Linux command for Text processing and searching

- **Nano, vi, jed command**:

Linux lets users edit files using a text editor like nano, vi, or jed. While most distributions include nano and vi, users must install jed manually. All these tools have the same command syntax:

- nano filename
- vi filename
- jed filename

If the target file doesn't exist, these editors will create one. We recommend nano if you want to quickly edit text files. Meanwhile, use vi or jed for scripting and programming.

- **Cat command:**

Concatenate or cat is one of the most used Linux commands. It lists, combines, and writes file content to the standard output. Here's the syntax:

cat filename.txt

There are various ways to use the cat command:

cat > filen.txt –creates a new file.

cat file1.txt file2.txt > file3.txt –merges file1.txt with file2.txt and stores the output in filename3.txt.

tac file.txt – displays content in reverse order.

- **Grep command:**

The global regular expression or grep command lets you find a word by searching the content of a file. This Linux command prints all lines containing the matching strings, which is useful for filtering large log files.

For example, to display lines containing blue in the notepad.txt file, enter:

grep blue notepad.txt

- **sed command:**

The sed command lets you find, replace, and delete patterns in a file without using a text editor. Here's the general syntax:

sed [option] 'script' input_file

The script contains the searched regular expression pattern, the replacement string, and subcommands. Use the s subcommand to replace matching patterns or d to delete them.

At the end, specify the file containing the pattern to modify. Here's an example of a command that replaces red in colors.txt and hue.txt with blue:

sed 's/red/blue' colors.txt hue.txt

- **head command:**

The head command prints the first ten lines of a text file or piped data in your command-line interface. Here's the general syntax:

head [option] [file]

For instance, to view the first ten lines of note.txt in the current directory, enter:

head note.txt

The head command accepts several options, such as:

-**n** – changes the number of lines printed. For example, head -n 5 shows the first five lines.

-c – prints the file's first customized number of bytes.

-q – disables headers specifying the file name.

- **Tail command:**

The tail command displays the last ten lines of a file, which is useful for checking new data and errors. Here's the syntax:

tail [option] [file]

For example, enter the following to show the last ten lines of the colors.txt file:

tail -n colors.txt

- **awk command:**

The awk command scans regular expression patterns in a file to retrieve or manipulate

matching data. Here's the basic syntax:

awk '/regex pattern/{action}' input_file.txt

The action can be mathematical operations, conditional statements like if, output expressions such as print, and a delete command. It also contains the $n notation, which refers to a field in the current line.

To add multiple actions, list them based on the execution order, separated using semicolons. For example, this command contains mathematical, conditional, and output **statements:**

awk -F':' '{ total += $2; students[$1] = $2 } END { average = total / length(students); print "Average:", average; print "Above average:"; for (student in students) if (students[student] > average) print student }' score.txt

- **sort command:**

The sort command rearranges lines in a file in a specific order. It doesn't modify the actual file and only prints the result as Terminal outputs. Here's the syntax:

sort [option] [file]

By default, this command will sort the lines in alphabetical order, from A to Z. To modify the sorting, use these options:

-o – redirects the command outputs to another file.

-r – reverses the sorting order to descending.

-n – sorts the file numerically.

-k – reorders data in a specific field.

- **Cut command:**

The cut command retrieves sections from a file and prints the result as Terminal outputs. Here's the syntax:

cut [option] [file]

Instead of a file, you can use data from standard input. To determine how the command sections the line, use the following options:

-f – selects a specific field.

-b – cuts the line by a specified byte size.

-c – sections the line using a specified character.

-d – separates lines based on delimiters.

You can combine these options, use a range, and specify multiple values. For example, this command extracts the third to fifth field from a comma-separated list:

cut -d',' -f3-5 list.txt

- **diff command:**

The diff command compares two files' content and outputs the differences. It is used to alter a program without modifying the code. Here's the general format:

diff [option] file1 file2

Below are some acceptable options:

-c – displays the difference between two files in a context form.

-u – shows the output without redundant information.

-i – makes the diff command case insensitive.

- **Tee command:**

The tee command writes the user's input to Terminal's output and files. Here's the basic syntax:

command | tee [option] file1

For example, the following pings Google and prints the output in Terminal, ping_result.txt,and the 19092024.txt file:

ping google.com | tee ping_result.txt 19092024.txt

- **locate command:**

The locate command lets you find a file in the database system. Add the -i option to turn off case sensitivity and an asterisk (*) to find content with multiple keywords. For example:

locate -i school*note

The command searches for files containing school and note, regardless of their letter case

- **Find command:**

Use the find command to search for files within a specific directory. Here's the syntax:

find [option] [path] [expression]

For example, to find a file called file1.txt within the directory folder and its subdirectories, use this command:

find /home -name file1.txt