SpaceVim英文文档

目　录

致谢

intro

SpaceVim

A community-driven vim distribution

Home | **About** | **Documentation** | **Development** | **Community** | **Sponsors**

About

Version

SpaceVim is a community-driven vim distribution that supports vim and Neovim. SpaceVim manages collections of plugins in layers. Layers make it easy for you, the user, to enable a new language or feature by grouping all the related plugins together. It got inspired by spacemacs.

Goals

- Provide a consistent user experience across platforms.
- Provide better default layer for different languages.

Principles

- Do not regress from origin
- Decide outcomes by weighing cost and benefit
- prefer usability over tradition if the benefits are overwhelming
- Give usability a chance

Credits & Thanks

This project exists thanks to all the people who have contributed to SpaceVim:

- @Gabirel and his Hack-SpaceVim
- @everettjf and his SpaceVimTutorial
- vimdoc generate doc file for SpaceVim
- Rafael Bodill and his vim-config
- Bailey Ling and his dotvim
- authors of all the plugins used in SpaceVim.

> 原文: *https://spacevim.org/about/*

Core Pillars

Core Pillars

Four core pillars: Mnemonic, Discoverable, Consistent and “Crowd-Configured”.

If any of these core pillars is violated open an issue and we’ll try our best to fix it.

Mnemonic

Key bindings are organized using mnemonic prefixes like b for buffer, p for project, s for search, h for help, etc…

Discoverable

Innovative real-time display of available key bindings. Simple query system to quickly find available layers, packages, and more.

Consistent

Similar functionalities have the same key binding everywhere thanks to a clearly defined set of conventions. Documentation is mandatory for any layer that ships with SpaceVim.

Crowd-Configured

Community-driven configuration provides curated packages tuned by power users and bugs are fixed quickly.

Highlighted features

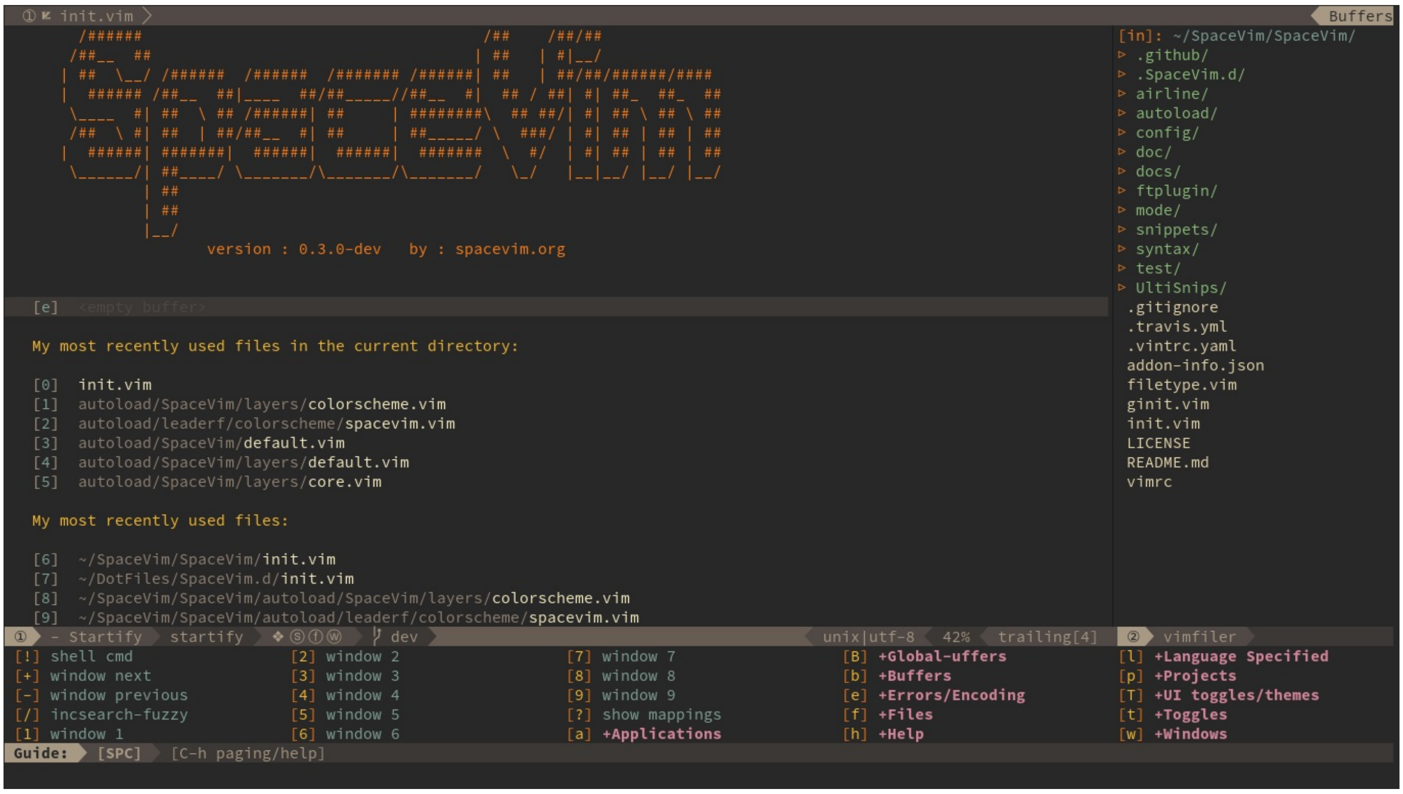
Highlighted features

- **Great documentation:** access documentation in SpaceVim with :h SpaceVim.
- **Minimalistic and nice graphical UI:** you’ll love the awesome UI and its useful features.
- **Keep your fingers on the home row:** for quicker editing with support for QWERTY and BEPO layouts.
- **Mnemonic key bindings:** commands have mnemonic prefixes like [Window] for all the window and buffer commands or [Unite] for the unite work flow commands.
- **Fast boot time:** Lazy-load 90% of plugins with [dein.vim]
- **Lower the risk of RSI:** by heavily using the space bar instead of modifiers.
- **Batteries included:** discover hundreds of ready-to-use packages nicely organised in configuration layers following a set of conventions.
- **Neovim centric:** Dark powered mode of SpaceVim

Screenshots

Screenshots

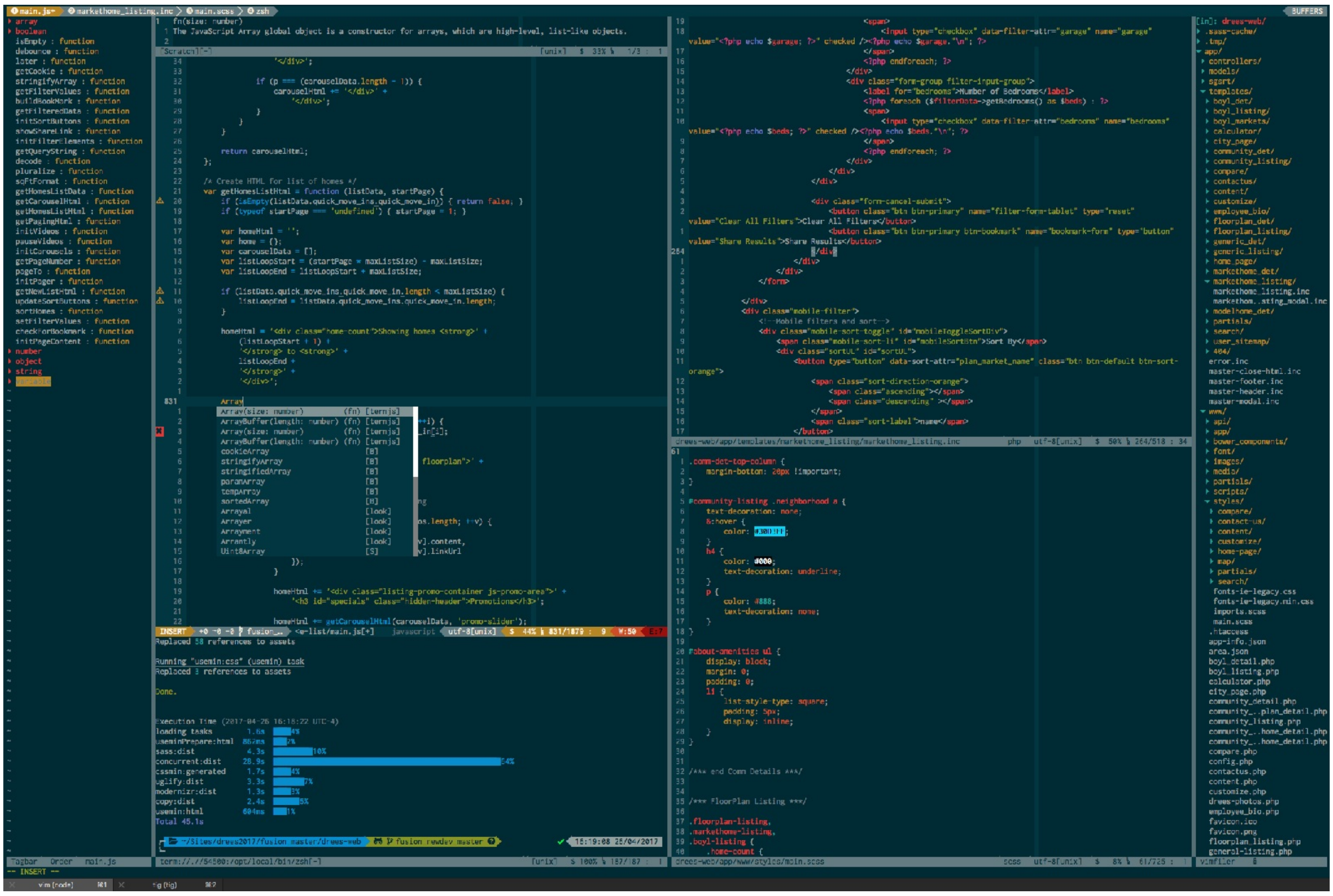
welcome page



```
 ① init.vim                                                                                              Buffers
       /######                                     /##     /##/##                 [in]: ~/SpaceVim/SpaceVim/
      /##__  ##                                   | ##    | #|__/                 ▷ .github/
     | ##  \__/ /######  /######  /####### /######| ##    | ##/##/######/####     ▷ .SpaceVim.d/
     |  ###### /##__  ##|____  ##/##_____//##__  #|  ## / ##| #| ##_  ##_  ##     ▷ airline/
      \____  #| ##  \ ## /######| ##     | ########\  ## ##/| #| ## \ ## \ ##     ▷ autoload/
      /##  \ #| ##  | ##/##__  #| ##     | ##_____/ \  ###/ | #| ## | ## | ##     ▷ config/
     |  ######| #######|  ######|  ######|  #######  \  #/  | #| ## | ## | ##     ▷ doc/
      \______/| ##____/ \_______/\_______/\_______/   \_/   |__|__/ |__/ |__/     ▷ docs/
              | ##                                                                ▷ ftplugin/
              | ##                                                                ▷ mode/
              |__/                                                                ▷ snippets/
                    version : 0.3.0-dev   by : spacevim.org                       ▷ syntax/
                                                                                  ▷ test/
   [e]  <empty buffer>                                                            ▷ UltiSnips/
                                                                                   .gitignore
   My most recently used files in the current directory:                           .travis.yml
                                                                                   .vintrc.yaml
   [0]  init.vim                                                                   addon-info.json
   [1]  autoload/SpaceVim/layers/colorscheme.vim                                   filetype.vim
   [2]  autoload/leaderf/colorscheme/spacevim.vim                                  ginit.vim
   [3]  autoload/SpaceVim/default.vim                                              init.vim
   [4]  autoload/SpaceVim/layers/default.vim                                       LICENSE
   [5]  autoload/SpaceVim/layers/core.vim                                          README.md
                                                                                   vimrc
   My most recently used files:

   [6]  ~/SpaceVim/SpaceVim/init.vim
   [7]  ~/DotFiles/SpaceVim.d/init.vim
   [8]  ~/SpaceVim/SpaceVim/autoload/SpaceVim/layers/colorscheme.vim
   [9]  ~/SpaceVim/SpaceVim/autoload/leaderf/colorscheme/spacevim.vim
 ① - Startify  startify  ❖ ⓢⓕⓦ  dev                      unix|utf-8  42%  trailing[4]  ② vimfiler
 [!] shell cmd            [2] window 2            [7] window 7            [B] +Global-uffers      [l] +Language Specified
 [+] window next          [3] window 3            [8] window 8            [b] +Buffers            [p] +Projects
 [-] window previous      [4] window 4            [9] window 9            [e] +Errors/Encoding    [T] +UI toggles/themes
 [/] incsearch-fuzzy      [5] window 5            [?] show mappings       [f] +Files              [t] +Toggles
 [1] window 1             [6] window 6            [a] +Applications       [h] +Help               [w] +Windows
 Guide:  [SPC]  [C-h paging/help]
```

working flow



Neovim on iTerm2 using the SpaceVim color scheme *base16-solarized-dark*

Depicts a common frontend development scenario with JavaScript (jQuery), SASS, and PHP buffers.

Non-code buffers show a Neovim terminal, a TagBar window, a Vimfiler window and a TernJS definition window.

to get more screenshots, see: [issue #415]

Who can benefit from this?

Who can benefit from this?

- **Elementary** vim users.
- Vim users pursuing a beautiful appearance.
- Vim users wanting to lower the risk of RSI.
- Vim users wanting to learn a different way to edit files.
- Vim users wanting a simple but deep configuration system.

Update and Rollback

Update and Rollback

Update SpaceVim itself

There are several methods of updating the core files of SpaceVim. It is recommended to update the packages first; see the next section.

Automatic Updates

NOTE: By default, this feature is disabled, It will slow down the startup of vim/neovim. If you like this feature, add `let g:spacevim_automatic_update = 1` to your custom configuration file.

SpaceVim will automatically check for a new version every startup. You must restart Vim after updating.

Updating from the SpaceVim Buffer

Use `:SPUpdate SpaceVim` in SpaceVim buffer, This command will open a buffer to show the process of updating.

Updating Manually with git

To update manually close Vim and update the git repository:

`git -C ~/.SpaceVim pull` .

Update plugins

Use `:SPUpdate` command will update all the plugins and SpaceVim itself. after `:SPUpdate` , you can assign plugins need to be updated. Use `Tab` to complete plugin names after `:SPUpdate` .

Get SpaceVim log

Use `:SPDebugInfo!` command will display the log of SpaceVim. You also can use `SPC h I` to open a buffer with issue template.

Custom Configuration

Custom Configuration

The very first time SpaceVim starts up, it will ask you to choose a mode,then create the `SpaceVim.d/init.toml` in your `HOME` directory. All Userconfiguration can be stored in your `~/.SpaceVim.d` directory.

`~/.SpaceVim.d/` will be added to `&runtimepath` of vim.

It is also possible to override the location of `~/.SpaceVim.d/` using the environmentvariable `SPACEVIMDIR` . Of course you can also use symlinks to change the location ofthis directory.

SpaceVim also support local config file for project, the init file is `.SpaceVim.d/init.toml` in the root of your project. `.SpaceVim.d/` will also be added into runtimepath.

All SpaceVim options can be found in `:h SpaceVim-config` , the key is same asthe option name(just remove `g:spacevim_` prefix).

Comprehensive documentation is available for each layer by `:h SpaceVim` .

if you want to add custom `SPC` prefix key bindings, you can add this to SpaceVim configuration file, **be sure** the key bindings is not used in SpaceVim.

```
call SpaceVim#custom#SPCGroupName(['G'], '+TestGroup')
call SpaceVim#custom#SPC('nore', ['G', 't'], 'echom 1', 'echomessage 1', 1)
```

Bootstrap Functions

SpaceVim provides two kinds of bootstrap functions for custom configurations and key bindings, namely `bootstrap_before` and `bootstrap_after` . To enable it you need to add `bootstrap_before = "myspacevim#before"` or `bootstrap_after = "myspacevim#after"` to `[options]` section in file `.SpaceVim.d/init.toml` . The difference is that these two functions will be called before or after the loading of SpaceVim’s main scripts as they named.

The bootstrap functions should be placed to the `autoload` directory in `runtimepath` , please refer to `:h autoload-functions` for further instructions. In our case, create file `.SpaceVim.d/autoload/myspacevim.vim` with contents for example

```
func! myspacevim#before() abort
    let g:neomake_enabled_c_makers = ['clang']
    nnoremap jk <esc>
endf

```

```
func! myspacevim#after() abort
    iunmap jk
endf
```

Vim Compatible Mode

This a list of different key bindings between SpaceVim and origin vim. If you still want to use this origin function, you can enable vimcompatible mode, via `vimcompatible = true` in `[options]` section.

- The s key does replace cursor char, but in SpaceVim it is the Window key bindings specific leader key by default (which can be set on another key binding in dotfile). If you still prefer the origin function of s, you can use an empty string to disable this feature.
  the option is `g:spacevim_windows_leader` , default value is `s` .
- The , key does repeat last f, F, t and T in vim, but in SpaceVim it is the language specified Leader key.
  the option is `g:spacevim_enable_language_specific_leader` , default value is 1.
- The q key does recording, but in SpaceVim it is used for smart close window.
  the option is `g:spacevim_windows_smartclose` , default value is `q` . If you still prefer the origin function of `q` , you can use an empty string to disable this feature.
- The Ctrl + a binding on the command line auto-completes variable names, but in SpaceVim it moves to the cursor to the beginning of the command.
  Send a PR to add the differences you found in this section.

Private Layers

This section is an overview of layers. A more extensive introduction to writing configuration layers can be found in SpaceVim's layers page (recommended reading!).

Purpose

Layers help collect related packages together to provide features. For example, the `lang#python` layer provides auto-completion, syntax checking, and REPL support for python files. This approach helps keep configuration organized and reduces overhead for the user by keeping them from having to think about what packages to install. To install all the `python` features the user has just to add the `lang#python` layer to their custom configuration file.

Structure

In SpaceVim, a layer is a single file. In a layer, for example, `autocomplete` layer, the file is `autoload/SpaceVim/layers/autocomplete.vim` , and there are there public functions:

- SpaceVim#layers#autocomplete#plugins(): return a list of plugins used in this plugins.

- `SpaceVim#layers#autocomplete#config()`: layer config, such as key bindings and autocmds.
- `SpaceVim#layers#autocomplete#set_variable()`: function for setting layer options.

Concepts

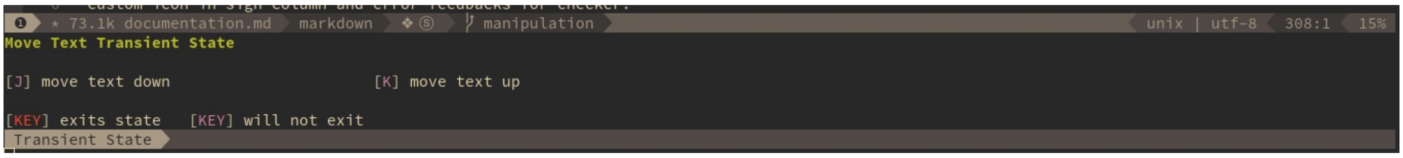
Concepts

Transient-states

SpaceVim defines a wide variety of transient states (temporary overlay maps)where it makes sense. This prevents one from doing repetitive and tediouspresses on the SPC key.

When a transient state is active, a documentation is displayed in thetransient state buffer. Additional information may as well be displayed in it.

Move Text Transient State:



```
 ★ 73.1k documentation.md  markdown    manipulation                unix | utf-8  308:1  15%
Move Text Transient State

[J] move text down                     [K] move text up

[KEY] exits state   [KEY] will not exit
 Transient State
```

Interface elements

Interface elements

SpaceVim has a minimalistic and distraction free UI:

- custom airline with color feedback according to current check status
- custom icon in sign column and error feedbacks for checker.

Colorschemes

The default colorscheme of SpaceVim is gruvbox.There are two variants of this colorscheme, a dark one and a light one. Some aspectsof these colorscheme can be customized in the custom configuration file, read `:h gruvbox` .

It is possible to define your default themes in your `~/.SpaceVim.d/init.toml` withthe variable colorschemes. For instance, to specify `desert` :

```
[options]
    colorscheme = "desert"
    colorscheme_bg = "dark"
```

Mappings	Description
`SPC T n`	switch to next random colorscheme listed in colorscheme layer.
`SPC T s`	select a theme using a unite buffer.

all the included colorscheme can be found in colorscheme layer.

NOTE:

SpaceVim use true colors by default, so you should make sure your terminal support true colors.for more information see: Colours in terminal

If your terminal do not supports true colors, you can disable SpaceVim true colors featurein `[options]` section:

```
    enable_guicolors = false
```

Font

The default font used by SpaceVim is DejaVu Sans Mono for Powerline. It is recommendedto install it on your system if you wish to use it.

To change the default font set the variable `guifont` in your `~/.SpaceVim.d/init.toml` file.

By default its value is:

```
guifont = 'DejaVu Sans Mono for Powerline:h11'
```

If the specified font is not found, the fallback one will be used (depends on your system).Also note that changing this value has no effect if you are running Vim/Neovim in terminal.

UI Toggles

Some UI indicators can be toggled on and off (toggles start with t and T):

Key Binding	Description
`SPC t 8`	highlight any character past the 80th column
`SPC t f`	display the fill column (by default `max_column` is 120)
`SPC t h h`	toggle highlight of the current line
`SPC t h i`	toggle highlight indentation levels (TODO)
`SPC t h c`	toggle highlight indentation current column
`SPC t h s`	toggle syntax highlighting
`SPC t i`	toggle indentation guide at point
`SPC t n`	toggle line numbers
`SPC t b`	toggle background
`SPC t t`	open tabs manager
`SPC T ~`	display ~ in the fringe on empty lines
`SPC T F`	toggle frame fullscreen
`SPC T f`	toggle display of the fringe
`SPC T m`	toggle menu bar
`SPC T t`	toggle tool bar

Statusline

The `core#statusline` layer provide a heavily customized powerline with the following capabilities:

- show the window number
- show the current mode
- color code for current state
- show the index of searching result
- toggle syntax checking info
- toggle battery info
- toggle minor mode lighters
- show VCS information (branch, hunk summary) (need git and VersionControl layer)

Key bindings	Description
`SPC [1-9]`	jump to the windows with the specific number

Reminder of the color codes for the states:

Mode	Color
Normal	Grey
Insert	Blue
Visual	Orange
Replace	Aqua

all the colors based on the current colorscheme

Some elements can be dynamically toggled:

Key Binding	Description
`SPC t m b`	toggle the battery status (need to install acpi)
`SPC t m c`	toggle the org task clock (available in org layer)(TODO)
`SPC t m m`	toggle the minor mode lighters
`SPC t m M`	toggle the major mode
`SPC t m n`	toggle the cat! (if colors layer is declared in your dotfile)(TODO)
`SPC t m p`	toggle the cursor position
`SPC t m t`	toggle the time
`SPC t m d`	toggle the date
`SPC t m T`	toggle the mode line itself
`SPC t m v`	toggle the version control info

nerd font installation:

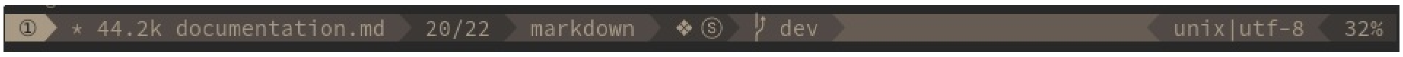
By default SpaceVim use nerd-fonts, please read the documentation of nerd fonts.

syntax checking integration:

When syntax checking minor mode is enabled, a new element appears showing the number of errors, warnings.

Search index integration:

Search index shows the number of occurrence when performing a search via `/` or `?`. SpaceVim integrates nicely the search status by displaying it temporarily when n or N are being pressed. See the 20/22 segment on the screenshot below.



```
① ⋆ 44.2k documentation.md  20/22  markdown  ❖ ⓢ  dev                unix|utf-8  32%
```

Battery status integration:

acpi displays the percentage of total charge of the battery as well as the time remaining to charge or discharge completely the battery.

A color code is used for the battery status:

Battery State	Color
Charging	Green
Discharging	Orange
Critical	Red

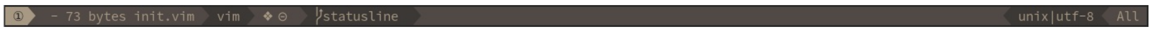
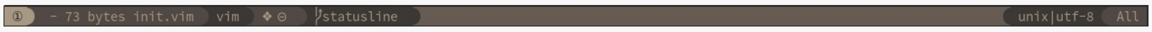
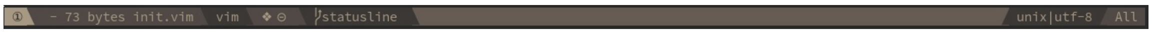
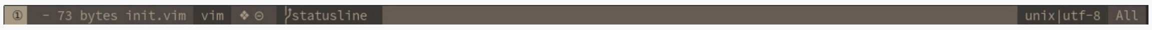
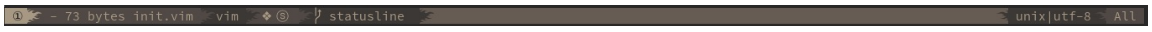
all the colors based on the current colorscheme

Statusline separators:

It is possible to easily customize the statusline separator by setting the `statusline_separator` variable in your custom configuration file and then redraw the statusline. For instance if you want to set back the separator to the well-known arrow separator add the following snippet to your configuration file:

```
statusline_separator = 'arrow'
```

here is an exhaustive set of screenshots for all the available separator:

Separator	Screenshot
`arrow`	
`curve`	
`slant`	
`nil`	
`fire`	

Minor Modes:

The minor mode area can be toggled on and off with `SPC t m m`

Unicode symbols are displayed by default. Add `statusline_unicode_symbols = false` to your custom configuration file, statusline will display ASCII characters instead (may be useful in terminal if you cannot set an appropriate font).

The letters displayed in the statusline correspond to the key bindings used to toggle them.

Key Binding	Unicode	ASCII	Mode

`SPC t 8`	⑧	8	toggle highlight of characters for long lines
`SPC t f`	ⓕ	f	fill-column-indicator mode
`SPC t s`	ⓢ	s	syntax checking (neomake)
`SPC t S`	Ⓢ	S	enabled in spell checking
`SPC t w`	ⓦ	w	whitespace mode

colorscheme of statusline:

current version only support `gruvbox` / `molokai` / `nord` / `one` / `onedark` , if you want tocontribute theme please check the template of a statusline theme.

```
" the theme colors should be
" [
"    \ [ a_guifg,  a_guibg,  a_ctermfg,  a_ctermbg],
"    \ [ b_guifg,  b_guibg,  b_ctermfg,  b_ctermbg],
"    \ [ c_guifg,  c_guibg,  c_ctermfg,  c_ctermbg],
"    \ [ z_guibg,  z_ctermbg],
"    \ [ i_guifg,  i_guibg,  i_ctermfg,  i_ctermbg],
"    \ [ v_guifg,  v_guibg,  v_ctermfg,  v_ctermbg],
"    \ [ r_guifg,  r_guibg,  r_ctermfg,  r_ctermbg],
"    \ [ ii_guifg, ii_guibg, ii_ctermfg, ii_ctermbg],
"    \ [ in_guifg, in_guibg, in_ctermfg, in_ctermbg],
" \ ]
" group_a: window id
" group_b/group_c: stausline sections
" group_z: empty area
" group_i: window id in insert mode
" group_v: window id in visual mode
" group_r: window id in select mode
" group_ii: window id in iedit-insert mode
" group_in: windows id in iedit-normal mode
function! SpaceVim#mapping#guide#theme#gruvbox#palette() abort
    return [
                \ ['#282828', '#a89984', 246, 235],
                \ ['#a89984', '#504945', 239, 246],
                \ ['#a89984', '#3c3836', 237, 246],
                \ ['#665c54', 241],
                \ ['#282828', '#83a598', 235, 109],
                \ ['#282828', '#fe8019', 235, 208],
                \ ['#282828', '#8ec07c', 235, 108],
                \ ['#282828', '#689d6a', 235, 72],
                \ ['#282828', '#8f3f71', 235, 132],
                \ ]
endfunction
```

this example is for gruvbox colorscheme, if you want to use same colors whenswitch between different colorschemes, you may need to set `custom_color_palette` in your custom configuration file. for example:

```
custom_color_palette = [
    ["#282828", "#a89984", 246, 235],
    ["#a89984", "#504945", 239, 246],
    ["#a89984", "#3c3836", 237, 246],
    ["#665c54", 241],
    ["#282828", "#83a598", 235, 109],
    ["#282828", "#fe8019", 235, 208],
    ["#282828", "#8ec07c", 235, 108],
    ["#282828", "#689d6a", 235, 72],
    ["#282828", "#8f3f71", 235, 132],
    ]
```

tabline

Buffers will be listed on tabline if there is only one tab, each item containsthe index, bufname and the filetype icon. if there are more than one tab, alltabs will be listed on the tabline. each item can be quickly accessed using `<Leader> number` . default `<Leader>` is `\` .

Key Binding	Description
`<Leader> 1`	Jump to index 1 on tabline
`<Leader> 2`	Jump to index 2 on tabline
`<Leader> 3`	Jump to index 3 on tabline
`<Leader> 4`	Jump to index 4 on tabline
`<Leader> 5`	Jump to index 5 on tabline
`<Leader> 6`	Jump to index 6 on tabline
`<Leader> 7`	Jump to index 7 on tabline
`<Leader> 8`	Jump to index 8 on tabline
`<Leader> 9`	Jump to index 9 on tabline

SpaceVim tabline also support mouse click, left mouse button will switch to buffer, middle button will delete the buffer.

NOTE: this feature is only supported in neovim with `has('tablineat')` .

Key Binding	Description
`<Mouse-left>`	Jump to the buffer
`<Mouse-middle>`	Delete the buffer

Tab manager:

You can also use `SPC t t` to open the tab manager windows.

key bindings within tab manager windows:

Key Binding	Description

`o`	Close or expand tab windows.
`r`	Rename the tab under the cursor.
`n`	Create new named tab below the cursor tab
`N`	Create new tab below the cursor tab
`x`	Delete the tab
`<C-S-Up>`	Move tab backward
`<C-S-Down>`	Move tab forward
`<Enter>`	Jump to windows under the cursor.

General Key bindings

General Key bindings

Window manager

Window manager

Windows manager key bindings can only be used in normal mode. The default leader is `s` , you cancan change it via `g:spacevim_windows_leader`

Key bindings	Description
`q`	Smart buffer close
`s` + `p`	Split nicely
`s` + `v`	:split
`s` + `g`	:vsplit
`s` + `t`	Open new tab (:tabnew)
`s` + `o`	Close other windows (:only)
`s` + `x`	Remove buffer, leave blank window
`s` + `q`	Remove current buffer, left buffer in the tabline will be displayed. If there is no buffer on the left, the right buffer will be displayed; if this is the last buffer in the tabline, then an empty buffer will be displayed.
`s` + `Q`	Close current buffer (:close)
`Tab`	Next window or tab
`Shift` + `Tab`	Previous window or tab
`<leader>` + `sv`	Split with previous buffer
`<leader>` + `sg`	Vertically split with previous buffer

SpaceVim has mapped normal `q` as smart buffer close, the normal func of `q` can be get by `<leader> q r`

Key	Mode	Action
`<leader>` + `y`	visual	Copy selection to X11 clipboard (“+y)
`Ctrl` + `c`	Normal	Copy full path of current buffer to X11 clipboard
`<leader>` + `Ctrl` + `c`	Normal	Copy github.com url of current buffer to X11 clipboard(if it is a github repo)
`<leader>` + `Ctrl` + `l`	Normal/visual	Copy github.com url of current lines to X11 clipboard(if it is a github repo)
`<leader>` + `p`	Normal/visual	Paste selection from X11 clipboard (“+p)
`Ctrl` + `f`	Normal	Smart page forward (C-f/C-d)
`Ctrl` + `b`	Normal	Smart page backwards (C-b/C-u)
`Ctrl` + `e`	Normal	Smart scroll down (3C-e/j)
`Ctrl` + `y`	Normal	Smart scroll up (3C-y/k)

`Ctrl` + `q`	Normal	`Ctrl` + `w`
`Ctrl` + `x`	Normal	Switch buffer and placement
`Up,Down`	Normal	Smart up and down
`}`	Normal	After paragraph motion go to first non-blank char (}^)
`<`	Visual/Normal	Indent to left and re-select
`>`	Visual/Normal	Indent to right and re-select
`Tab`	Visual	Indent to right and re-select
`Shift` + `Tab`	Visual	Indent to left and re-select
`gp`	Normal	Select last paste
`Q` / `gQ`	Normal	Disable EX-mode ()
`Ctrl` + `a`	Command	Navigation in command line
`Ctrl` + `b`	Command	Move cursor backward in command line
`Ctrl` + `f`	Command	Move cursor forward in command line

File Operations

File Operations

Key	Mode	Action
`<leader>` + `cd`	Normal	Switch to the directory of the open buffer
`<leader>` + `w`	Normal/visual	Write (:w)
`Ctrl` + `s`	Normal/visual/Command	Write (:w)
`:w!!`	Command	Write as root (%!sudo tee > /dev/null %)

Editor UI

Editor UI

Key	Mode	Action
F2	*All*	Toggle tagbar
F3	*All*	Toggle Vimfiler
<leader> + num	Normal	Jump to the buffer with the num index
<Alt> + num	Normal	Jump to the buffer with the num index, this only works in neovim
<Alt> + h / <Left>	Normal	Jump to left buffer in the tabline, this only works in neovim
<Alt> + l / <Right>	Normal	Jump to Right buffer in the tabline, this only works in neovim
<leader> + ts	Normal	Toggle spell-checker (:setlocal spell!)
<leader> + tn	Normal	Toggle line numbers (:setlocal nonumber!)
<leader> + tl	Normal	Toggle hidden characters (:setlocal nolist!)
<leader> + th	Normal	Toggle highlighted search (:set hlsearch!)
<leader> + tw	Normal	Toggle wrap (:setlocal wrap! breakindent!)
g0	Normal	Go to first tab (:tabfirst)
g$	Normal	Go to last tab (:tablast)
gr	Normal	Go to previous tab (:tabprevious)
Ctrl + <Dow>	Normal	Move to split below (j)
Ctrl + <Up>	Normal	Move to upper split (k)
Ctrl + <Left>	Normal	Move to left split (h)
Ctrl + <Right>	Normal	Move to right split (l)
*	Visual	Search selection forwards
#	Visual	Search selection backwards
, + Space	Normal	Remove all spaces at EOL
Ctrl + r	Visual	Replace selection
<leader> + lj	Normal	Next on location list
<leader> + lk	Normal	Previous on location list
<leader> + S	Normal/visual	Source selection

Native functions

`Native functions`

Key	Mode	Action
`<leader>` + `qr`	Normal	Same as native `q`
`<leader>` + `qr/`	Normal	Same as native `q/`, open cmdwin
`<leader>` + `qr?`	Normal	Same as native `q?`, open cmdwin
`<leader>` + `qr:`	Normal	Same as native `q:`, open cmdwin

Bookmarks management

Bookmarks management

Key	Mode	Action
`m` + `a`	Normal	Show list of all bookmarks
`m` + `m`	Normal	Toggle bookmark in current line
`m` + `n`	Normal	Jump to next bookmark
`m` + `p`	Normal	Jump to previous bookmark
`m` + `i`	Normal	Annotate bookmark

As SpaceVim use above bookmarks mappings, so you can not use `a` , `m` , `n` , `p` or `i` registers to mark current position, but other registers should works will. if you really need to use these registers, you can add `nnoremap <leader>m m` to your custom configuration, then you use use `a` registers via `\ma`

Fuzzy finder

Fuzzy finder

SpaceVim provides five kinds of fuzzy finder, each of them is configured in a layer(`unite` , `denite` , `leaderf` , `ctrlp` and `fzf` layer).These layers have the same key bindings and features. But they need different dependencies.

User only need to load one of these layers, then will be able to get thesefeatures.

Key bindings

Key bindings	Discription
`<Leader> f <space>`	Fuzzy find menu:CustomKeyMaps
`<Leader> f e`	Fuzzy find register
`<Leader> f f`	Fuzzy find file
`<Leader> f h`	Fuzzy find history/yank
`<Leader> f j`	Fuzzy find jump, change
`<Leader> f l`	Fuzzy find location list
`<Leader> f m`	Fuzzy find output messages
`<Leader> f o`	Fuzzy find outline
`<Leader> f q`	Fuzzy find quick fix
`<Leader> f r`	Resumes Unite window

But in current version of SpaceVim, leaderf/ctrlp and fzf layer has not be finished.

Feature	unite	denite	leaderf	ctrlp	fzf
menu: CustomKeyMaps	**yes**	**yes**	no	no	no
register	**yes**	**yes**	no	**yes**	**yes**
file	**yes**	**yes**	**yes**	**yes**	**yes**
yank history	**yes**	**yes**	no	no	**yes**
jump	**yes**	**yes**	no	**yes**	**yes**
location list	**yes**	**yes**	no	no	**yes**
outline	**yes**	**yes**	**yes**	**yes**	**yes**
message	**yes**	**yes**	no	no	**yes**
quickfix list	**yes**	**yes**	no	**yes**	**yes**
resume windows	**yes**	**yes**	no	no	no

Key bindings within fuzzy finder buffer

key bindings	Mode	description
Tab / <C-j>	-	Select next line
Shift + Tab / <C-k>	-	Select previous line
jk	Insert	Leave Insert mode (Only for denite/unite)
Ctrl + w	Insert	Delete backward path
Enter	-	Run default action
Ctrl + s	-	Open in a split
Ctrl + v	-	Open in a vertical split
Ctrl + t	-	Open in a new tab
Ctrl + g	-	Exit unite

Denite/Unite normal mode key bindings

key bindings	Mode	description
Ctrl + h/k/l/r	Normal	Un-map
Ctrl + l	Normal	Redraw
Tab	Normal	Select actions
Space	Normal	Toggle mark current candidate, up
r	Normal	Replace (‘search’ profile) or rename
Ctrl + z	Normal/insert	Toggle transpose window

The above key bindings only are part of fuzzy finder layers, please read the layer’s documentation.

Discovering

Discovering

Mappings

Mappings guide

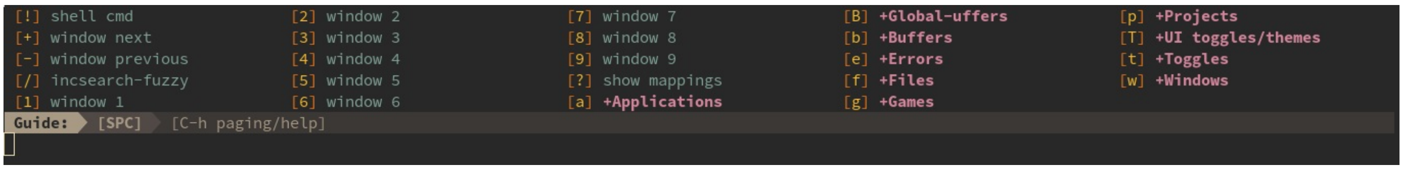
A guide buffer is displayed each time the prefix key is pressed in normal mode. It lists the available key bindings and their short description.The prefix can be `[SPC]` , `[Window]` , `[denite]` , `<leader>` and `[unite]` .

The default key of these prefix is:

Prefix name	custom option and default value	description
`[SPC]`	NONE / `<Space>`	default mapping prefix of SpaceVim
`[Window]`	`g:spacevim_windows_leader` / `s`	window mapping prefix of SpaceVim
`<leader>`	default vim leader	default leader prefix of vim/neovim

By default the guide buffer will be displayed 1000ms after the key has been pressed. You can change the delay by setting `'timeoutlen'` option to your liking (the value is in milliseconds).

for example, after pressing `<Space>` in normal mode, you will see :



```
[!] shell cmd          [2] window 2       [7] window 7        [B] +Global-uffers   [p] +Projects
[+] window next        [3] window 3       [8] window 8        [b] +Buffers         [T] +UI toggles/themes
[-] window previous    [4] window 4       [9] window 9        [e] +Errors          [t] +Toggles
[/] incsearch-fuzzy    [5] window 5       [?] show mappings   [f] +Files           [w] +Windows
[1] window 1           [6] window 6       [a] +Applications   [g] +Games
Guide: [SPC] [C-h paging/help]
```

this guide show you all the available key bindings begin with `[SPC]` , you can type `b` for all the buffer mappings, `p` for project mappings, etc.

after pressing `<C-h>` in guide buffer, you will get paging and help info in the statusline.

key	description
`u`	undo pressing
`n`	next page of guide buffer
`p`	previous page of guide buffer

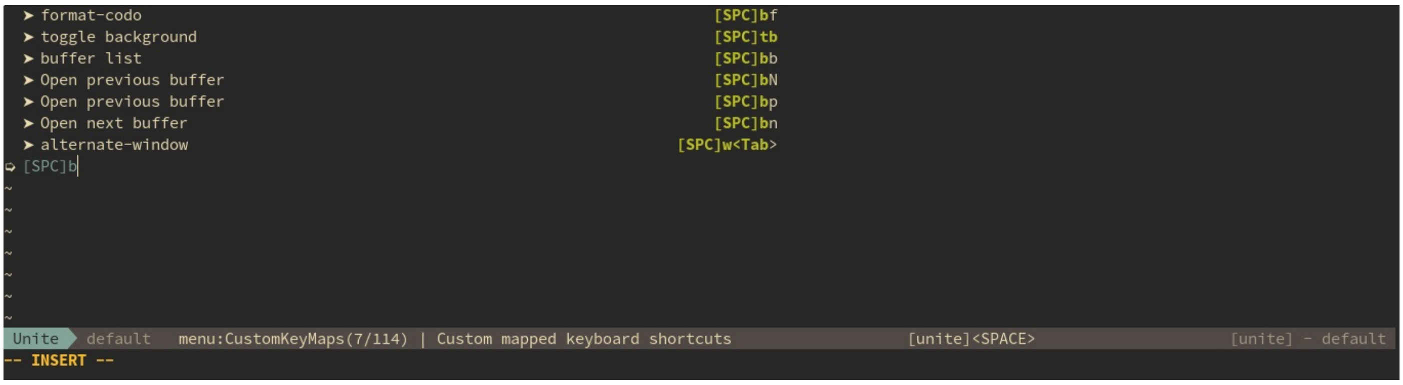
to defined custom SPC mappings, use `SpaceVim#custom#SPC()` . here is an example:

```
call SpaceVim#custom#SPC('nnoremap', ['f', 't'], 'echom "hello world"', 'test custom SPC', 1)
```

Unite/Denite describe key bindings

It is possible to search for specific key bindings by pressing `?` in the root of guide buffer.

To narrow the list, just insert the mapping keys or description of what mapping you want, Unite/Denite will fuzzy find the mappings, to find buffer related mappings:



```
 ➤ format-codo                                                              [SPC]bf
 ➤ toggle background                                                        [SPC]tb
 ➤ buffer list                                                              [SPC]bb
 ➤ Open previous buffer                                                     [SPC]bN
 ➤ Open previous buffer                                                     [SPC]bp
 ➤ Open next buffer                                                         [SPC]bn
 ➤ alternate-window                                                    [SPC]w<Tab>
➪ [SPC]b
~
~
~
~
~
~
~
Unite  default   menu:CustomKeyMaps(7/114) | Custom mapped keyboard shortcuts          [unite]<SPACE>          [unite] - default
-- INSERT --
```

then use `<Tab>` or `<Up>` and `<Down>` to select the mapping, press `<Enter>` will execute that command.

Getting help

Denite/Unite is powerful tool to unite all interfaces. it was meant to be like Helm for Vim. These mappings is for getting help info about functions, variables etc:

Mappings	Description
`SPC h SPC`	discover SpaceVim documentation, layers and packages using unite
`SPC h i`	get help with the symbol at point
`SPC h k`	show top-level bindings with which-key
`SPC h m`	search available man pages

Reporting an issue:

Mappings	Description
`SPC h I`	Open SpaceVim GitHub issue page with pre-filled information

Available layers

All layers can be easily discovered via `:SPLayer -l` accessible with `SPC h l`.

Available plugins in SpaceVim

All plugins can be easily discovered via `<leader> l p`.

Add custom plugins

If you want to add plugin from github, just add the repo name to the SpaceVim option `custom_plugins` :

```
[[custom_plugins]]
name = 'lilydjwg/colorizer'
merged = 0
```

Toggles

both the toggles mappings started with `[SPC] t` or `[SPC] T` . you can find it in the mapping guide.

Navigating

Navigating

Point/Cursor

Navigation is performed using the Vi key bindings `hjkl` .

Key Binding	Description
`h`	move cursor left (origin vim key, no mappings)
`j`	move cursor down (origin vim key, no mappings)
`k`	move cursor up (origin vim key, no mappings)
`l`	move cursor right (origin vim key, no mappings)
`H`	move cursor to the top of the screen (origin vim key, no mappings)
`L`	move cursor to the bottom of the screen (origin vim key, no mappings)
`SPC j 0`	go to the beginning of line (and set a mark at the previous location in the line)
`SPC j $`	go to the end of line (and set a mark at the previous location in the line)
`SPC t -`	lock the cursor at the center of the screen

Vim motions with vim-easymotion

quick-jump-link mode (TODO)

https://github.com/easymotion/vim-easymotion/issues/315

Similar to easymotion or `f` in vimperator for firefox, this mode allows one to jump to any link in help file with two key strokes.

mapping	description
`o`	initiate quick jump link mode in help buffer

Unimpaired bindings

Mappings	Description
`[ SPC`	Insert space above
`] SPC`	Insert space below
`[ b`	Go to previous buffer
`] b`	Go to next buffer
`[ f`	Go to previous file in directory

] f	Go to next file in directory
[l	Go to the previous error
] l	Go to the next error
[c	Go to the previous vcs hunk (need VersionControl layer)
] c	Go to the next vcs hunk (need VersionControl layer)
[q	Go to the previous error
] q	Go to the next error
[t	Go to the previous frame
] t	Go to the next frame
[w	Go to the previous window
] w	Go to the next window
[e	Move line up
] e	Move line down
[p	Paste above current line
] p	Paste below current line
g p	Select pasted text

Jumping, Joining and Splitting

The `SPC j` prefix is for jumping, joining and splitting.

Jumping

Key Binding	Description
SPC j 0	go to the beginning of line (and set a mark at the previous location in the line)
SPC j $	go to the end of line (and set a mark at the previous location in the line)
SPC j b	jump backward
SPC j f	jump forward
SPC j d	jump to a listing of the current directory
SPC j D	jump to a listing of the current directory (other window)
SPC j i	jump to a definition in buffer (denite outline)
SPC j I	jump to a definition in any buffer (denite outline)
SPC j j	jump to a character in the buffer (easymotion)
SPC j J	jump to a suite of two characters in the buffer (easymotion)
SPC j k	jump to next line and indent it using auto-indent rules
SPC j l	jump to a line with avy (easymotion)
SPC j q	show the dumb-jump quick look tooltip (TODO)

`SPC j u`	jump to a URL in the current window
`SPC j v`	jump to the definition/declaration of an Emacs Lisp variable (TODO)
`SPC j w`	jump to a word in the current buffer (easymotion)

Joining and splitting

Key Binding	Description
`J`	join the current line with the next line
`SPC j k`	go to next line and indent it using auto-indent rules
`SPC j n`	split the current line at point, insert a new line and auto-indent
`SPC j o`	split the current line at point but let point on current line
`SPC j s`	split a quoted string or s-expression in place
`SPC j S`	split a quoted string or s-expression, insert a new line and auto-indent

Window manipulation

Window manipulation key bindings

Every window has a number displayed at the start of the statusline and can be quickly accessed using `SPC number` .

Key Binding	Description
`SPC 1`	go to window number 1
`SPC 2`	go to window number 2
`SPC 3`	go to window number 3
`SPC 4`	go to window number 4
`SPC 5`	go to window number 5
`SPC 6`	go to window number 6
`SPC 7`	go to window number 7
`SPC 8`	go to window number 8
`SPC 9`	go to window number 9

Windows manipulation commands (start with `w`):

Key Binding	Description
`SPC w TAB`	switch to alternate window in the current frame (switch back and forth)
`SPC w =`	balance split windows
`SPC w b`	force the focus back to the minibuffer (TODO)
`SPC w c`	Distraction-free reading current window

SPC w C	Distraction-free reading other windows via vim-choosewin
SPC w d	delete a window
SPC u SPC w d	delete a window and its current buffer (does not delete the file) (TODO)
SPC w D	delete another window using vim-choosewin
SPC u SPC w D	delete another window and its current buffer using vim-choosewin (TODO)
SPC w t	toggle window dedication (dedicated window cannot be reused by a mode) (TODO)
SPC w f	toggle follow mode (TODO)
SPC w F	create new tab(frame)
SPC w h	move to window on the left
SPC w H	move window to the left
SPC w j	move to window below
SPC w J	move window to the bottom
SPC w k	move to window above
SPC w K	move window to the top
SPC w l	move to window on the right
SPC w L	move window to the right
SPC w m	maximize/minimize a window (maximize is equivalent to delete other windows) (TODO, now only support maximize)
SPC w M	swap windows using vim-choosewin
SPC w o	cycle and focus between tabs
SPC w p m	open messages buffer in a popup window (TODO)
SPC w p p	close the current sticky popup window (TODO)
SPC w r	rotate windows forward
SPC w R	rotate windows backward
SPC w s or SPC w -	horizontal split
SPC w S	horizontal split and focus new window
SPC w u	undo window layout (used to effectively undo a closed window) (TODO)
SPC w U	redo window layout (TODO)
SPC w v or SPC w /	vertical split
SPC w V	vertical split and focus new window
SPC w w	cycle and focus between windows
SPC w W	select window using vim-choosewin

Buffers and Files

Buffers manipulation key bindings

Buffer manipulation commands (start with `b`):

Key Binding	Description
`SPC TAB`	switch to alternate buffer in the current window (switch back and forth)
`SPC b .`	buffer transient state
`SPC b b`	switch to a buffer (via denite/unite)
`SPC b d`	kill the current buffer (does not delete the visited file)
`SPC u SPC b d`	kill the current buffer and window (does not delete the visited file) (TODO)
`SPC b D`	kill a visible buffer using vim-choosewin
`SPC u SPC b D`	kill a visible buffer and its window using ace-window(TODO)
`SPC b C-d`	kill other buffers
`SPC b C-D`	kill buffers using a regular expression(TODO)
`SPC b e`	erase the content of the buffer (ask for confirmation)
`SPC b h`	open *SpaceVim* home buffer
`SPC b n`	switch to next buffer avoiding special buffers
`SPC b m`	open *Messages* buffer
`SPC u SPC b m`	kill all buffers and windows except the current one(TODO)
`SPC b p`	switch to previous buffer avoiding special buffers
`SPC b P`	copy clipboard and replace buffer (useful when pasting from a browser)
`SPC b R`	revert the current buffer (reload from disk)
`SPC b s`	switch to the *scratch* buffer (create it if needed)
`SPC b w`	toggle read-only (writable state)
`SPC b Y`	copy whole buffer to clipboard (useful when copying to a browser)
`z f`	Make current function or comments visible in buffer as much as possible (TODO)

Create a new empty buffer

Key Binding	Description
`SPC b N h`	create new empty buffer in a new window on the left
`SPC b N j`	create new empty buffer in a new window at the bottom
`SPC b N k`	create new empty buffer in a new window above
`SPC b N l`	create new empty buffer in a new window below
`SPC b N n`	create new empty buffer in current window

Special Buffers

In SpaceVim, there are many special buffers, these buffers are created by plugins or SpaceVim itself. And these buffers are not listed.

Files manipulations key bindings

Files manipulation commands (start with f):

Key Binding	Description
`SPC f b`	go to file bookmarks
`SPC f c`	copy current file to a different location(TODO)
`SPC f C d`	convert file from unix to dos encoding
`SPC f C u`	convert file from dos to unix encoding
`SPC f D`	delete a file and the associated buffer (ask for confirmation)
`SPC f E`	open a file with elevated privileges (sudo edit)(TODO)
`SPC f f`	open file
`SPC f F`	try to open the file under point
`SPC f o`	open a file using the default external program(TODO)
`SPC f R`	rename the current file(TODO)
`SPC f s`	save a file
`SPC f S`	save all files
`SPC f r`	open a recent file
`SPC f t`	toggle file tree side bar
`SPC f T`	show file tree side bar
`SPC f y`	show and copy current file absolute path in the cmdline

Vim and SpaceVim files

Convenient key bindings are located under the prefix `SPC f v` to quickly navigate between Vim and SpaceVim specific files.

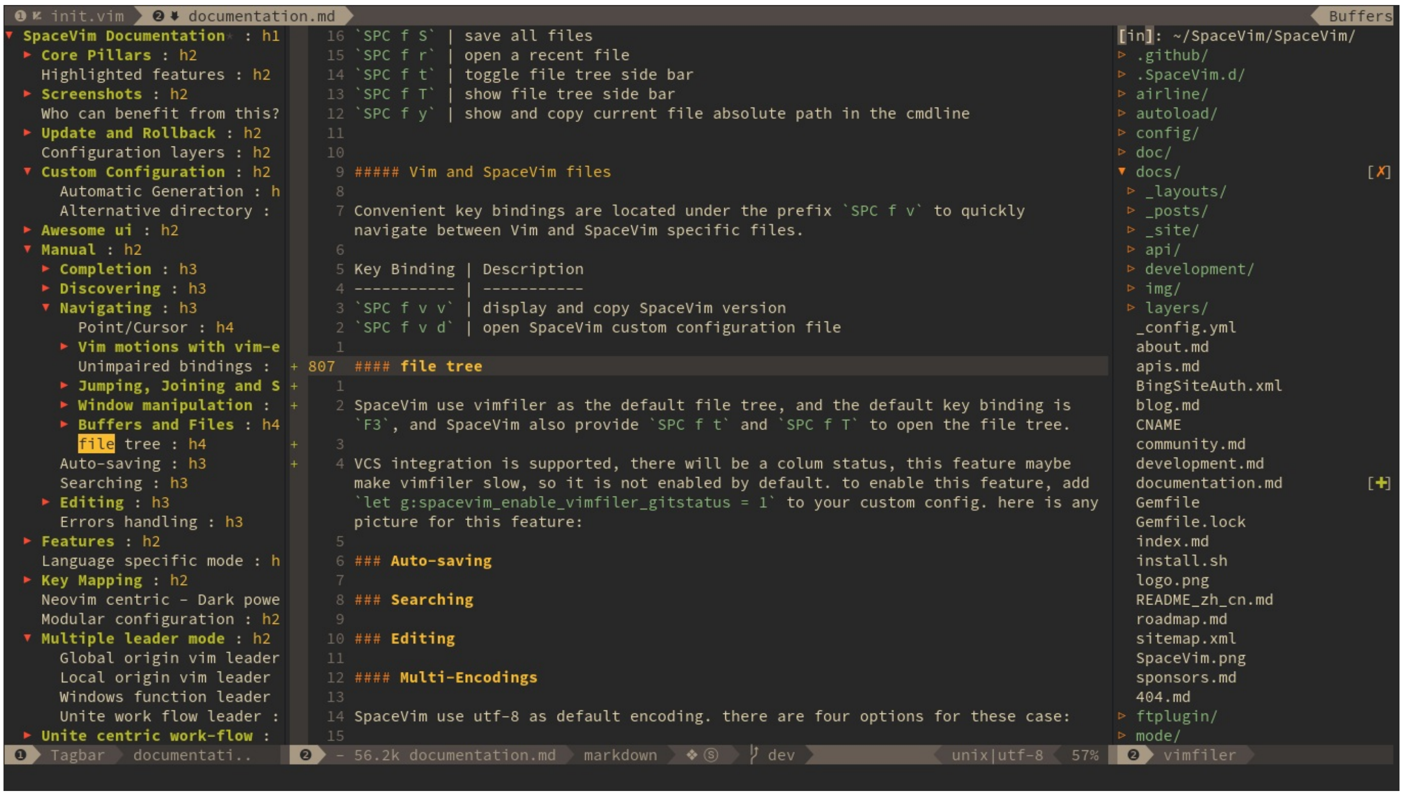
Key Binding	Description
`SPC f v v`	display and copy SpaceVim version
`SPC f v d`	open SpaceVim custom configuration file

File tree

SpaceVim use vimfiler as the default file tree, and the default key binding is `F3`, and SpaceVim also provide `SPC f t` and `SPC f T` to open the file tree. to change the file explore to nerdtree:

```
# the default value is vimfiler
filemanager = "nerdtree"
```

VCS integration is supported, there will be a column status, this feature maybe make vimfiler slow, soit is not enabled by default. to enable this feature, add `enable_vimfiler_gitstatus = true` to your custom config.here is any picture for this feature:



```
init.vim   documentation.md                                                    Buffers
```

```
▼ SpaceVim Documentation : h1
  ▶ Core Pillars : h2
    Highlighted features : h2
  ▶ Screenshots : h2
    Who can benefit from this?
  ▶ Update and Rollback : h2
    Configuration layers : h2
  ▼ Custom Configuration : h2
      Automatic Generation : h
      Alternative directory :
  ▶ Awesome ui : h2
  ▼ Manual : h2
    ▶ Completion : h3
    ▶ Discovering : h3
    ▼ Navigating : h3
        Point/Cursor : h4
      ▶ Vim motions with vim-e
        Unimpaired bindings :
      ▶ Jumping, Joining and S
      ▶ Window manipulation :
      ▶ Buffers and Files : h4
        file tree : h4
      Auto-saving : h3
      Searching : h3
    ▶ Editing : h3
      Errors handling : h3
  ▶ Features : h2
    Language specific mode : h
  ▶ Key Mapping : h2
    Neovim centric - Dark powe
    Modular configuration : h2
  ▼ Multiple leader mode : h2
      Global origin vim leader
      Local origin vim leader
      Windows function leader
      Unite work flow leader :
  ▶ Unite centric work-flow :
```

```
`SPC f S` | save all files
`SPC f r` | open a recent file
`SPC f t` | toggle file tree side bar
`SPC f T` | show file tree side bar
`SPC f y` | show and copy current file absolute path in the cmdline


##### Vim and SpaceVim files

Convenient key bindings are located under the prefix `SPC f v` to quickly
navigate between Vim and SpaceVim specific files.

Key Binding | Description
----------- | -----------
`SPC f v v` | display and copy SpaceVim version
`SPC f v d` | open SpaceVim custom configuration file

#### file tree

SpaceVim use vimfiler as the default file tree, and the default key binding is
`F3`, and SpaceVim also provide `SPC f t` and `SPC f T` to open the file tree.

VCS integration is supported, there will be a colum status, this feature maybe
make vimfiler slow, so it is not enabled by default. to enable this feature, add
`let g:spacevim_enable_vimfiler_gitstatus = 1` to your custom config. here is any
picture for this feature:

### Auto-saving

### Searching

### Editing

#### Multi-Encodings

SpaceVim use utf-8 as default encoding. there are four options for these case:

```

```
[in]: ~/SpaceVim/SpaceVim/
▷ .github/
▷ .SpaceVim.d/
▷ airline/
▷ autoload/
▷ config/
▷ doc/
▼ docs/                              [✗]
  ▷ _layouts/
  ▷ _posts/
  ▷ _site/
  ▷ api/
  ▷ development/
  ▷ img/
  ▷ layers/
  _config.yml
  about.md
  apis.md
  BingSiteAuth.xml
  blog.md
  CNAME
  community.md
  development.md
  documentation.md                   [+]
  Gemfile
  Gemfile.lock
  index.md
  install.sh
  logo.png
  README_zh_cn.md
  roadmap.md
  sitemap.xml
  SpaceVim.png
  sponsors.md
  404.md
▷ ftplugin/
▷ mode/
```

```
Tagbar  documentati..   - 56.2k documentation.md  markdown  dev   unix|utf-8  57%   vimfiler
```

File tree navigation

Navigation is centered on the `hjkl` keys with the hope of providing a fast navigation experience like in vifm:

Key Binding	Description
`<F3>` or `SPC f t`	Toggle file explorer
Within *file tree* buffers	
`<Left>` or `h`	go to parent node and collapse expanded directory
`<Down>` or `j`	select next file or directory
`<Up>` or `k`	select previous file or directory
`<Right>` or `l`	open selected file or expand directory
`Ctrl` + `j`	Un-map
`Ctrl` + `l`	Un-map
`E`	Un-map
`N`	Create new file under corsor
`yy`	Copy file full path to system clipboard
`yY`	Copy file to system clipboard
`P`	Paste file to the position under the cursor
`.`	toggle visible ignored files

`sv`	Split edit
`sg`	Vertical split edit
`p`	Preview
`i`	Switch to directory history
`v`	Quick look
`gx`	Execute with vimfiler associated
`'`	Toggle mark current line
`V`	Clear all marks
`Ctrl` + `r`	Redraw

Open file with file tree.

If there is only one file buffer opened, a file is opened in the active window, otherwise we need to use vim-choosewin to select a window to open the file.

Key Binding	Description
`l` or `Enter`	open file in one window
`sg`	open file in an vertically split window
`sv`	open file in an horizontally split window

Commands starting with g

Commands starting with g

after pressing prefix `g` in normal mode, if you do not remember the mappings, you will see the guide which will tell you the functional of all mappings starting with `g` .

Key Binding	Description
`g#`	search under cursor backward
`g$`	go to rightmost character
`g&`	repeat last ":s" on all lines
`g'`	jump to mark
`g*`	search under cursor forward
`g+`	newer text state
`g,`	newer position in change list
`g-`	older text state
`g/`	stay incsearch
`g0`	go to leftmost character
`g;`	older position in change list
`g<`	last page of previous command output
`g<Home>`	go to leftmost character
`gE`	end of previous word
`gF`	edit file under cursor(jump to line after name)
`gH`	select line mode
`gI`	insert text in column 1
`gJ`	join lines without space
`gN`	visually select previous match
`gQ`	switch to Ex mode
`gR`	enter VREPLACE mode
`gT`	previous tag page
`gU`	make motion text uppercase
`g]`	tselect cursor tag
`g^`	go to leftmost no-white character
`g_`	go to last char
``g` ``	jump to mark
`ga`	print ascii value of cursor character

`gd`	goto definition
`ge`	go to end of previous word
`gf`	edit file under cursor
`gg`	go to line N
`gh`	select mode
`gi`	insert text after ‘^ mark
`gj`	move cursor down screen line
`gk`	move cursor up screen line
`gm`	go to middle of screenline
`gn`	visually select next match
`go`	goto byte N in the buffer
`gs`	sleep N seconds
`gt`	next tag page
`gu`	make motion text lowercase
`g~`	swap case for Nmove text
`g<End>`	go to rightmost character
`g<C-G>`	show cursor info

Commands starting with z

Commands starting with z

after pressing prefix `z` in normal mode, if you do not remember the mappings, you will see the guide which will tell you the functional of all mappings starting with `z` .

Key Binding	Description
`z<Right>`	scroll screen N characters to left
`z+`	cursor to screen top line N
`z-`	cursor to screen bottom line N
`z.`	cursor line to center
`z<CR>`	cursor line to top
`z=`	spelling suggestions
`zA`	toggle folds recursively
`zC`	close folds recursively
`zD`	delete folds recursively
`zE`	eliminate all folds
`zF`	create a fold for N lines
`zG`	mark good spelled(update internal-wordlist)
`zH`	scroll half a screenwidth to the right
`zL`	scroll half a screenwidth to the left
`zM`	set `foldlevel` to zero
`zN`	set `foldenable`
`zO`	open folds recursively
`zR`	set `foldlevel` to deepest fold
`zW`	mark wrong spelled
`zX`	re-apply `foldlevel`
`z^`	cursor to screen bottom line N
`za`	toggle a fold
`zb`	redraw, cursor line at bottom
`zc`	close a fold
`zd`	delete a fold
`ze`	right scroll horizontally to cursor position
`zf`	create a fold for motion
`zg`	mark good spelled

`zh`	scroll screen N characters to right
`zi`	toggle foldenable
`zj`	mode to start of next fold
`zk`	mode to end of previous fold
`zl`	scroll screen N characters to left
`zm`	subtract one from `foldlevel`
`zn`	reset `foldenable`
`zo`	open fold
`zr`	add one to `foldlevel`
`zs`	left scroll horizontally to cursor position
`zt`	cursor line at top of window
`zv`	open enough folds to view cursor line
`zx`	re-apply foldlevel and do “zV”
`zz`	smart scroll
`z<Left>`	scroll screen N characters to right

Searching

Searching

With an external tool

SpaceVim can be interfaced with different searching tools like:

- rg - ripgrep
- ag - the silver searcher
- pt - the platinum searcher
- ack
- grep
  The search commands in SpaceVim are organized under the `SPC s` prefix with the next key is the tool to use and the last key is the scope. For instance `SPC s a b` will search in all opened buffers using `ag`.

If the last key (determining the scope) is uppercase then the current word under the cursor is used as default input for the search. For instance `SPC s a B` will search with word under cursor.

If the tool key is omitted then a default tool will be automatically selected for the search. This tool corresponds to the first tool found on the system of the list `g:spacevim_search_tools`, the default order is `rg`, `ag`, `pt`, `ack` then `grep`. For instance `SPC s b` will search in the opened buffers using `pt` if `rg` and `ag` have not been found on the system.

The tool keys are:

Tool	Key
ag	a
grep	g
ack	k
rg	r
pt	t

The available scopes and corresponding keys are:

Scope	Key
opened buffers	b
buffer directory	d
files in a given directory	f
current project	p

It is possible to search in the current file by double pressing the second key of the sequence, for instance `SPC s a a` will search in the current file with `ag`.

Notes:

- rg, ag and pt are optimized to be used in a source control repository but they can be used in an arbitrary directory as well.
- It is also possible to search in several directories at once by marking them in the unite buffer.

  Beware if you use `pt`, TCL parser tools also install a command line tool called `pt`.

Useful key bindings

Key Binding	Description
`SPC r l`	resume the last completion buffer
``SPC s ` ``	go back to the previous place before jump
Prefix argument	will ask for file extensions

Searching in current file

Key Binding	Description
`SPC s s`	search with the first found tool
`SPC s S`	search with the first found tool with default input
`SPC s a a`	ag
`SPC s a A`	ag with default input
`SPC s g g`	grep
`SPC s g G`	grep with default input
`SPC s r r`	rg
`SPC s r R`	rg with default input

Searching in buffer directory

Key Binding	Description
`SPC s d`	searching in buffer directory with default tool
`SPC s D`	searching in buffer directory cursor word with default tool
`SPC s a d`	searching in buffer directory with ag
`SPC s a D`	searching in buffer directory cursor word with ag
`SPC s g d`	searching in buffer directory with grep
`SPC s g D`	searching in buffer directory cursor word with grep
`SPC s k d`	searching in buffer directory with ack
`SPC s k D`	searching in buffer directory cursor word with ack
`SPC s r d`	searching in buffer directory with rg

`SPC s r D`	searching in buffer directory cursor word with rg
`SPC s t d`	searching in buffer directory with pt
`SPC s t D`	searching in buffer directory cursor word with pt

Searching in all loaded buffers

Key Binding	Description
`SPC s b`	search with the first found tool
`SPC s B`	search with the first found tool with default input
`SPC s a b`	ag
`SPC s a B`	ag with default input
`SPC s g b`	grep
`SPC s g B`	grep with default input
`SPC s k b`	ack
`SPC s k B`	ack with default input
`SPC s r b`	rg
`SPC s r B`	rg with default input
`SPC s t b`	pt
`SPC s t B`	pt with default input

Searching in an arbitrary directory

Key Binding	Description
`SPC s f`	search with the first found tool
`SPC s F`	search with the first found tool with default input
`SPC s a f`	ag
`SPC s a F`	ag with default text
`SPC s g f`	grep
`SPC s g F`	grep with default text
`SPC s k f`	ack
`SPC s k F`	ack with default text
`SPC s r f`	rg
`SPC s r F`	rg with default text
`SPC s t f`	pt
`SPC s t F`	pt with default text

Searching in a project

Key Binding	Description
`SPC /` or `SPC s p`	search with the first found tool

`SPC *` or `SPC s P`	search with the first found tool with default input
`SPC s a p`	ag
`SPC s a P`	ag with default text
`SPC s g p`	grep
`SPC s g p`	grep with default text
`SPC s k p`	ack
`SPC s k P`	ack with default text
`SPC s t p`	pt
`SPC s t P`	pt with default text
`SPC s r p`	rg
`SPC s r P`	rg with default text

Hint: It is also possible to search in a project without needing to open a file beforehand. To do so use `SPC p p` and then `C-s` on a given project to directly search into it like with `SPC s p`. (TODO)

Background searching in a project

Background search keyword in a project, when searching done, the count will be shown on the statusline.

Key Binding	Description
`SPC s j`	searching input expr background with the first found tool
`SPC s J`	searching cursor word background with the first found tool
`SPC s l`	List all searching result in quickfix buffer
`SPC s a j`	ag
`SPC s a J`	ag with default text
`SPC s g j`	grep
`SPC s g J`	grep with default text
`SPC s k j`	ack
`SPC s k J`	ack with default text
`SPC s t j`	pt
`SPC s t J`	pt with default text
`SPC s r j`	rg
`SPC s r J`	rg with default text

Searching the web

Key Binding	Description
`SPC s w g`	Get Google suggestions in vim. Opens Google results in Browser.
	Get Wikipedia suggestions in vim. Opens Wikipedia page in Browser.

	(TODO)

Note: to enable google suggestions in vim, you need to add `let g:spacevim_enable_googlesuggest = 1` to your custom Configuration file.

Searching on the fly

Key Binding	Description
`SPC s g G`	Searching in project on the fly with default tools

key binding in FlyGrep buffer:

Key Binding	Description
`<Esc>`	close FlyGrep buffer
`<Enter>`	open file at the cursor line
`<Tab>`	move cursor line down
`<S-Tab>`	move cursor line up
`<Bs>`	remove last character
`<C-w>`	remove the Word before the cursor
`<C-u>`	remove the Line before the cursor
`<C-k>`	remove the Line after the cursor
`<C-a>` / `<Home>`	Go to the beginning of the line
`<C-e>` / `<End>`	Go to the end of the line

Persistent highlighting

SpaceVim uses `g:spacevim_search_highlight_persist` to keep the searched expression highlighted until the next search. It is also possible to clear the highlighting by pressing `SPC s c` or executing the ex command `:noh` .

Highlight current symbol

SpaceVim supports highlighting of the current symbol on demand and add a transient state to easily navigate and rename these symbol.

It is also possible to change the range of the navigation on the fly to:

- buffer
- function
- visible area
  To Highlight the current symbol under point press `SPC s h`.

Navigation between the highlighted symbols can be done with the commands:

Key Binding	Description

Binding	
`*`	initiate navigation transient state on current symbol and jump forwards
`#`	initiate navigation transient state on current symbol and jump backwards
`SPC s e`	edit all occurrences of the current symbol
`SPC s h`	highlight the current symbol and all its occurrence within the current range
`SPC s H`	go to the last searched occurrence of the last highlighted symbol

In highlight symbol transient state:

Key Binding	Description
`e`	edit occurrences (`*`)
`n`	go to next occurrence
`N` / `p`	go to previous occurrence
`b`	search occurrence in all buffers
`/`	search occurrence in whole project
`Tab`	toggle highlight current occurrence
`r`	change range (function, display area, whole buffer)
`R`	go to home occurrence (reset position to starting occurrence)
Any other key	leave the navigation transient state

Editing

Editing

Paste text

Auto-indent pasted text

Text manipulation commands

Text related commands (start with `x`):

Key Binding	Description	
`SPC x a &`	align region at &	
`SPC x a (`	align region at (	
`SPC x a )`	align region at)	
`SPC x a [`	align region at [	
`SPC x a ]`	align region at]	
`SPC x a {`	align region at {	
`SPC x a }`	align region at }	
`SPC x a ,`	align region at ,	
`SPC x a .`	align region at . (for numeric tables)	
`SPC x a :`	align region at :	
`SPC x a ;`	align region at ;	
`SPC x a =`	align region at =	
`SPC x a ¦`	align region at ¦	
`SPC x a`		align region at
`SPC x a a`	align region (or guessed section) using default rules (TODO)	
`SPC x a c`	align current indentation region using default rules (TODO)	
`SPC x a l`	left-align with evil-lion (TODO)	
`SPC x a L`	right-align with evil-lion (TODO)	
`SPC x a r`	align region using user-specified regexp (TODO)	
`SPC x a m`	align region at arithmetic operators `(+-*/)` (TODO)	
`SPC x c`	count the number of chars/words/lines in the selection region	
`SPC x d w`	delete trailing whitespaces	

SPC x d SPC	Delete all spaces and tabs around point, leaving one space
SPC x g l	set lanuages used by translate commands (TODO)
SPC x g t	translate current word using Google Translate
SPC x g T	reverse source and target languages (TODO)
SPC x i c	change symbol style to lowerCamelCase
SPC x i C	change symbol style to UpperCamelCase
SPC x i i	cycle symbol naming styles (i to keep cycling)
SPC x i -	change symbol style to kebab-case
SPC x i k	change symbol style to kebab-case
SPC x i _	change symbol style to under_score
SPC x i u	change symbol style to under_score
SPC x i U	change symbol style to UP_CASE
SPC x j c	set the justification to center (TODO)
SPC x j f	set the justification to full (TODO)
SPC x j l	set the justification to left (TODO)
SPC x j n	set the justification to none (TODO)
SPC x j r	set the justification to right (TODO)
SPC x J	move down a line of text (enter transient state)
SPC x K	move up a line of text (enter transient state)
SPC x l d	duplicate line or region (TODO)
SPC x l s	sort lines (TODO)
SPC x l u	uniquify lines (TODO)
SPC x o	use avy to select a link in the frame and open it (TODO)
SPC x O	use avy to select multiple links in the frame and open them (TODO)
SPC x t c	swap (transpose) the current character with the previous one
SPC x t w	swap (transpose) the current word with the previous one
SPC x t l	swap (transpose) the current line with the previous one
SPC x u	set the selected text to lower case (TODO)
SPC x U	set the selected text to upper case (TODO)
SPC x w c	count the number of occurrences per word in the select region (TODO)
SPC x w d	show dictionary entry of word from wordnik.com (TODO)
SPC x TAB	indent or dedent a region rigidly (TODO)

Text insertion commands

Text insertion commands (start with `i`):

Key binding	Description
`SPC i l l`	insert lorem-ipsum list
`SPC i l p`	insert lorem-ipsum paragraph
`SPC i l s`	insert lorem-ipsum sentence
`SPC i p 1`	insert simple password
`SPC i p 2`	insert stronger password
`SPC i p 3`	insert password for paranoids
`SPC i p p`	insert a phonetically easy password
`SPC i p n`	insert a numerical password
`SPC i u`	Search for Unicode characters and insert them into the active buffer.
`SPC i U 1`	insert UUIDv1 (use universal argument to insert with CID format)
`SPC i U 4`	insert UUIDv4 (use universal argument to insert with CID format)
`SPC i U U`	insert UUIDv4 (use universal argument to insert with CID format)

Increase/Decrease numbers

Key Binding	Description
`SPC n +`	increase the number under point by one and initiate transient state
`SPC n -`	decrease the number under point by one and initiate transient state

In transient state:

Key Binding	Description
`+`	increase the number under point by one
`-`	decrease the number under point by one
Any other key	leave the transient state

Tips: you can increase or decrease a value by more that once by using a prefix argument (i.e. `10 SPC n +` will add 10 to the number under point).

Replace text with iedit

SpaceVim uses powerful iedit mode to quick edit multiple occurrences of a symbol or selection.

Two new modes: `iedit-Normal` / `iedit-Insert`

The default color for iedit is `red` / `green` which is based on the current colorscheme.

iedit states key bindings

State transitions:

Key Binding	From	to
`SPC s e`	normal or visual	iedit-Normal

In iedit-Normal mode:

`iedit-Normal` mode inherits from `Normal` mode, the following key bindings are specific to `iedit-Normal` mode.

Key Binding	Description
`Esc`	go back to `Normal` mode
`i`	switch to `iedit-Insert` mode, same as `i`
`a`	switch to `iedit-Insert` mode, same as `a`
`I`	go to the beginning of the current occurrence and switch to `iedit-Insert` mode
`A`	go to the end of the current occurrence and switch to `iedit-Insert` mode
`<Left>` / `h`	Move cursor to left
`<Right>` / `l`	Move cursor to right
`0` / `<Home>`	go to the beginning of the current occurrence
`$` / `<End>`	go to the end of the current occurrence
`D`	delete the occurrences
`S`	delete the occurrences and switch to iedit-Insert mode
`gg`	go to first occurrence
`G`	go to last occurrence
`n`	go to next occurrence
`N`	go to previous occurrence
`p`	replace occurrences with last yanked (copied) text
`<Tab>`	toggle current occurrence

In iedit-Insert mode:

Key Binding	Description
`Esc`	go back to `iedit-Normal` mode
`<Left>`	Move cursor to left
`<Right>`	Move cursor to right
`<C-w>`	delete words before cursor
`<C-K>`	delete words after cursor

Examples

Commenting

Comments are handled by nerdcommenter, it’s bound to the following keys.

Key Binding	Description
`SPC ;`	comment operator
`SPC c h`	hide/show comments
`SPC c l`	comment lines
`SPC c L`	invert comment lines
`SPC c p`	comment paragraphs
`SPC c P`	invert comment paragraphs
`SPC c t`	comment to line
`SPC c T`	invert comment to line
`SPC c y`	comment and yank
`SPC c Y`	invert comment and yank

Tips: To comment efficiently a block of line use the combo `SPC ; SPC j l`

Multi-Encodings

SpaceVim use utf-8 as default encoding. there are four options for these case:

- fileencodings (fencs): ucs-bom,utf-8,default,latin1
- fileencoding (fenc): utf-8
- encoding (enc): utf-8
- termencoding (tenc): utf-8 (only supported in vim)
  to fix messy display: `SPC e a` is the mapping for auto detect the file encoding. after detecting file encoding, you can run the command below to fix the encoding:

```
set enc=utf-8
write
```

Errors handling

Errors handling

SpaceVim uses neomake to gives error feedback on the fly. The checks are only performed at save time by default.

Errors management mappings (start with e):

Mappings	Description
`SPC t s`	toggle syntax checker
`SPC e c`	clear all errors
`SPC e h`	describe a syntax checker
`SPC e l`	toggle the display of the list of errors/warnings
`SPC e n`	go to the next error
`SPC e p`	go to the previous error
`SPC e v`	verify syntax checker setup (useful to debug 3rd party tools configuration)
`SPC e .`	error transient state

The next/previous error mappings and the error transient state can be used to browse errors from syntax checkers as well as errors from location list buffers, and indeed anything that supports vim's location list. This includes for example search results that have been saved to a location list buffer.

Custom sign symbol:

Symbol	Description	Custom option
`✖`	Error	`g:spacevim_error_symbol`
`➤`	warning	`g:spacevim_warning_symbol`
`?`	Info	`g:spacevim_info_symbol`

Managing projects

Managing projects

Projects in SpaceVim are managed by vim-projectionist and vim-rooter, vim-rooter will find the root of the project when a `.git` directory or a `.projections.json` file is encountered in the file tree.

project manager commands start with `p` :

Key Binding	Description
`SPC p '`	open a shell in project’s root (with the shell layer)

Searching files in project

Key Binding	Description
`SPC p f`	find files in current project
`SPC p /`	fuzzy search for text in current project
`SPC p k`	kill all buffers of current project
`SPC p t`	find project root
`SPC p p`	list all projects

EditorConfig

EditorConfig

SpaceVim has support for EditorConfig, a configuration file to “define and maintain consistent coding styles between different editors and IDEs.”

To customize your editorconfig experience, read the editorconfig-vim package’s documentation.

Vim Server

Vim Server

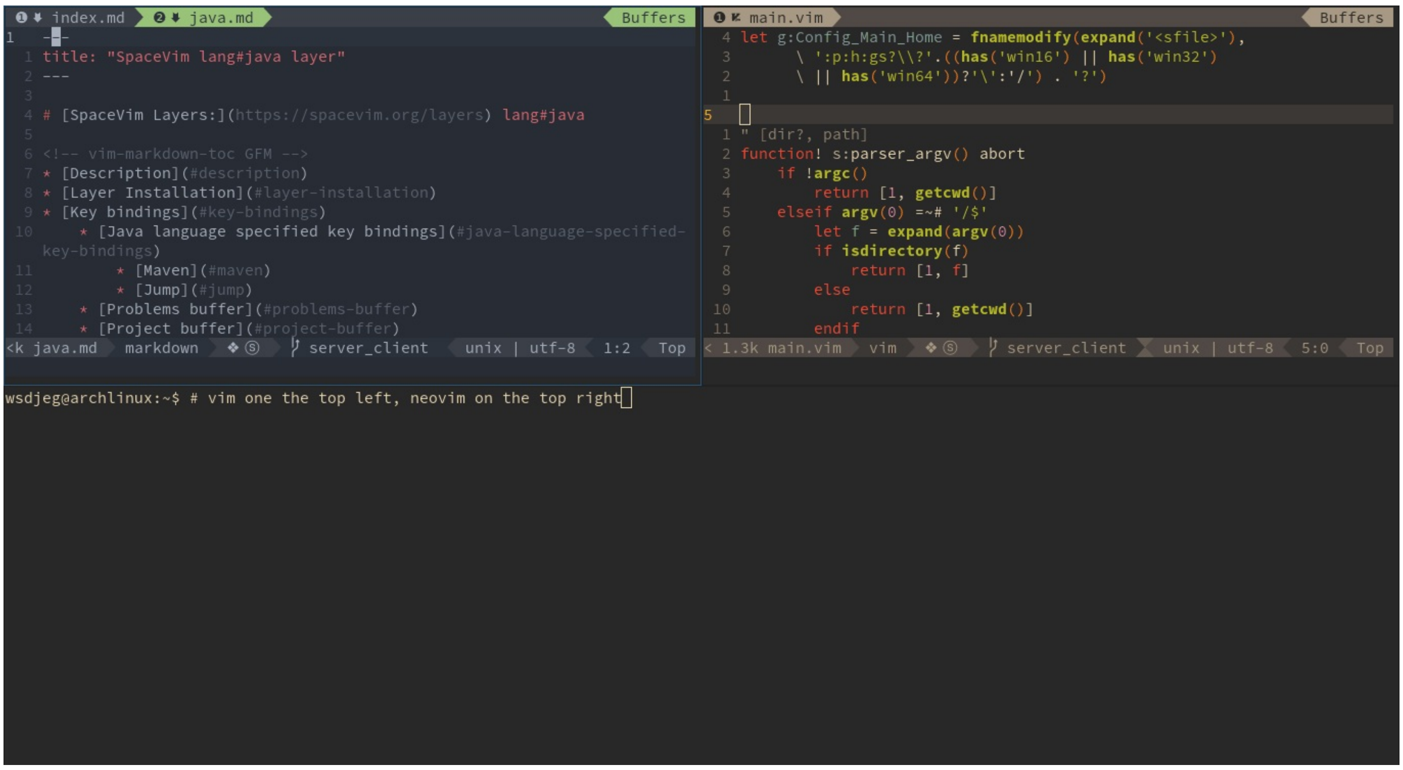
SpaceVim starts a server at launch. This server is killed whenever you close your Vim windows.

Connecting to the Vim server

If you are using neovim, you need to install neovim-remote, then add this to your bashrc.

```
export PATH=$PATH:$HOME/.SpaceVim/bin
```

Use `svc` to open a file in the existing Vim server, or using `nsvc` to open a file in the existing neovim server.



```
 index.md   java.md                                   Buffers
--
title: "SpaceVim lang#java layer"
---

# [SpaceVim Layers:](https://spacevim.org/layers) lang#java

<!-- vim-markdown-toc GFM -->
* [Description](#description)
* [Layer Installation](#layer-installation)
* [Key bindings](#key-bindings)
    * [Java language specified key bindings](#java-language-specified-
key-bindings)
        * [Maven](#maven)
        * [Jump](#jump)
    * [Problems buffer](#problems-buffer)
    * [Project buffer](#project-buffer)
<k java.md  markdown  server_client  unix | utf-8  1:2  Top
```

```
 main.vim                                             Buffers
let g:Config_Main_Home = fnamemodify(expand('<sfile>'),
      \ ':p:h:gs?\\?'.((has('win16') || has('win32')
      \ || has('win64'))?'\':'/') . '?')


" [dir?, path]
function! s:parser_argv() abort
    if !argc()
        return [1, getcwd()]
    elseif argv(0) =~# '/$'
        let f = expand(argv(0))
        if isdirectory(f)
            return [1, f]
        else
            return [1, getcwd()]
        endif
< 1.3k main.vim  vim  server_client  unix | utf-8  5:0  Top
```

```
wsdjeg@archlinux:~$ # vim one the top left, neovim on the top right
```

Achievements

Achievements

issues

Achievements	Account
100th issue(issue)	BenBergman
1000th issue(PR)	sei40kr

Stars, forks and watchers

Achievements	Account
First stargazers	monkeydterry
100th stargazers	ShaneDelmore
1000th stargazers	dongkui0712
2000th stargazers	EvgeneOskin
3000th stargazers	zerdon
4000th stargazers	sfwatergit
5000th stargazers	robgrzel