Topics To be Covered

Week	Broader Topic	Lecture	Topics	Tools to be covered
1	Editors	31-35	1. Editors 1.1 features 1.2 vi editor 1.3 Modes of vi 1.3 Practice Questions 2. Vim editor 2.1 Installation 2.2 Modes of Vim 2.3 Basic Navigation 2.4 Practice Questions 3. Editor: Nano 3.1 Open a file 3.2 Basic Navigation 3.3 Practice Questions 4. Difference between vi, vim and nano 5. “sed” command	Linux

1. Editors:

There are several popular text editors available for Linux and Unix-like operating systems, each with its own set of features and advantages. Here are some commonly used text editors:

- Vim: Vim (Vi Improved) is a powerful and highly configurable text editor known for its modal editing capabilities. It has a steep learning curve but offers extensive customization, syntax highlighting, plugins, and support for various programming languages.
- Emacs: Emacs is a versatile and extensible text editor that provides a wide range of features, including customizable keybindings, syntax highlighting, integrated documentation, and support for programming languages. It also has a built-in Lisp interpreter that allows for extensive customization.
- Nano: Nano is a simple and user-friendly text editor that is easy to learn and use. It provides basic editing functionality, such as syntax highlighting, search and replace, and file navigation. Nano is often the default editor for many Linux distributions.
- Sublime Text: Sublime Text is a popular cross-platform text editor known for its speed and flexibility. It offers a modern and visually appealing interface, multiple selections, powerful search and replace capabilities, and an extensive set of plugins and themes.
- Atom: Atom is an open-source, hackable text editor developed by GitHub. It provides a highly customizable user interface, integrated package manager, and a wide range of plugins for enhanced functionality. Atom is known for its community-driven development and active ecosystem.
- Visual Studio Code: Visual Studio Code (VS Code) is a lightweight and feature-rich source code editor developed by Microsoft. It offers a wide range of extensions, powerful debugging capabilities, integrated

Git support, and an intuitive user interface. VS Code is highly popular among developers.

These are just a few examples of text editors available for Linux. Each editor has its own strengths and caters to different user preferences and requirements. It's worth exploring and trying out different editors to find the one that best suits your needs.

1.1 features of editor

Certainly! Here are some common features found in text editors:

- Syntax Highlighting: Text editors often provide syntax highlighting, which applies different colors or formatting to different parts of the code to improve readability and understanding.
- Auto-Indentation: Editors can automatically adjust the indentation level of code based on the programming language's syntax, making it easier to write well-formatted and organized code.
- Code Completion: Editors may offer code completion features, suggesting possible keywords, function names, and variable names as you type. This feature can save time and reduce typing errors.
- Find and Replace: Text editors usually provide find and replace functionality to search for specific text within a document and replace it

with another value. This feature can be helpful for making global changes in the code.

- Multiple Cursors and Selections: Advanced editors allow for multiple cursors or selections, enabling simultaneous editing or manipulation of multiple lines or sections of code.
- Split View and Tabs: Editors may support splitting the view into multiple panes or using tabs to work on different files simultaneously, making it easier to navigate and compare code.
- Version Control Integration: Many editors have built-in support for popular version control systems like Git, allowing you to perform common version control operations without leaving the editor.
- Plug-in and Extension Ecosystem: Text editors often provide extension or plug-in systems that allow users to customize and extend the editor's functionality with additional features, themes, and language support.
- Snippets and Templates: Editors can provide a way to insert pre-defined code snippets or templates for common programming constructs, reducing repetitive typing and improving productivity.
- Command Palette and Keyboard Shortcuts: Advanced editors often have a command palette or a quick access menu that allows you to execute commands or functions using keyboard shortcuts, enhancing efficiency and workflow.
- Split Editing: Some editors support split editing, allowing you to work on different parts of a document simultaneously, which can be helpful for referencing or editing different sections of code.
- Integrated Development Environment (IDE) Features: Certain text editors, such as Visual Studio Code and Atom, offer IDE-like features such as integrated debugging, integrated terminals, build systems, and more, providing a comprehensive development environment.

These are just some of the common features found in text editors. Different editors may have additional unique features and functionalities. It's important

to explore different editors to find the one that aligns with your preferences and workflow.

1.2 Linux file editor (vi)

In Linux, one of the commonly used file editors is vi, which is a powerful and versatile text editor. It is a command-line editor that comes pre-installed on most Linux distributions.

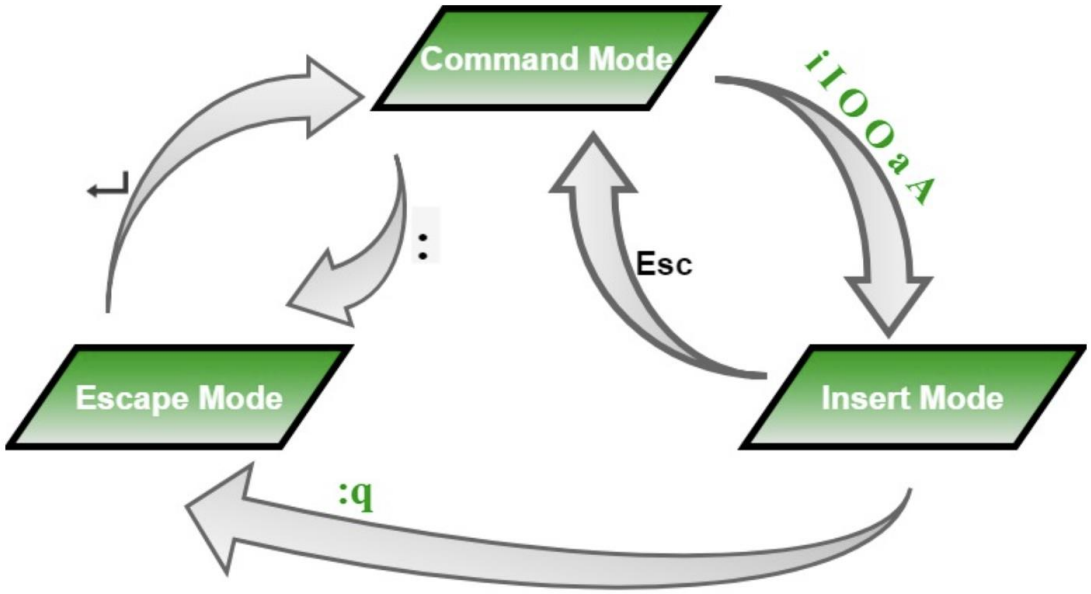
Here are some key features and commands of the vi editor:

1. **Opening a File:**

   ```
   vi filename
   ```

This command opens the specified file in the vi editor. If the file doesn't exist, a new file with that name will be created.

1.3 Modes in vi:



- **Command Mode**: This is the default mode when you open a file in vi. In this mode, you can navigate, search, and issue commands by typing single-character commands.
- **Insert Mode**: This mode allows you to insert and edit text in the file. Press i to enter Insert mode.
- **Visual Mode**: This mode is used for selecting and manipulating blocks of text. Press v to enter Visual mode.

3. **Basic Navigation:**
    - Use the arrow keys to move the cursor.
    - h: Move left.
    - j: Move down.
    - k: Move up.
    - l: Move right.
    - G: Move to the end of the file.
    - gg: Move to the beginning of the file.
    - :line_number: Move to a specific line number.

4. **Saving and Exiting:**
    - To save changes and exit vi, press Esc to enter Command mode and then type :wq or :x.
    - To exit vi without saving changes, press Esc to enter Command mode and then type :q!.

5. **Editing:**
    - i: Enter Insert mode to insert text before the cursor.
    - a: Enter Insert mode to append text after the cursor.
    - o: Open a new line below the current line and enter Insert mode.
    - dd: Delete the current line.
    - yy: Yank (copy) the current line.
    - p: Paste the copied or deleted text after the current line.

- u: Undo the last change.
- Ctrl + r: Redo a change that was undone.

6. **Searching and Replacing:**
    - /search_text: Search forward for search_text.
    - ?search_text: Search backward for search_text.
    - n: Move to the next occurrence of the search text.
    - N: Move to the previous occurrence of the search text.
    - :%s/old_text/new_text/g: Replace all occurrences of old_text with new_text in the entire file.

The vi editor has many more features and commands, but these should give you a good starting point for editing files in Linux using vi. It is a powerful editor, but it may have a learning curve if you're new to it. With practice and exploration, you'll become more comfortable and efficient with vi.

1.4 Practice Questions

1. Opening and Navigating Files:

- How can you open a file named "example.txt" in vi?
- What command can you use to move the cursor to the end of the file?
- How can you move the cursor to a specific line in the file?

2. Inserting and Editing Text:
    - How can you enter Insert mode to start inserting text at the current cursor position?
    - What command can you use to delete the current line in vi?
    - How can you undo the last change made in vi?
3. Saving and Exiting:
    - What command saves changes and exits vi simultaneously?
    - How can you exit vi without saving any changes made to the file?
    - Is it possible to save changes in vi without exiting the editor? If yes, how?
4. Searching and Replacing:
    - How can you search for a specific word in vi?
    - What command can you use to replace all occurrences of a word in the file?
    - How can you move to the next occurrence of a searched word in vi?
5. Visual Mode and Editing Blocks of Text:
    - How can you enter Visual mode in vi?
    - What is the purpose of Visual mode in vi?
    - Can you copy and paste a block of text using vi? If yes, how?

2. Editor: Vim

2.1 Installation and Configure vim in Our Linux System

To install vim on Debian-based Linux like Ubuntu run the command:

sudo apt-get install vim

Vim (Vi IMproved) is an enhanced version of the vi editor and is available on most Unix-like systems, including Linux. It is designed to be a highly configurable and powerful text editor, providing advanced features and customization options.

Here are some key features and commands of the Vim editor:

2.1 Opening a File:

vim filename

This command opens the specified file in the Vim editor. If the file doesn't exist, a new file with that name will be created.

2.2 Modes in Vim:

- **Normal Mode**: This is the default mode when you open a file in Vim. In this mode, you can navigate, issue commands, and perform various editing operations.
- **Insert Mode**: This mode allows you to insert and edit text in the file. Press i to enter Insert mode.
- **Visual Mode**: Similar to vi, this mode is used for selecting and manipulating blocks of text. Press v to enter Visual mode.

2.3 Basic Navigation:

- Use the arrow keys to move the cursor.

- h: Move left.
- j: Move down.
- k: Move up.
- l: Move right.
- G: Move to the end of the file.
- gg: Move to the beginning of the file.
- :line_number: Move to a specific line number.

4. **Saving and Exiting:**
    - To save changes and exit Vim, press Esc to enter Normal mode and then type :wq or :x.
    - To exit Vim without saving changes, press Esc to enter Normal mode and then type :q!.

5. **Editing:**
    - i: Enter Insert mode to insert text before the cursor.
    - a: Enter Insert mode to append text after the cursor.
    - o: Open a new line below the current line and enter Insert mode.
    - dd: Delete the current line.
    - yy: Yank (copy) the current line.
    - p: Paste the copied or deleted text after the current line.
    - u: Undo the last change.
    - Ctrl + r: Redo a change that was undone.

6. **Searching and Replacing:**

- /search_text: Search forward for search_text.
- ?search_text: Search backward for search_text.
- n: Move to the next occurrence of the search text.
- N: Move to the previous occurrence of the search text.
- :%s/old_text/new_text/g: Replace all occurrences of old_text with new_text in the entire file.

Vim provides many advanced features and customization options, including split windows, syntax highlighting, plugins, and extensive configuration files. It offers a steep learning curve, but with practice and exploration, you can harness its full power as a text editor.

2.4 Practice Questions

1. Opening and Navigating Files:
    - How can you open a file named "example.txt" in Vim?
    - What command can you use to move the cursor to the end of the file?
    - How can you move the cursor to a specific line in the file?
2. Inserting and Editing Text:
    - How can you enter Insert mode to start inserting text at the current cursor position in Vim?
    - What command can you use to delete the current line in Vim?
    - How can you undo the last change made in Vim?
3. Saving and Exiting:
    - What command saves changes and exits Vim simultaneously?
    - How can you exit Vim without saving any changes made to the file?
    - Is it possible to save changes in Vim without exiting the editor? If yes, how?

4. Searching and Replacing:
    - How can you search for a specific word in Vim?
    - What command can you use to replace all occurrences of a word in the file?
    - How can you move to the next occurrence of a searched word in Vim?
5. Visual Mode and Editing Blocks of Text:
    - How can you enter Visual mode in Vim?
    - What is the purpose of Visual mode in Vim?
    - Can you copy and paste a block of text using Vim? If yes, how?

3. Editor: nano

Nano is a simple and user-friendly text editor available on Linux systems. It is designed to be easy to use, especially for beginners, and provides basic editing capabilities. It is often included as a default text editor in many Linux distributions.

Here are some key features and commands of the Nano editor:

3.1 Opening a File:

```
nano filename
```

This command opens the specified file in the Nano editor. If the file doesn't exist, a new file with that name will be created.

3.2 Basic Navigation:

- Use the arrow keys to move the cursor.
- Ctrl + B: Move one page up.

- Ctrl + F: Move one page down.
- Ctrl + Y: Move one line up.
- Ctrl + V: Move one line down.
- Ctrl + _ (underscore): Go to a specific line number.

3. **Saving and Exiting:**
    - To save changes and exit Nano, press Ctrl + O, then press Enter to confirm, and finally press Ctrl + X to exit.
    - To exit Nano without saving changes, press Ctrl + X and then press N when prompted to confirm.

4. **Editing:**
    - Ctrl + K: Cut the current line.
    - Ctrl + U: Paste the cut line.
    - Ctrl + J: Justify the current paragraph.
    - Ctrl + W: Search for a specific word or phrase.
    - Ctrl + \: Replace a word or phrase.

5. **Miscellaneous:**
    - Ctrl + G: Display the help menu, which shows the available commands in Nano.
    - Ctrl + C: Show the current cursor position (line and column numbers).
    - Ctrl + R: Read a file into the current buffer.

Nano offers a minimalistic and straightforward interface for editing text files. It lacks some of the advanced features of editors like Vim, but it is easy to use and provides a basic set of functionalities for editing text files.

33 a. Practice Questions

- Opening and Editing a File:
- How can you open a file named "example.txt" in Nano?
- What command can you use to move the cursor to the beginning of a line?
- How can you insert text at the current cursor position in Nano?
- How can you cut a line in Nano?
- How can you paste a cut line in Nano?
- How can you search for a specific word or phrase in Nano?
- How can you replace a word or phrase in Nano?

4. Difference between vi, nano and vim editors

1. **Feature Set:**
    - vi and vim are both powerful text editors with a wide range of features suitable for more advanced users. They offer extensive customization options, syntax highlighting, support for multiple windows, and various navigation and editing commands.
    - nano is a more straightforward and user-friendly editor designed for basic text editing. It has a simpler feature set, making it easier to learn and use, but lacks some of the advanced functionalities of vi/vim.

2. **Learning Curve:**

    - vi and vim have a steeper learning curve compared to nano. They require familiarity with their command-driven interface and keyboard shortcuts. However, once mastered, they offer greater efficiency and productivity for experienced users.
    - nano is designed to be more accessible for beginners, with a menu-based interface and more intuitive keybindings. It has a gentler learning curve and is generally easier to start using.

1. **Availability**:

    - vi is typically available on most Unix-like systems, including Linux, as it is a standard text editor that comes with many distributions.
    - vim is an improved version of vi and is compatible with vi commands. It is often installed by default on modern Linux distributions but may need to be installed separately on some systems.
    - nano is not as widely available as vi/vim, but it is still commonly found on many Linux distributions and is often included as the default text editor.

2. **Customization:**

    - Both vi/vim and nano can be customized to some extent, but vim offers more extensive customization options. It has a highly configurable setup with support for plugins and advanced customization through its .vimrc configuration file.
    - nano has a simpler configuration system, allowing users to customize basic settings such as keybindings and color schemes.

In summary, vi/vim is a powerful and feature-rich text editor with a steep learning curve, while nano is a simpler and more beginner-friendly editor. The choice between them depends on your level of comfort with command-driven interfaces and the complexity of your text editing needs.

5. "sed" command

The sed command (short for "stream editor") is a powerful text manipulation tool available on Unix-like systems, including Linux. It is primarily used for performing operations on text files, such as searching, replacing, inserting, and deleting lines of text.

Here are some key features and common uses of the sed command:

1. Syntax:

   sed [options] 'command' input_file

2. Basic Usage:
   - By default, sed reads the input file line by line and applies the specified command(s) to each line.
   - The modified output is usually displayed on the console. To save the changes to the original file, you can use the -i option followed by an extension for creating a backup file (e.g., -i.bak) or -i alone for in-place editing without creating a backup.
3. Common Operations:
   - **Search and Replace**: Use the s command to search for a pattern and replace it with another string.
   - For example:

     sed 's/pattern/replacement/' input_file

   - **Delete Lines**: Use the d command to delete lines based on a condition. For example, to delete all lines containing a specific pattern:

     Copy codesed '/pattern/d' input_file

- Insert and Append Lines: Use the i or a command to insert or append lines before or after a specific line number or pattern. For example:

  sed '3iNew line' input_file sed '/pattern/aNew line' input_file

4. Options:
   - -n: Suppress automatic printing of the pattern space. Useful when using sed for selective printing.
   - -e or --expression: Specify multiple commands or scripts to be executed.
   - -r or -E: Enable extended regular expressions (support for extended regex syntax).

The sed command supports more advanced features like regular expressions, range-based operations, conditional branching, and more. It is a versatile tool for performing complex text transformations efficiently. For detailed information and additional usage examples, refer to the sed manual page (man sed) or the online documentation.