Editing

Editing in shell scripting refers to modifying text files or scripts using command-line tools available in the shell environment. Shell scripts are collections of commands written in a scripting language (like Bash) that automate tasks.

How editing works in shell scripting:

- **Text Editors**: Shell scripts often involve creating or modifying text files. You can use text editors like "nano," "vim," or "emacs" directly from the command line to open and edit files.
- **Creating New Files:** To create a new file, you use a text editor with the file name as an argument. For example, nano newfile.txt will open a new file named "newfile.txt" in the "nano" editor.
- **Modifying Existing Files:** To modify an existing file, you use a text editor with the file name as an argument. For instance, nano existingfile.txt will open the "existingfile.txt" file in the "nano" editor for editing.
- **Editing Content**: Once the file is open in the text editor, you can navigate through the file using keyboard shortcuts, make changes to the content, and save the changes.
- **Saving Changes**: After making changes, save the modified file. In "nano," you might press "Ctrl + O" to write the changes to the file and "Ctrl + X" to exit the editor.
- **Shell Commands for Editing**: Besides using text editors, shell scripting also allows you to manipulate text files using various commands. For example:
- **echo:** Appends text to a file or displays text on the screen.
- **cat:** Displays the content of a file.
- **sed and awk**: Text processing tools for finding, replacing, and manipulating text within files.
- **grep**: Searches for specific patterns in files.

- **Variable Substitution**: Within shell scripts, you can use variables to store and manipulate text. Variables are placeholders that can be replaced with their values.
- **Here Documents:** Shell scripts often use "here documents" to include multi-line content within a script without needing to create separate files.
- **Permissions:** Remember to consider file permissions when editing files. Make sure you have the necessary permissions to read and write to the files you're editing.

Shell scripting and editing are crucial skills for automating tasks and managing configurations in a command-line environment

Overview of vi

An overview of the vi text editor. vi is a powerful and widely used text editor found on Unix-based systems. It is known for its efficiency and versatility, although its interface might be initially challenging for new users. Here's a brief overview of vi:

1. Modes:

vi operates in different modes: Normal mode, Insert mode, and Command-line mode.

Normal mode: This is the default mode where you can navigate and manipulate text.

Insert mode: In this mode, you can directly type and edit text.

Command-line mode: Used for entering commands like saving, quitting, and searching.

2. Basic Navigation in Normal Mode:

‘h’: Move cursor left.

‘j’: Move cursor down.

‘k’: Move cursor up.

‘l’: Move cursor right.

‘w’: Move to the beginning of the next word.

‘b’: Move to the beginning of the previous word.

‘0’: Move to the beginning of the current line.

‘$’: Move to the end of the current line.

‘gg’: Move to the beginning of the file.

‘G’: Move to the end of the file.

‘:{line_number}’: Move to a specific line number.

3. Editing in Normal Mode:

‘x’: Delete the character under the cursor.

‘dd’: Delete the current line.

‘yy’: Copy the current line.

‘p’: Paste the copied or deleted content.

‘u’: Undo the last action.

‘Ctrl + r’: Redo.

4. Inserting and Editing in Insert Mode:

Press ‘i’ in Normal mode to enter Insert mode at the cursor.

Press ‘a’ in Normal mode to enter Insert mode after the cursor.

Press ‘o’ in Normal mode to open a new line below and enter Insert mode.

Press ‘O’ in Normal mode to open a new line above and enter Insert mode.

Press ‘Esc’ to return to Normal mode from Insert mode.

5. Saving and Quitting:

‘:w’: Save changes.

‘:q’: Quit the editor.

‘:wq’: Save and quit.

‘:q!’: Quit without saving (force quit).

6. Search and Replace in Command-line Mode:

‘:/pattern’: Search for a specific pattern.

‘:%s/pattern/replace/g’: Replace all occurrences of a pattern with another string.

These are just some of the basic commands in vi. The editor offers many more features and commands that can be useful for both simple text editing and more complex tasks. Remember that mastering vi might take some practice, but once you become proficient, it can be a powerful tool for working with text files in a command-line environment.

Vi Commands

List of common vi commands categorized by the modes in which they are used:

Normal Mode Commands:

- **Navigation:**

  ‘h’: Move cursor left.

  ‘j’: Move cursor down.

  ‘k’: Move cursor up.

  ‘l’: Move cursor right.

  ‘w’: Move to the beginning of the next word.

  ‘b’: Move to the beginning of the previous word.

  ‘0’: Move to the beginning of the current line.

  ‘$’: Move to the end of the current line.

  ‘gg’: Move to the beginning of the file.

‘G’: Move to the end of the file.

‘:{line_number}’: Move to a specific line number.

‘%’: Move to the matching parenthesis or brace.

Editing:

‘x’: Delete the character under the cursor.

‘dd’: Delete the current line.

‘yy’: Copy the current line.

‘p’: Paste the copied or deleted content after the cursor.

‘P’: Paste before the cursor.

‘u’: Undo the last action.

‘Ctrl + r’: Redo.

‘.’: Repeat the last change.

Searching:

‘/pattern’: Search for a pattern forward.

‘?pattern’: Search for a pattern backward.

‘n’: Move to the next search result.

‘N’: Move to the previous search result.

Marks and Movement:

‘ma’: Set mark 'a' at the current cursor position.

‘'a’: Jump to the mark 'a'.

‘`a’: Jump to the beginning of the line containing mark 'a'.

‘'’: Jump back to the position before the latest jump.

‘Ctrl’ + ‘o’: Jump to older position.

‘Ctrl’ + ‘i’: Jump to newer position.

Copy and Paste Between Files:

‘:e {filename}’: Open a new file for editing.

‘:vsp {filename}’: Open a new file in a vertical split.

‘:sp {filename}’: Open a new file in a horizontal split.

‘:bnext’ or ‘:bn’: Move to the next buffer.

‘:bprev’ or ‘:bp’: Move to the previous buffer.

‘:bd’: Close the current buffer (close the file).

‘:ls’: List all open buffers.

‘:b {buffer_number}’: Switch to a specific buffer.

Insert Mode Commands:

❖ **Inserting Text:**

Press ‘i’ to enter Insert mode before the cursor.

Press ‘I’ to enter Insert mode at the beginning of the line.

Press ‘a’ to enter Insert mode after the cursor.

Press ‘A’ to enter Insert mode at the end of the line.

Press ‘o’ to open a new line below and enter Insert mode.

Press ‘O’ to open a new line above and enter Insert mode.

Press ‘Esc’ to return to Normal mode.

Command-Line Mode Commands:

✓ **Saving and Quitting:**

‘:w’: Save changes.

‘:q’: Quit the editor.

‘:wq’ or ‘:x’: Save and quit.

‘:q!’: Quit without saving (force quit).

‘:wqa’: Save changes in all open windows and quit.

✓ **Search and Replace:**

‘:%s/pattern/replace/g’: Replace all occurrences of a pattern with another string.

‘:s/pattern/replace/’: Replace the first occurrence of a pattern in the current line.

‘:s/pattern/replace/g’: Replace all occurrences of a pattern in the current line.

✓ **Moving Between Files:**

‘:e {filename}’: Edit a different file.

‘:sp {filename}’: Open a file in a horizontal split.

‘:vsp {filename}’: Open a file in a vertical split.

‘:b {buffer_number}’: Switch to a specific buffer.

These are just some of the most common vi commands. Remember that vi has a wide range of features and commands that can be quite powerful once you become comfortable with the editor's workflow.