HPC-Café: Effective editing with Vim

Jan Eitzinger, 09.11.2021
Some history 😊

- Where it all started: **ed** line editor (1969, Ken Thompson, Bell Labs)
  
  *the most user-hostile editor ever created*

- **ex** line editor (1976, Bill Joy, 1BSD), there is still an **ex** mode in vim. **ex** included a visual mode that was the birth of **vi**.

- Vi Improved (**Vim**) (1991, Bram Moolenar, Amiga)

  *Stevie (ST Editor for VI Enthusiasts) (1987, Tim Thompson, Atari ST)*

**Vim is by far the most popular vi clone!**
What this talk is not about

- This is **no introduction** to vim! Not for beginners and not for advanced users.
- There will be **no exhaustive list of commands** and tips and tricks! Well, maybe a few.

This talk introduces the **mindset** and **habits** that **make vim a powerful tool**!

And don’t worry. This will be a short talk! I hope …
Vim resources

- https://vimawesome.com  Repository for vim plugins
- http://vimcasts.org  Vim screencasts
- http://vimcolors.com  Repository for vim color schemes
- https://vim.fandom.com/wiki/Vim_Tips_Wiki  Vim tips repository

- Seven habits of effective text editing (with vim) by the master himself
  https://moolenaar.net/habits.html
All those plugins and tools …

- Use a **plugin manager**! I use vim-plug:
  
  https://github.com/junegunn/vim-plug

- **Fuzzy find** everything!
  - General-purpose command-line fuzzy finder
    
    https://github.com/junegunn/fzf
  
  - Vim plugin to integrate fzf
    
    https://github.com/junegunn/fzf.vim
  
  - Fast recursive regex grep tool
    
    https://github.com/BurntSushi/ripgrep
  
  - Simple and fast find alternative
    
    https://github.com/sharkdp/fd
And more of it

- Use **completion**, a lot! And all with just `<TAB>`
  https://github.com/lifepillar/vim-mucomplete

- Use a **language server protocol** enabled **linter**
  https://github.com/dense-analysis/ale

- If you do not want to spent time fiddling (**Defaults everyone can agree on**)  
  https://github.com/tpope/vim-sensible

- You ask for my (current) vim configuration? Here you go

Honorable mentions: **vim-easymotion**, **autopairs**, **vim-unimpaired**
Learning how to use vim is a constant process

- You should not and cannot learn all vim commands!
- Learn only those commands that make your editing more effective.

**Basic steps**

1. While editing, look for actions you spend quite a bit of time on.
2. Find an editor command that will do the action quicker.
3. Train using the command. Do this until it becomes a habit.

- You must constantly learn and improve!
- Don’t overdo it!
- Use the excellent built-in docs!
Some preliminary info for the demo

- This demo uses vim in my personal configuration!  
  *I will mark keystrokes that are not standard vim.*

- Some specific adoptions in my configuration:
  - `<space>` is the leader key
  - `<tab>` maps to `<C-d>`: scroll down half a page
  - `<S-tab>` maps to `<C-u>`: scroll up half a page
  - `<leader>a` close quickfix windows
  - `<leader>q` close current buffer

Vim’s power comes from adopting it to you and your requirements.

**Make vim YOUR tool!**
Part 1: The vim way

Vim is optimized for repetition! Make actions and movements **repeatable**!

Commands used:

- `<h>`  `<j>`  `<k>`  `<l>`  Move cursor one character left, down, up, right
- `<.`>  Repeat last action (aka micro macro)
- `<A>`  Compound command for `<$a>`
- `<f{char}>`  Search forward for `{char}`
- `<;>`  Repeat last movement
- `<s>`  Delete character under cursor and enter Insert mode

**Optimal case:** One keystroke to move, one keystroke to execute!
Part 2: Act, Repeat, Reverse

- The act and repeat routine often leads to **trigger-happy** editing
- Vim makes it easy to **back out** where you **went to far**!

**Commands used:**

- `<*>` Search forward for word under cursor
- `<cw>` Change word under cursor
- `<n>` Cycle forward through matches
- `<.` Repeat action
- `<u>` Reverse previous action
- `<C-r>` Reverse last undo
- `<,>` Reverse last search in line
Part 3: From great to awesome - On the fly macros

Commands used:

- `<qa>`  
  Start recording macro and store to register a

- `<q>`  
  Stop recording macro

- `<@a>`  
  Playback macro in register a

- `<@@>`  
  Playback last executed macro

Tips for robust macros

- Normalize cursor position
- Strike target with repeatable motion
- One step back can mean three steps forward

I map `<@q>` to `<Q>` in order to do the more memorable `<qq>`, `<q>`, `<Q>`
Part 4: Getting around within a file

- **Keep your hands on the home row!** There are no arrow keys 😊
- Navigate by searches and direct jumps

Commands used:

- `<f{char}>` Search forward in line for `{char}`
- `</{pattern}>` Search `{pattern}` forward in file
- `<gg>` Jump to start of file
- `<G>` Jump to end of file
- `<zz>` Center cursor position in window
- `<{num}G>` Jump to line
- `<TAB>` Scroll down half a page (default `<C-d>`)  
- `<S-TAB>` Scroll up half a page (default `<C-u>`)
Part 4: Getting around within a file cont.

Commands used:

\([>, <]\)  \(\langle\%\rangle\)  
Jump to next, previous start of function block  
Jump to matching braces

Using vim-easymotion plugin:

\(<\text{leader}\text{-s}>\)  
Mark locations matching two characters

Use a tags browser! I use https://github.com/preservim/tagbar.
Part 5: Getting around within a project

- The Nerdtree plugin provides a nice file browser.
- Index your project using ctags!

Commands used:

\(<c-]\) \> Jump to definition of symbol under cursor
\(<c-o]\) \> Jump back to previous location
\(<leader>e]\) \> FZF search files
\(<leader>b]\) \> FZF search open buffers
\(<leader>g]\) \> Recursive pattern search with ripgrep
\(<leader>T]\) \> Search tags
Part 6: Manage your screen

Vim has support for tabs. A tab may hold multiple windows. I do not use tabs, ever.

Commands used:

`:sp{lit}>` Split screen horizontally
`:vsp{lit}>` Split screen vertically
`<C>h|j|k|l` Move cursor to window on the left, bottom, top, right
`<arrow keys>` Resize windows
`<leader>a` Close all Quickfix/Location list windows
Part 7: Don’t be religious about this

- In some cases copy and paste with the mouse is the fastest option
- Vim supports all styles of editing, at least in the GUI versions
- Other editors (VSCode) often offer a vim editing mode
Other awesome features I did not talk about

- Visual Mode and Visual-Block Mode
- Registers for Copy/Paste and Macros
- Command mode
- Patterns and Search/Substitute commands
- Build integration and Quickfix list
- Built-in spell checker
- Other plugins I often use:
  - `vim-fugitive`
  - `vim-interestingwords`
  - `vim-commentary`
- For the looks: `lightline.vim` and `gruvbox` color scheme