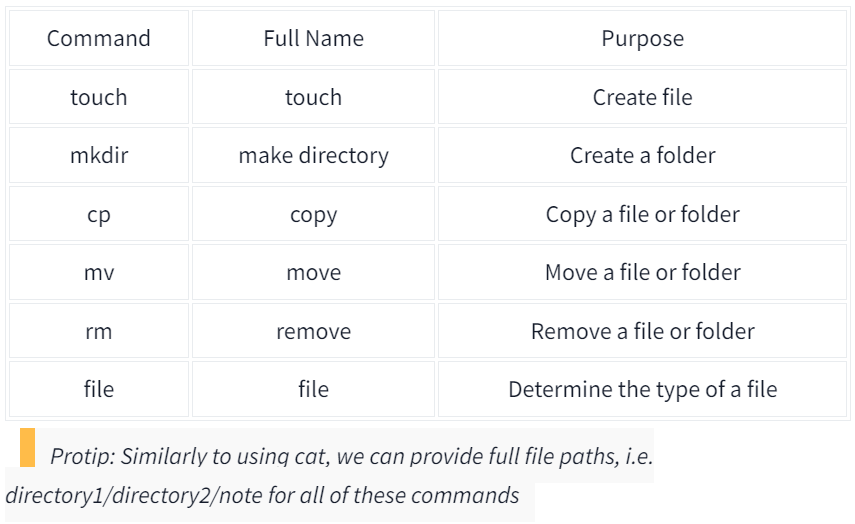
* sudo openvpn “filename”
  + Used to connect to TryHackMe. “filename” refers to the .ovpn file that TryHackMe gave you
* Reboot
  + Reboots the computer
* ifconfig
  + Equivalent for ipconfig in Windows
* Echo
  + Prints information
  + | base 64 -d - decodes a statement in base 64
* Whoami
  + Figures out what user name you are surently logged in as
* Ls
  + listing(finding out the contents of any files or folders)
* Cd
  + Change directory
* Cat
  + Concatenate (fantastic way for us to output the contents of files (not just text files!))
* Pwd
  + Print working directory
* Searchsploit
  + Run `searchsploit` command with a keyword related to vulnerability - e.g., `searchsploit apache`
  + Searchsploit will display list of exploits and CVE numbers (if available) - Each entry prefixed with unique ID - CVE number displayed alongside ID and exploit name
  + To View details of specific exploit use the`-x` flag and exploit ID - e.g., `searchsploit -x /path/to/exploit/directory/exploit\_id.rb
    - Replace /path/to/exploit/directory/exploit\_id.rb with the actual path and filename of the exploit you want to view.
* Find
  + We can use find to look for a specific file that we don’t know where its located or if we don’t know the file name we can use a wildcard (\*) to search for anything that has a “.txt”
  + Examples:
    - find -name passwords.txt
    - find -name \*.txt
* Grep
  + Allows us to search the contents of files for specific values such as a IP address or a prefix
  + Examples:
    - grep "81.143.211.90" access.log
    - Grep “THX” access.log
* - - help
  + Give all possible options that a common accepts. This is used behind the command
* Man (short for manual)
  + The manual pages are a great source of information for both system commands and applications available on a Linux machine
* The touch command takes exactly one argument -- the name we want to give the file we create. For example, we can create the file "note" by using touch note. It's worth noting that touch simply creates a blank file. You would need to use commands like echo or text editors such as nano to add content to the blank file.
* This is a similar process for making a folder, which just involves using the mkdir command and again providing the name that we want to assign to the directory.
* rm is extraordinary out of the commands that we've covered so far. You can simply remove files by using rm. However, you need to provide the -R switch alongside the name of the directory you wish to remove.\*\*\*Example: rm -R (name of directory), include without parenthesis
* Starting with cp, this command takes two arguments: 1. the name of the existing file and 2. the name we wish to assign to the new file when copying. cp copies the entire contents of the existing file into the new file. In the screenshot below, we are copying "note" to "note2".
* Moving a file takes two arguments, just like the cp command. However, rather than copying and/or creating a new file, mv will merge or modify the second file that we provide as an argument. Not only can you use mv to move a file to a new folder, but you can also use mv to rename a file or folder.
* So far, the files we have used in our examples haven't had an extension. Without knowing the context of why the file is there -- we don't really know its purpose. Enter the file command. This command takes one argument. For example, we'll use file to confirm whether or not the "note" file in our examples is indeed a text file, like so file note.
* su
  + Switch users
    - Can use the -l switch to inherit a lot more properties of the new user
* nano
  + Edits a file
    - nano “filename”
    - Ctrl is represented by ^ on linux
    - VIM is a similar tool for text editing
* wget
  + Downloads a specific file if you know the web address
    - wget “url”
* scp
  + Secure copy through ssh and provides both authentication and encryption
    - scp “file name” “source” “destination”
    - If not logged into the computer it looks like this: “destination” “document name”
* ps
  + Provide a list of running processes
* ps aux
  + See the processes run by other users and those that don't run from a session (i.e. system processes)
* top
  + gives you real-time statistics about the processes running on your system instead of a one-time view
* Kill
  + Kills a command as long as you give the associated PID
  + Processes you can send when it is killed:
    - SIGTERM - Kill the process, but allow it to do some cleanup tasks beforehand
    - SIGKILL - Kill the process - doesn't do any cleanup after the fact
    - SIGSTOP - Stop/suspend a process
* systemctl
  + Allows us to interact with the **systemd** process/daemon.systemctl is an easy to use command that takes the following formatting: systemctl [option] [service]For example, to tell apache to start up, we'll use systemctl start apache2Same with if we wanted to stop apache, we'd just replace the [option] with stop (instead of start like we provided)
    - Start
    - Stop
    - Enable
    - Disable
* Less
  + terminal pager program used to view the contents of a text file one screen at a time. less allows backward navigation in the file and forward navigation
* whois
  + Searches for an object in an RFC 3912 database
  + Syntax: whois <domain/IP>
* Traceroute
  + it allows you to see every intermediate step between your computer and the resource that you requested.
    - Syntax: traceroute <destination>
* Dig
  + Allows us to manually query recursive DNS servers of our choices for information about domains
    - dig <domain> @<dns-server-ip>

**Processes and States**

* Processes can run in two states: foreground and background
* Commands run in the terminal (e.g., echo) run in the foreground by default
* Background processes do not display output in the terminal

**Backgrounding a Process**

* Use the & operator at the end of a command to run it in the background
  + Example: echo "Hi THM" & (returns process ID instead of output)
* Useful for long-running tasks like file copying, allowing further commands
* Can background a running process with Ctrl + Z
  + Example: backgrounding a script that keeps looping output

**Foregrounding a Process**

* Use the fg command to bring a backgrounded process back to the foreground
  + Allows interacting with the process and seeing its output
* Can check for background processes with ps aux

**Users can schedule actions/tasks to run after system boot, e.g.:**

* Running commands
* Backing up files
* Launching programs like Spotify, Chrome

**Cron Process and Crontabs:**

* Cron is a process started at boot for managing cron jobs(In Linux, automated tasks are commonly referred to as "cron jobs". The cron is a time-based job scheduler in Unix-like operating systems that allows users to schedule tasks (known as "cron jobs" or "cron entries") to run periodically at fixed times, dates, or intervals.)
* Crontab is a special file with formatting recognized by cron to execute commands

**Crontab Format (6 values required):**

1. MIN - Minute to execute (0-59)
2. HOUR - Hour to execute (0-23)
3. DOM - Day of month to execute (1-31)
4. MON - Month to execute (1-12)
5. DOW - Day of week to execute (0-6, 0=Sunday)
6. CMD - Command to execute

Example: Back up Documents folder every 12 hours 0 \*/12 \* \* \* cp -R /home/cmnatic/Documents /var/backups/

**Using Wildcards (\*):**

* + can be used if value doesn't matter
* 0 \*/12 \* \* \* runs command every 12 hours regardless of date

**Resources**:

* Crontab Generator (web tool to generate formatting)
* Cron Guru (cron schedule expression editor)

**Editing Crontabs:**

* crontab -e (opens editor to edit user's crontab)

**Introducing Packages & Software Repos**

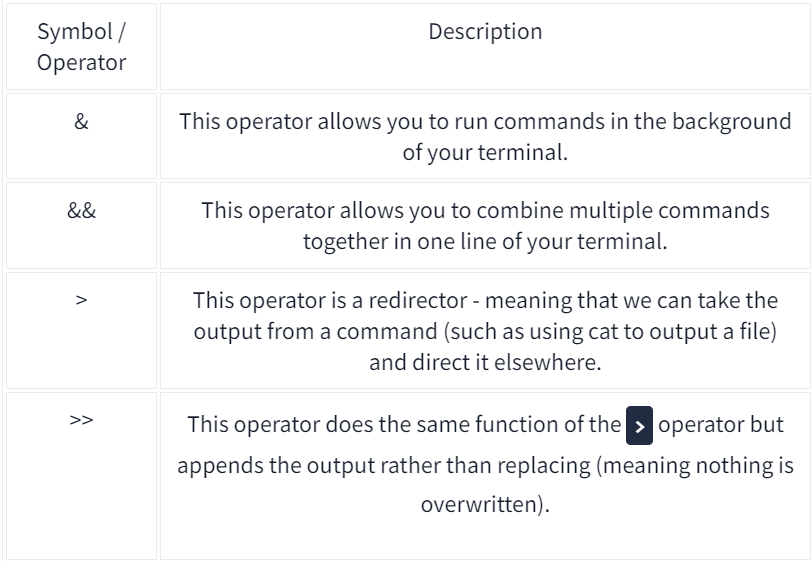
* Developers submit software to "apt" repository for community
* Highlights Linux's user accessibility and open source tools
* ls command on Ubuntu 20.04 shows gateway/registry files
* Can add community repositories to extend OS capabilities
* Additional repos added via add-apt-repository or listing another provider

**Managing Your Repositories (Adding and Removing)**

* apt command installs/manages software packages
* Adding repos manually (instead of add-apt-repository)
* Adding Sublime Text 3 repo as example
* GPG (Gnu Privacy Guard) keys verify software integrity
* Steps:
  + Download GPG key: wget -qO - https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
  + Add repo to sources list
    - Create /etc/apt/sources.list.d/sublime-text.list
    - Add repo info to file
  + Update apt: apt update
  + Install software: apt install sublime-text
* Removing:
  + add-apt-repository --remove ppa:PPA\_Name/ppa
  + Or delete added file
  + apt remove [software-name]

**Common Directories**

* /etc
  + Stores system configuration files and settings
  + Contains files used by the operating system and various programs
  + Examples:
    - **sudoers**: List of users/groups permitted to run commands as root via sudo
    - **passwd**: Stores user account information and e
    - rypted passwords (uses sha512 encryption)
* /var
  + stores data that is frequently accessed or written by services or applications running on the system
    - For example, log files from running services and applications are written here (**/var/log**), or other data that is not necessarily associated with a specific user (i.e., databases and the like)
    - the two types of log files below that are of interest:
      * access log
      * error log
* /root
  + Home directory for the root (superuser) account
  + Unlike /home which contains home directories for regular users
  + Logical assumption is that root's home would be /home/root but convention is to have root's home at /root instead
  + Contains root user's personal files, configurations, etc.
* /tmp
  + Short for "temporary", the /tmp directory is used to store data that is only needed to be accessed once or twice. Similar to the memory on your computer, once the computer is restarted, the contents of this folder are cleared out.



**Shell Operators**

* The "&" shell operator allows us to execute a command and have it run in the background (such as a large file copy)
* We can use "&&" to make a list of commands to run for example command1 && command2. However, it's worth noting that command2 will only run if command1 was successful.
* Operator ">" This operator is what's known as an output redirector. What this essentially means is that we take the output from a command we run and send that output to somewhere else. \*\*\*\*\*Note: If the file already exists, the contents will be overwritten!
* The >> operator allows to append the output to the bottom of the file — rather than replacing the contents

**RustScan**

**Fast port scanner built in Rust**

Syntax: **rustscan [FLAGS] [OPTIONS]**

Options:

* -a <addresses> - List of comma separated CIDRs, IPs, or hosts to be scanned
* -p <ports> - List of comma separated ports to be scanned
* -r <port ranges> - Range of ports with format start-end
* -t <time in milliseconds> - Timeout before a port is assumed to be closed

[default: 1500]

* --tries <tries> - number of tries before a port is assumed to be closed

[default: 1]

**Mosquitto**

* Installation: sudo apt-get install mosquitto mosquitto-clients -y
* **mosquitto\_sub**
  + **Used for subscribing to topics**
  + Options:
    - -h <hostname> - Host to be connected to
    - -p <port number> - Port number to connect through
    - -t <topic> - MQTT topic to subscribe to
      * “#” is a wildcard command to listen to everything
* **mosquitto\_pub**
  + **Used for publishing simple messages to subscribers**
  + Options:
    - -h <hostname> - Host to be connected to
    - -p <port number> - Port number to connect through
    - -t <topic> - MQTT topic to publish the message
    - -m <message> - Send a single message from the command line
* When connecting to

**Netcat**

**The nc (or netcat) utility is used for just about anything under the sun involving TCP, UDP, or UNIX-domain sockets. It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, do port scanning, and deal with both IPv4 and IPv6. Unlike telnet(1), nc scripts nicely, and separates error messages onto standard error instead of sending them to standard output, as telnet(1) does with some.**

Syntax: **nc [destination] [port]**

[-I length]

[-i interval]

[-M ttl]

[-m minttl]

[-O length]

[-P proxy\_username]

[-p source\_port]

[-q seconds]

[-s source]

[-T keyword]

[-V rtable]

[-W recvlimit]

[-w timeout]

[-X proxy\_protocol]

[-x proxy\_address[:port]]

[-Z peercertfile]

**SSH**

**Used for logging into a remote machine and for executing commands on a remote machine**

Syntax: **ssh [Username]@[Hostname/IP]**

**Extra Information**

If a password hash starts with $6$ it is sha512crypt (Unix variant)