Final Assignment

Question 1

Awk

Description: A command-line utility that processes and analyzes text files by applying pattern-based rules.

Syntax: awk 'pattern {action}' filename

- Examples:
  - Print the first field of a file separated by colon: awk -F: '{print $1}' filename.txt
  - Find and print lines containing a specific word: awk '/searchword/ {print}' filename.txt
  - Calculate the total sum of a column: awk '{sum += $1} END {print sum}' filename.txt

Cat

Description: A command that concatenates and displays the contents of files.

Syntax: cat filename1 filename2 ...

- Examples:
  - Display the contents of a single file: cat filename.txt
  - Concatenate two files and display the result: cat file1.txt file2.txt
  - Append the contents of one file to another: cat file1.txt >> file2.txt

CP

Description: A command that copies files and directories.

Syntax: cp source_file destination_file

- Examples:
  - Copy a file to a new location: cp file.txt /path/to/destination/
  - Copy a directory and its contents recursively: cp -r directory /path/to/destination/
  - Copy multiple files to a directory: cp file1.txt file2.txt /path/to/destination/

Cut

Description: A command that extracts columns or fields from a file.

Syntax: cut -d 'delimiter' -f field filename

- Examples:
  - Extract the first field of a file separated by colon: cut -d: -f1 filename.txt
  - Extract a range of fields: cut -d: -f1-3 filename.txt
  - Extract fields based on character positions: cut -c1-5 filename.txt

Grep

Description: A command that searches for a pattern in a file and displays matching lines.

Syntax: grep pattern filename

- Examples:
  - Search for a word in a file and display matching lines: grep 'searchword' filename.txt
  - Search for a word recursively in a directory: grep -r 'searchword' /path/to/directory/
  - Search for a pattern and count the number of occurrences: grep -c 'pattern' filename.txt

Head

Description: A command that displays the first few lines of a file.

Syntax: head filename

- Examples:
  - Display the first 10 lines of a file: head filename.txt
  - Display the first 20 lines of a file: head -n 20 filename.txt
  - Display the first few lines of multiple files: head file1.txt file2.txt

LS

Description: A command that lists the contents of a directory.

Syntax: ls options directory

- Examples:
  - List the contents of the current directory: ls
  - List the contents of a directory with detailed information: ls -l /path/to/directory/
  - List all files in a directory, including hidden files: ls -a /path/to/directory/

Man

Description: A command that displays the manual page of a command or utility.

Syntax: man command

- Examples:
  - Display the manual page of the ls command: man ls
  - Display the manual page of the awk command: man awk
  - Display the manual page of the cp command: man cp

Mkdir

Description: A command that creates a new directory.

Syntax: mkdir directory

- Examples:
  - Create a new directory in the current directory: mkdir new_directory
  - Create a new directory with parent directories: mkdir -p /path/to/new/directory
  - Create multiple directories at once: mkdir directory1 directory2 directory3

Mv

Description: A command that moves or renames files and directories.

Syntax: mv source_file destination_file

- Examples:
  - mv file.txt backup/: Move the file file.txt to the backup/ directory.
  - mv oldfile.txt newfile.txt: Rename the file oldfile.txt to newfile.txt.
  - mv directory1/* directory2/: Move all files from directory1/ to directory2/.

Tac

Description: A command that displays the contents of a file or input stream in reverse order (i.e., from bottom to top).

Syntax: tac [OPTION]... [FILE]...

- Example:
  - tac file.txt: Display the contents of file.txt in reverse order.
  - cat file.txt | tac: Pipe the contents of file.txt to tac to display them in reverse order.
  - tac file1.txt file2.txt: Display the contents of file1.txt and file2.txt in reverse order.

Tail

Description: A command that displays the last few lines of a file or input stream.

Syntax: tail [OPTION]... [FILE]...

- Example:
  - tail file.txt: Display the last 10 lines of file.txt.
  - tail -n 5 file.txt: Display the last 5 lines of file.txt.
  - tail -f file.txt: Continuously display the last few lines of file.txt as the file is updated.

Touch

Description: A command that creates an empty file or updates the modification time of an existing file.

Syntax: touch [OPTION]... FILE...

- Example:
  - touch file.txt: Create an empty file called file.txt.
  - touch -d "2022-01-01" file.txt: Update the modification time of file.txt to January 1, 2022.
  - touch *.txt: Create empty files with the .txt extension in the current directory.

Tr

Description: A command that translates or deletes characters from a file or input stream.

Syntax: tr [OPTION]... SET1 [SET2]

- Example:

- tr 'a-z' 'A-Z' < file.txt: Translate all lowercase letters to uppercase letters in file.txt.
- echo "hello world" | tr -s ' ': Collapse multiple spaces into a single space in the input string "hello world".
- cat file.txt | tr -d '\r' > newfile.txt: Remove all carriage returns from file.txt and save the result to a new file called newfile.txt.

Tree

Description: A command that displays the directory structure of a path in a tree-like format.

Syntax: tree [OPTION]... [DIRECTORY]

- Example:
  - tree: Display the directory structure of the current directory.
  - tree -d: Display only the directories in the directory structure.
  - tree -L 2 /path/to/directory: Display the directory structure of /path/to/directory up to a depth of 2 levels.

Question 2

How to work with multiple terminals open?

In Linux, you can open multiple terminals by launching a new terminal window or tab. Most desktop environments have a shortcut key to open a new terminal window, and some allow you to split a single terminal window into multiple panes. You can also use tools like tmux or screen to create and manage multiple terminal sessions within a single window.

How to work with manual pages?

To view the manual page for a command, you can use the man command followed by the command name. For example, to view the manual page for the ls command, you can run man ls. The manual page is divided into sections, with each section providing different types of information about the command. You can navigate through the manual page using the arrow keys, page up and page down keys, or the q key to exit.

How to parse (search) for specific words in the manual page

To search for specific words in the manual page, you can use the / key followed by the search term. For example, if you want to search for the term "recursive" in the ls manual page, you can type /recursive and press Enter. This will highlight all occurrences of the term in the manual page. You can navigate between the occurrences using the n and N keys.

How to redirect output (> and |)

The > symbol is used to redirect the output of a command to a file. For example, if you want to save the output of the ls command to a file called filelist.txt, you can run ls > filelist.txt. The | symbol is used to pipe the output of one command as input to another command. For example, if you want to list the files in the current directory and then count the number of lines in the output, you can run ls | wc -l.

How to append the output of a command to a file

To append the output of a command to a file instead of overwriting it, you can use the >> symbol instead of >. For example, if you want to append the output of the ls command to a file called filelist.txt, you can run ls >> filelist.txt.

How to use wildcards for copying and moving multiple files at the same time

Wildcards are used to match multiple files or directories with a single pattern. The most commonly used wildcards are *, which matches any sequence of characters, and ?, which matches any single character. For example, if you want to list all files in the current directory that have a .txt extension, you can run ls *.txt. You can also use wildcards in combination with other commands, such as cp or rm, to copy or delete multiple files at once.

How to use brace expansion for creating entire directory structures in a single command

Brace expansion is used to generate multiple strings from a pattern. It allows you to create entire directory structures or lists of filenames with a single command. To use brace expansion, you can enclose the pattern in curly braces {} and separate the options with commas. For example, if you want to create three directories called folder1, folder2, and folder3, you can run mkdir folder{1,2,3}. This will create the three directories in one command.