EDIÇÃO DE TEXTO COM

VI / EX

ÍNDICE

1. Introdução

Qualquer ficheiro de texto ASCII, um programa ou um documento, pode ser criado e/ou modificado, usando um editor de texto. O funcionamento dos editores pode ser classificado de várias maneiras, sendo as mais usuais as descritas de seguida:

- Edição por linha: a edição é efectuada através duma sequência de perguntas e respostas. A acção desejada sobre o ficheiro só é efectuada depois do utilizador introduzir uma linha de comando. A visualização da alteração não é automática. Um exemplo de um editor de linha é o edlin do MS-DOS.

- Edição por écran: neste tipo de editor, a acção desejada sobre o ficheiro é imediatamente efectuada assim que o utilizador dá o comando. A visualização, neste caso, já é automática, ou seja, assim que alteramos o texto vemos o écran respectivo a ser alterado.

No sistema operativo UNIX existem dois editores: ed e o ex. O primeiro é um editor de linha enquanto que o ex pode tomar vários aspectos.

O editor ex tem várias formas de ser invocado: ex,edit,vedit,vi e view. A úniva difernça entre elas é a maneira de se iniciar a edição. Assim, enquanto que ex e o edit são

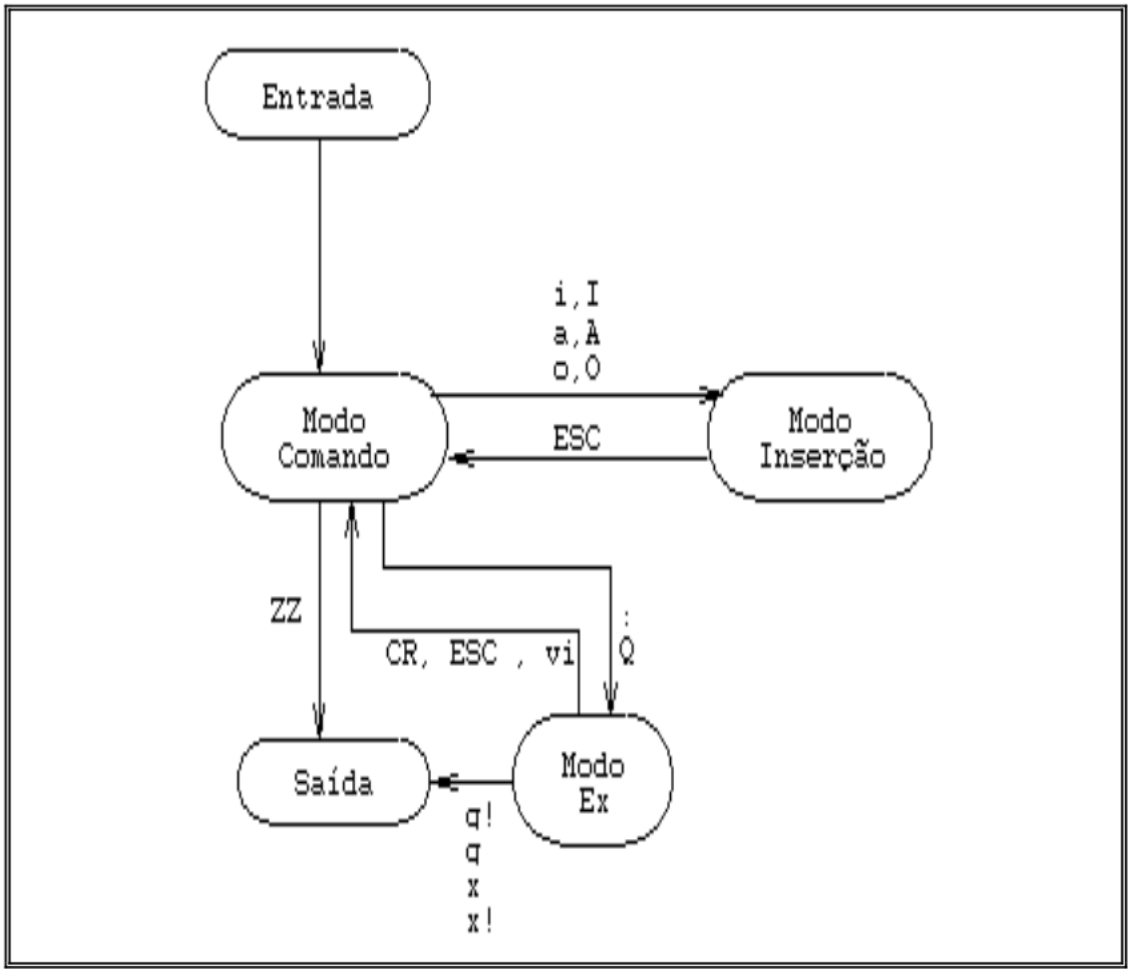
editores de linha, os restantes são do tipo editores de écran. O editor que nos vai interessar, devido a ser o mais utilizado, é o vi. Os restatntes serão referenciados no fim deste manual.

O vi (do inglês "visual") combina as caracteristicas de um editor de linha com as caracteristicas de um editor de écran, de modo a obter um vasto leque de poderosas ferramentas de edção de texto.

O vi não é um processador de texto nem permite incorporar gráficos nos nossos textos. No entanto é um editor independente do tipo de terminal que se está a usar, bastando para isso, definir a variável de ambiente TERM (o que é normalmente feito de forma automática durante o login).

2. Modos

Existem três modos principais de funcionamento do editor vi: inserção, comando e modo ex. Por defeito, sempre que entramos no vi ele fica em modo comando. A figura que se segue pretende mostrar como é que se passa de um modo para o outro, sendo uma primeira abordagem a este editor. Os comandos que permitem mudar de modos serão explicados a seu tempo.



3. Interacção

Existem várias mensagens e sinais que o editor nos transmite:

apito Pode-se ouvir um apito nas situações seguintes:

- Fazer um comando errado

- Interromper um comando com DEL

- Sempre que premirmos DEL

Nota: Sempre que ouvirmos o apito, sabemos que nos encontramos em modo comando.

~ O vi mostra este caracter na 1ª coluna, em todas as linhas depois do fim do buffer.

@ Algumas vezes aparece este caracter a substituir uma linha que foi apagada. Este caracter não faz parte do texto, servindo só de indicador.

última linha Esta linha é usada para mostrar :

- Mensagens

- Comandos que comecem por por / ,? ,! e :

- Output dos comandos do ex.

- Indicação do modo de trabalho (ver secção das opções).

4. Terminal

Já se referenciou o facto de o editor vi usar a variavel de ambiente TERM para identificar o tipo de terminal com que vai trabalhar. O modo de definir essa variável depende do interpretador de comandos utilizado:

Bourne Shell

Temos que inserir duas linhas de comando no nosso ficheiro *.profile*.

```
TERM=tipo

export TERM
```

C Shell

Temos que inserir uma linha de comando no nosso ficheiro *.login*.

```
setenv TERM tipo
```

Nota: "*tipo*" é o tipo de terminal utilizado (vt100, wy30, etc).

5. Entrada e Saída

A sintaxe do vi é: vi [opções] [ficheiro ...]

Algumas das opções são:

-r

Usa-se quando o sistema "vai abaixo" durante a edição dum ficheiro. O vi tenta recuperar a última versão salva do ficheiro.Se não indicarmos o ficheiro o vi mostra uma lista de ficheiros salvos.

-R

É a opção de "read-only".Podemos editar o ficheiro mas não alterá-lo.

-wn

Edita o ficheiro com uma janela de dimensão *n*.

+r

Edita o ficheiro e coloca o cursor no início da linha *n*.

+

Edita o ficheira e coloca o cursor na última linha.

+/string

Edita o ficheiro e move o cursor para a primeira ocorrência de *string*.

+"comando do ex"

Edita o ficheiro, executando o comando especificado.

Para sair do editor temos várias hipoteses:

ZZ

Sai do editor e actualiza o ficheiro se este tiver sido alterado.

:q

Sai do editor se o ficheiro não tiver sido alterado.

:q! Sai do editor sem actualizar as alterações efectuadas no ficheiro.

:x Sai do editor actualizando as alterações efectuadas.

:x! Se o editor tiver sido invocado com a opção -R ,embora o ficheiro seja "read-only", actualiza as alterações.

No caso do ficheiro não ter permissão de escrita só o administrador de sistema poderá, com este comando, sair do ficheiro actualizando-o.

Q Volta ao modo ex. Volta-se ao vi executando o comando *vi*.

6. cursor

Direita

l <Barra de espaços> ou <seta para direita>

Move o cursor um caracter para a direita. Se este comando for precedido por um número o cursor mover-se-á tantos caracteres para a direita quantos os indicados pelo número. Quase todos os comandos podem ser precedidos por um número, razão pela qual daqui para a frente só se referenciar aqueles que não o permitam.

Esquerda

h <BS> ou <seta para a esquerda>

Próxima linha

j Ctrl-N <LF> <seta para baixo> <RETURN> +

Todos estes comandos posicionam o cursor na próxima linha na mesma coluna onde se encontrava anteriormente excepto *<RETURN>* e + . Estes últimos, colocam o cursor no início da próxima linha.

Linha anterior

k Ctrl-P <seta para cima> -

Movem o cursor para linha anterior na mesma coluna, excepto o comando - que move o cursor para o início da linha acima da corrente.

Princípio da linha

^ 0

Enquanto que o comando *0* coloca o cursor no primeiro caracter da linha corrente, o comando ^ move o cursor para o primeiro caracter da linha que não é um espaço ou um tab. Não é possivel preceder este comando com um número.

Fim de linha

$

Coloca o cursor sobre o último caracter da linha corrente. Se este comando for precedido por um número *n* o cursor irá para o fim da *n-1* linha abaixo da linha corrente.

Linha n

[n]G

Coloca o cursor no princípio da linha indicada por *n*. Se o número for omitido o cursor ficará posicionado no início da última linha do ficheiro.

Coluna n

[col]|

Move o cursor para a coluna *col* da linha corrente. Se *col* for omitido o cursor é movido para a 1ª coluna da linha corrente.

Próxima palavra

w W

Move o cursor para o início da próxima palavra. O conceito de palavra varia consoante nós utilizamos o comando *w* ou o comando *W*. Assim, utilizando *w*, uma palavra é definida como um conjunto de caracteres alfanuméricos separados por pontuação ou caracteres separadores (tab, newline ou espaços). Se utilizarmos *W* uma palavra é um conjunto de caracteres excepto os separadores.

Palavra anterior

b B

Move o cursor para o princípio da palavra anterior. Aqui também existe, tal como no comando anterior, duas definições de palavras consoante se utiliza a letra minúscula ou maiúscula. Se o cursor já estiver posicionado numa palavra ele é colocado no início dela.

Fim de palavra

e E

Coloca o cursor no fim da próxima palavra, sendo as diferenças entre *e* e *E* análogas às dos casos anteriores

Oração

( )

Coloca o cursor no princípio (*(*) ou no fim (*)*) de uma oração. Entende-se por oração uma sequência de caracteres separados por pontuação (. ? !) seguidos por 2 espaços ou um newline. As orações são também delimitadas por parágrafos e delimitadores de secções(ver à frente).

Parágrafo

{ }

Move o cursor para o início ({) ou fim (}) de um parágrafo. Por defeito um parágrafo é definido pelas macros do formatador de texto nroff (.IP, .LP, .P, .QP e .bp). Um parágrafo também começa após linhas vazias.

Secção

[[ ]]

Move o cursor para o início (*[[*) ou fim (*]]*) de uma secção. Por defeito uma secção é definida pelas macros do nroff .NH e .SH. Uma secção também começa com um formfeed(Ctrl-L) ou por um { no início de uma linha. Neste último caso este comando é bastante útil quando se escreve um programa na linguagem C.

Parêntesis e Chavetas - ( ) [ ] { }

%

Testa se os parêntises estão balanceados. Se o cursor estiver posicionado num destes caracteres o comando *%* posiciona o cursor no caracter correspondente((-) [-] {-}). No caso de não exitir (não está balanceado) o editor envia um sinal sonoro.

Topo do écran

[offset]H

Move o cursor para o canto superior esquerdo do écran (topo). Se for precedido por *offset* o cursor irá para para *offset-1* linha a contar do topo do écran.

Meio do écran

M

Move o cursor para a linha correspondente ao meio do écran.

Fim do écran

[offset]L

Move o cursor para a última linha do écran. Se for precedido de *offset* o cursor ficará colocado na linha *offset-1* a contar do fim do écran.

Marca

'letra `letra

Podem-se criar marcas virtuais em qualquer ponto do nosso ficheiro através do comando *m* . Bastará fazer *mletra* , onde *letra* é um caracter de a-z . Assim, *'letra* posiciona o cursor no início da linha que contém a marca, enquanto que *`letra* move o cursor para a posição exacta da marca.

7. Écran

Página

[tamanho]Ctrl-U [tamanho]Ctrl-D

Ctrl-U faz scroll de meio écran para cima e Ctrl-D faz scroll de meio écran para baixo. Se for indicado o tamanho os comandos fazem scroll correspondente ao tammanho indicado. Este valor de *tamanho* é memorizado e em futuras utilizações destes comandos, será esse o tamanho por defeito.

Scroll

Ctrl-F Ctrl-B Ctrl-y Ctrl-e

Ctrl-F faz scroll de uma página (écran) para a frente (*F*orward) e Ctrl-B faz scroll de uma página para trás (*B*ackward).

Ctrl-y faz scroll de uma linha para baixo, deixando o cursor na mesma linha e Ctrl-e faz scroll de uma linha para cima, deixando o cursor na mesma linha.

Status

Ctrl-G

Mostra na linha de status do editor (última linha) o nome do ficheiro que se está a editar, se o ficheiro foi modificado ou não, a linha corrente, o número de linhas do ficheiro e a percentagem do ficheiro (em linhas) da posição do cursor.

Re-desenhar écran

Ctrl-R Ctrl-L

O comando *Ctrl-L* redesenha todo o écran enquanto que *Ctrl-R* só elimina as linhas que são indicadas por @: quando se apaga uma linha por vezes aparece o caracter @ no sitio da linha.

8. inserção

Qualquer destes comandos permitem entrar noutro modo de funcionamento do vi - modo inserção. Só se sai deste modo premindo a tecla de <escape> entrando assim em modo de inserção.

Inserir texto

i a

O comando *i* permite inserir texto imediatamente antes do cursor enquanto que o comando *a* permite inserir texto na posição imediatamente a seguir ao cursor.

Inserir no fim da linha corrente

A

Inserir no início da linha corrente

I

Abrir nova linha

o O

Abre uma nova linha e permite inserir texto a partir dessa posição . O comando *o* abre uma linha abaixo da linha corrente enquanto que o comando *O* abre uma linha acima da linha corrente.

9. Remoção

Caracter

x X

O comando *x* apaga o caracter onde o cursor está colocado e o comando *X* apaga o caracter imediatamente atrás da posição do cursor.

Linha

dd

Apaga a linha corrente

Até ao fim da linha

D

Apaga o texto desde a posição do cursor até ao fim da linha corrente.

Uma das muitas facilidades deste editor é a conjugação de comandos. Assim, no caso de querermos apagar texto, podemos conjugar os comandos de movimento de cursor com a letra *d* (*dmovimento de cursor).*

exemplos:

dw apaga uma palavra

dfa apaga texto até a próxima ocorrência de *a* (ver à frente)

d0 apaga texto desde a posição do cursor até ao início da linha corrente

d`a apaga o texto compreendido entre a posição do cursor e a marca *a*.

10. Modificação

Desfazer a última inserçào ou remoção de texto

u U

O comando *u* desfaz a última inserção ou remoção de texto, enquanto que o comando *U* repõe a linha currente tal como ela estava antes independentemente do número de alterações que lhe fizeram.

Repetir o último comando de inserir ou apagar texto

. *(ponto)*

O comando . repete o último comando de inserção ou de remoção.

Mudar texto

cmovimento do cursor C cc

Estes comandos permitem a mudança do texto indicado, pelo texto inserido até se premir ESC. O fim do texto a mudar é apontado pelo caracter $. O comando *cmovimento do cursor* muda todo o texto até ao sitio indicado pelo movimento do cursor; O comando *C* modifica o texto até ao fim da linha corrente; O comando *cc* modifica toda a linha corrente.

Substituição de texto

rcaracter R s S

O comando *r* seguido de um caracter substitui o caracter corrente pelo indicado , sem sair para o modo de inserção. O comando *R* ou *s* substitui o caracter corrente pelo texto inserido até se premir ESC. Por último, o comando *S* substitui a linha corrente pelo texto inserido. Quer se utilize o comando *s* ou *S* o fim do caracter/texto a substituir é indicado pelo caracter $.

Mapear caracter

~

Se o caracter sob o cursor corresponder a uma letra, este comando transforma-a em maiúscula se ela for minúscula e vice-versa.

Filtro

!movimento de cursor cmd

Filtra o texto abrangido pelo cursor até ao *movimento de cursor* pelo comando do sistema operativo *cmd.*

exemplo:

!5jsort Ordena 5 linhas através do comando sort

Assim que se introduzir o comando de movimento do cursor o ! é passado para a última linha do écran, ficando à espera de um comando do sistema operativo.

Junção de linhas

J

Este comando junta a próxima linha com a linha corrente

Deslocação de texto

>movimento de cursor <movimento de cursor >> <<

Move o texto ou a linha corrente (>> <<)um número de espaços indicados pela opção shiftwidth (ver à frente). O comando > ou >> move para a direita enquanto que os outros movem o texto para a esquerda.

11. Blocos

O vi tem disponiveis 26 buffers (a-z), 9 buffers (1-9) organizados em stack e 1 buffer sem nome. Sempre que for apagado texto o editor coloca esse texto no seu stack, ao qual nós temos acesso. O buffer sem nome é um buffer que contém o último texto apagado.

Pôr texto

["caracter alfanumerico]p ["caracter alfanumerico]P

Estes comandos poem o texto dos buffers indicados por *caracter alfanumerico* no nosso ficheiro. Se for omitido o nome do buffer, considera-se que o buffer é o buffer sem nome. O comando *P* coloca o texto antes da linha/caracter corrente e o comando *p* coloca o texto depois da linha/caracter corrente. A diferença entre colocar o texto antes/depois do caracter corrente ou linha corrente, depende se o texto do buffer contem uma parte de uma linha ou contém uma ou mais linhas

Escrever num buffer

["letra]ymovimento do cursor ["letra]yy ["letra]Y

Todos estes comandos copiam o texto indicado para o buffer indicado por *letra*. Se o nome do buffer for omitido será considerado o buffer sem nome. Se o buffer indicado for uma letra minúscula, mesmo que não esteja vazio, o editor escreve nesse buffer. No entanto se nós pretendermos escrever o texto no fim do buffer basta-nos indicar o nome desse buffer com uma letra maiúscula. O bloco de texto a guardar depende do comando. Assim temos que ymovimento do *cursor* abrange todo o texto entre a posição corrente do cursor e a posição indicada pelo movimento. O comando *yy* ou o *Y* escreve apenas a linha corrente.

exemplos:

12yy escreve 12 linhas no buffer sem nome

"Ay´m acrescenta ao buffer a o texto compreendido entre o cursor e a marca m.

12. Pesquisa

Pesquisa

/[conjunto]/[offset]RETURN ?/[conjunto]?[offset]RETURN

O comando / pesquisa na direcção da posição corrente para o fim do ficheiro enquanto que ? pesquisa na direcção inversa. Sempre que chegar ao fim ou ao princípio do ficheiro e não tiver encontrado o *conjunto,* o vi regressa ao princípio ou ao fim, respectivamente e prosegue a pesquisa. No caso de não encontrar o *conjunto* o editor indica-o na linha de status.

conjunto pode conter alguns metacaracteres e denomina-se expressão regular:

^ o conjunto está no início de uma linha

$ o conjunto está no fim de uma linha

. Qualquer caracter excepto o newline

[] um dos caracteres do conjunto []

\< o conjunto está no início de um palavra

\> o conjunto está no fim de uma palavra

Exemplos de comandos de pesquisa:

a[a-f,A-S]d todas as palavras cujo 1° caracter é um a, o 3° um d e o 2° um caracter compreendido entre a e f ou A e S.

[^a-b]. todas as palavras que não comecem por a ou por b.

^abc a palavra abc e que esteja no início de uma linha.

abc$ a palavra abc que esteja no fim de uma linha

Exemplos do uso de offset:

/palavra/- linha anterior à que contem palavra

/palavra/+4 4ª linha a contar da linha que contem palavra

Próximo conjunto

n / *N* ?

n ou / pesquisam na direcção do último comando de pesquisa enquanto que *N* ou ? pesquisam na direcção inversa à indicada pelo último comando de pesquisa.

Pesquisar caracter

fcaracter Fcaracter tcaracter Tcaracter ; ,

O comando *fcaracter* pesquisa o caracter na linha corrente a partir da posição do cursor para o fim e, no caso desse caracter existir, coloca o cursor nessa posição. O comando *Fcaracter* pesquisa desde a posição do cursor para o princípio. Os comandos *tcaracter* e *Tcaracter* são semelhantes mas não posicionam o cursor sobre o *caracter*. O comando (;) repete a pesquisa do caracter e (,) repete a pesquisa em direcção contrária.

13. Ex

Se estivermos em modo comando e fizermos *:* entramos em modo ex. Muitos dos comandos que se podem executar neste modo, são identicos aos do editor de linha ed. Saimos deste modo com RETURN ou ESC.

Estes comandos são nomes em inglês e o vi aceita as suas abreviaturas. Por exemplo: podemos substituir o comando read por *r*, o comando substitute por s, o comando set por se, etc.

A sintaxe geral destes comandos e´a seguinte:

[endereço][comando][!][parametros][count][flags]

Um endereço é um conjunto de linhas e para referenciar esse conjunto temos alguns caracteres especiais:

. representa a linha corrente

n representa a linha n

$ representa a última linha do buffer

% representa todo o buffer. É uma abreviatura de 1,$

+n -n representa um offset relativo à linha corrente. Por exemplo, as formas .+3 +3 ou +++ são semelhantes e referenciam a 3ª linha abaixo da linha corrente.

/conjunto/

?conjunto? representa a linha que contém o *conjunto*

´x representa a linha que contém a marca x

Exemplos:

1,5 Da linha 1 até à linha 5

1,/ola/ Da linha 1 até à linha que contém a palavra ola

,100 Da linha corrente até à 100ª linha abaixo

Editar vários ficheiros

Existem alguns comandos que permitem a fácil edição de vários ficheiros simultaneamente.

args Mostra a lista dos ficheiros que estão a ser editados

Exemplo:

fich1 fich2 [fich3] fich4

Aqui o ficheiro corrente é fich3.

f Dá várias indicações: o nome do ficheiro corrente, se foi modificado ou não,se é "read-only", o número de linha corrente, o número de linhas do ficheiro e a percentagem do ficheiro (em linhas).

f fich

O ficheiro corrente passa a ser *fich*, o qual não está editado e se quisermos actualizá-lo teremos que fazer o comando :w!

n Edita o próximo ficheiro da lista

n! Edita o próximo ficheiro da lista sem actualizar as últimas alterações.

n[+comando_de_editor] lista_de_ficheiros

lista_de_ficheiros passa a ser a nova lista de ficheiros e assim que se editar o 1°, o comando *comando_do_editor* é executado.

rew Volta-se às condições iniciais - edição do 1° ficheiro

rew! É o mesmo que o comando anterior só que não actualiza o ficheiro corrente.

Comando de edição

Existem dois caracteres especiais # e % que representam o nome ficheiro anteriormente editado e o nome ficheiro corrente, respectivamente. Estes caracteres podem ser utilizados em qualquer comando do modo ex.

e fich edita o ficheiro *fich* excepto se ele fôr um ficheiro binário,um directório ou um device. Se omitirmos *fich,* o ficheiro a editar será o ficheiro corrente.

e! fich edita *fich* sem actualizar o ficheiro corrente.

e +n fich edita *fich* posicionando o cursor na linha *n*.

*Ctrl-^* é equivalente a :e#RETURN

Comandos de escrita

w fich Escreve as alterações feitas no buffer no ficheiro *fich*, caso ele exista. Se omitirmos *fich, o ficheiro a actualizar é o ficheiro corrente.*

w>> fich Acrescenta o buffer ao fim do ficheiro *fich.*

w! fich Escreve o buffer no ficheiro *fich* mesmo que este exista.

w !cmd Passa as linhas especificadas, para o comando do sistema operativo *cmd*. O output do comando não é inserido no ficheiro corrente.

Comandos de leitura

r fich Copia o texto de *fich* para o ficheiro corrente. Se *fich* fôr omitido o ficheiro considerado é o ficheiro corrente.

r !cmd Lê o output do comando do sistema operativo *cmd* para a linha abaixo da corrente.

Comandos globais e de substituição

A sintaxe do comando global é a seguinte:

g/conjunto/cmds

Exemplos:

g/s1/p Mostra todas as linhas que contêm a string *s1*.

g/s1/s//s2/ Substitui a primeira ocorrência de *s1* em todas as linhas que a contêm, por *s2*.

g/s1/s//s2/g

Substitui todas as ocorrências de *s1* por *s2*.

g/s1/s//s2/gp

Igual ao anterior, só que mostra todas as linhas alteradas.

g/s1/s//s2/gc

Antes de substituir *s1* por *s2* pede a confirmação. Em caso afirmativo responde-se com y.

g/s0/s/s1/s2/g

Em todas as linhas que contêm *s0* substitui *s1* por *s2*.

g!/conjunto/cmd

v/conjunto/cmd

Executa o comando do editor *cmd* para todas as linhas que não contenham *conjunto*.

s/s1/s2/opções

Substitui *s1* por *s2* .Se a opção g(global) fizer parte desta linha de comando todas as ocorrências de *s1* são substituidas por *s2*. A outra opção,c (confirm) pede a confirmação antes da substituição.

Comandos de manipulação de texto

co linha copia as linhas especificadas, para a linha *linha*. *Se* linha fôr 0, representa a primeira linha. Se fôr (.) representa a linha corrente.

Exemplo:

1,4 co 5 copia o conjunto de linhas da 1 à 4, para a linha 5

m linha move as linhas especificadas para a linha *linha*.

Exemplo:

,10 m 20 move 10 linhas a partir da corrente para a linha 20.

d remove o texto indicado pelo endereço.

Exemplo:

20,$ d apaga todas as linhas a partir da linha 20, inclusivé.

Nota: co é a breviatura de copy, m de move e d de delete.

Comandos de saída para o shell

cd dir O directório corrente passa ser *dir*.

sh Cria um sub-shell.

!comando Executa o comando do sistema operativo *comando* e volta ao editor.

!! Repete o último comando .

Outros comandos

abbr mapeia o 1° argumento pelos restantes.

Exemplo:

:abbr Ola Ola,como vai?

Sempre que estivermos em modo de inserção e escrevermos a palavra Ola, assim que premirmos ESC, Ola será substituido por Ola,como vai?.

unabbr Desfaz o comando anterior.

map map! Mapeia qualquer caracter ou sequência de escapes numa sequência de comandos. O comando *map* funciona em modo comando e o comando *map!* funciona em modo de inserção.

unmap Desfaz o comando *map*.

preserve Este comando só é usado em emergências, quando o comando *w* não funciona. O buffer é actualizado via sistema operativo e é guardado no directório /usr/preserve. Pode-se recuperar o ficheiro chamando o vi com a opção -r.

recover fich

recupera o ficheiro *fich* que tinha sido actualizado via *preserve*.

Opções

Para consultarmos o estado das opções do vi fazemos o comando *:set all* ou abreviando *:se all* , e ele mostra uma lista das opções disponiveis.

Se quisermos activar a opção fazemos:

:se opção

Para desactivar,

:se noopção

Se a opção necessitar de um valor numérico ou de uma string:

:se opção=valor

Para consultar o estado de determinada opção:

:se opção?

De seguida indicam-se as opções mais importantes.

autoindent ai defeito: noai

Serve para estruturar um texto. Faz a identação automática e para andar para trás na identação, em modo de inserção, faz-se Ctrl-D.

directory dir defeito: dir=/tmp

Especifica qual o directório na qual o vi guarda o buffer temporário. Se o directório indicada não possuir permissão de escrita, o editor sai abruptamente.

ignorecase ic defeito: noic

Faz a distinção das letras minúsculas e maiúsculas aquando da pesquisa de uma expressão regular.

list defeito: nolist

Quando a opção está *list* os caracteres especiais são visualizados. Por exemplo, o TAB aparece-nos com ^I e o fim de linha aparece como $.

magic defeito: magic

Quando está *magic, t*odos os metacaracteres numa expressão regular são interpretados como tal. Para retirar o seu significado usa-se a barra invertida (\). Se a opção está *nomagic* só os caracteres ^ e $ são considerados como metacaracteres. Aqui a barra invertida tem um significado diferente: força os caractere que por defeito são metacaracteres, a serem interpretados como tal.

mesg defeito: nomesg

Esta opção impede que um utilizador comunique connosco via write enquanto estivermos no editor. No caso de querermos a comunicação fazemos :se mesg.

number nu defeito: nonumber

Quando activada, precede cada linha do buffer pelo número correspondente.

paragraphs para defeito: para=IPLPPPQPP TPbp

Especifica os delimitadores de parágrafos para os comandos { e }.

report *defeito: report=5*

Qualquer comando que modifique mais do que o número de linhas indicadas nesta opção, avisa o utilizador de quantas linhas é que alterou.

scroll *defeito: scroll=12 (1/2 écran)*

Determina o número de linhas a fazer scroll, quando se executa o comando Ctrl-D.

sections *defeito: sections=SHNHHH HU*

Especifica quais os delimitadores para os comandos [[e]].

shell *defeito: shell=/bin/sh*

Determina qual o programa a ser executado quando se cria um sub-shell (:sh).

shiftwidth sw *defeito: sw=8*

Usado para determinar quantos espaços se desloca o cursor quando se executam os comandos >> ou << ou ainda, quando em modo de inserção e com a opção autoident, se faz Ctrl-D.

showmatch sm defeito: nosm

Quando esta opção está activa, sempre que se escreve (ou { o cursor é movido para o seu par ( ) ou }) fica aí 1 segundo e volta ao sítio da inserção.

showmode defeito: noshowmode

Quando activa indica-nos, no canto inferior direito do écran, se estamos em modo de inserção, no modo de substituição ou em modo comando.

tabstop ts defeito: ts=8

Determina quantos caracteres é que constituem um TAB.

term defeito: term=tipo de terminal

Esta opção serve para indicar ao vi qual o tipo de terminal que se está a utilizar(ver terminfo e termcap).

terse defeito: noterse

Se esta opção estiver activa todas as mensagens de erro do editor aparecem-nos numa forma reduzida.

warn *defeito: warn*

Esta opção permite o editor enviar a mensagem [No write since last change] sempre que nós executarmos um comando de shell (!comando).

wrapscan ws *defeito: ws*

Por defeito, quando o editor pesquisa uma expressão regular e chega ao fim sem encontrá-la, a pesquisa inicia-se a partir do início do ficheiro , e vice-versa. Se fizermos :se nows o editor não pesquisa a expressão regular além do fim ou do início do ficheiro.

14. Algo mais

Sempre que o se chama o vi, este inicializa-se de acordo com as inicializações por defeito e com aquelas que existem num ficheiro chamado .exrc. Este ficheiro encontra-se no directório $HOME.

Exemplo de um ficheiro .exrc:

```
se number

map ^a :se ai^M
```

Pelo que se vê, todas as linhas do ficheiro a editar são automaticamente numeradas e existe uma macro(^a) que seleciona a opção autoident. Quando pretendemos inserir um caracter de controlo temos que estar em modo de inserção e fazer Ctrl-V. A todo o caracter inserido depois de Ctrl-V é lhe retirado o significado especial. Não se pode utilizar os seguintes caracters: Ctrl-S e Ctrl-Q. Assim podemos criar as macros que quisermos para todas as vezes que utilizarmos o vi.

15. Invocações

Tal como se falou na introdução, além de invocar o editor Ex como vi, podemos também evocá-lo das seguintes maneiras:

view Igual ao vi só que a flag de readonly é inicialmente activada.

vedit É semelhante ao vi, sendo mais aconselháda aos principiantes. Põe activas as seguintes opções: showmode, novice, report=1.

edit É uma variante do Ex para utilizadores casuais.

16. Limitações

O *vi* tem os seguintes limites:

- 250k linhas num ficheiro
- 510 caracteres por linha
- 256 caracteres por linha de comando
- 100 caracteres para um comando de shell
- 63 caracteres para uma string

Nota: estes valores podem ser diferentes na versão do *vi* que está a utilizar. Inserimos esta informação para servir apenas de guia. Quando tiver problemas de tamanhos lembre-se destes valores.