How to Linux

```
Who am I?
https://www.linkedin.com/in/propatriavigilans/
https://raymondrizzo.com
```

Least Privilege and Root

You should be using a user account that does not have permissions beyond what you need to do. However, at times you will need to elevate your privileges by doing one of the following:

- Run a command at a higher privilege level, if allowed, using `sudo`: `$ sudo mkdir /media/cdrom`
- Change to a different user using `su`: `$ su privileged-user`
- Change to root, if you know the password: `$ su`

More information on Commands

This will not be an exhaustive, or remotely complete explanation for the commands listed herein.

For further information on any command seen below, use one of the following methods:

- You can usually use the command, followed by `--help` for more information: `$ cat --help`
- You can use the Linux man-pages to find more information: `$ man cat`
    - Some entries may exist in more than one section (see: https://en.wikipedia.org/wiki/Man_page).
    - To see what information is available, use the `-k` option: `man -k "^zip$"`

Directory Navigation

- Move up one level: `$ cd ..`
- Move directly to a directory: `$ cd /etc`
- Move directory to a nested directory: `$ cd /usr/share`
- Move directly to the file system root: `$ cd /`
- Change directory to you home directory: `$ cd`
    - or: `$ cd ~`
- Print current working directory: `$ pwd`

Directory Listing

- Basic directory listing: `$ ls`
- Long listing format: `$ ls -l`
- Long listing format / show all: `$ ls -la`

File Creation

- Create an empty file: `$ touch example.file.1`
- Create a new directory: `$ mkdir test.directory`

File Copy

- Create a copy of a file with a specific name: `$ cp example.file.1 example.file.1.copied`
- Copy a file to other directory, preserving name: `$ cp example.file.1 /tmp/.`

File Move

- You cannot directly rename a file, but you can move it to change its name:`$ mv example.file.1 example.file.1.moved`
- Move a file to other directory, preserving name: `$ mv example.file.1.moved /tmp/.`

Echo, Redirects, Cat, and Pipes

- You can echo text to the screen, which is often useful in shell scripts: `$ echo "Look at me!"`
- Redirecting this output to a file will create a file if one does not already exist: `$ echo "This will be in example.file.2" > example.file.2`
- Cat will allow you to view the contents of this new file: `$ cat example.file.2`
- What do you expect the behavior of this to be?: `$ echo "different text for example.file.2" > example.file.2`
- A standard redirect will overwrite the contents of a file: `$ cat example.file.2`
- To append to a file, you must use `>>`: `$ echo "This will be in example.file.1" >> example.file.1`
- This will allow text/data to be added without overwriting: `$ echo "This will also be in example.file.1" >> example.file.1`
- Each time you append something, it will be added with a newline character at the end: `$ cat example.file.1`
- You can pipe the output of one command to another to make you life easier: `$ ps -ef | grep -i bash`
- At times, a file will contain too much information to be displayed in a single screen. You can pipe the output of cat to `more` to see each screen at a time, or use `less` to scroll back and forward through the file: `$ cat example.file.1 | less`

Symbolic (Symlink) and Hard Links

- Creating a symlink makes a logical pointer in the file system to the original file: `$ ln -s example.file.1 example.file.1.symlink`
- When referencing the symlink, you will actually be using the original file: `$ cat example.file.1.symlink`
- You can see the symlink and the file that it is pointed to: `$ ls -l`
- Creating a hard link essentially duplicates the file: `$ ln example.file.1 example.file.1.hardlink`
- The same information is shown from either: `$ cat example.file.1.symlink`
- Updating either the original or the hard link will update both: `$ echo "I'm here too!" >> example.file.1.symlink`
- Example: `$ cat example.file.1`
- Example: `$ cat example.file.1.hardlink`
- Example: `$ cat example.file.1.symlink`

Delete files

- Deleting the source file will break a symbolic link: `$ rm example.file.1`
- ls will show that the symbolic link is now broken: `$ ls -la`
- The file cannot be located: `$ cat example.file.1.symlink`
- The hard link is still working: `$ cat example.file.1.hardlink`
- Restoring the original file fixes the symlink: `$ cp example.file.1.hardlink example.file.1`
- Delete all files in current directory: `$ rm *`
- Delete directory: `$ rm -r ./test.directory`

Hidden Files

- Creating a file with a leading dot will make it hidden: `$ touch .example.file.hidden`
- A normal directory list will not display hidden files: `$ ls -l`
- However, the show all flag will: `$ ls -la`

Note: *A directory is also considered a file, and can be hidden the same way.*

Alias

- You can create an alias to make some commands easier for you: `$ alias cls="clear"`
- For example, this will clear your screen using the same command as CMD.EXE does on Windows: `$ cls`

Identifying Files, Permissions and Ownership

Everything in Linux is a file, whether it's a .txt, directory, port, or a hard disk.

If you are having difficulty running a shell script of binary file, ensure that you have the correct permissions for your user/group.

```
drwxr-xr-x. 4 example-user example-user  4096 May 23 13:36 .
drwxr-xr-x. 4 root         root          4096 May 23 13:36 ..
-rw-r--r--. 1 example-user example-user   220 May 23 13:36 .bash_logout
-rw-r--r--. 1 example-user example-user  5551 May 23 13:36 .bashrc
-rw-r--r--. 1 example-user example-user  3526 May 23 13:36 .bashrc.original
drwxr-xr-x. 5 example-user example-user  4096 May 23 13:36 .config
-rw-r--r--. 1 example-user example-user 11759 May 23 13:36 .face
lrwxrwxrwx. 1 example-user example-user     5 May 23 13:36 .face.icon -> .face
drwxr-xr-x. 3 example-user example-user  4096 May 23 13:36 .java
-rw-r--r--. 1 example-user example-user   807 May 23 13:36 .profile
-rw-r--r--+ 1 example-user example-user 10877 May 23 13:36 .zshrc
|\ /\ /\ /| | \__________/ \__________/ \___/ \__________/  \___/
| |  |  | | |      |             |        |        |         |- File name
| |  |  | | |      |             |        |        |-------- Last modified
| |  |  | | |      |             |        |------------------------- Assigned group
| |  |  | | |      |-------------------------------------------------- Owner
| |  |  | | |----------------------------- Number of hard links to the file
| |  |  | |- ACL (Access Control List) Permissions used to protect the file
| |  |  |--------------------------------- Permissions for all other users
| |  |---------------------------------- Permissions for the assigned group
| |------------------------------------ Permissions allowed for the user/owner
| ---------------------------------------------------- Directory or a file
```

```
                                                    d = directory
                                                    - = file
```

Permissions for User, Group and Other are assigned per file and per category.

The available permissions are:

- Read: Able to read file
- Write: Able to overwrite and delete file
- Execute: Able to run file

Permissions (Basic)

Permissions can be set for user, group, other in one shot by using the octal values associated with the permissions.

If you're not familiar, I suggest reading up on binary to decimal conversions.

```
Permission r w x
Value      4 2 1
```

- To allow the user/owner full access, and to remove all access from others: `$ chmod 700 example.file.1`
- To allow the user/owner only read and write access, and to remove all access from others: `$ chmod 600 example.file.1`

Individual permissions can be added for user/group/other.

- Add execute permissions for user/owner: `$ chmod u+x example.file.1`
- Add write permissions for group users: `$ chmod g+w example.file.1`
- Add read permissions for all other users: `$ chmod o+r example.file.1`

Individual permissions can also be removed for user/group other.

- Remove read permissions for all other users: `$ chmod o-r example.file.1`

File Ownership

- Change file owner user: `$ chown username example.file.1`
- Change file owner group: `$ chgrp`

File Execution

Usually, you will be able to execute most files in the `/bin`, `/sbin`, `/usr/bin`, and `/usr/sbin` directories without directly specifying their full location, as these are in the `PATH` environment variable.

When a file that is marked executable is not locatable in the `PATH` variable, you will need to specify the full location.

```
┌──(example-user@kvm)-[/tmp]
└─$ ls -la execute.me
-rwxr-xr-x 1 example-user example-user 35 May 23 14:35 execute.me

┌──(example-user@kvm)-[/tmp]
└─$ execute.me
execute.me: command not found

┌──(example-user@kvm)-[/tmp]
└─$ ./execute.me
I've executed!
```

The above example is referring to execute.me by using `./` to reference the current directory.

Permissions (Advanced)

ACL (Access Control List) permissions are used to protect files at a more granular level. This will only verify that your kernel is able to use ACLs, and not explain how to compile a kernel to allow them.

Note: This can also be used with groups.

- First, you need to know what file systems are mounted: `$ mount`
- For the next part, you need to know which kernel is running on your system: `$ uname -a`
- From that point, you can determine if the kernel is using ACLs or not: `$ cat /boot/config-5.4.0-109-generic | grep -i acl`
- Check to see if the ACL package is installed: `$ dpkg -l | grep -i acl`
- Install it if needed: `$ sudo apt-get install acl`
- Make a test directory: `$ sudo mkdir /tmp/test.directory`
- Remove execute permissions for other users: `$ sudo chmod o-x /tmp/test.directory`
- You should not be able to access the directory: `$ cd /tmp/test.directory`
- *Modify (-m)* the permissions on the directory to allow your user: `$ setfacl -m u:$USER:rwx /tmp/test.directory`
- You should be able to access the directory: `$ cd /tmp/test.directory`
- Create a subdirectory: `$ sudo mkdir /tmp/test.directory/subdirectory`
- You should be able to access the subdirectory: `$ cd /tmp/test.directory/subdirectory`
- But not create a file: `$ touch this`
- Set a default ACL to apply to any new files and subdirectories: `$ setfacl -m d:u:$USER:rwx /tmp/test.directory`
- Create a second subdirectory: `$ sudo mkdir /tmp/test.directory/subdirectory2`
- You should be able to access the subdirectory: `$ cd /tmp/test.directory/subdirectory2`
- As well as create a file: `$ touch this`

Locating Files and Directories

- One way to find the location of an executable is with which: `$ which ls`
- Find all .txt files with find: `$ find / -type f -name *.txt 2>/dev/null`
- Find all log directories: `$ find / -type d -name *log* 2>/dev/null`

One way to ensure less output to your session is to use `2>/dev/null` to redirect all errors to null.

Logs

The most common, non application specific, logs you will be gathering information from are:

- `/var/log/syslog`
- `/var/log/messages`
- `/var/log`
- `/var/log/auth.log`
- `/var/log/secure`
- `/var/log/kern.log`
- `/var/log/cron`

Other application specific logs are usually located within `/var/log` or `/opt/log`, but can be located anywhere the developer decided. Check your documentation to be sure.

Finding Information in Logs

There are a few methods of finding the information that you're looking for in a log:

- You can use `tail`, optionally specifying the number of lines, to find the last entries: `$ tail -n 20 /var/log/auth.log`
- You can use `tail` and follow the log to track events in real-time: `$ tail -f /var/log/auth.log`
- You can use `head` to grab the head of the file: `$ head -n 20 /var/log/auth.log`
- You can combine `cat` and `grep` to find specific information within: `$ cat /var/log/auth.log | grep -i -C 2 root`

In the above examples -i searches case-insensitive, and -C 4 provides 2 lines of context before and after each match.

Finding Information in Journald

Another way of finding system logs is if you're using journald.

- Follow works the same as `tail -f`: `$ sudo journalctl -f`
- Searching by regex here can yield a ton of useful information if you know what you are looking for: `$ sudo journalctl -g root`
- However, this is usually too much information, and you will waint to use `-u` to filter by unit (specifying the name of the service): `$ sudo journalctl -u postgres@14-main.service -f`

Filesystems

- List block devices: `$ lsblk`
- Show currently mounted filesystems: `$ mount`
- Mount an iso file: `$ sudo mount Centos.iso /media/cdrom`
- Unmounting a location: `$ sudo umount /media/cdrom`

Network Connectivity

Changing IP addressing on Linux varies by distribution, and is beyond the scope of this. However, here are some common references.

- CentOS/Red Hat: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/sec-configuring_ip_networking_with_nmtui
- Ubuntu: https://ubuntu.com/server/docs/network-configuration

Steps to Verify Network Connectivity

1. Check your interface to ensure an IP address is set: `$ ip addr`
2. Check your interface status: `$ ip link`
3. Check your ARP cache: `$ ip neigh`
4. Test your local IPv4 stack with: `$ ping -c 3 127.0.0.1`
    - Alternatively you may test your IPv6 stack with: `$ ping ::1`
5. Ping your static/DHCP address: `$ ping -c 3 xxx.xxx.xxx.xxx`
6. Find your default gateway and local routing table: `$ ip route`
7. Ping your default gateway: `$ ping -c 3 xxx.xxx.xxx.xxx`
8. Ping any remote IP addresses that you are attempting to connect to: `$ ping -c 3 xxx.xxx.xxx.xxx`
9. If you are using DNS to resolve hostnames:
    1. Ensure DNS nameservers are configured `$ cat /etc/resolv.conf`
    2. Ping each DNS server: `$ ping -c 3 xxx.xxx.xxx.xxx`
    3. Attempt to resolve the FQDN/Hostname of the server you're connecting to: `$ dig the-server`
10. If you are running a service locally that is not responding remotely
    1. Ensure that the service is running: `$ sudo ps -ef | grep -i service-name`
    2. Ensure that the service is listening: `$ sudo netstat -tunap | grep -i service-name`
    3. Attempt to connect to the service locally: `$ nc 127.0.0.1 port-number-of-service`
    - Usually entering some random text pressing enter should give you some feedback. As long as the port isn't closed, you usually at least make the connection.