Presentatie vi en vim

Table of Contents

Inleiding

Vi is een teksteditor.

ed

1969

Ken Thompson

line **ed**itor

vi

1977

Bill Joy

visual editor

vim

1992

Bram Molenaar, nl, 1961 – 2023

vi **im**proved

Neovim - nvim

2015

Neovim is a Vim-based text editor engineered for extensibility and usability, to encourage new applications and contributions

LazyVim - nvim

2023

IDE

ed

1969

Ken Thompson

line editor

ed example 1

https://www.geeksforgeeks.org/linux-unix/ed-command-in-linux-with-examples/

1. Launch ed:

```
ed
```

2. Type a few lines, entering insert mode along the way:

```
a
Hello world
This is line 2
.
```

3. Print all lines:

```
,p
```

4. Save the file:

```
w example.txt
```

5. Exit ed:

```
q
```

ed example 2

1: Start ed with the example.txt file:

ed example.txt

2: Print all lines

,p

3: Quit ed:

q

ed example 3

https://www.redhat.com/en/blog/introduction-ed-editor

vi

1977

Bill Joy

visual editor

Bill Joy is quoted in an interview on his process of writing *ex* and *vi*:

> "It took a long time. It was really hard to do because you've got to remember that I was trying to make it usable over a 300 baud modem. That's also the reason you have all these funny commands. It just barely worked to use a screen editor over a modem. It was just barely fast enough. A 1200 baud modem was an upgrade. 1200 baud now is pretty slow. 9600 baud is faster than you can read. 1200 baud is way slower. So the editor was optimized so that you could edit and feel productive when it was painting slower than you could think. Now that computers are so much faster than you can think, nobody understands this anymore."

Geen pijltjes toetsen, home, end, insert, delete

Geen muis

vim

1992

Bram Molenaar, nl, 1961 – 2023

vi i**m**proved

https://www.vim.org/

Vim is 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.

$ vim

Waarom vim?

https://medium.com/quick-programming/why-you-should-finally-learn-vim-yes-even-in-2025-bf5ae142bc7d

- **Vim is Stupid Fast**
- **Working in Vim is Fun**
- **It’s Always There**
- **You’ll Feel Like a Wizard**
- **You Don’t Need to Master It All For the Benefits**

What is Vim and Why use Vim?

https://medium.com/@fay_jai/what-is-vim-and-why-use-vim-54c67ce3c18e

So what makes Vim special?

Compared to other text editors, 2 aspects make Vim stand out:

1 - Efficiency

Core to Vim's belief is that most people spend more time **editing** existing text than **writing** new text. This is especially true for software engineers who are often tasked with enhancing and maintaining existing code.

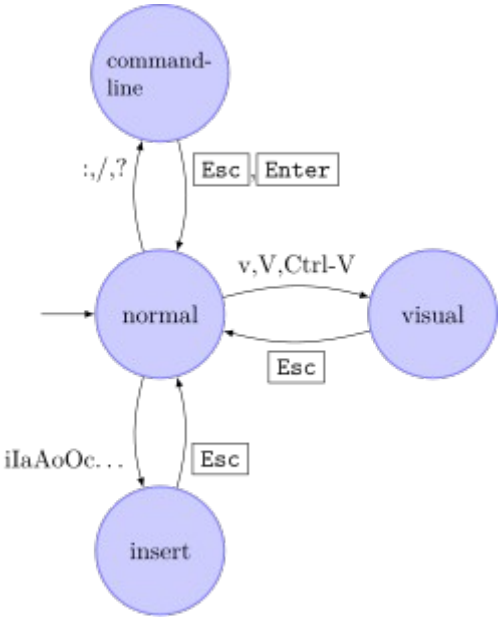
2 – Ubiquity (overal aanwezig)

This is also a pretty cool aspect of Vim, which is that it's everywhere. It's available on basically every major platform you can think of. Whether you're using a Mac, Windows, or some Linux distribution, Vim's got you covered.

https://www.freecodecamp.org/news/vim-editor-modes-explained/

Because Vim is focused on changing existing code just as much as writing new code, it is split into several modes that each have different purposes.

https://en.wikipedia.org/wiki/Vim_%28text_editor%29



Vim modes:

Mode	From	key(s)	comment
Normal mode(or command mode)	any of the other modes	ESC /* Once, or multiple times. */	Normal mode is the central mode from which you can switch to other modes.
Insert mode	Normal mode, …?	i // Insert text at the cursor a // Insert text directly behind the cursor I // Go to the start of the current line and switch to insert mode. A // Go to the end of the current line and switch to insert mode o // Open a line below the cursor. O // Open a line above the cursor.	Vim displays an — INSERT — message at the bottom left corner of the screen.
Visual mode	Normal mode, …?	v	Visual mode lets you select areas character by character, line by line, or block by block Vim displays an 'VISUAL' message at the bottom left corner of the screen.
Command line mode	Normal mode, …?	:	
Replace Mode		R	

Normal Mode

By default, Vim starts in “normal” mode. Normal mode can be accessed from other modes by pressing `Esc` or `<C-[>`.

In Normal mode key presses don’t work as one would expect. That is, they don’t insert text into the document; instead, certain key presses can:

Move the cursor

- **h** move one character left
- **j** move one row down
- **k** move one row up
- **l** move one character right

As many vim commands, row movement can be prefixed by a number to move s everal lines at a time:

- **4j** move 4 rows down
- **6k** move 6 rows up

Basic word movements:

- **w** move to beginning of next word
- **b** move to previous beginning of word
- **e** move to end of word
- **W** move to beginning of next word after a whitespace
- **B** move to beginning of previous word before a whitespace
- **E** move to end of word before a whitespace

Beginning/End of line movement:

- **0** move to the beginning of the line
- **$** move to the end of the line

Manipulate text

Enter other modes

Normal mode is where one should spend most of their time while using Vim. Remember, this is what makes Vim different.

In normal mode, there are multiple ways to move around an open file. In addition to using the cursor keys to move around, you can use `h` (left), `j` (down), `k` (up), and `l` (right) to move as well. This particularly helps touch typists who don’t like leaving the home row when making changes.

You can also make changes to single characters in normal mode. For example, to replace a single character, move your cursor over it and press `r`, and then the character you want to replace it with. Similarly, you can delete single characters by moving your cursor over it and pressing `x`.

To perform an undo, press `u` in normal mode. This undoes changes up to the last time you were in normal mode. If you want to redo (*i.e.*, undo your undo) press `Ctrl+r` in normal mode.

Insert Mode

This is the second most used mode, and will be the most familiar behavior to most people. Once in insert mode, typing inserts characters just like a regular text editor. You can enter it by using an insert command from normal mode.

Insert commands include:

- `i` for '**i**nsert', this immediately switches vim to insert mode
- `a` for '**a**ppend', this moves the cursor after the current character and enters insert mode
- `o` inserts a new line below the current line and enters insert mode on the new line

These commands have an uppercase variety too:

- `I` moves the cursor to the beginning of the line and enters insert mode
- `A` moves the cursor to the end of the line and enters insert mode
- `O` inserts a new line above the current one and enters insert mode on the new line

There are so many more ways of inserting text in Vim that can't be listed here but these are the simplest. Also, beware of staying in insert mode for too long; Vim is not designed to be used in insert mode all the time.

To leave insert mode and return to normal mode, press `Esc` or `<C-[>`

Visual Mode

Visual mode is used to make selections of text, similar to how clicking and dragging with a mouse behaves. Selecting text allows commands to apply only to the selection, such as copying, deleting, replacing, and so on.

To make a text selection:

- Press `v` to enter visual mode, this will also mark a starting selection point
- Move the cursor to the desired end selection point; vim will provide a visual highlight of the text selection

Visual mode also has the following variants:

- `V` to enter visual line mode, this will make text selections by line
- `<C-V>` to enter visual block mode, this will make text selections by blocks; moving the cursor will make rectangle selections of the text

To leave visual mode and return to normal mode, press `Esc` or `<C-[>`.

The visual mode actually has multiple subtypes: *visual*, *block-visual* and *linewise-visual*

- *visual*: like described above. Enter by pressing `v`
- *block-visual*: select any rectangular region. Enter by pressing `<ctrl>+v`
- *linewise-visual*: always select full lines. Enter by pressing `<shift>+v`

Command Mode

Command mode has a wide variety of commands and can do things that normal mode can't do as easily. To enter command mode type ':' from normal mode and then type your command which should appear at the bottom of the window. For example, to do a global find and replace type `:%s/foo/bar/g` to replace all 'foo' with 'bar'

- `:` Enters command mode
- `%` Means across all lines
- `s` Means substitute
- `/foo` is regex to find things to replace
- `/bar/` is regex to replace things with
- `/g` means global, otherwise it would only execute once per line

Vim has a number of other methods that you can read about in the help documentation, `:h` or `:help`.

Replace Mode

Replace mode allows you replace existing text by directly typing over it. Before entering this mode, get into normal mode and put your cursor on top of the first character that you want to replace. Then press 'R' (capital R) to enter replace mode. Now whatever you type will replace the existing text. The cursor automatically moves to the next character just like in insert mode. The only difference is that every character you type will replace the existing one.

How-to

Cut and paste:

1. Position the cursor where you want to begin cutting.
2. Press v (or upper case V if you want to cut whole lines). /* Enter Visual mode? */
3. Move the cursor to the end of what you want to cut.
4. Press d (delete).
5. Move to where you would like to paste.
6. Press P to paste before the cursor, or p to paste after.

Copy and paste

can be performed with the same steps, only pressing y instead of d in step 4.

The name of the mark used is related to the operation (d:delete or y:yank).

1. Position the cursor where you want to begin cutting.
2. Press v (or upper case V if you want to cut whole lines). /* Enter Visual mode? */
3. Move the cursor to the end of what you want to cut.
4. Press y (yank).
5. Move to where you would like to paste.
6. Press P to paste before the cursor, or p to paste after.

Zoeken

https://stackoverflow.com/questions/2287440/how-to-do-case-insensitive-search-in-vim

To do case insensitive matching:

You can use the `\c` escape sequence anywhere in the pattern. For example:

`/\ccopyright` or `/copyright\c` or even `/copyri\cght`

To do the inverse (case *sensitive* matching), use `\C` (capital C) instead.

Hex editor

https://vi.stackexchange.com/questions/2232/how-can-i-use-vim-as-a-hex-editor

Use the xxd command to transform a file in Vim to hex representation:

```
:%!xxd
```

: enters command-line mode, % matches whole file as a range, ! filters that range through an external command, xxd is that external shell command

Giving an output like this, this is split into octet count/line (octets per line may be changed with parameter -c on xxd command), hex representation, and text representation:

Once you make the changes (in the hex part), you can go back to text with -r command on xxd:

```
:%!xxd -r
```

Unicode Characters

https://jdhao.github.io/2020/10/07/nvim_insert_unicode_char/

Learning vim

$ vimtutor

Neovim

https://neovim.io/

2015

Neovim is a Vim-based text editor engineered for extensibility and usability, to encourage new applications and contributions

apt install neovim

...

$ nvim

Start nvim, skip user config and plugins:

$ nvim –clean

Lazy vim

2023

https://www.lazyvim.org/

Features

- 🔥 Transform your Neovim into a full-fledged IDE
- 💤 Easily customize and extend your config with lazy.nvim
- 🚀 Blazingly fast
- 🧹 Sane default settings for options, autocmds, and keymaps
- 📦 Comes with a wealth of plugins pre-configured and ready to use

Config:

$ nvim -u ~/.config/nvim/lua/config/lazy.lua

Start nvim:

$ nvim

command

:LazyHealth

command:

:LazyExtras

x