Basic Linux commands:

File Operations: `cp`, `mv`, `rm`, `touch`

cp (copy files and directories)

- **r (or -R)**: Recursively copy directories, including all subdirectories and files within them. Essential when copying a directory and its contents.
- **i**: Interactive mode. Prompts the user before overwriting existing files. This can prevent accidental data loss.
- **v**: Verbose mode. Displays detailed output of the copy operation, showing files as they are copied.
- **-preserve=attributes**: Preserves specified attributes (e.g., mode, ownership, timestamps) of the files being copied. Without this, the copied files may adopt the current time as their modification time and potentially change other attributes.

mv (move or rename files and directories)

- **i**: Interactive mode. The system prompts for confirmation before overwriting an existing file, which can prevent unintended data loss.
- **v**: Verbose mode. Shows detailed output of the move operation, including files being moved.
- **u**: Updates. Moves files only when the source is newer than the destination or when the destination file does not exist.
- **n**: No overwriting. Prevents an existing file from being overwritten.

rm (remove files or directories)

- **i**: Interactive mode. Prompts for confirmation before each deletion, offering protection against accidental data loss.
- **f**: Force. Silently ignores nonexistent files and arguments, and it forces the deletion of files without prompting.

- **r (or -R)**: Recursive. Allows for the deletion of directories and their contents recursively.
- **v**: Verbose mode. Displays detailed output of what is being removed.

touch (change file timestamps or create files)

- **a**: Changes only the access time of the file. This is useful for updating the last read time without altering when the file was last modified.
- **m**: Modifies only the modification time of the file. Handy for changing when the file was last written to, without affecting the read time.
- **c**: No create. Does not create any files that do not already exist. Touch typically creates a new file if the named file does not exist, but with `c`, it will only update existing files.
- **t [TIMESTAMP]**: Sets the access and modification times to the specified TIMESTAMP. The format of TIMESTAMP is usually `[[CC]YY]MMDDhhmm[.ss]`, allowing precise control over the date and time settings.

Navigation: `cd`, `ls`, `pwd`

cd (change directory)

- The `cd` command doesn't have flags in the traditional sense since its primary function is to change the current working directory. However, there are a few important usage patterns:
  - **cd**: Without any arguments, it changes the directory to the user's home directory.
  - **cd [directory]**: Changes the current directory to the specified directory.
  - **cd ..**: Moves the working directory up one level (to the parent directory).
  - **cd -**: Changes the directory to the previous working directory, allowing quick toggling between two directories.

ls (list directory contents)

- **l**: Long listing format. Shows detailed information about files and directories, including permissions, number of links, owner, group, size, and timestamp.

- **a**: Includes entries that start with a dot (.) in the listing. By default, these "hidden" files are not shown.
- **h**: When used with `l`, it displays file sizes in a human-readable format (e.g., KB, MB).
- **R**: Recursively lists subdirectories encountered.

pwd (print working directory)

- The `pwd` command prints the full pathname of the current working directory. It's straightforward and does not have multiple options

Directory Operations: `mkdir`, `rmdir`

mkdir (make directories)

- **p**: Allows the creation of nested directories in a single command. If the directories already exist, no error is reported. This option is particularly useful for setting up a new directory structure that may have multiple levels.
- **v**: Verbose mode. The command prints a message for each directory it creates, giving you immediate feedback on what's being done.
- **m [MODE]**: Sets the file mode (permissions) for the new directories. This option allows you to specify the permissions for the newly created directories at the time of creation, using the same syntax as the `chmod` command. For example, `mkdir -m 755 dirname` creates a directory with read, write, and execute permissions for the owner, and read and execute permissions for the group and others.

rmdir (remove empty directories)

- **p**: Removes directory and its parent directories if they are empty. This option is handy for cleaning up a nested directory structure without leaving empty parent directories.
- **v**: Verbose mode. Like with `mkdir`, this option provides immediate feedback by printing a message for each directory that is removed.

Additional Notes:

- **rmdir** is strict about only removing empty directories. If a directory contains files or subdirectories, `rmdir` will refuse to remove it. This behavior ensures

that you don't accidentally delete files. To remove non-empty directories and their contents recursively, you would typically use `rm -r` instead.

- **mkdir**, with its `p` option, complements `rmdir -p` by allowing the creation of complex directory trees in one step, which can then be carefully dismantled with `rmdir -p` as needed, ensuring no empty directories are left behind.

Text Manipulation: `cat`, `grep`, `sed`

cat (concatenate and display files)

- Used to display the content of files to the standard output. It can also concatenate multiple files into one output stream.
- Common flags include:
  - **n**: Number all output lines, which is helpful for viewing files with line numbers.
  - **E**: Displays a dollar sign (`$`) at the end of each line, making end-of-line (EOL) characters visible.

grep (search for patterns in files)

- Searches files or standard input globally for lines matching a specified pattern and prints them.
- Essential flags include:
  - **i**: Ignores case distinctions in both the pattern and the input files, making the search case-insensitive.
  - **v**: Inverts the search, displaying lines that do not match the pattern.
  - **r (or -R)**: Recursively searches files in directories and subdirectories.

sed (stream editor)

- Used for filtering and transforming text in a stream (a file or input from a pipeline).
- Key usages involve:
  - **s/pattern/replacement/**: Substitutes the first occurrence of the pattern in each line with the replacement.

- **i**: Edits files in-place, saving the changes back to the original file without needing to redirect the output.
- **'5d'**: Deletes the 5th line from the input stream, demonstrating sed's ability to not just edit but also selectively remove lines based on pattern matching or line numbers.

Text editors `vim, nano`

Vim

Vim operates in multiple modes, with the primary ones being Normal mode (for navigating and manipulating text) and Insert mode (for typing text).

- **Typing Text**: Press `i` to enter Insert mode, then type your text. Press `Esc` to return to Normal mode.
- **Undo**: In Normal mode, press `u` to undo the last change. Press `Ctrl-r` to redo changes.
- **Save**: From Normal mode, type `:w` then press `Enter` to save changes.
- **Paste**: In Normal mode, position the cursor where you want to paste, then press `p` to paste after the cursor or `P` to paste before.
- **Delete**: In Normal mode, press `x` to delete the character under the cursor, or `dd` to delete the entire line.
- **Save/Exit**: Type `:wq` in Normal mode and press `Enter` to save changes and exit.
- **Exit without Saving**: Type `:q!` in Normal mode and press `Enter` to exit without saving changes.

Nano

Nano operates in a single mode, making it more straightforward for simple editing tasks.

- **Typing Text**: Just type directly into the editor to insert text.
- **Undo**: Press `Alt-U` to undo the last change. Nano may not support redo.
- **Save**: Press `Ctrl-O`, then press `Enter` to save changes.
- **Paste**: If you have text copied in your clipboard, simply right-click in the terminal (if terminal supports it) or use terminal's paste shortcut (often `Ctrl-`

`Shift-V`) to paste.

- **Delete**: Use `Backspace` to delete the character before the cursor, or `Ctrl-K` to cut the entire line (this also copies the line, allowing you to paste it).
- **Save/Exit**: Press `Ctrl-X`, then press `Y` when prompted to save changes, and `Enter` to confirm the filename.
- **Exit without Saving**: Press `Ctrl-X`, then press `N` when prompted to save changes.

NOTICE:
You can change to the editor you are used to by running:

sudo update-alternatives --config editor

Open your
`~/.bashrc` file in a text editor, for example, using Nano itself:

```
edit ~/.bashrc
```

```
export EDITOR='nano'
```

```
source ~/.bashrc
```

System Information: `uname`, `df`, `ps`

df (disk filesystem)

The `df` command displays information about the disk space usage of mounted filesystems.

- **h (human-readable)**: Displays disk space in a human-readable format, making it easier to read and understand (e.g., KB, MB, GB).
- **T (type)**: Includes the type of each filesystem in the output. This can be particularly useful for identifying different types of filesystems mounted on the system, such as ext4, xfs, or nfs.
- **i (inodes)**: Displays inode information instead of block usage. Since every file and directory consumes an inode, this option can help diagnose when

you're running out of inodes, which might not be obvious from the disk space usage alone.

- **-total**: Provides a grand total usage for all filesystems listed. This can give you a quick summary of the overall disk usage status without needing to sum up the usage manually.

ps (process status)

The `ps` command shows information about active processes on a system.

- **e (all processes)**: Displays information about all processes. This flag is useful for getting a comprehensive overview of what's running on the system, not just the processes associated with the current user or terminal.
- **f (full format)**: Provides a full-format listing. This detailed view includes additional information like the UID (User ID), PID (Process ID), PPID (Parent Process ID), and more, offering deeper insights into each process.
- **u [user_name]**: Displays processes for a specific user. You can use this flag to filter the process list by a particular user, which is helpful in multi-user environments or when tracking down a user-specific issue.
- **-forest**: Displays parent-child relationships in a tree format. This visualization can be invaluable for understanding the hierarchy of processes and how they're related to each other.

Typical use cases for uname:

The `uname` command in Unix and Linux operating systems is used to display system information. When run without any options, `uname` shows the operating system name.

- To see the kernel name:

```
uname -s
```

- To get the kernel release:

```
uname -r
```

- To display all available system information:

```
uname -a
```