Introduction to Linux Commands

Welcome to this introductory guide on Linux commands. If you have never used Linux before, this guide will help you learn some of the most basic and essential commands to get started with navigating and managing the Linux environment.

Section 1: Getting Started

Opening the terminal: Find and open the terminal application on your Linux distribution. The terminal is where you will enter commands to interact with the operating system. In our case, we will be using Ubuntu to run Linux.

Section 2: Navigating the Filesystem

1. **pwd** (Print Working Directory): Displays the current directory you are in.
   ```
   $ pwd
   ```
2. **ls** (List): Lists files and directories in the current directory.
   ```
   $ ls
   ```
3. **dir** (Directory): Display files in the current directory.
   ```
   $ dir
   ```
4. **cd** (Change Directory): Changes the current directory to a specified directory.
   To move to a specific directory
   ```
   $ cd /path/to/directory
   ```
   To move one directory up:
   ```
   $ cd ..
   ```
   To move to the root directory:
   ```
   $ cd /
   ```
   To move to the user's home directory:
   ```
   $ cd ~
   ```

Section 3: Managing Files and Directories

5. **mkdir** (Make Directory): Creates a new directory.
   ```
   $ mkdir new_directory_name
   ```
6. **rmdir** (Remove Directory): Deletes an empty directory.
   ```
   $ rmdir directory_name
   ```
7. **touch**: Creates a new, empty file.
   ```
   $ touch new_file.txt
   ```
8. **cp** (Copy): Copies files or directories.

To copy a file:

```
$ cp <source_file> <destination_file>
$ cp source_file.txt destination_file.txt
```

To copy a directory and its contents

```
$ cp -r <source_dir_name> <destination_dir_name>
$ cp -r source_dir destination_dir
```

r copies the files in the directory recursively

9. **mv** (Move): Moves or renames files and directories.

   To move a file:

   ```
   $ mv <source_file> <destination_directory>/
   $ mv source_file.txt destination_directory/
   ```

   To rename a file or directory:

   ```
   $ mv <old_name_file> <new_name_file>
   $ mv old_name.txt new_name.txt
   ```

10. **rm** (Remove): Deletes files.

    Remove single file:

    ```
    $ rm file_name.txt
    ```

    Remove multiple files:

    ```
    $ rm file1.txt file2.txt file3.txt
    ```

    Remove directory and all its contents.

    ```
    $ rm -r directory_name
    ```

    r recursively removes subdirectories and files

11. **clear** (Clear): Clears the terminal. Another way to achieve the same effect is by using the keyboard shortcut Ctrl + L.

    ```
    $ clear
    ```

Section 4: Viewing and Editing Files

12. **cat** (To join two files and store in output file).

    ```
    $ cat file1 file2 > file3
    ```

    cat (To displays the contents of a file).

    ```
    $ cat file_name.txt
    ```

13. **nano**: Text editors for creating or modifying files.

    To create or edit a file using nano:

    ```
    $ nano file_name.txt
    ```

Once in the editor, you can type and edit the text as needed. The bottom of the screen displays a list of available commands, with the **^** symbol representing the **Ctrl** key. Some common commands include:

- **^X**: Exit the editor. If you have unsaved changes, you'll be prompted to save them before exiting.
- **^O**: Save the current file (Write Out).
- **^W**: Search for text within the file (Where Is).
- **^K**: Cut a line of text (Cut Text).
- **^U**: Paste the previously cut text (Uncut Text).

vim is another text editor, in comparison to nano, vim has a steeper learning powerful but offers more advanced features and capabilities for efficient text editing. When you first open a file with vim, you are in Normal mode.

14. **vim**: Text editors for creating or modifying files.
    To create or edit a file using vim:
    ```
    $ vim file_name.txt
    ```

Essential commands to help you get started:

- **i**: Enter Insert mode, where you can type and edit text.
- **Esc**: Return to Normal mode from any other mode.
- **:w** Save the current file (Write) while in Command mode (accessed by pressing : in Normal mode).
- **:q** Exit the editor (Quit) while in Command mode. If you have unsaved changes, you'll need to force quit or save the changes first.
- **:wq** or **:x** Save and exit the editor (Write and Quit) while in Command mode.
- **:q!** Force quit the editor without saving changes while in Command mode.

Section 5: Getting Help

15. **man** (Manual): Displays the manual pages for specified commands.
    ```
    $ man command_name
    ```

16. **--help** or **-h**: Many commands support the --help or -h option, which provides a summary of the command usage and available options.
    ```
    $ command_name -help
    $ command_name -h
    ```

17. **info**: Displays more extensive documentation on some commands and programs. It offers a hierarchical, hypertext-like format for easy navigation.
    ```
    $ info command_name
    ```

These are the basic commands that every beginner should learn to get started with Linux. As you gain experience, you will come across more advanced commands and concepts, which will help you become more proficient in using Linux.

Section 6: Resources

Online resources: Ubuntu and Linux have vast online communities where you can find help, tutorials, and documentation. Some popular platforms include:

- Ubuntu Forums (https://ubuntuforums.org/)
- Ask Ubuntu (https://askubuntu.com/)
- Stack Overflow (https://stackoverflow.com/)
- Official Ubuntu documentation (https://ubuntu.com/tutorials)