**VI (V**isual **I**nterface**)– Editor de textos para Linux**

**Introdução**

"vi" é a sigla para "**V**isual **I**nterface". A origem desse nome se deve ao seguinte fato: quando o vi foi criado (começo da década de 80), não era comum existirem editores de textos como nos dias de hoje. Naquela época, você digitava um texto mas não podia vê-lo! Isso mesmo! Em 1992, foi criado o vim (Vi IMitator), um clone fiel ao vi, porém com muitas outras funcionaliades, que só foram sendo adicionadas. Algum tempo depois, o vim passou a ser chamado de `Vi IMproved' (vi melhorado).

O vim é um dos editores de textos mais utilizados no mundo Unix. Em alguns sistemas, existe um link simbólico (/bin/vi) apontando para o /usr/vim. Em outros, o /bin/vi é o executável, só que executa diretamente o vim. Muita gente acha que usa vi, mas na verdade utiliza o vim, e eles têm algumas diferenças. O que você verá abaixo fala sobre o vim.

O vim é um editor de textos muito poderoso, ele pode: abrir vários arquivos ao mesmo tempo, possui sistema de autocorreção, auto-identação, seleção visual, macros, seleção vertical de texto, uso de expressões regulares, sintaxe colorida, e muito mais. Ele não é exclusivo do Unix, ou seja, pode ser executado em outras plataformas, como Amiga, MacOS, Sun, Windows entre outras.

Existe também o gvim, que é o vim em modo gráfico, com todas as funcionalidades do vim em pleno funcionamento, o que muda é apenas o modo gráfico mesmo.

O vim possui vários modos, ou seja, estados em que ele se encontra. São eles: modo de inserção, comandos, linha de comando, visual, busca e reposição. Abordarei os dois principais:

**Modo de inserção e de comandos**

Para identificar o modo (estado) do vim, basta visualizar o rodapé da tela.

Agora, vamos à prática. Para executar o vim, utilize:

$ vi => Abre o vim vazio, sem nenhum arquivo e exibe a tela de apresentação.  
$ vi arquivo => Abre o arquivo de nome "arquivo".  
$ vi arquivo + => Abre o arquivo de nome "arquivo", com o cursor no final do mesmo.  
$ vi arquivo +10 => Abre o arquivo de nome "arquivo", com o cursor na linha 10.  
$ vi arquivo +/Copag => Abre o arquivo de nome "arquivo", na primeira ocorrência da palavra "Copag".

Ao executar o vim, ele inicia diretamente em modo de comando. Para comprovar, é só olhar na última linha (rodapé) e não vai haver nada lá. Isso quer dizer que você não conseguirá escrever nada, pode digitar a vontade que só vai ouvir beeps. Para começar a escrever, pressione "i" em seu teclado. O vim entra em modo de inserção, que você comprova (como falado anteriormente) pelo rodapé da tela, onde fica a seguinte marcação:

- - -- INSERT --

Suponha que você já digitou o bastante, e quer salvar, por segurança. Pressione a tecla ESC para voltar em modo de comandos. E veja os comandos para salvar/sair:

:w => Salva o arquivo que está sendo editado no momento.  
:q => Sai.  
:wq => Salva e sai.  
:x => Idem.  
ZZ => Idem.  
:w! => Salva forçado.  
:q! => Sai forçado.  
:wq! => Salva e sai forçado.

Então, você editou uma boa quantidade de textos e quer salvar:

:w

Agora, quer voltar a editar o texto:

i

Lembre que utilizando o "i" para inserção, a mesma se inicia inserindo texto antes do cursor. Veja agora outros subcomandos de inserção de texto:

A => Insere o texto no fim da linha onde se encontra o cursor  
o => Adiciona uma linha vazia abaixo da linha corrente  
O => Adiciona uma linha vazia acima da linha corrente  
Ctrl + h => Apaga último caracter à esquerda

Voltando ao modo de comando:

Veja agora subcomandos para movimentação pelo texto:

Ctrl + f => Passa para a tela seguinte.  
Ctrl + b => Passa para a tela anterior.  
H => Move o cursor para a primeira linha da tela.  
M => Move o cursor para o meio da tela.  
L => Move o cursor para a última linha da tela.  
h => Move o cursor para caracter a esquerda.  
j => Move o cursor para linha abaixo.  
k => Move o cursor para linha acima.  
l => Move o cursor para caracter a direita.  
w => Move o cursor para o início da próxima palavra (não ignorando a pontuação).  
W => Move o cursor para o início da próxima palavra (ignorando a pontuação).  
b => Move o cursor para o início da palavra anterior (não ignorando a pontuação).  
B => Move o cursor para o início da palavra anterior (ignorando a pontuação).  
0 (zero) => Move o cursor para o início da linha corrente.  
^ => Move o cursor para o primeiro caracter não branco da linha.  
$ => Move o cursor para o fim da linha corrente.  
nG => Move o cursor para a linha de número "n"

(susbstitua n pelo número da linha)..  
G => Move o cursor para a última linha do arquivo.

**Copiando e colando textos no vim (utilizando o mouse)**

Selecione o texto necessário com o botão esquerdo do mouse. Quando você for colar, saiba que o texto será colado a partir de onde se encontra o cursor (esse que aparece, às vezes piscando e às vezes não, quando você está digitando). Para colar, depois de ter selecionado o texto, você pode utilizar uma dessas opções:

1) Pressionando o botão direito do mouse;  
2) Pressionando o botão direito + botão esquerdo juntos;  
3) Pressionando o botão do meio do mouse (mouse de 3 botões);

Observação: Lembre-se que o vim deve estar no modo de inserção.

**Usando o modo visual do vim**

Entre no modo visual: v  
Agora, utilize as teclas direcionais (setas) do teclado, para selecionar o texto desejado.  
Pressione e cole, utilizando a tecla "p" (paste).

Veja agora como apagar um determinado texto:

Utilizando normalmente as teclas Backspace/Delete, ou entrando em modo visual (v) e pressionando a tecla Delete.

Você pode remover até o final de uma palavra, utilizando: dw  
Pode também remover até o final de uma frase: d$

**Desfazendo uma ação**

É claro que você pode desfazer uma ação que você considera errado, ou que errou ao digitar o texto. É só utilizar: u  
Se você precisar voltar o texto na tela, utilize as teclas Ctrl + r.

**Subcomandos para localização de texto**

/palavra => Procura pela palavra ou caracter acima ou abaixo do texto.  
?palavra => Move para a ocorrência anterior da palavra (para repetir a busca use "n").  
n => Repete o último comando utilizando / ou ?.  
N => Repete o último comando / ou ? ao contrário (baixo para cima).  
Ctrl+g => Mostra o nome do arquivo, o número da linha corrente e o total de linhas.

**Mais opções para remoção de caracteres**

x => Apaga o caracter onde o cursor estiver.  
dd => Apaga a linha inteira onde o cursor estive  
D => Apaga a linha a partir da posição do cursor até o fim.  
J => Une a linha corrente à próxima.  
:5dd => Removeas próximas 7 linhas a partir da posição do atual do cursor (qualquer número).

**Mais para copiar e colar**

:yy => Copia a linha onde o cursor se encontra.  
:5yy => Copia as próximas 5 linhas a partir da posição atual do cursor.  
:p => Cola o que foi copiado na linha abaixo do cursor atual.

**Opções para substituição de textos**

rCARACTER => Substitui o caracter onde o cursor se encontra pelo caracter especificado em CARACTER.  
RTEXTO => Substitui o texto corrente pelo texto digitado (sobrepõe).  
cw => Remove a palavra corrente para substituição.  
cc => Remove a linha corrente para substituição.  
C => Substitui o restante da linha corrente, esperando o texto logo após o comando.  
J => Une a linha corrente à próxima.  
:s/velho/novo => Substitui a primeira ocorrência de "velho" por "novo" na linha corrente.  
:% s/velho/novo => Substitui em todo o arquivo (%) a primeira ocorrência de "velho" por "novo" em cada linha.  
:% s/velho/novo/g => Substitui em todo o arquivo (%), todas (g) as ocorrências de "velho" por "novo".  
:% s/velho/novo/gc => Igual ao anterior, mas pedindo confirmação para cada substituição.  
:% s/^String[0-9]//gc => Expressões regulares também funcionam, como no sed.  
:% s/./\u&/gc => Converte para maiúsculas (\u) o primeiro caracter (.) de cada linha.

**Abreviações**

:ab => Mostra todas as abbr.  
:abc[lear] => Remove todos.  
:iab => Apenas para modo de inserção.  
:iabc[lear] => Tira todos de inserção.  
:cab => Apenas p/modo de comando ( : ).  
:cabc[lear] => Tira todos os modos de comando.  
:una vc => Tira ab para vc.

Observação: Pontuação, espaço ou o ENTER, disparam a expansão de uma abreviação. Porém, Ctrl+] também pode ser usado, para expandir sem adicionar caracteres.

**Opções para o comando SET**

:set  
autowrite aw => Salva a cada alteração.  
backspace bs => Comportamento backspace (1 ou 2).  
errorbell eb => Campainha de erro.  
expandtab et => Troca tab por espacos.  
fileformat=dos ff => Converte o arquivo para DOS.  
hidden hid => Preserva o buffer.  
hlsearch hls => Elumina a última procura.  
ignorecase ic => Case insensitive na busca.  
incsearch is => Ilumina procura enquanto digita.  
laststatus=2 => Mostra linha de estado.  
lazyredraw lz => Não redesenha em macros.  
lines=N => Múmero de linhas na tela.  
magic => Usar mágicas na procura de padrões.  
number nu => Mostra núm da linha.  
report=N => Mostra aviso quando N linhas mudaram (0=sempre).  
showcmd => Mostra o comando que se está fazendo.  
showmatch sm => Mostra o casamento de {},[],().  
smartcase scs => Assume "noic" quando tiver maiúsculas.  
textwidth=N => Quebra de linha do texto.  
undolevels ul=N => Guarde os N últimos comandos para desfazer (padrão=1000).  
vb t\_vb= => Retira o "beep" de erro.

**Agora invertendo maiúsculas/minúsculas**

5~ => Inverte os 5 próximos caracteres.  
g~$ => Inverte todos os caracteres até o fim da linha.  
seleciona, u => Converte para minúsculas.  
seleciona, U => Converte para maiúsculas.  
seleciona, ~ => Inverte.

Observação: Onde está escrito "seleciona", é para fazer utilizando o modo visual (v).

Agora veja como definir coluna de quebra de linha (problema que eu tive quando iniciei no aprendizado do vim):

:set textwidth=N

Se você já estiver num arquivo pronto:

:set wm=5 => O número 5 aqui são as colunas que serão "cortadas".  
gqG => Até o final do arquivo.

Vamos ver agora o que podemos fazer pressionando a tecla "Ctrl":

É claro que é segurando Ctrl + .

No modo de COMANDO:

A => Incrementa um número (Add)  
X => Decrementa um número  
S => ScrollLock  
L => Redesenha tela  
V => Modo visual (Visual Vertical)  
G => Status do arquivo  
M => Início da próxima linha  
E => Linha abaixo sem mover cursor  
Y => Linha acima sem mover cursor  
N => Próxima linha (Next)  
P => Linha anterior (Previous)  
F => PageDown (Forward)  
B => PageUp (Backyard)  
U => PageUp / 2 (Up)  
D => PageDown / 2 (Down)

Agora, no modo de INSERÇÃO:

A => Insere o último texto inserido  
I => TAB  
S => ScrollLock  
H => BackSpace  
T => 2 tab's no início da linha (Two Tabs)  
V => Anula expansão do próximo caractere  
J => Enter - quebra de linha  
M => Enter - quebra de linha  
L => Redesenha tela  
R => Insere conteúdo do registrador [a-z] (Veja abaixo)  
K => Insere um dígrafo (Veja abaixo)  
N => Procura palavra no texto atual (Next)  
P => Procura palavra no texto atual (Previous)  
Y => Copia caractere que está acima (Yank)

Veja os caracteres especiais:

ga => Mostra o código da letra sobre o cursor.  
:dig => Mostra todos os dígrafos disponíveis (tabela).

Exemplos: Para fazer um º, use Ctrl+K,-,o ("Ctrl"+"K"+"-"+"o").  
Para fazer um ½, use Ctrl+K,1,2 ("Ctrl"+"K"+"1"+"2").

**Trabalhando com arquivos e janelas múltiplas**

Você pode abrir múltiplos arquivos, por exemplo:

$ vim arquivo1 arquivo2

E pode alternar entre as janelas. Veja:

:wn => Grava o atual e vai ao próximo.  
:wN => Grava o atual e vai ao anterior.  
:args => Mostra todos os arquivos atuais.  
:qa => Sai de todas as janelas de uma vez.  
:all => Abre todos os arquivos em janelas individuais.

Tecla chave das janelas = Crtl+W  
j, seta abaixo => Move para janela abaixo.  
k, seta acima => Move para janela acima.  
o => Apenas esta janela, fecha todas as outras (Only).  
+, - => Muda o tamanho da janela.  
= => Deixa todas as janelas com tamanhos iguais.

**Os registradores**

"[a-z] => Use o registrador [a-z] para o próximo delete, cópia ou cola.  
:reg => Mostra o conteúdo de todos os registradores.  
:reg [a-z] => Mostra o conteúdo do registradores [a-z].

Observação: O [a-z] pode ser: 0-9a-z%#:.-="

Marcas:

m[a-z] => Marca em [a-z] a posição corrente do cursor.  
`[a-z] => Vai até a marca [a-z].  
`` => Vai até a posição anterior ao último pulo (alterna).  
:marks => Mostra as marcas ativas.

**Fazendo gravação de seqüência de comandos**

q[a-z] => Inicia a gravação de uma seqüência no registrador [a-z].  
q[A-Z] => Inicia a gravação, adicionando no registrador [a-z].  
q => Pára a gravação.  
@[a-z] => Executa a seqüência do registrador [a-z] (5 vezes? 5@a)

Dica: Pode-se colocar o @[a-z] dentro da própria gravação do q[a-z]! Assim ele é executado recursivamente. Muito útil quando há uma procura de padrões na gravação. faz para todas as ocorrências.

**Mapeamentos**

:map :r!date => Mapeamento em modo de comando.  
:imap :r!date => Mapeamento em modo de inserção.  
:cmap r!date => Mapeamento em modo linha de comando.  
:vmap :r!date => Mapeamento em modo visual.

Exemplos:

"html: negrito no trecho selecionado  
:vmap d`pa # html: negrito no trecho selecionado

"liga/desliga autoIndent  
:map ,si :set ai!:echo "autoIndent="&ai

"mostrar os espaços em branco no fim das linhas  
:map / \*$^M

Através dos mapeamentos é possível "encurtar" comandos, ou seja, abreviá-los. Conheça as sintaxes:

Comment => Ciano  
Constant => Roxo  
Identifier => Ciano  
PreProc => Azul escuro  
Special => Vermelho  
Statement => Amarelo  
String => Roxo  
Type => Verde

Todo => Preto, fundo marrom  
Error => Branco, fundo vermelho  
Ignore => Preto, fundo preto! - esconde

**Utilizando o recurso de expandtab**

Mas, o que isso faz? Transforma todos os TABs em espaços. Podemos ativar dentro do próprio vim, utilizando o comando:

:set expandtab

Para desabilitar:

:set noexpandtab

Podemos colocar também no arquivo ~/.vimrc a seguinte linha:

set expandtab

O arquivo ~/.vimrc pode ser usado para muitas configurações, e essa é uma delas. Existe ainda o arquivo ~/.exrc, mas não entraremos em detalhes.

Podemos incluir a saída de um comando no vim, utilizando:

:r!comando

Por exemplo:

:r!rpm -q kernel

Incluiria o seguinte resultado, dentro do seu texto (isso na minha máquina):

kernel-2.4.18-3

Dicas diversas do vim:

:xit => Igual :wq, mas só grava se tiver sido alterado algo no arquivo.

:map N\_ARQ ^R=expand("%:t:r")^M  
Imprime no arquivo o próprio nome do arquivo editado quando N\_ARQ é digitado.

Agora, veja como alinhar o texto:

:left  
:right  
:center

E para fazer uma busca de 2 palavras ao mesmo tempo:  
/palavra1\|palavra2

O vim é um editor de textos com centenas de opções, comandos, strings... Enão dá para abordar tudo aqui. Por isso, a melhor maneira de conhecê-lo a fundo é usando-o.

Agora, veja quem colaborou com a criação das dicas, e de onde eu tirei tanta informação:

*Revista do Linux (Ed. 04 - Vi, vim e venci - Aurélio);  
Página do Aurélio (www.verde666.org);  
Dicas-L (expandtab) (www.Dicas-L.unicamp.br);  
Dicas enviadas pelo amigo Uziel (uhfn@ig.com.br)  
Vários tutoriais recolhidos pela internet.*

Agradeço a todos aqueles que contribuiram direta e/ou indiretamente com os tutoriais.

*Dicas desenvolvidas e gentilmente cedidas por* [*Copag*](mailto:copag@uai.com.br)*. Publicado em 30/06/2003.*