shell

In Linux, a **shell** is a command-line interface (CLI) that allows users to interact with the operating system. It acts as a middle layer between the user and the operating system kernel, interpreting commands and executing them.

Types of Shells in Linux

Linux supports various types of shells, each with unique features and syntax. The most common ones are:

1. Bourne Shell (sh)

- **Path:** `/bin/sh`
- **Features:**
  - The original Unix shell developed by Stephen Bourne.
  - Basic scripting capabilities.
  - Limited interactivity compared to modern shells.
- **Use case:** Foundational scripting and backward compatibility.

2. Bash (Bourne Again Shell)

- **Path:** `/bin/bash`
- **Features:**
  - Superset of Bourne Shell with added features.
  - Command-line editing, history, and scripting capabilities.
  - Widely used as the default shell in most Linux distributions.
- **Use case:** General-purpose scripting and user interaction.

3. Z Shell (zsh)

- **Path:** `/bin/zsh`
- **Features:**
  - Advanced scripting capabilities.
  - Highly customizable with themes and plugins.
  - Improved auto-completion and command correction.
- **Use case:** Power users and developers looking for enhanced functionality.

4. Korn Shell (ksh)

- **Path:** `/bin/ksh`
- **Features:**

    - Combines features of Bourne Shell and C Shell.
    - Advanced scripting capabilities and performance optimization.
- **Use case:** Preferred in enterprise systems and older environments.

5. C Shell (csh)

- **Path:** `/bin/csh`
- **Features:**
    - Syntax similar to the C programming language.
    - Limited scripting capabilities.
- **Use case:** Useful for users familiar with C-style syntax.

6. tcsh

- **Path:** `/bin/tcsh`
- **Features:**
    - An enhanced version of C Shell.
    - Supports command-line editing and scripting.
- **Use case:** Replacement for `csh` with interactive improvements.

7. Fish (Friendly Interactive Shell)

- **Path:** `/usr/bin/fish`
- **Features:**
    - User-friendly, designed for interactive use.
    - Rich auto-suggestions and syntax highlighting.
    - No backward compatibility with Bourne Shell scripts.
- **Use case:** Beginners or users who prefer a modern and intuitive shell.

How to Check Your Current Shell

You can determine which shell you are currently using by running:

```
echo $SHELL
```

How to Switch Between Shells

To check available shells on your system:

```
cat /etc/shells
```

1. To temporarily switch to a shell (e.g., `sh`):
   `sh`

2. To permanently change the default shell:
   `sudo chsh ubuntu -s /path/to/shell`

CLI editor

In Linux, a **CLI editor** is a text editor that operates entirely within the terminal or command-line interface. These editors are lightweight, fast, and often used for editing configuration files, writing scripts, or programming directly from the terminal.

Popular CLI Editors in Linux

1. Nano

- **Command:** `nano <filename>`
- **Features:**
  - Simple and beginner-friendly.
  - On-screen help for common commands.
  - Syntax highlighting support for programming languages.
  - Basic operations like cut, copy, paste, and search.
- **Use case:** Ideal for quick edits and beginners.
- **Exit Tip:** Use `Ctrl + X` to exit.

---

2. Vim (Vi Improved)

- **Command:** `vim <filename>` or `vi <filename>`
- **Features:**
  - Powerful and highly configurable.
  - Modes of operation:
    - **Normal mode:** Navigate and manipulate text.
    - **Insert mode:** Enter and edit text.

        - **Command mode:** Execute editor commands (e.g., save, exit).
    - Syntax highlighting, macros, and plugins.
    - Steeper learning curve but very efficient for advanced users.
- **Use case:** Programming, large-scale text editing.
- **Basic Commands:**
    - Enter **insert mode**: Press `i`.
    - Save: `:w`
    - Quit: `:q`
    - Save and quit: `:wq`
    - Quit without saving: `:q!`

Basic commands

Command	Description
`ls`	List files and directories in the current location.
`pwd`	Show the current working directory.
`cd <directory>`	Change to the specified directory.
`mkdir <directory>`	Create a new directory.
`rmdir <directory>`	Remove an empty directory.
`rm <file>`	Remove a file.
`rm -r <directory>`	Remove a directory and its contents recursively.
`cp <source> <dest>`	Copy a file or directory.
`mv <source> <dest>`	Move or rename a file or directory.
`touch <file>`	Create an empty file or update the timestamp.
`cat <file>`	Display the contents of a file.

Package Management (Debian-based Systems)

Command	Description
`apt update`	Update the list of available packages.
`apt upgrade`	Upgrade all installed packages.
`apt install <package>`	Install a specific package.
`apt remove <package>`	Remove a specific package.