The `grep` command in Linux is a powerful and widely used tool for searching and matching patterns within text files. Its name comes from the ed command `g/re/p`, which means "global search for a regular expression and print matching lines." Here are detailed notes on how to use the `grep` command:

Basic Syntax:

```bash

grep [options] pattern [file(s)]

```

- `pattern`: The regular expression or plain text pattern you want to search for.

- `file(s)`: The file or files in which to search for the pattern. If not specified, `grep` reads from standard input (stdin).

Common Options:

- `-i` or `--ignore-case`: Ignore case distinctions in both the pattern and input files.

- `-r` or `--recursive`: Read all files under each directory recursively.

- `-n` or `--line-number`: Prefix each line of output with the 1-based line number within its input file.

- `-v` or `--invert-match`: Invert the sense of matching, select non-matching lines.

- `-c` or `--count`: Suppress normal output; instead, print a count of matching lines for each input file.

- `-l` or `--files-with-matches`: Suppress normal output; instead, print the name of each input file with at least one matching line.

- `-w` or `--word-regexp`: Select only those lines containing matches that form whole words.

- `-A NUM` or `--after-context=NUM`: Print NUM lines of trailing context after matching lines.

- `-B NUM` or `--before-context=NUM`: Print NUM lines of leading context before matching lines.

- `-C NUM` or `--context=NUM` or `-NUM`: Print NUM lines of output context.

Examples:

1. Search for a pattern in a file:

```bash

grep "pattern" filename

```

2. Search for a pattern case-insensitively:

```bash

grep -i "pattern" filename

```

3. Search recursively in all files under a directory:

```bash

grep -r "pattern" directory

```

4. Display line numbers for matching lines:

```bash

grep -n "pattern" filename

```

5. Invert the match, i.e., display lines that do not contain the pattern:

```bash

grep -v "pattern" filename

```

6. Count the number of lines that match the pattern:

```bash

grep -c "pattern" filename

```

7. Display only the names of files with matching lines:

```bash

grep -l "pattern" \*

```

8. Search for whole words:

```bash

grep -w "word" filename

```

9. Display lines with context (lines before and after the match):

```bash

grep -A 2 -B 2 "pattern" filename

```

10. Combine multiple options:

```bash

grep -i -n -r "pattern" directory

```

Or, using short options:

```bash

grep -inr "pattern" directory

```

These are just some basic examples, and `grep` has many more options and use cases. It's a powerful tool for text processing and searching in the Linux command line.