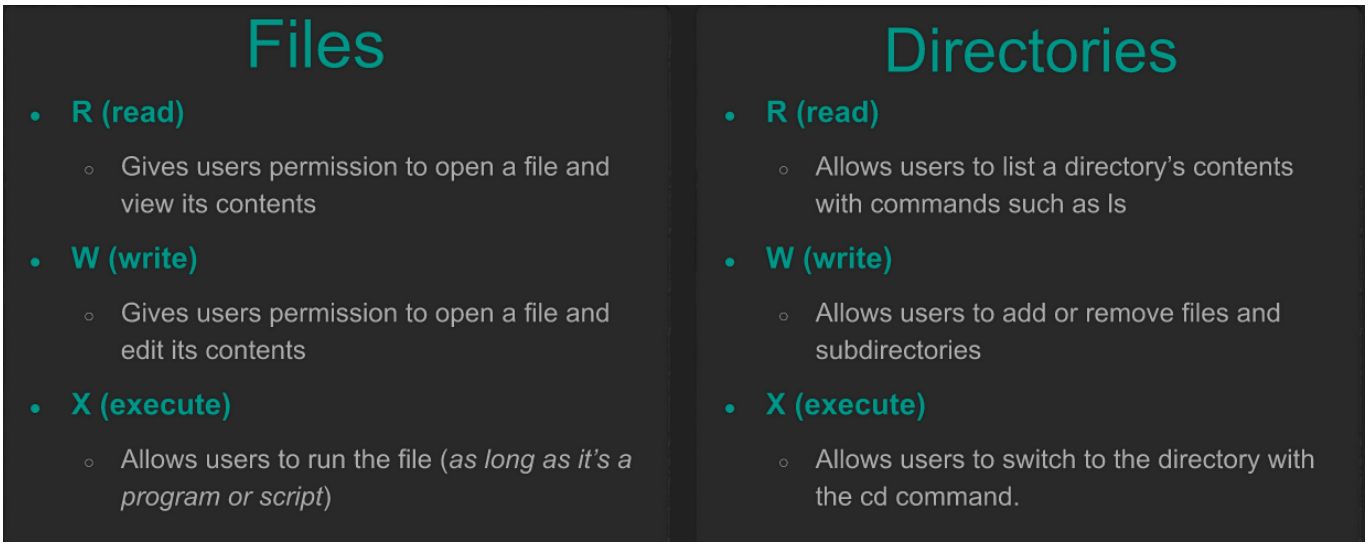
Ls -l output review



Linux File Permissions



Linux File Permissions | Files vs Directories



Linux File Permissions | The chmod command

- the **chmod (change mode)** command is used to change permissions on files and directories.
- It has this Syntax: **chmod permissions file/directory**
- You can use it in **two** ways to change file permissions:
  - Symbolic notation
  - Numeric notation

Linux File Permissions | Symbolic Notation

Table 5-2 Symbolic notation

Category	Operator	Permission
u (user)	+ (add to existing permissions)	r (read)
g (group)	- (remove from existing permissions)	w (write)
o (other)	= (assign absolute permissions)	x (execute)
a (all)	One of the preceding operators	One or more of the preceding permissions

Examples:

- `chmod u+x script.sh`
- `chmod o-x script.sh`
- `chmod u=rwx,g=rw,o=r script.sh`

Linux File Permission | Numeric Notation

Table 5-3 Numeric notation

Permission	Numeric value
---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
rw-	6
rwx	7

Permission	Value
Read	4
Write	2
Execute	1

Example:

```
chmod 766 script.sh
chmod 700 script.sh
chmod 555 script.sh
```