

Vim para noobs

William Oliveira

VIM para noobs

Um livro guia/referência para quem deseja aprender a utilizar o editor de textos Vim

William Oliveira

Esse livro está à venda em <http://leanpub.com/vimparanoobs>

Essa versão foi publicada em 2018-12-09



Esse é um livro [Leanpub](#). A Leanpub dá poderes aos autores e editores a partir do processo de Publicação Lean. [Publicação Lean](#) é a ação de publicar um ebook em desenvolvimento com ferramentas leves e muitas iterações para conseguir feedbacks dos leitores, pivotar até que você tenha o livro ideal e então conseguir tração.

© 2018 William Oliveira

Dedicado a toda comunidade de desenvolvedores e desenvolvedoras de software do Brasil

Conteúdo

Básico	1
Aprender a usar o básico	1
Não esquecer	1
Instalação do Vim	2
Modos de operação	3
Formando comandos	4
Abrindo um arquivo	5
Abrindo a partir de caminhos dentro do arquivo	5
Abrindo o arquivo em uma linha específica	5
Alternando entre arquivos	6
Se movimentando pelo texto	7
Inserindo texto	8
Deletando texto	9
Copiar e colar dentro do VIM	10
Recortar e colar	11
Recortar e colar dentro do Vim	11
Copiando para o clipboard no Vim	12
Resolvendo o problema de clipboard do Vim com o X11	12
Copiando e colando para o clipboard no Vim	12
Replace	14
Substituição simples	14
Substituir em massa	14
Desfazendo as coisas	15
Buscar texto dentro do arquivo	16

CONTEÚDO

Executando comandos externos	17
Salvando e fechando arquivos	18
Salvar como...	19
Alguns comandos legais para edição	20
Criando Marcas	21
Coisas legais de se fazer com marcas	21
Criar uma marca Global	21
Criando Folders (ou dobras no texto)	22
Personalizando o Vim	23
Geradores de vimrc files	23
Salvando suas preferências	24
Exemplo de arquivo .vimrc	24
Comandos e atalhos personalizados	25
Mudando o tema	26
Temas legais	27
Onde conseguir mais temas?	27
Adicionando sintaxes	28
Exemplos de vimrc	29
Plugins	30
Instalando plugins com o Vundle	31
Buscando um plugin com o Vundle	31
Removendo plugins	32
Listando os plugins instalados	32
Atualizando os plugins	33
Alguns plugins maneiros	34
Onde encontrar mais plugins?	35
Comandos uteis no .vimrc	36
Trabalhando com Janelas no VIM	37
trabalhando com abas no Vim	38

CONTEÚDO

Referências do livro	39
---------------------------------------	-----------

Basico

O Vim possui algumas características bem diferentes dos editores que já estamos acostumados, por isso é importante:

Aprender a usar o básico

Rode no Terminal

```
1 vimtutor
```

E faça os exercícios até o final.

Não esquecer

Como conseguir ajuda

Se usar o comando `:h` ou `:help` será aberto um helper.

Você pode usar `:h` comando para saber mais sobre o comando que irá executar

Letras maiúsculas nos comandos

Quando encontrar uma referência a um comando com letra maiúscula, é letra maiúscula mesmo (SHIFT + Letra)!

Ex.:

G - para pular para o final do arquivo.

Você deve pressionar o SHIFT+G.

Todo comando (também chamado de operador) deve ser executado no [Normal ou Command Mode](#)¹. Portanto se acostume a digitar o texto e pressionar ESC para poder executar um comando.

¹[/modos.md](#)

Instalação do Vim

Em sistemas baseados no Debian

```
1 sudo apt-get install vim
```

Em outros sistemas operacionais.²

²<http://www.vim.org/download.php>

Modos de operação

Os modos de operação no Vim são os seguintes:

- Para acessar o modo de inserção pressione i
- Para voltar ao normal mode pressione esc
- Para executar um comando em command mode digite :command
- Para acessar o visual mode pressione v

Onde:

- O modo de inserção é onde você passa o tempo digitando texto
- O modo normal é onde você vai formatar o texto (deletar, copiar e colar, replace, etc)
- Modo de comandos é uma maneira de executar comandos para o Vim, configurar o editor e muito mais
- Modo visual serve para seleção de grandes blocos de texto

Formando comandos

A maneira de agilizar a utilização de comandos no Vim é criando-os de acordo com sua necessidade.

A forma de se executar os comandos segue o seguinte padrão:

operator count motion

Onde:

operator - Operador

count - Contador para repetir o mesmo comando

motion - Movimento ou direção para onde executar

Ex.:

1 d4\$

Deleta 4 linhas até o final (d, deletar, 4, contador, \$, até o final da linha).

Abrindo um arquivo

```
1 vim arquivo  
  
ou  
  
1 vim caminho/arquivo
```

Abrindo a partir de caminhos dentro do arquivo

Caso esteja em um arquivo com referência a outro arquivo, você pode abrir o mesmo usando ‘CTRL-w-f’ com o cursor sobre o caminho.

Ex.:

Considere que está lendo um arquivo e encontrar no meio do texto:

```
1 /etc/hosts
```

Se usar o comando ‘CTRL-w-f’ em cima dessa linha, o Vim irá abrir o arquivo hosts.

Abrindo o arquivo em uma linha específica

Conseguimos descobrir onde está uma palavra ou bloco fazendo o seguinte (no linux):

```
1 grep -n "palavra" arquivo
```

Será exibido no Terminal a linha com o termo pesquisado:

```
1 numero_da_linha:termo_pesquisado
```

Então você consegue fazer:

```
1 vim +numero_da_linha nome_do_arquivo
```

E ele abre direto na linha específica.

OBS: Isso não é uma mágica do Vim, outros editores fazem a mesma coisa no Linux. ;)

Alternando entre arquivos

Podemos abrir outro arquivo, enquanto estamos no Vim, usando `:e nome ou caminho/nome`

Quando estamos editando mais de um arquivo podemos usar `CTRL+6` para alternar entre um e outro.

Se movimentando pelo texto

h: Esquerda

j: Baixo

k: Cima

l: Direita

Para ir ao final de um arquivo use: G

Para voltar ao topo: gg

Para mover ao final de uma palavra: e

Para mover ao final de uma palavra voltando o cursor: ge

Para mover até o começo da próxima palavra: w

Para mover até o começo de uma palavra voltando o cursor: b

Para mover ao próximo caractere específico use: f[caractere]

Para mover ao começo de uma linha, use: 0

Para mover ao final de uma linha, use: \$

Para pular ao final do parágrafo: }

Para pular para o fechamento de um parentese/colchete/chaves: %

Para saber sua posição no documento: CTRL+G

Podemos voltar onde estávamos a partir de qualquer local fazendo: numero_da_linha + G

O Vim mantém um histórico de alterações e para mover o cursor até a última alteração use g;, para mover para a próxima na lista g,

Inserindo texto

i - Entra no modo de inserção no local onde o cursor se encontra

I - Entra no modo de inserção no começo da linha

a - Entra no modo de inserção na frente de onde o cursor se encontra

A - Entra no modo de inserção no final da linha

o - Adiciona uma linha abaixo de onde estiver o cursor e entra em modo de inserção

O - Adiciona uma linha acima e entra em modo de inserção

ESC - Volta ao modo normal/comando

v - Entra no modo visual

Deletando texto

x - Deleta o caractere onde o cursor estiver

X - Deleta o caractere de trás do cursor

dw - Deleta a partir do cursor até o começo da próxima palavra

d\$ - Deleta do cursor até o fim da linha

dd - Deleta a linha inteira, incluindo o [enter] que estiver no final

C - Deleta de onde o cursor estiver até o final da linha e entra em modo de inserção a partir dali

ce - Deleta do cursor até o final da palavra e entra em modo de inserção

Copiar e colar dentro do VIM

y - Para copiar e p para colar

yy - Para copiar uma linha inteira

A maneira mais fácil de fazer o copy & paste é entrar no modo visual (v), selecionar o texto e então pressionar o y.

Recortar e colar

Recortar e colar dentro do Vim

Para recortar e colar use algum comando para exclusão (x, dd, dw, d\$ ou algum comando [formatado por você](#)³), movimente o cursor até o local onde deseja colar e use o comando p.

A ultima coisa deletada será inserida depois do cursor.

Uma maneira de copiar grandes blocos é usar o modo visual (v), selecionar o bloco, usar um d e colar usando o p.

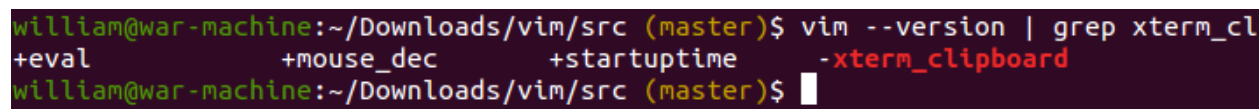
³[/comandos.md](#)

Copiando para o clipboard no Vim

Antes de conhecer os comandos de copiar/colar, execute esse teste no seu terminal:

```
1 vim --version | grep xterm_clipboard
```

Caso apareça igual a imagem abaixo, você se ferrou! - Brinks

A terminal window with a dark background. The prompt is 'william@war-machine:~/Downloads/vim/src (master)'. The command 'vim --version | grep xterm_cl' is entered. The output shows several features like '+eval', '+mouse_dec', and '+startuptime', but the line for clipboard support shows '-xterm_clipboard' in red, indicating it is not supported. The prompt returns to 'william@war-machine:~/Downloads/vim/src (master)\$'.

Clipboard test Vim

Resolvendo o problema de clipboard do Vim com o X11

O Vim não atende o X11 em [copiar e colar para o clipboard](#)⁴. Existem maneiras de contornar esse problema, como a desse [link](#)⁵. Eu optei por instalar uma versão do Vim com suporte ao clipboard, o [vim-gnome](#)⁶. Não tem nada de mais em relação ao Vim “normal” (pelo menos até agora não notei), porém não tive mais problemas com o clipboard.

Agora sim...

Copiando e colando para o clipboard no Vim

”+y : Copia um conteúdo marcado (com v) pra o clipboard.

”+p : Cola o conteúdo do clipboard para o Vim.

Se for copiar uma linha inteira, pode usar ”+yy, por exemplo. O “+ é um [registrador](#)⁷ do Vim para o Clipboard e você pode criar seus comandos igual acontece com os [outros operadores](#)⁸ do Vim.

Ex.:

⁴<http://vimcasts.org/blog/2013/11/getting-vim-with-clipboard-support/>

⁵<http://vimcasts.org/episodes/accessing-the-system-clipboard-from-vim/>

⁶<http://packages.ubuntu.com/precise/vim-gnome>

⁷<http://usevim.com/2012/04/13/registers/>

⁸[comandos.html](#)

1 "+5yy

Copia 5 linhas do texto.

Replace

Substituição simples

Use `rx`, onde `x` é o novo caractere a ser inserido onde estiver o cursor.

Outra maneira é fazer com `R`. Executando dessa maneira, irá aparecer na barra inferior o alerta – `REPLACE` –, então basta digitar a nova palavra.

Substituir em massa

`:s/antiga/nova` - Substitui a ocorrência de *antiga* para *nova* na mesma linha

`:#, # s/atiga/nova` - Substitui a ocorrência desde `#` até `#` linha do arquivo (um range)

`:% s/antiga/nova` - Substitui em todo o arquivo

`:% s/antiga/nova/gc` - Substitui no arquivo inteiro, mas solicita confirmação a cada ocorrência

Desfazendo as coisas

u - Desfaz a ultima alteração

U - Desfaz todas as mudanças efetuadas em uma linha inteira

CTRL+R - Refaz

Buscar texto dentro do arquivo

/termo_a_ser_pesquisado - Pesquisa para baixo do arquivo

?/termo - Pesquisa para cima do arquivo

// - Busca o ultimo termo pesquisado

/palavra/+numero - Posiciona o cursor *numero* de linhas após a ocorrência da palavra

Ao entrar no modo de busca, o VIM deixa o */palavra* na barra inferior, então podemos usar:

n - Para a próxima ocorrência N - Para a ocorrência anterior

Executando comandos externos

Digite `:! comando` para executar um comando externo ao editor.

Ex.:

```
1  :! ls
```

Vai executar o comando `ls` no Terminal.

Você pode editar um texto, salvar, digitar `:!git add arquivo`. Bom né?

Salvando e fechando arquivos

:q - Sai sem salvar. Será solicitado confirmação se existirem alterações não salvas.

:q! - Sai sem salvar descartando as alterações

:w - Salva o arquivo (escreve)

:wq - Salva e sai do arquivo

:x - Salva e sai do arquivo

ZZ - Salva e sai do arquivo

Salvar como...

Se você estiver em um arquivo e desejar salvar criando um novo, pode usar algo parecido com um “Salvar como...”.

Execute ‘:w nome ou caminho/nome’.

Se quiser salvar e sair, pode usar o ‘:wq nome ou caminho/nome’.

Alguns comandos legais para edição

`vU` : Deixa o caractere maiúsculo

`vu` : Deixa o caractere minúsculo

Você pode usar `v`, selecionar todo o texto e pressionar `u` ou `U` depois.

`:r arquivo` : Insere o conteúdo de arquivo na linha atual

`:sh` : Retorna temporariamente para o Shell.

`:ab texto textocompleto` : Define texto como abreviação de textocompleto

`:set autoindent` : Liga a indentação automática

`:set smartindent` : Liga a indentação inteligente

`:set shiftwidth=4` : Define o tamanho da indentação em 4 espaços

`ctrl-t`, `ctrl-d` : Indenta/Remove a indentação (modo de inserção)

`>>` : Como pressionar um tab

`<<` : Como voltar um tab

Criando Marcas

Marcas são como um [teleport](#)⁹ ;D

Você deixa configurado um ponto com uma marca e depois pode voltar a ela ou utilizar para outras coisinhas legais.

Criando uma marca:

ma - A marca (m) a foi criada onde o cursor estiver

Imagine que esteja em outra parte do texto. Para voltar basta pressionar: `a

Coisas legais de se fazer com marcas

Marque um local e delete até ele com

1 d'marca

Criar uma marca Global

Uma marca Global é um ponto onde você pode voltar até ele vindo até mesmo de outro arquivo.

mA - Com A maiusculo podemos acessar de qualquer local

⁹<http://www.wowhead.com/spell=48020/demonic-circle-teleport>

Criando Folders (ou dobras no texto)

Podemos pegar todo um bloco e ocultar ele. É como dobrar uma parte do texto.

Ao criar um folder de 10 linhas, por exemplo, o texto ficará assim:

```
1  +-- 10 linhas -----
```

Para isso faríamos:

```
1  zf10j
```

Para abrir novamente essas linhas, movemos o cursor até o local com `+-- 10...` e executamos:

```
1  zo
```

Uma maneira mais fácil de criar os folders é entrando no modo visual (v), selecionando o bloco e pressionando `zf`.

Isso vai criar um folder na seleção ativa.

Comandos:

`zfap` - Cria uma dobra para o parágrafo atual

`zf/palavra` - Cria uma dobra até a “palavra”

`zo` - Abre a dobra onde o cursor estiver

`zR` - Abre todas as dobras do arquivo atual

`zc` - Fecha a dobra onde o cursor estiver

`zd` - Apaga o folder (o conteúdo não é apagado)

`zj` - Desce até a próxima dobra

`zk` - Sobe até a próxima dobra

Personalizando o Vim

Para configurar o Vim, usamos alguns comandos junto com `:set`.

Bastaria digitar os comandos:

`:set autowrite` ou `:set aw` - Salva o arquivo a cada alteração

`:set errorbell` ou `:set eb` - Apita cada vez que você errar um comando.

Por exemplo.

O Vim possui um arquivo de configuração, o `.vimrc`.

Esse arquivo fica oculto em sua pasta `/home`.

Geradores de vimrc files

Caso você já manje de `vimrc` e deseja só usar algum pronto com coisas maneiras pode ser interessante dar uma olhada nesses links:

- vimconfig.com¹⁰
- vim-bootstrap.com¹¹

Se você nunca usou ou não conhece muito sobre Vim, o melhor é ir criando o seu próprio `.vimrc`. Não tem problema olhar [o .vimrc dos outros](#)¹² e inspirar-se neles, mas criando o seu você vai aprender mais e também vai deixar perfeito para as suas necessidades.

¹⁰<http://vimconfig.com/>

¹¹<http://vim-bootstrap.com/>

¹²</exemplos-de-vimrc.html>

Salvando suas preferências

Para salvar suas preferências você pode usar:

```
1 vim ~/.vimrc
```

E deixar seus comandos `:set` lá, porém nesse arquivo não precisa digitar `:set` configuração bastando `set` configuração.

Exemplo de arquivo .vimrc

```
1 syntax o
2 set encoding=utf8
3 set number
4 set tabstop=2
5 set eb
6 set autowrite
7 set expandtab
8 set hlsearch
9 set ignorecase
10 set incsearch
11 set showcmd
12 set showmatch
```

Comandos e atalhos personalizados

Com a personalização conseguimos criar comandos no VIM através do .vimrc.

Ex.:

Mapa para os comandos de trocar de Janela com CTRL+j/k/h/l para facilitar a vida.

```
1 map <C-j> <C-W>j
2 map <C-k> <C-W>k
3 map <C-h> <C-W>h
4 map <C-l> <C-W>l
```

Usando:

map - Criamos um Mapeamento em modo comando

imap - Mapeamento em modo de inserção.

cmap - Mapeamento em modo de linha de comando

vmap - Mapeamento no modo visual

Mudando o tema

Para encontrar um esquema de cores maneiro para o seu Vim, você pode acessar o site vimcolors.com¹³.

Alguns temas tem suporte ao **Vundle**, portanto será o mesmo processo de instalação que os **plugins**¹⁴.

Será necessário criar uma pasta `~/.vim`, se a mesma ainda não existir.

Os temas podem ser clonados dentro de `~/.vim/colors` para melhor organização, porém o Vundle, por exemplo, coloca dentro de `./vim/bundle`.

Normalmente, na documentação do próprio tema tem um passo a passo de instalação, mas basicamente você sempre vai fazer:

```
1 cd ~/.vim/colors #Pasta com esquemas de cores
2 git clone esquemadecores.git
```

E adicionar a linha do esquema no seu `.vimrc`.

Ex. do [esquema de cores](#)¹⁵ que eu uso atualmente:

```
1 colorscheme monokai
```

¹³<http://vimcolors.com/>

¹⁴</instalando-os-plugins-com-vundle.md>

¹⁵<https://github.com/sickill/vim-monokai>

Temas legais

Os que eu mais curti:

- <https://github.com/sickill/vim-monokai>
- <https://github.com/altercation/vim-colors-solarized>
- <https://github.com/marcopaganini/termschool-vim-theme>
- <https://github.com/joshdick/onedark.vim>

Onde conseguir mais temas?

- <http://vimcolors.com/>
- <http://www.vimninja.com/2012/08/26/10-vim-color-schemes-you-need-to-own/>

Você pode criar seu próprio tema [aqui](#)¹⁶.

¹⁶<http://mswift42.github.io/themecreator/>

Adicionando sintaxes

O mesmo que acontece com os [temas](#)¹⁷ ocorre com as sintaxes das linguagens, algumas poderão ser facilmente instaladas com o Vundle, outras serão, também facilmente, instaladas manualmente.

Basicamente você poderia fazer isso em uma instalação manual:

```
1 cd ~/.vim/syntax
2 git clone arquivo_sintaxe
```

Porém o Vundle também coloca esse tipo de arquivo dentro de ~/.vim/bundle.

E adicionar ao .vimrc

```
1 au FileType type call SyntaxFoo()
```

Para encontrar a sintaxe da sua linguagem basta dar uma Googlada: `vim syntax <language>`

¹⁷[/mudando-o-tema.md](#)

Exemplos de vimrc

As configurações do seu Vim são muito pessoais. Algumas coisas legais eu já passei, porém tem muito mais! Da uma olhada em:

- https://pt.wikibooks.org/wiki/Vim/Como_editar_prefer%C3%A2ncias
- <https://github.com/skwp/dotfiles#vim—whats-included>
- <https://github.com/aureooms/dotfiles/blob/master/.vimrc>
- <https://github.com/pedrofranceschi/vimfiles/blob/master/vimrc>
- <https://github.com/Floby/vim-config/blob/master/vimrc>
- <https://github.com/alessioalex/dotfiles/blob/master/vimrc>
- <https://github.com/smith/vim-config/blob/master/vimrc>

Plugins

Existem várias maneiras de se instalar um plugin no Vim.

A que eu achei mais legal foi usando o [Vundle](#)¹⁸, pois acabou sendo parecido com o uso do Package Control com Sublime.

A instalação do Vundle é simples. Você vai criar uma pasta chamada `.vim` em `/home` (se ela ainda não existir), clonar o projeto com Git e configurar todo o resto no seu `.vimrc`.

Para ativar corretamente o Vundle é necessário deixar as seguintes linhas no seu `.vimrc`:

```
1  "Configure Vundle
2  set nocompatible
3  "filetype off
4
5  "Vundle config
6  set rtp+=~/vim/bundle/Vundle.vim
7  call vundle#begin()
8
9  "need this to install Vundle
10 Plugin 'gmarik/Vundle.vim'
11
12 " ---- Plugins ----
```

Todos os plugins você pode adicionar abaixo da linha onde deixei: `" ---- Plugins ----`.

¹⁸<https://github.com/VundleVim/Vundle.vim>

Instalando plugins com o Vundle

Imagine que você vai instalar um plugin de Markdown Preview, esse [aqui](#)¹⁹.

Basta adicionar no seu `.vimrc`:

```
1 Plugin 'JamshedVesuna/vim-markdown-preview'
```

Abrir o Vim e rodar o comando:

```
1 :PluginInstall
```

Agora é só aguardar a instalação.

Buscando um plugin com o Vundle

Para pesquisar um nome de plugin ou saber se determinado plugin é instalável via Vundle você pode fazer:

```
1 :PluginSearch nome_do_plugin
```

Serão listados os plugins encontrados, então basta usa o comando:

i - Para instalar o plugin.

Você move o cursor até a linha do nome do plugin e pressiona i.

¹⁹<https://github.com/JamshedVesuna/vim-markdown-preview>

Removendo plugins

Você pode deixar o Vundle deletar os plugins que não são utilizados com:

```
1 :PluginClean
```

Ele irá pedir permissão a cada opção de delete.

Caso queira deixar tudo nas mãos do Vundle basta rodar:

```
1 :PluginClean!
```

Listando os plugins instalados

Basta rodar:

```
1 :PluginList
```

Atualizando os plugins

Caso vá atualizar todos os plugins basta rodar:

```
1 :PluginUpdate
```

Ou `:PluginUptade nome_do_plugin` para atualizar somente um.

Para outros comandos do Vundle, acesse a [documentação oficial](#)²⁰.

²⁰<https://github.com/VundleVim/Vundle.vim>

Alguns plugins maneiros

Alguns plugins legais de se utilizar no seu Vim:

- JavaScript Syntax : [vim-java-syntax](#)²¹
- JavaScript snippets : [vim-javascript-snippets](#)²²
- JSLint : [jslint.vim](#)²³
- JSBeautify : [vim-jsbeautify](#)²⁴
- Jade : [vim-jade](#)²⁵
- Stylus : [vim-stylus](#)²⁶
- Sass/SCSS : [vim-haml](#)²⁷
- HTML5 & CSS3 : [CSS more readable](#)²⁸
- EMMET : [emmet-vim](#)²⁹
- Node autocomplete : [vim-node](#)³⁰
- Markdown Preview : [vim-markdown-preview](#)³¹
- Git : [vim-gitgutter](#)³²
- Git flow : [vim-fugitive](#)³³
- Facilitar o trabalho com Janelas e Arquivos : [ack.vim](#)³⁴
- Sidebar mais fácil : [NERD Tree](#)³⁵
- Um autocomplete com base no que você está digitando : [supertab](#)³⁶
- Alinhamento de texto : [tabular-vim](#)³⁷
- Python autocompletion : [jedi-vim](#)³⁸
- Barra inferior com algumas informações bem úteis : [Airline](#)³⁹

²¹<https://github.com/jelera/vim-javascript-syntax>

²²<https://github.com/grvcoelho/vim-javascript-snippets>

²³<https://github.com/hallettj/jslint.vim>

²⁴<https://github.com/maksimr/vim-jsbeautify>

²⁵<https://github.com/digitaltoad/vim-jade>

²⁶<https://github.com/wavded/vim-stylus>

²⁷<https://github.com/tpope/vim-haml>

²⁸http://www.vim.org/scripts/script.php?script_id=3220

²⁹<https://github.com/mattm/emmet-vim>

³⁰<https://github.com/moll/vim-node>

³¹<https://github.com/JamshedVesuna/vim-markdown-preview>

³²<https://github.com/airblade/vim-gitgutter>

³³<https://github.com/tpope/vim-fugitive>

³⁴<https://github.com/mileszs/ack.vim>

³⁵<https://github.com/scrooloose/nerdtree>

³⁶<https://github.com/ervandew/supertab>

³⁷<http://vimcasts.org/episodes/aligning-text-with-tabular-vim>

³⁸<https://github.com/davidhalter/jedi-vim>

³⁹<https://github.com/vim-airline/vim-airline>

Onde encontrar mais plugins?

- <http://vimawesome.com/>
- <http://www.vim.org/scripts/>
- <http://www.bestofvim.com/plugin/>
- <http://code.tutsplus.com/series/vim-essential-plugins-net-19224>

Comandos uteis no .vimrc

Você pode achar vários comandos úteis dando uma olhada [nesses arquivos de .vimrc](#)⁴⁰.

No meu .vimrc só coloquei o que estou usando até agora:

```
1  "Cute ;D
2  syntax on
3  set encoding=utf8
4  set number
5  set tabstop=2
6  set eb
7  set expandtab
8  set hlsearch
9  set ignorecase
10 set incsearch
11 set showcmd
12 set showmatch
13 set textwidth=80
14 set cursorline
15 filetype indent on
16
17 " Smart way to move between windows
18 map <C-j> <C-W>j
19 map <C-k> <C-W>k
20 map <C-h> <C-W>h
21 map <C-l> <C-W>l
22
23 " space open/closes folds
24 nnoremap <space> za
25
26 "Theme
27 colorscheme monokai
```

⁴⁰[/exemplos-de-vimrc.html](#)

Trabalhando com Janelas no VIM

`:vsp` : Divide a janela verticalmente

`:sp` : Divide horizontalmente

Você pode abrir arquivos diretamente na outra metade da janela com: `comando arquivo`.

Ex.:

```
1 :vsp arquivo_x
```

Para alternar entre as duas metades você pode usar `CTRL+w` seguido da direção da janela.

Ex.:

```
1 CTRL+w+l
```

Vai para a janela da direita.

Se você quiser ficar somente com uma janela ativa use o comando `:only` na janela em questão.

Para fechar uma janela use: `:quit`

trabalhando com abas no Vim

`:tab nome_do_arquivo` : Abre o arquivo em uma nova aba

`:tabnew` : Abre uma nova aba em branco

`gt` : Vai para a próxima aba

`gT` : Volta a aba anterior

`CTRL+Page Down` : Mesmo que `gt`

`CTRL+Page Up` : Mesmo que `gT`

`:tabc` : Fecha a aba atual

`:tab split` : Copia o conteúdo da aba atual em uma nova aba

Referências do livro

Usei os seguintes links para coletar toda a informação do livro (alguns exemplos veio da minha cabeça mesmo errando e arrumando ;P):

- <https://pt.wikibooks.org/wiki/Vim>
- http://vim.wikia.com/wiki/Example_vimrc
- <http://doughblack.io/words/a-good-vimrc.html>
- <http://vimcasts.org/>
- <http://vimawesome.com/>
- <http://www.bestofvim.com/>
- <http://www.vimninja.com/>
- <http://aurelio.net/vim/>
- <http://vim.spf13.com/#vimrc>
- <http://vim-bootstrap.com>
- <https://code.google.com/p/vimbook/>
- <https://cassiobotaro.gitbooks.io/vimbook/> (Edição em Gitbook do livro acima)
- <http://aurelio.net/vim/vi-vim-venci.html>
- <http://oli.me.uk/2013/06/29/equipping-vim-for-javascript/>
- <http://tableless.com.br/manipulando-janelas-e-arquivos-no-vim/>
- <https://blog.butecopensource.org/vim-tips-trabalhando-com-abas/>
- http://www.tocadoelfo.com.br/2009/06/99-comandos-do-vim-que-todo-programador_15.html