Comandos básicos do Vim

Contents [Hide]


```
Comandos básicos do Vim
Autor: José Cláudio Faria
Data : 2013/12/08 - 22:34:23
```

Este resumo dos comandos básicos do Vim foram coletados em vários sites da internet, no manual do Vim e em livros.

Quando decidi usar (de forma eficiente) o Vim, achei que fazer um resumo dos recursos mais usados seria uma boa forma de aprendizado, além de poder ser útil para consultas futuras.

Vários pessoas já aprenderam e aperfeiçoaram o uso do Vim usando esse material. Então decidi dar uma formatação melhor (usando markdown no Vim) e divulgar na web.

Espero que seja útil como foi (e creio que ainda será por muitos anos) para mim.

OBS: ^ é abreviação para CTRL

1. MODOS

O Vim, diferentemente da grande maioria dos editores, é baseado em **modos**. Entender o que são e como usar esse modos é fundamental para seu uso.

```
Normal   - O que é digitado (e executado) são comandos (atalhos default ou definidos pelo usuário)

Inserção - O que é digitado é o texto.
           Deverá estar visível uma linha na parte de baixo: -- INSERÇÃO --

Execução - O que é digitado (e executado) são comandos.
           Após digitar : (dois pontos) irá aparecer na parte de baixo do editor uma linha para digitar o comando.
           Teste com:
           :ls
           !ls
           ESC para sair
```

Ao entrar no Vim, você estará sempre no modo Normal!

1.1. Entrar em modo de inserção

```
i  - Insere na posição atual, vem de 'insert'
I  - Insere no começo da linha
a  - Acrescenta na posição atual, vem de 'append'
A  - Acrescenta ao final da linha
o  - Insere na linha abaixo, já criando uma nova linha
O  - Insere na linha acima, já criando uma nova linha
cc - Deleta a linha e entra no modo de inserção, vem de 'clear'
C  - Deleta linha a frente do cursor e entra em modo de inserção, vem de 'clear'
s  - Deleta caracter e entra no modo de inserção 'substitute'
S  - Deleta linha e entra no modo de inserção
```

1.2. Sair do modo de inserção

```
ESC
^[
```

1.3. Salvando e saindo

```
:w           - Salva as alterações, abreviação de 'write'
:w arquivo   - Salva as alterações no arquivo especificado, como 'Save as'.
               Se estiver editando algo que já existe (velho) ele (o velho) é que ficará no buffer ativo para edição.
               Existe um atalho (^6) que permite alternar entre o novo e o velho.
:sav arquivo - Salva as alterações no arquivo especificado, como 'Save as'.
               O novo arquivo é que fica no buffer ativo para edição.
:q           - Sai do Vim 'quit'
:q!          - Força a saída sem salvar as alterações
```

```
:wq          - Salva as alterações e sai
```

2. MOVIMENTANDO

A forma mais comum para movimentar o cursor ainda é usando as setinhas do teclado, utilize-as à vontade. Todavia, com o tempo você vai perceber que as setinhas ficam longe demais das letras e que você precisa movimentar todo o seu braço para alcançá-las. Para evitar este esforço, tente o seguinte:

```
j - Baixo, sendo a tecla do seu indicador
k - Cima
l - Direita
h - Esquerda
```

Observe a naturalidade do comandos de movimentação com a mão direita no teclado!

Quando você está digitando uma linha longa e utiliza quebra de linha automática (wordwrap), ao movimentar para a linha de baixo você vai perceber que ele realmente vai para a próxima linha, e não para a posição abaixo do cursor. Para considerar o wordwrap, prefixe o movimento com *g*:

```
gk                - Vai para a posição acima, considerando o wordwrap
g[seta pra baixo] - Vai para a posição abaixo, considerando o wordwrap
g$                - Vai até o final da linha, considerando o wordwrap
0                 - Volta ao começo da linha
^                 - Volta ao começo da linha (duas vezes já que se trata de um acento)
$                 - Vai até o final da linha
w                 - Avança até a próxima palavra
e                 - Avança até o fim da palavra atual
b                 - Retorna ao início da palavra
f[caractere]      - Pressione f seguido de algum caractere para posicionar
                    o cursor na próxima ocorrência desse caractere
                    (não digite os colchetes)
t[caractere]      - A mesma coisa para o t, mas posiciona um caractere antes
                    do caractere pressionado (não digite os colchetes)
gg                - Retorna à primeira linha (tecle o gê duas vezes mesmo)
G                 - Vai até a última linha
:[núm. linha]     - Vai até a linha especificada (não digite os colchetes)
´´                - Volta até onde você estava antes de pular de posição (´ espaço ´ espaço)
```

Para quem ainda está se adaptando, as teclas seguintes também funcionam: *Home*, *Page Up*, *Page Down* e *End*!

2.1. Scrolling

```
^e  - Uma linha abaixo no buffer
^e5 - 5 linhas abaixo no buffer
^y  - Uma linha acima no buffer
^y5 - 5 linhas acima no buffer
^f  - Página abaixo (foward)
^b  - Página acima (backward)
```

3. REPETIÇÃO DE COMANDOS

Para repetir um comando, simplesmente prefixe-o com o número de vezes que deseja repetí-lo. Exemplo:

```
3w                   - Avança três palavras
10k                  - Sobe dez linhas
2t"                  - Coloca o cursor antes da segunda aspas
3i[escreve e dá ESC] - O que você digitar será inserido 3 vezes
                       (não digite os colchetes e não se esqueça de dar ESC)
```

4. DELETANDO

```
Inserção - Utilizamos o backspace e o delete para as correções triviais
Normal   - Podemos fazer exclusões mais elaboradas, combinando o delete
           com os comandos que já vimos acima.
x/X - Apaga o caracter sob o cursor/antes do cursor
D   - Paga da posição atual até o fim da linha
J   - Junta duas linhas (não importa a posição do cursor, 5J irá juntar cinco linhas contíguas)
dd  - Apaga toda a linha
dj  - Apaga 2 linhas abaixo
dk  - Apaga 2 linhas acima
dw  - Apaga até o fim da palavra
dt" - Apaga da posição atual até o fechamento das aspas
5db - Apaga cinco palavras para trás
^w  - Em Modo de Inserção (diferente dos comandos acima em Modo Normal),
      funciona como o backspace, mas apaga até o começo da palavra
      (seria o mesmo que db em modo normal)
d[algum comando de posicionamento] - Combina o comando com qualquer outro comando
                                     de posicionamento, veja os exemplos abaixo:
```

5. MODO VISUAL

No Vim utilizamos o Modo Visual, com a tecla *v* seguida de algum comando de movimentação:

```
v    - Inicia ou termina o Modo Visual, utilize por exemplo as setinhas para marcar o texto
V    - Inicia o Modo Visual, mas para toda a linha
^V   - Inicia o Modo Visual colunado
vw   - Inicia o Movo Visual e marca a próxima palavra
```

```
v$   - Inicia o Movo Visual e marca até o fim da linha
v%   - Estando com o cursor em cima de um delimitador, "{}[]()", inicia o modo visual e seleciona tudo que estiver entre
       dois delimitadores, eles inclusive
vib  - Estando entre dois delimitadores em uma posição qualquer, "{}[]()", inicia o modo visual e seleciona o conteudo
       entre dois delimitadores
vi"  - Inicia o modo visual e seleciona o conteudo entre duas " "
vi'  - Inicia o modo visual e seleciona o conteudo entre duas ' '
vu   - Torna minúsculo o caracter/seleção
vU   - Torna maiúsculo o caracter/seleção
vwu  - Marca a palavra e utiliza o comando u para deixá-la em minúsculo
vwU  - A mesma coisa para deixá-la em maiúsculo
ggVG - Volta até o início do arquivo (gg), inicia o Modo Visual de Linhas
       (V) e seleciona até o final do arquivo (G)
```

É possível utilizar *u* e *U* em qualquer tipo de seleção!

6. DESFAZENDO ALTERAÇÕES

```
u  - Desfazer (vem de undo)
^r - Refazer  (vem de redo)
```

7. COPIANDO

Para copiar um texto no Vim você pode utilizar a seleção seguida do comando de cópia, ou utilizar a cópia combinada a algum comando de movimentação:

```
yy  - Copia toda a linha (o y vem do termo yank, algo como arrancar em português)
Y   - Copia toda a linha
yw  - Copia até o fim da palavra
y2j - Copia mais duas linhas abaixo
"+y - Copia para a área de transferência (ficando disponível para outros aplicativos)
v[faz alguma seleção]y - Copia a seleção realizada (não digite os colchetes)
```

8. COLANDO

```
p    - Cola a partir da posição atual (o primeiro caractere colado fica após o cursor)
P    - Cola na posição atual (o primeiro caractere colado fica onde está o cursor)
[p   - Colar antes
]p   - Colar depois
"+gp - Colar da área de transferência (disponível de outros aplicativos)
```

Em casos onde toda uma linha foi copiada, o pê minúsculo cola abaixo e o maiúsculo acima.

Uma forma alternativa e eficiente de colar o conteúdo da área de transferência (outro aplicativo) é usar o atalho do console *CTRL+SHIFT+V* no meu caso.

9. REGISTRADORES

Boas dicas em: http://pt.wikibooks.org/wiki/Vim/Usando_registros

Os acostumados em *CTRL-C* e *CTRL-V* perceberão que os métodos acima não permitem a troca de conteúdo com outros programas. Isso acontece porque ao copiar o texto com *y*, o conteúdo é colocado em um registrador anônimo (disponível somente no Vim).

```
:registers (ou :reg) lista o conteúdo dos registradores
```

Este recurso lhe dá, adicionalmente, a flexibilidade de ter diversos itens copiados simultaneamente. Para especificar o registrador onde você deseja disponibilizar o conteúdo copiado, prefixe o comando com aspas duplas (“) seguido do registrador (qualquer letra ou número).

Além das cópias explícitas com *y*, os registradores também são usados obscuramente pelo próprio Vim. Quando você deleta um texto, ele vai para o registrador anônimo, podendo nesse caso funcionar como um recurso de recortar.

9.1. Detalhes dos registradores

```
0     - Tem o último texto copiado
1 a 9 - Ficam os textos deletados (1 o mais recente, 9 o mais antigo)
-       sinal de menos, fazendo referência a algo pequeno, ficam os textos de menos de uma linha
        que tenham sido deletados. A exceção é o underscore, que não coloca o conteúdo em nenhum registrador
"dd   - Desta forma a linha deletada não vai para nenhum registrador
```

9.2. Manipulando registros

```
:let @a=@_ ..... Limpa o registro a
:let @a="" ..... Limpa o registro a
:let @a=@" ..... Salva registro sem nome *N*
:let @*=@a ..... Copia o registro para o buffer de colagem
:let @*=@: ..... Copia o ultimo comando para o buffer de colagem
:let @*=@/ ..... Copia a última busca para o buffer de colagem
:let @*=@% ..... Copia o nome do arquivo para o buffer de colagem
:reg       ..... Mostra o conteúdo de todos os registros
```

9.3. Copiar

```
"myy    - Copia toda a linha no registrador "m"
"mY     - Copia toda a linha no registrador "m"
"jye    - Copia até o fim da palavra no registrador "j"
"+yy    - Copia toda a linha na área de transferência
          (o sinal de mais deve ser digitado mesmo, ele é o registrador)
ggVG"+y - Vai até o início do arquivo (gg), inicia o Modo Visual de Linhas (V),
```

```
         vai até a última linha (G) e copia para a área de transferência (¿+y)
```

9.4. Colar

```
"+P         - Cola da área de transferência (o sinal de mais deve ser digitado)
"mp         - Cola do registrador eme
5"mp        - Cola do registrador eme cinco vezes (teclar SPC após ")
^r+registro - Cola no modo de inserção
^r++        - Cola o conteudo da área de transferência no modo de inserção (Ctrl+r+Shift++)
```

10. BUSCAR

```
/txt - (txt = termo a ser buscado)
n    - Localiza a próxima ocorrência
N    - Localiza a ocorrência na direção contrária
*    - Localiza palavra sob o cursor
funcionalidades:
:set hlsearch   ..... Destaca todos os termos encontrados (highlight)
:set nohlsearch ..... Como tantas outras opções do Vim, o prefixo ¿no¿ desabilita a funcionalidade
```

Para ignorar a diferença entra maiúsculas e minúsculas, basta incluir \c no termo da busca: /\ctxt - Realiza uma busca case insensitive do termo digitado /\Ctxt - Cê maiúsculo força a diferenciação de maiúsculas e minúsculas

```
:set ignorecase ..... Configura todas as buscas como case insensitive
```

É possível utilizar expressões regulares como padrão de busca!

11. SUBSTITUIR

A substituição de termos segue basicamente o padrão de expressões regulares, no formato *s/antes/depois/*. Ao substituir um termo por outro no Vim, precisamos ainda especificar onde a alteração deve ser realizada; para alterar um termo em todo documento, utilizamos o simbolo de porcentagem:

```
:%s/antes/depois/ ..... Substitui a primeira ocorrência dos termos localizados em todas as linhas
```

Os primeiros dois pontos iniciam um comando e o símbolo de porcentagem especifica onde a busca deverá ocorrer (nesse caso, em todo o documento). A barra funciona apenas como delimitador, o que é comum para quem está familiarizado com expressões regulares.

Caso o termo localizado apareça mais de uma vez na mesma linha, somente a primeira ocorrência é substituída, sendo necessário o uso da flag de Global Matching:

```
:%s/antes/depois/g  .... Com a letra g no final, especificamos a flag necessária para alterarmos
                         todas as ocorrências do termo em todas as linhas do documento
:%s/antes/depois/gc .... Com as letras gc no final, especificamos as flags necessárias para alterarmos
                         todas as ocorrências do termo em todas as linhas do documento, porém, com confirmação
                         de alteração ou não de cada ocorrência
```

Da mesma forma que utilizamos a porcentagem para abranger todo o documento, podemos especificar intervalos de linhas:

```
:1,10s/antes/depois/g ..... Altera todas as ocorrências entre as linhas 1 e 10
```

Nesses casos eu prefiro primeiro realizar a seleção com Modo Visual de Linhas e em seguida aplicar a seleção. Tente o seguinte:

```
Vkkk:s/antes/depois/g - Iniciamos o Modo Visual de Linhas (V), subimos três linhas (kkk),
                       começamos um comando (:) e realizamos a substituição (s/antes/depois/g)
```

As substiuições também suportam metacaracteres de expressões regulares, inclusive backreferences para recuperar o que foi casado em um grupo. Este assunto é um tanto quanto longo e merece melhor atenção em algum post futuro ou lendo o próprio manual.

Se existir uma seleção não se pode informar % antes do *s* ... :'<,'> s/antes/depois/gc

12. ABRINDO ARQUIVOS EM BUFFERS

Uma vez dentro do Vim, você pode carregar outros arquivos no que é chamado de Buffer.

```
:edit arquivo ..... Carrega o arquivo especificado em outro buffer
:e arquivo    ..... Por comodidade, basta utilizar a abreviação
gf            ..... Abre o arquivo especificado no texto onde está o cursor.
                     Exemplo: ~/.vimrc (tecle gf com o cursor em cima)
:buffers ..... Lista os buffers abertos, indicando o número do buffer e o caminho do arquivo
:ls      ..... Sinônimo para listar os buffers abertos
:buffer[número] ..... Troca para o buffer do número especificado (não digite os colchetes)
:b[número]      ..... Troca para o buffer do número especificado (não digite os colchetes)
```

Como lembrar os números é impraticável, você pode digitar parte do nome do arquivo (o suficiente para que ele seja o único encontrado). Vamos supor que você esteja com os arquivos consulta.php e consulta.tpl abertos em seus respectivos buffers:

```
:b php ..... Para abrir o buffer do arquivo consulta.php (abreviação de :buffer)
:b tpl ..... Para abrir o buffer do arquivo consulta.tpl
```

O acionamento dos buffers não está relacionado às extensão dos arquivos, apenas ao fato do padrão digitado identificar unicamente cada buffer. Imagine agora os arquivos Conexao.inc.php e Alunos.inc.php:

```
:b Con ..... Para abrir o buffer do arquivo Conexao.inc.php
:b Alu ..... Para abrir o buffer do arquivo Alunos.inc.php
```

Os termos *¿Con¿* e *¿Alu¿* já foram suficientes para identificar o buffer desejado.

Se ao invés de um arquivo você especificar um diretório para o comando *:edit*, o Vim abrirá uma lista de arquivos e diretórios para que você possa navegar e localizar o arquivo desejado. Utilize a tecla Enter para abrir o arquivo.

```
:bd         ..... Para fechar o buffer corrente (abreviação de :bdelete)
:bd[número] ..... Para fechar o buffer especificado (não utilize os colchetes)
```

13. TABS

```
:tabedit file ..... Abre arquivo em outra tab
:tabnew  file ..... Faz a mesma coisa
^PageUp/PageDown - navegação entre as tabs (antes/depois). (incompatível com tmux)
gt - Próxima
gT - Prévia
nt - Vai para tab 'n'
:tabmn ..... Move tab para posição n
:tabm0 ..... Move a tab atual para a posição 0
```

14. DIVIDINDO A JANELA

```
:split ..... Divide a janela na horizontal
:sp    ..... Divide a janela na horizontal
:vsplit ..... Divide a janela na vertical
:vs     ..... Divide a janela na vertical
:split arquivo - Abre o arquivo após dividir a janela horizontalmente (o mesmo para :vsplit na vertical)
^w[movimento] - Para focar outra janela; o movimento significa as setas direcionais ou uma das teclas hjkl
                (não digite os colchetes)
^wq - Para fechar a janela atual (o arquivo continua aberto no buffer)
^w= - Configura as janelas com o mesmo tamanho (se estiver utilizando o plugin Tabbar, terá um resultado estranho,
      já que as abas têm sua própria janela)
^w- - Diminui o tamanho da janela focada
^w+ - Aumenta o tamanho da janela focada
```

15. CONFIGURAÇÕES BÁSICAS

Finalizando, um breve comentário sobre as configurações que uso:

```
:colorscheme slate ..... Troca o esquema de cores para um de fundo escuro (especialmente agradável na versão gráfica)
:syntax on         ..... Habilita o highlight de sintaxe, praticamente a única coisa que eu utilizava para programar em outras IDEs mais completas
:set tabstop=4     ..... Configura a largura visível de tabulações com \t
:set expandtab     ..... Utiliza espaços ao invés de \t para tabulação
:set shiftwidth=4  ..... Configura o número de espaços na tabulação
:set smarttab      ..... Habilita facilidades na tabulação, útil para mim ao dar backspace para remover tabulações com espaço
:set number        ..... Mostra o número das linhas
```

As configurações são relativas à execução do Vim, algumas individuais por buffer. Para não ter que refazer as configurações toda vez que abrir o Vim, basta colocá-las no arquivo ~/.vimrc. No arquivo, não é necessário colocar os dois pontos a cada comando.

Por muito tempo utilizei o GVim, versão gráfica do Vim, e mantinha algumas configurações extras pra ele no ~/.gvimrc:

```
:set guioptions-=m        ..... Para remover a barra de menu
:set guioptions-=T        ..... Para remover a barra de ferramentas
:set guifont=Monospace\ 8 ..... Porque a fonte padrão é enorme!
```

Atualmente tenho usado o Vim no terminal mesmo, para poder desfrutar de um fundo transparente ou com alguma imagem legal :)

16. MARCAS

Para fazer marcas (referências) no texto, tecle *m* seguido de uma letra. Por exemplo, *ma* marca com *a* a linha onde se encontra o cursor.

Se você deletar a linha contendo a marca ela também será deletada.

O comando :delmarks (ou :delm) pode ser usado para deletar markas específicas.

```
--------------------------------------------------------------------------------
 Command          Description
--------------------------------------------------------------------------------
 :delmarks a    ..... Delete mark a
 :delmarks a-d  ..... Delete marks a, b, c, d
 :delmarks abxy ..... Delete marks a, b, x, y
 :delmarks aA   ..... Delete marks a, A
 :delmarks!     ..... Delete all lowercase marks for the current buffer (a-z)
--------------------------------------------------------------------------------
```

17. MACRO

É uma mão-na-roda para tarefas repetitivas que possuem um padrão.

```
qletra - Para inciar o modo de gravação (ex: qm)
q      - Encerra gravação da macro
@letra - Executa a macro (ex: @m)
```

As macros ficam visíveis nos registadores do Vim.

18. MISC

18.1. Dividindo a janela

```
^ws - Divide a janela atual em duas (:split)
^wo - Faz a janela atual ser a única (:only)
```

Caso tenha duas janelas e use o atalho acima *wo* lembre-se de salvar tudo ao fechar, pois apesar de a outra janela estar fechada o arquivo ainda estará carregado, portanto faça:

```
:wall ..... Salva todos 'write all'
:qall ..... Fecha todos 'quite all'
```

18.2. Abrindo e fechando janelas

```
^wn - Abre uma nova janela, sobrepondo a atual (:new)
^wq - Fecha a janela atual, e termina após a última (:quit)
^wc - Fecha a janela atual (:close)
```

18.3. Manipulando janelas

```
^ww - Alterna entre janelas (salta de uma para outra)
^wj - Desce uma janela j
^wk - Sobe  uma janela k
^wr - Rotaciona janelas na tela
^w+ - Aumenta o espaço da janela atual
^w- - Diminui o espaço da janela atual
```

18.4. File Explorer

Para abrir o gerenciador de arquivos do Vim use:

```
:Vex ..... Abre o file explorer verticalmente
:e.  ..... Abre o file explorer na janela atual após abrir chame a ajuda <F1>
```

Para abrir o arquivo sob o cursor em nova janela coloque a linha abaixo no seu ~/.vimrc

```
let g:netrw_altv = 1
```

Caso queira pode mapear um atalho "no caso abaixo F2" para abrir o File Explorer.

```
map <F2> <esc>:Vex<cr>
```

Maiores informações:

```
:help buffers
:help windows
```

18.5. Dicas

Caso esteja editando um arquivo e nele houver referência a outro arquivo tipo:

```
/etc/hosts
```

Você pode usar este comando para abrir uma nova janela com o arquivo citado

```
^wf
```

Mas lembre-se que posicionar o cursor sobre o nome do arquivo (veja também mapeamentos)

18.6. Conhecendo o tipo de arquivo

```
:set filetype? - para conhecer o tipo do arquivo.
                 Muito útil para criar arquivos 'snippets' (trechos) para o plugin snipmate
```

19. LATEX, BIBITEX E JABREF

Integração do programa JabRef com o (g)Vim.

Estando o (g)Vim em modo de inserção (ou não) é possível inserir uma citação a partir do JabRef no (g)Vim. Para isso inicie o (g)Vim como abaixo:

```
vim --servername vim nome_do_arquivo.tex
gvim --servername vim nome_do_arquivo.tex
```

Para ficar permanente é necessário criar/editar o arquivo .bashrc no home:

```
alias vim='vim --servername vim'
```

20. CODIFICAÇÃO: VERIFICANDO E ALTERANDO (UTF8, LATIN1)

```
:set fileencodings<ENTER>        ..... Ver o tipo de codificação
:set fileencodings=utf8<ENTER>   ..... Alterar a codificação para utf8
:set fileencodings=latin1<ENTER> ..... Alterar a codificação para latin1
```

21. ATALHOS (MAO NA RODA) ESTANDO NO MODO DE INSERÇAO

```
^o   - O Vim entra em um modo de inserção especial "(inserção)" permitindo uma única tarefa
       nos outros modos (normal e comando), após isso, imediatamente retorna ao modo de INSERÇÃO normal
^x^l - Abre uma janela popup permitindo escolher qualquer linha já existente no arquivo
^y   - Repete, caracter por caracter, a linha acima a do cursor
^e   - Repete, caracter por caracter, a linha abaixo da do cursor
^w   - Apaga (na linha) a última palavra digitada anterior ao cursor
^u   - Apaga (na linha) da posição do cursor até o início da linha
^t   - Indenta linha atual
^d   - Desidenta linha atual
```

22. CORES (COLORIMENTO, SYNTAX, HIGHLIGHTERS)

No Debian os esquemas de cores ficam armazenados na pasta: /usr/share/vim/vim73 (ou vim74)/colors

Existe um excelente plugin com muitas opções modernas de cores que recomendo: Colour-Sampler-Pack.

Nele minha meu esquema de cores preferido é o **jellybeans**, pois se adapta muito bem tanto no terminal (Vim) quanto sob GVim ou mesmo fora do ambiente gráfico X, mantendo o mesmo padrão de cores em todas as situações.

23. IMPRESSÃO

```
:harcopy
:ha
:ha=3
```

24. MAPEAMENTO E REMAPEAMENTO

Mapeamentos são um ponto importante do Vim, com eles podemos controlar ações com quaisquer teclas, mas antes temos que saber que:

```
---------------------------------------------------------
Tecla    : Tecla mapeada
---------------------------------------------------------
<CR>     : Enter
<ESC>    : Escape
<LEADER> : normalmente \
<BAR>    : | pipe
<CWORD>  : Palavra sob o cursor
<CFILE>  : Arquivo sob o cursor
<CFILE>< : Arquivo sob o cursor sem extensão
<SFILE>  : Conteúdo do arquivo sob o cursor
<LEFT>   : Salta um caractere para esquerda
<UP>     : Equivale clicar em 'seta acima'
<M-F4>   : A tecla (ALT = M) mais a tecla F4
<C-f>    : Control f
<BS>     : Backspace
<space>  : Espaço
<TAB>    : Tab
---------------------------------------------------------
```