`sed`, which stands for stream editor, is a powerful command-line tool for text manipulation in Linux. It operates on a line-by-line basis, processing one line at a time and applying specified operations. Here are some commonly used `sed` commands with brief explanations:

1. Substitution (`s/old/new/`):

- The most common `sed` command is the substitution command, which replaces occurrences of a specified pattern (old) with a new string (new).

```bash

sed 's/old/new/' filename

```

- Example: Replace the first occurrence of "apple" with "orange" in a file.

```bash

sed 's/apple/orange/' filename

```

2. Global Substitution (`s/old/new/g`):

- This variation of the substitution command replaces all occurrences of the specified pattern on each line.

```bash

sed 's/old/new/g' filename

```

- Example: Replace all occurrences of "apple" with "orange" in a file.

```bash

sed 's/apple/orange/g' filename

```

3. Delete Lines (`/pattern/d`):

- Delete lines that match a specified pattern.

```bash

sed '/pattern/d' filename

```

- Example: Delete lines containing the word "error" from a file.

```bash

sed '/error/d' filename

```

4. Print Specific Lines (`n; p`):

- Print specific lines by specifying line numbers.

```bash

sed -n '2,5p' filename

```

- Example: Print lines 2 to 5 from a file.

```bash

sed -n '2,5p' filename

```

5. Search and Replace with a Range (`/start/,/end/s/old/new/g`):

- Apply substitution only within a specific range of lines.

```bash

sed '/start/,/end/s/old/new/g' filename

```

- Example: Replace "apple" with "orange" only between lines containing "start" and "end."

```bash

sed '/start/,/end/s/apple/orange/g' filename

```

6. Insert a Line Before (`i\text`):

- Insert a new line before a matching pattern.

```bash

sed '/pattern/i\new\_line' filename

```

- Example: Insert the line "This is a new line" before lines containing "pattern."

```bash

sed '/pattern/i\This is a new line' filename

```

7. Append a Line After (`a\text`):

- Append a new line after a matching pattern.

```bash

sed '/pattern/a\new\_line' filename

```

- Example: Append the line "This is a new line" after lines containing "pattern."

```bash

sed '/pattern/a\This is a new line' filename

```

8. Replace Only on Lines Matching a Pattern (`/pattern/s/old/new/g`):

- Replace occurrences only on lines that match a specific pattern.

```bash

sed '/pattern/s/old/new/g' filename

```

- Example: Replace "apple" with "orange" only on lines containing "pattern."

```bash

sed '/pattern/s/apple/orange/g' filename

```

These are just a few examples of what `sed` can do. The tool is extremely versatile and can be used for more complex text manipulations and scripting in the Linux environment.