# AN INTRODUCTION TO COMMAND LINE LINUX FOR ABSOLUTE BEGINNERS

How to not be scared when typing in the shell.

# WELCOME TO SHELL

- Welcome to the shell. We're going to be covering the Bourne Again Shell, also called "bash". Almost everything discussed here will work in other shells though.
- Most modern linux systems use bash.
- There are other shells such as fish, zsh, and ksh, they are similar but may have different features
- All shells display output from something called "STDOUT", or standard output, and take input from something called "STDIN", or standard input.

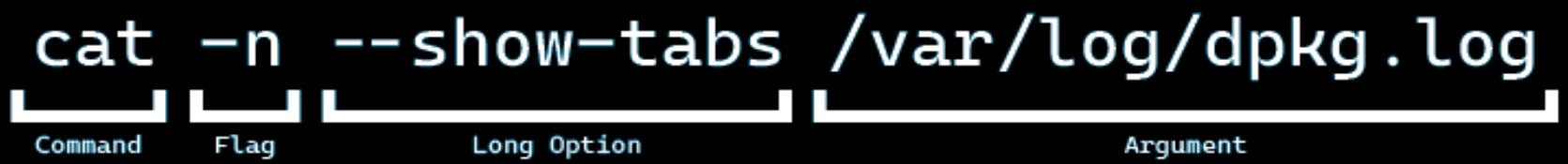
- A quick note: using the "clear" command will clear your terminal so that you can start over with a blank screen if it becomes too crowded

# WHERE AM I?

- Now that you're in the shell, where are we?
- Use the "pwd" command to find out what directory you're in.
- "pwd" stands for "Print Working Directory"
- If we want to see what's in our current directory, we can use the "ls" command
- "ls" lists the files and directories in your current working directory.
- "ls" also has a few options that may be useful, such as "ls –a" to list everything, including hidden directories, and "ls –l" that lists things such as file size, creation date, permissions, and more.

# WE HAVE OPTIONS

- Short options or flags are single character things we can pass to a command to do things. They're generally prepended by a single dash character.
- Example: ls –a
- There are also things called long options, longs, or long flags that are generally full words prepended by two dashes.
- Example: ls –-all
- These long options are generally just more verbose ways to represent short flags.
- Arguments are longer strings that can be passed to a command for things like file input, text to send, and more. Arguments are usually passed after all flags.
- Example: "mkdir directory"


cat -n --show-tabs /var/log/dpkg.log
Command
Flag
Long Option
Argument

# DIRECTIONS FOR DIRECTORIES

- We're not just limited to our current working directory, we can move around. We can use the "cd" command to change directories. It's followed by a path.
- "cd" accepts full paths, but also relative paths for things in your current directory, for example "cd /home/username/directory" works from anywhere, but if we were in the home directory we can just use "cd directory"
- We can also create a new directory using the "mkdir" command followed by the name of the directory we want to create
- If we want to go up a directory, back to it's parent directory without typing in the whole path, we can use the ".." path to show that we want to go up one level. "cd .."

# ADDITIONS AND DELETIONS

- Lets create a file
- We can use the “touch” command to create a file that doesn’t already exist.
- Lets create a new file: touch file.txt
- Use “ls” to show that there’s a new file called file.txt
- Now lets remove it. We can use the “rm” command to delete or “remove” files.
- Rm file.txt

# ECHO “ARROW” >> KNEE.TXT

- Arrows allow us to redirect command output to files
- >> will append data to a file
- > will overwrite all data in a file with new data
- Can filter errors, can pass data into a command, and lots more.
- Lets write some data to a file using the echo command to print some data to stdout and then redirect said stdout to a file.
- Echo “Hello, World!” > hello.txt
- The “echo” command simply outputs text to STDOUT, like the print() command in many programming languages.

# THE CAT

- There's a cat in your computer!
- Cat stands for "concatenate" and can be used to view the contents of a file. Just give cat a file and it will print the entire thing to stdout.
- Sometimes we have long files. One way to get to these files is to look at the head and the tail of the cat.
- The head command will print the first 10 lines of a file
- The tail command will print the last 10 lines.
- The number of lines displayed can be changed using the –n option.

# PLUMBING

- The pipe character “ | ” can be used to redirect command output to other commands.
- This is a feature that was copied from UNIX. It’s designed to allow users to chain many small commands together to create bigger and better programs.
- It is one of the fundamental features of all UNIX-like operating systems.

# MORE OR LESS

- So, you want to scroll through a long file rather than viewing the top, bottom, or all of the file?
- We can do this by passing the cat of the file through a command called more
- More is what's called a pager. A pager allows you to scroll through a file using the space bar to advance forwards.
- Cat /var/log/dpkg.log |more
- Press Q to quit out of more
- There's also less, which is a more advanced version of more with some advanced features.

# SEARCHING

- We can search through a file. To find lines containing only certain strings.
- Simply pass the cat of a file through a command called grep and give it a string to search for.
- Cat /var/log/dpkg.log | grep gcc

# IT'S NOT VIM

- "Ed is the standard text editor"
- Sometimes we need to edit a text file, or browse a text file.
- Instead of using vim, we are instead going to learn how to use nano because it's a bit more beginner-friendly.
- Ctrl+o to save
- Ctrl+x to exit
- That's how easy nano can be!

# USERS AND PERMISSIONS

- Having multiple users is a big feature of linux
- Whoami can be used to figure out what user you are currently logged in as.
- Why can't I edit this file? Lets see who owns it. ls -l to list the files in the current directory with more details.
- There are several default users on linux, the most important user (besides you) is the root account.
- Root can do anything. Also called the superuser account.
- Root can change any file, remove any file, change any setting, do anything. Root is an extremely powerful account.

# I AM ROOT

- Sometimes you need to run a command as root
- This should almost always be done using sudo, because the root account is almost always disabled for security reasons
- You must have rights to do sudo, this can be set up by a user with root access
- Some systems don't have sudo, and instead have su
- Su allows you to access superuser functions and gain a full root shell
- Su should not be used unless absolutely necessary

# PROCESSES

- All commands that run on a Linux system create what's known as a process. A process is a way that Linux can manage the memory associated with said program and mange access to various system resources.
- Processes can run in the foreground or the background. Foreground programs are things that you are actively working on, such as nano, ls, or other commands we've discussed here. Background commands are things like a web server, etc.
- We can see the current processes running on a system using the "ps" command.
- You'll notice the PID field, which is the ID that identifies processes on a Linux system.

# SUSPEND AND RESUME

- We can suspend processes that are currently running in the foreground so that we can go do other things using the “ctrl+z” keyboard shortcut.
- Let’s try it with nano. Open a new nano instance and then minimize it using the ctrl+z command.
- Let’s run that “ps” command again, and we can then see our nano instance there.
- If we want to return out background command to the foreground, we can use the “fg” command.

# IT'S MURDER

- We can kill processes if they're not behaving correctly
- Foreground processes can be killed using the "ctrl+c" keyboard shortcut.
- The C does not stand for copy, it stands for Cancel.
- We can kill background processes using their PID and the "kill" command.
- There are two methods of killing a process, asking it nicely to stop, and outright ending it.
- kill and then the process ID is the gentile way of asking a process to stop
- If that doesn't work, we can force it to kill using "kill -9"
- Force killing a process may lose unsaved data, and will leave opened files, locks, and resources in use hanging, which could cause errors.
- Kill -9 should only be used as a last resort.

# ~ IS WHERE THE HEART IS

- ~ is a reference for your home directory, can be used to shorten things like /home/username/text.txt to ~/text.txt
- !! Is the bang bang command. It can be used to repeat the previous command. This is most useful for sudo !! To re-run a command as root.
- !$ will repeat the first argument after the previous command. This is useful for running multiple commands on a single file, such as echo “hello” > text.txt and then cat !$
- $ is the expansion character. It can be used for variables.
- * is the wildcard character. It can be used to match parts of a file name. example: rm *.png to remove all png files in a directory. Or, rm IMG-*.png remove all files with the prefix IMG- with file type png.

# WHERE TO GET HELP

- The man pages
- --help
- Google, DuckDuckGo, Bing, Yahoo!