Virtual Box Signin – sysadmin / cybersecurity

 pwd: Displays the current working directory.

 ls: Lists the directories and files in that current directory.

 cd: Navigates into a directory.

 mkdir: Makes a directory.

 rmdir: Removes a directory.

 touch: Creates an empty file.

 rm: Removes a file.

 clear: Clears the terminal history on the page.

 file path: Identifies a unique location in a file system.

 absolute paths: A path that includes the top directory of the file system.

 relative paths: A path that begins from the current directory.

 cp: Copies files.

 mv: Moves files.

Multiple preview commands can assist with these tasks.

* To preview and scroll through a whole file, we can use the more or less commands.
  + more: Used to view a file one page at a time. To move to the next page, press space bar.
  + less: This command is similar to more. It allows you to scroll up and down a file.
* To preview a file by a certain number of lines, use the head or tail commands.
  + head: Displays the top 10 lines of a file.
  + tail: Displays the bottom 10 lines of a file.
* 10 is the default number of lines, but this number can easily be modified.
  + The syntax for head is: head -number file.
  + For example: head -50 logfile.txt will show the top 50 lines in the logfile.txt file.
  + The tail command uses the same syntax.

CTRL C can get out of a locked screen.

su: switch user

Can use ‘sudo touch [file name]’

Can use ‘ego’ to preview image files – ego [file name].

Commands for concatenating and redirecting:

* cat: Concatenates and combines multiple files together.
* >: Writes to a file, and overwrites file if the file name already exists.
* >>: Writes to a file, and appends to the file if the file name already exists.

The command to display the man page is very simple: man <command>.

* For example: man ls will display the man page of the ls command.

**Syntax for Finding a File**

1. find -type f
   * We'll start with the command used to find **all files**. In this specific example, we are finding all files in our current directory and its subdirectories.
   * In order to do this, we will use the option -type and the required parameter f to indicate that we are searching for files.
2. find -type f -name log.txt
   * Now we will find a **specific file**. In this example, we are finding the files called log.txt in our current directory and its subdirectories.
   * In order to do this, we are using an option, -name, to search for an exact match of the specified parameter log.txt.
3. find -type f -iname log.txt
   * To remove case sensitivity and find all relevant files regardless of whether the file name matches the parameter's case, we will change the -name option to -iname. This option requires a parameter identifying what you are looking for.
   * We can use the above example to find the files called log.txt (lowercase) and LOG.TXT (uppercase) in your current directory and its subdirectories.
4. find -type f -iname \*.txt
   * In this example, we are using a symbol known as a **wildcard** in order to search for all files that end with .txt.
   * The \* wildcard symbol indicates that any file ending with .txt will be included in the results, regardless of what comes before .txt. Using wildcards with find is known as a wildcard search.
   * We can run the above example to find all files ending with lowercase .txt or uppercase .TXT in our current directory and its subdirectories.
   * At times, you may need to search for part of a file name.
     + For example, you may want to look for all file names that begin with a certain date, regardless of what the file name ends with.
     + This can be done with a wildcard, signified by an asterisk \*.

There is a command called grep that allows us to search within a file or multiple files to find a specific data point.

* grep, which stands for "global regular expression print," is a command to search for data inside of files.
* grep by default returns the entire line that the desired data point is found in.
* grep by default will only search for data in the current directory and not in sub-directories.

**Syntax for grep**

1. <grep data\_point File\_to\_search\_inside>
   * In this basic syntax, grep is used to find a specific data point within a **single** file.
   * Next, we'll run through a few examples of the grep command, highlighting the various features and options we can use.
2. grep bob log1.txt
   * In this example, we are using grep to find the lines in which the user bob is mentioned in the file log1.txt.
     + grep: The command being run.
     + bob: The specific data point being searched for.
     + log1.txt: The file being searched for the data point.
   * After this command is run, all the lines where the data point bob was found inside of the file log1.txt will be displayed.
   * If no matches of bob are found in the file, nothing will be returned.
3. grep bob \*.txt
   * In this next example, we are using grep to find a specific data point within **multiple** files.
   * Specifically, we are using grep to find where bob exists within in all .txt files.
     + bob is the specific data point being searched for.
     + \*.txt is the wildcard. \* indicates that the command will search through all files that end with .txt.
   * After this command is run, it will display the files where the value of bob was found, followed by the lines where it was found inside of all the .txt files.
4. grep -i bob \*.txt
   * This grep command can be used to find a **case-insensitive** specific data point within multiple files.
   * Specifically, this command finds the lines where the user bob or BOB exist in all TXT files.
     + grep: The command being run.
     + -i: The option for grep that indicates case insensitivity.
     + bob: The specific data case-insensitive point being searched for.
     + \*.txt is the wildcard of \* that indicates it will search through all files that end with .txt.
5. grep -il bob \*.txt
   * In the final example, we are showing that grep can be used to find the **file name** that contains a specific data point.
   * Specifically, this command only outputs the **file name** where the user of bob or BOB exist within all TXT files. When this command is run, it will only display the names of the file that contain bob.
     + grep: The command being run.
     + -il: Two options placed together.
       - i: An option for grep that indicates case insensitivity.
       - l: An additional option that indicates to only return the file name.
       - Note that i and l are two separate options. However, we can place them under a single hyphen.
     + bob: The specific data point.
     + \*.txt: The wildcard.
     + Atlanta, Georgia, USA
6. To use awk to only display the state, Georgia, we will run the following:

awk -F, '{print $2}' awk.txt

* + The syntax (after the pipe): awk -F(delimiter) '{print $(field\_number)}'
    - awk: Indicator telling your operating system to run the awk command.
    - -F,: Option for doing field separation with awk.
      * The value that comes right after -F is the parameter signifying how the file separates out the data points. In the example, the parameter is a comma (,).
      * This parameter is also known as the **delimiter**.
      * **Note:** The parameter needs to come directly after the -F, with no spaces.
    - The Dog Chased the Ball

1. Next we will use sed to replace the word "Ball" with "Cat":

cat sed.txt | sed s/Ball/Cat/

* + The syntax is:

cat (file name) (pipe) sed s/(old value)/(replacement value)/

1. Run the command. It replaces the word Ball with Cat with the following result:
   * The Dog Chased the Cat

if [ <condition> ]: if this condition is true...

then: run the following command(s).

else: if [ <condition> ] is false, run the following command(s).

fi: ends the if statement.

 = : Checks if two strings are equal.

 != : Checks if two strings are not equal.

 -eq : Checks if two integers are equal.

 -ne : Checks if two integers are not equal.

 -gt : Checks if one integer is greater than another.

 -lt : Checks if one integer is less than another.

 -d /path\_to/directory : Checks for existence of a directory.

 -f /path\_to/file : Checks for existence of a file.

Shell use nano #!/bin/bash