A Guide to the Vim Editor: From Basic to Intermediate

Welcome to Vim! It's a highly configurable text editor built to enable efficient text editing. It is an improved version of the vi editor distributed with most UNIX systems. Its power lies in its "modal" design and its extensive set of keyboard commands.

1. The Most Important Concept: Modal Editing

Unlike most editors, where typing letters immediately inserts text, Vim operates in several **modes**. The two you'll use constantly are:

- **Normal Mode:** This is the default mode when you open Vim. In this mode, keys on the keyboard are shortcuts for commands (like moving the cursor, deleting text, copying, pasting, etc.). You don't insert text directly. To return to Normal mode from any other mode, just press the **Esc** key.
- **Insert Mode:** In this mode, Vim behaves like a regular editor like Notepad or VS Code. Whatever you type is inserted into the file.

This modal system is the key to Vim's efficiency. You spend most of your time in Normal mode, navigating and manipulating text, and only switch to Insert mode when you need to type new content.

How to open a file: vim filename.txt

2. The Absolute Basics: Getting In and Out

- **Exit Vim:** The most famous beginner problem!
  1. Make sure you are in **Normal Mode** (press Esc).
  2. Type :q and press Enter. This means "quit".
  3. If you have unsaved changes, Vim will stop you.
     - To quit **without saving**: :q!
     - To **write** (save) and quit: :wq
     - To save and quit (alternative): :x (only writes if there are changes)

3. Basic Navigation (Normal Mode)

Forget the mouse. In Vim, you navigate with your keyboard. While you *can* use the arrow keys, the "home row" keys are faster once you get used to them.

- h: move left
- j: move down
- k: move up

- l: move right

Moving within a line:

- 0: move to the beginning of the line
- ^: move to the first non-whitespace character of the line
- $: move to the end of the line

Moving between words:

- w: move to the start of the next word
- b: move to the start of the previous word
- e: move to the end of the current word

Moving around the file:

- gg: go to the very first line of the file
- G: go to the very last line of the file
- [line_number]G: go to a specific line number (e.g., 15G goes to line 15)

4. Basic Editing (Switching to Insert Mode)

To start typing, you need to switch from Normal mode to Insert mode.

- i: **i**nsert text before the cursor
- a: **a**ppend text after the cursor
- I: **I**nsert text at the beginning of the current line
- A: **A**ppend text at the end of the current line
- o: open a new line **o**ver (below) the current line and enter Insert mode
- O: open a new line **O**ver (above) the current line and enter Insert mode

Remember: After you're done typing, press Esc to go back to Normal mode.

5. Intermediate Editing: The Verbs and Nouns

Vim's real power comes from combining "operators" (verbs) with "motions" (nouns).
Verbs: d (delete), c (change), y (yank/copy)
Nouns (Motions): w (word), s (sentence), p (paragraph), $ (end of line), etc.
Structure: [verb][motion]

Deleting Text (d)

- dw: **d**elete **w**ord (from the cursor to the start of the next word)
- db: **d**elete **b**ackward to the start of the word
- d$: **d**elete to the end of the **l**ine

- dd: **d**elete the entire current **l**ine

Changing Text (c)

"Change" is like delete, but it leaves you in Insert mode.

- cw: **c**hange **w**ord (deletes the word and puts you in Insert mode)
- c$: **c**hange to the end of the **l**ine
- cc: **c**hange the entire **l**ine

Yanking (Copying) and Pasting (y and p)

- yw: **y**ank **w**ord
- yy: **y**ank the entire **l**ine
- p: **p**aste the copied/deleted text *after* the cursor
- P: **p**aste *before* the cursor

Combining with Numbers: You can prefix commands with a number to repeat them.

- 5j: move down 5 lines
- d2w: **d**elete **2 w**ords
- 3yy: **y**ank **3 l**ines
- 2dd: delete 2 lines

Undoing and Redoing

- u: **u**ndo the last change
- Ctrl + r: **r**edo the last change

6. Intermediate Techniques

Search and Replace

- /pattern: search forward for pattern
  - Press n to go to the next occurrence.
  - Press N to go to the previous occurrence.
- ?pattern: search backward for pattern
- **:%s/old/new/g**: The classic search and replace.
  - : enters Command-line mode.
  - % means "on every line in the file".
  - s stands for "substitute".
  - old is the text to find.
  - new is the text to replace it with.
  - /g means "global", i.e., replace every occurrence on the line, not just the first one.

- To confirm each change, add c: :%s/old/new/gc

Visual Mode

Visual mode allows you to select text before running a command on it.

- v: enter character-wise **v**isual mode. Move the cursor to select text.
- V: enter line-wise **V**isual mode. Selects whole lines.
- Ctrl + v: enter **v**isual block mode. Selects rectangular blocks of text.

Once text is selected, you can use an operator on it, like d (delete), y (yank), or c (change).

Use Case: Prepending text to a block of lines.

1. Go to the start of the block.
2. Press Ctrl + v to enter block mode.
3. Use j to select down across all the lines.
4. Press I (capital i).
5. Type the text you want to insert (e.g., //).
6. Press Esc. The text will be inserted at the beginning of every selected line.

Working with Multiple Files

- :e filename: **e**dit a new file.
- :ls: **ls**t all open buffers (files in memory).
- :bn: go to the **n**ext **b**uffer.
- :bp: go to the **p**revious **b**uffer.
- :sp filename: **sp**lit the window horizontally to open a new file.
- :vsp filename: **v**ertically **sp**lit the window.

You can navigate between split windows using Ctrl + w followed by a direction key (h, j, k, l).

Example Problems & Use Cases

1. **Problem:** You have a variable named temp_variable and you want to rename it to user_id everywhere in the file.
   - **Solution:** In Normal Mode, type :%s/temp_variable/user_id/g and press Enter.
2. **Problem:** You want to delete the next 5 lines of code.
   - **Solution:** In Normal Mode, type 5dd.
3. **Problem:** You need to copy a function (say, 15 lines long) and paste it below.
   - **Solution:** Move your cursor to the first line of the function. Type 15yy. Move

    your cursor to where you want to paste it. Type p.
4. **Problem:** You have a list of items and you want to comment them all out (in C, add // at the start).
    - **Solution:** Use the visual block mode trick described above. Ctrl+v, select the lines, press I, type // , and press Esc.

This guide covers the core commands you need to be productive in Vim. The key is practice. Try to use Vim for all your text editing for a week, and these commands will start to become muscle memory. Happy Vimming!