Manual de VI

1 – Aprendiendo vi.

Existen algunas reglas básicas para los usuarios novatos de vi. Primero, ten un resumen de los comandos a mano y a todas horas. Segundo, consigue un buen libro que comprenda todos los aspectos de vi. Este documento no es la mejor forma para aprender vi (al menos no hasta el momento).

1.1 - ¿Qué juegos me ayudarán a aprender vi?

Esto podría parecer un poco estúpido, pero existen realmente algunos juegos para sistemas Unix que pueden ayudarte a enseñarte vi. Estos juegos te ayudarán principalmente con lo básico. Aunque no conozco ningún juego que te ayude con todos los comandos de vi, conozco algunos que te enseñarán a usar jul para moverte con el cursor por el documento. NetHack, es un juego que te ayudará concretamente a esto, es un juego bastante completo y que te puede entretener y divertir durante mucho tiempo. Algunos otros, sin desmerecerlos, son: rouge, moria, omega, worm, and snake.

1.2 - ¿Cuál es la diferencia entre los modos Comando e Inserción?

A menudo calificado como una de los principales problemas de vi, y calificado igualmente a menudo como uno de sus grandes fuertes, vi diferencia entre un "*modo comando*" y un "*modo inserción*". Entender esta diferencia es VITAL para aprender vi. Cuando uno llama a vi, él se inicia en modo comando. En este modo, uno se puede mover por el fichero, y realizar comandos para cambiar ciertas áreas del texto, cortar, copiar y pegar secciones del texto y muchas más cosas. El modo inserción es donde uno puede realmente insertar texto. En otras palabras, el modo comando se usa para moverse por el documento, y el modo inserción se usa para introducir texto en el documento.

Comandos tales como: **a, i, c, C, O, o** y otros le pasarán a uno del modo inserción al modo comando. <**Escape**> o **^C** le sacarán a uno del modo inserción para devolverle al modo comando. Familiarízate con esta distinción. Es una de las cosas que hacen a vi diferentes del resto de otros editores.

1.3 – Un momento, mi teclado no tiene la tecla "ESC" ¿qué hago?

Intenta teclear **^[** en su lugar, si tu teclado tiene una tecla <**Meta**>, intenta esa. Si no funciona ninguna de estas, intenta **^3**.

1.4 - ¿Qué son todos esos gusanillos (~)?

Están ahí solamente para permitirte saber donde acaba tu fichero, no forman parte realmente de tu documento, y no te tienes que preocupar por ellos.

1.5 – No estoy acostumbrado a utilizar hjkl, ¿alguna sugerencia?

Primero, si tu terminal está correctamente configurado, deberías ser capaz de poder utilizar las teclas de cursor. Sin embargo, si piensas que estarás utilizando vi mucho, entonces tiene más sentido que aprendas hjkl, ya que son más rápido de teclear.

1.6 - ¿Cómo me salgo sin salvar?

:q! Lo hará. Si vi parece haber frozen, asegúrate de que no has pulsado **^S** por error. Para deshacer un **^S**, teclee **^Q**.

1.7 - ¿Cómo inserto un fichero?

:r <fichero>

Por ejemplo, para insertar el fichero /etc/motd, teclee **:r /etc/motd** desde el modo comando.

Esto insertará el fichero especificado en la posición o línea en la que estés situado en el fichero en el que estás trabajando. Si especificas un número antes de la **r**, insertará el fichero especificado en dicha línea del fichero actual.

1.8 - ¿Cómo busco texto?

/<texto> buscará hacia delante. **?<texto>** buscará hacia atrás. **??** o **//** repetirá la última búsqueda. Además, en vi, **n** buscará la siguiente ocurrente. **N** repetirá la última búsqueda pero cambiando el sentido de búsqueda. Las expresiones regulares pueden ser utilizadas dentro de las búsquedas.

1.9 - ¿Cómo búsco una secuencia de control?

/^V^<secuencia>

^V le dirá a vi que tome el siguiente carácter literalmente, y no lo coja como un comando.

1.10 - ¿Cómo formateo el texto?

Si tu computadora tiene el programa **fmt** instalado, todo lo que necesitas hacer es teclear !}fmt desde el modo inserción. Esto volverá a justificar el texto desde la posición actual hasta el final de párrafo. Si tu máquina no tiene **fmt**, necesitas encontrar un programa similar.

1.11 - ¿Cómo copio texto?

De acuerdo, esto puede ser un poco complicado. Intentar sacar en claro las ideas que puedas y también es muy conveniente que experimentes con lo que vayas aprendiendo.

"<letra>yy copiará una línea de texto al registro <*letra*>. (Un registro podríamos decir que es una posición de memoria intermedia en la que vi guarda el texto que eliminaste, copiaste o marcaste para que se guardase en dicha posición de memoria). <*letra*> debe estar entre **a** y **z**.

"<letra>dd eliminará una línea y la colocará en el registro <*letra*>.

Puedes usar un número antes de ***dd*** o ***yy*** para especificar el número de líneas que deseas eliminar o copiar respectivamente. Utilizando una <*letra*> en mayúsculas, añadirá el texto al registro de la misma letra pero en minúscula, manteniendo lo que tuviese dicho registro anteriormente.

"<letra>p pondrá el texto a continuación del cursor.

"<letra>P pondrá el texto antes del cursor.

Si el registro contiene el principio o final de una línea, la línea se colocará en otra línea según sea apropiado. La **Y** puede ser utilizada como abreviatura para **yy**. Además, **y$**, **yH**, **yM**, etc son válidas, tales y como son los comandos equivalentes **d**. Para cortado y pegado rápido, no hace falta especificar registro. En este caso, no está permitido el añadir cosas al registro, y el registro será eliminado si se ejecuta otro comando de borrado (incluyendo la **x**)

Por ejemplo, para mover el párrafo anterior, uno debería irse a la parte superior o primera línea del párrafo, teclear **"a4dd**, mover el cursor a la posición en la cual uno desea colocar el párrafo, y entonces teclear **"ap** para poner el párrafo debajo de la línea actual.

También es posible seleccionar partes de líneas en lugar de líneas enteras. Se pueden establecer un carácter inicial (en mitad, final o principio de línea) un carácter final (en mitad, final o principio de línea) y llevarnos (mediante copia o eliminado) dicho trozo de texto al registro que deseemos. Trabajando así con distintos bufers en cada uno de los cuales podríamos tener un trozo de texto correspondiente a una función o un mensaje de error que se repite en el código fuente de un programa o script. Para hacer esto:

1. Sitúese encima del carácter inicial que quiere llevarse al registro y pulse **m<marca>** (donde marca es un carácter entre a y z). Esto se utiliza para marcar trozos de texto, que viene definidos por una posición o marca inicial (comando **m**)

2. Sitúese un carácter después del último carácter del bloque que quiere marcar y pulse **“<letra_registro>y`<marca>** (para copiar el bloque al registro <letra>) o **“<letra_registro>d`<marca>** (para eliminar el bloque y guardarlo en registro <letra_registro>)
3. Ya estamos listo para pegar el registro en la posición del texto que deseemos. Solamente tendremos que colocarnos un carácter antes de la posición en la que queramos volcar el registro o bufer y teclear **“<letra_registro>p**.

Aunque la secuencia para trabajar con marcas y distintos registros o bufers de memoria es un poco farragosa, con la práctica se acaba automatizando y resulta casi imprescindible y realmente útil. Nos evitamos tener que contar líneas para llevarnos una porción de texto a un determinad buffer o registro. Este tipo de atajos de teclado son los que diferencias a los auténticos expertos de los vulgares usuarios de vi.

1.12 – He cometido un error, ¿cómo deshacer lo hecho?

u deshará el último comando. **U** deshará los cambios de la línea actual. **:e!** volverá a cargar el documento actual desde la última vez que se salvó sin salvar ningún cambio. Además, el texto eliminado se almacena en los registros numerados desde 1 a 9. “<número>p restaurará la enésima cosa borrada. Puedes buscar rápidamente en los registros probando uno, pulsando **u** si no es el que buscas y probando con el siguiente. (Para hacer este proceso más rápido, vi utiliza **.** que difiere ligeramente del uso normal. En lugar de repetir el último comando, intentará con el siguiente registro, de forma que todo lo que necesitas hacer es: **“1p u. u. u.** etc. hasta que deshagas el borrado que quieras deshacer.)

1.13 – ¿Como escribir las sistintas secciones de un fichero en ficheros distintos?

:[m],[n]w <nombre_de_fichero> salvará desde la línea **m** a la línea **n** al fichero **nombre_de_fichero**. Este método de numeración de líneas funcionar con casi todos los comandos **:**. Si usas **:[m],[n]w >> <nombre_de_fichero>** se añadirán las líneas al fichero.

1.14 - ¿Qué son los comandos “dos puntos”?

Los comandos que siguen a los dos puntos (**:**), son comandos del editor ex. Estos permiten mucha flexibilidad y potencia. Por ejemplo, existen muchas formas de buscar y remplazar, y todos ellos tienen algunas similitudes (de hecho, ellos son de alguna forma los mismos …)

2 - ¿Cómo buscas y sustituyes?

Bien, existen unos pocos métodos. El más simple es:

:s/viejo/nuevo/g

Pero esto solo actúa sobre la línea actual… Así que:

:%s/viejo/nuevo/g

En general

:[rango]s/viejo/nuevo/[cgi]

Donde [rango] es cualquier rango, incluyendo **números**, **$** (final de fichero), **.** (la posición o línea actual), **%** (fichero actual), o simplemente dos números con una barra entre ellos. (o incluso +5 para representar las siguientes cinco líneas). [cgi] es cualquiera de las tres c, g, i o ninguna de ellas. **c** le dice a vi que te pregunte antes de realizar el cambio (confirm), **g** para que realice varias substituciones en una misma línea si se diese el caso, **i** para decirle a vi que no haga distinciones entre mayúsculas y minúsculas en la búsqueda. Si no especificas ninguno de estos caracteres después de la barra final, solamente se cambiará la primera ocurrencia de cada línea.

Mi método favorito es:
:g/foobar/s/bar/baz/g Esto busca foobar, y lo cambia por foobaz. Dejará sin cambios a jailbars, con el otro método no sería así. Este es mi método favorito, pero es un poco complicado de recordar. Por supuestos que también puedes utilizar patrones de búsqueda que sean expresiones regulares, y otros pocos comandos en la parte que sustituye al texto. Si utilizas **\(** y **\)** en el patrón para formar una secuencia de escape, puedes hacer muchas cosas nifty.

Por ejemplo:
:g/\(foo\)\(bar\)/s/\2/\1baz/g cambiará foobar por foobaz.
Las secuencias especiales permitidas son:

& cualquier cosa que encaje con la búsqueda o patrón

\[1-9] The contents of the 1st-9th \(\) pair

\u El siguiente carácter se pasará a mayúsculas

\U Los caracteres hasta **\e** o **\E** se convertirán a mayúsculas

\l El siguiente carácter se pasará a minúsculas

\L Los caracteres hasta **\e** o **\E** se convertirán a minúsculas

\[eE] finaliza la selección para convertir a mayúsculas o minúsculas

2.1 - ¿Cómo ejecuto un programa desde dentro de vi?

:!cmd ejecutará el comando **cmd**. **:sh** ejecutará un shell interactivo. Dentro de este shell, podrías si quisieses, ejecutar vi de nuevo. Esto es particularmente útil cuando estas editando makefiles y ficheros de configuración de programas en un intento de hacer que un programa compile. La ventaja sobre **:e** es que no necesitas salvar el fichero, y que estarás en el mismo sitio cuando salgas del shell (aconsejo salvar el fichero de todos modos …)

2.2 - ¡Hubo un apagón mientras escribía con el vi!

Bien, deberías haber recibido un correo con un mensaje al respecto, pero deberías ser capaz de recuperar el fichero tecleando **vi –r <nombre_fichero>** donde *<nombre_fichero>* es el nombre del fichero que estabas editando en el momento del apagón o caída del sistema. **vi –r** te dará una lista de ficheros que pueden ser recuperados.

2.3 - ¿Trucos para hacer más amigable la programación con vi?

:set ai hará que vi realice la autoindentación por ti.

:set sw=# donde # es un número que fijará el shiftwidth (ancho de tabulador).

Puedes utilizar <<, >> para desplazar una línea "***shifwidth***" caracteres a la izquierda o a la derecha respectivamente. Además, puedes utilizar **<%** para desplazar un conjunto {, (, [a la izquierda o **>%** para hacer el desplazamiento a la derecha. Debes estar en la parte superior del {, }, (,), [o] específico del par para desplazarlos.

:set sm mostrará el {, (or [correspondiente cuando escribes el },),] que lo cierra. Es la mejor forma de que no se te quede un carácter de bloque {, (, [sin su pareja.

:set lisp realizará algunos cambios que son útiles para la programación en lisp. () se moverá hacia atrás y forth a través de las expresiones-s, y {} se moverá sin detenerse en los átomos.

2.4 - Macros -- ¿Cómo las escribo?

Una macro es un nombre de comando que representa un conjunto de operaciones. Las macros suelen ser utilizadas cuando se ejecutan una serie de comandos frecuentemente y siempre en el mismo orden.

El nombre de la macro puede ser una o más letras o una tecla de función. Las macros se pueden definir para **modo comando** o **modo texto** según se desee.

Es importante señalar que la macro debe ser una palabra completa. No funciona cuando la macro se encuentra formando parte de una palabra. Para definir las macros se puede hacer de uno de las dos formas:

- Macros en **modo comando**: :**map [#] nombre_macro definición**
- Macros en **modo texto**: **:map ! [#] nombre_macro definición**

donde *nombre_macro* puede ser de hasta 10 caracteres (aunque es preferible que dicho nombre sea exclusivamente una letra) y *definición* de hasta 100 caracteres. Si se utilizan múltiples comandos en la *definición*, no es necesario que estén separados. Existen una serie de caracteres de control que se pueden utilizar dentro de la *definición* de las macros:

CARACTERES DE CONTROL	**CÓDIGO**
Escape	Ctrl-V
Nueva Línea	Ctrl-J
Retorno de Carro	Ctrl-M
Tecla de Escape	Ctrl-[

Los nombres de las macros no deberían ser letras de vi que se utilicen en modo comando, como por ejemplo la "**a**" para añadir, la "**o**" para insertar líneas, …

Recuerda utilizar ^V antes de cualquier carácter de control que pudieses utilizar.

:unmap <lhs> eliminará la macro.

:map! <lhs> <rhs> hará que <lhs> inserte <rhs> en el texto del documento.

2.5 - ¿Cómo hago que una tecla de función se comporte como una Macro?

Si en la definición de la macro se especifica **#n** donde **n** es un número entre 0 y 9, será mapeado a la tecla de función apropiada.

Ejemplo: **:map #1 :set all** → Cuando pulsemos la tecla de función F1 será igual que si tecleasemos ":set all" para que nos muestre todas las variables de entorno.

2.6 - ¿Hay alguna forma para abreviar texto?

Sí ,por supuesto. Esto es vi, puede hacer cualquier cosa.

:ab email ellidz@midway.uchicago.edu

lo hará, de forma que cuando teclees email como una palabra específica, se extenderá a la palabra entera sin abreviar (ellidz@midway.uchicago.edu)

2.7 - ¿Cómo chequeo la ortografía del documento actual?

Aquí está una macro para hacerlo. Estas deberían ser puestas en tu fichero .exrc. Es una bonita macro bastante simle, simplemente llama al comando **ispell** con el fichero actual. Por supuesto, para usar esto necesitas **ispell** en tu sistema. Para usarlo, simplemente teclea **V** con vi. (V no es utilizada por vi, por lo que es una buena tecla para esta función.)

map V :w^M:!ispell % ^M:e!^M^M

El Segundo ^M está puesto para que no sea necesario pulstar return después de que sea hecho el chequeo del fichero.

2.8 – Tengo un terminal hardcopy, ¿puedo todavía usar vi?

Está bien, espero que nadie pregunte realmente esto…

vi se iniciará en un modo específico, llamado el "modo abierto" en esta situación. Las cosas funcionan más o menos de la misma forma. Los caracteres eliminados aparecerán en tu pantalla en \'s. vi actuará como si el tamaño de la ventana fuese una única línea. **^r** redibujará la línea actual. **Z** redibuja la ventana alrededor de la línea actual.

3 – Información online sobre vi y bibliografía

Existe una FAQ con frecuentes preguntas la cual proporciona una lista de todos los ficheros que están online en el archivo vi. También da una lista de direcciones de espejos o mirrors del archivo vi. La dirección principal es alf.uib.no, pero está archivada en otros muchos, muchos sitios. Por favor, chequea esa faq antes de realizar un ftp a él, ya que es posible que exista un sitio mucho más cerca y rápido.

Un sitio mirror está disponible en: cs.uwp.edu /pub/vi y otro en monu6.cc.monash.edu.au/pub/Vi. Este sitio tiene muchos, muchos ficheros sobre vi, incluyendo unos pocos clones. Tiene también la distribución UCB de vi, y muchas macros útiles. Échales un vistazo.

Aquí te presento una pequeña bibliografía (en inglés por supuesto) de libros sobre este editor:

- "***The ultimate guide to the VI and EX Text Editors***" Hewlett Packard Company (authors) The Benjamin/Cummings Publishing Company, Inc. ISBN 0-8053-4460-8
- "***A Practical guide to the Unix System***" Mark G. Sobell Benjamin Cummings Publisher
- "***Learning the Vi Editor***" Linda Lamb O'Reilly & Associates ISBN 0-937175-67-6
- "***Unix power tools***" (particularly for macros) O'Reilly & Associates
- "***An Introduction to Display Editing with Vi' & 'EX reference Manual'***" in UNIX programmers Manual vol. II Bill Joy Berkeley Software

4 – Macros y trucos tontos para vi

Esta sección es para trucos y macros de vi tontos. En la actualidad, cualquier truco y macro de vi interesante es aceptable, mientras no sea demasiado grande. Añadiré cualquier truco o macro que se me sugiera que crea razonable.

4.1 – Trucos de vi tontos

xp

Esto elimina el carácter situado bajo el cursor, y lo pone a continuación. En otras palabras, intercambia la posición de dos caracteres.

ddp

Similar a xp, pero intercambia líneas.

uu

Deshace y vuelve a hacer el último cambio. (Esto te llevará a la última modificación del fichero sin cambiar nada del mismo.)

yyp

Duplica una línea

:g/.*/m0

Esto invertirá el orden de las líneas del fichero actual. m0 es el comando ex para mover la línea a la línea 0.

:v/./d o :g/^$/d

Elimina las líneas en blanco del fichero

:g/^[<ctrl-v><tab>]*$/d

Elimina todas las líneas que tienen solamente espacios en blanco.

:v/./.,/./-1join

Remplaza múltiples líneas en blanco por una única línea en blanco.

4.2 – Macros tontas

Nota: <crtl-x> significa que mientras mantienes pulsada la tecla control, pulses a continuación la tecla x.

1. Intercambiar un carácter con el carácter correspondiente de la línea superior: **map * kxjphxkP**
2. Partir una línea que es demasiado larga: **map g $80<ctrl-v><ctrl-v>|F r<ctrl-v><enter>**
3. Cambiar la mayúsculas por minúsculas y a la inversa en casi todas las palabras:
   map v ywmno<ctrl-v><esc>P:s/./\~/g<ctrl-v><enter>0"nDdd`n@n
4. Poner ` y ' alrededor de la palabra actual: **map *` i`<ctrl-v><esc>ea'<ctrl-v><esc>**
5. Poner ' y ' alrededor de la palabra actual: **map *' i'<ctrl-v><esc>ea'<ctrl-v><esc>**
6. Poner " y " alrededor de la palabra actual: **map *" i"<ctrl-v><esc>ea"<ctrl-v><esc>**
7. Poner ` y ' alrededor de la palabra actual: **map! `` <ctrl-v><esc>bi`<ctrl-v><esc>ea'**

8. Partir una línea en la posición en la que está el cursor, y poner un **>** al principio de la línea siguiente. Si el particionamiento de palabra al final de línea está habilitado, podría partir la última palabra de la primera línea:
   map g may0O<ctrl-v><esc>P`ay$:s/./ /g0i><ctrl-v><esc>`aPa<ctrl-v><esc>D
9. Inserta un carácter: **map g i$<ctrl-v><esc>r**
10. Hacer que control-x funcione como cortar, control-v como pegar, control-p como copiar. Deberás marcar el principio del área como **m** (utilice mm). (control-c no puede ser remapeado).
    map <ctrl-v><ctrl-x> "zd`m
    map <ctrl-v><ctrl-p> "zy`m
    map <ctrl-v><ctrl-v><ctrl-v><ctrl-v> "zP
11. Centra una línea: **map = 080i <ctrl-v><esc>$78hd0^D:s/ / /g<ctrl-v><enter>$p**
12. Redefine el tabulador de forma que inserta 5 espacios en lugar de una marca de tabulador:
    map! <ctrl-v><ctrl-i> <space><space><space><space><space>
13. Cambia la línea actual y la última línea (repítalo para invertir el fichero): **map v Gdd"Pj**
14. Imprime el documento a la impresora por defecto (para BSD sustituya lp por lpr): **map v 1G!Glp<ctrl-v><enter>u**
15. Fije # para activar y desactivar los número de línea:
    map \o# o:set nu<ctrl-v><enter>:set nonu<ctrl-v><esc>-:
    map \o# "wp<ctrl-v><enter>
    map \d# "w2dd
    map \x# "xdd@x"xpk
    map # ma3L\o#\x#\d#`a:<ctrl-v><enter>

5 – Referencia alfabética rápida de vi

Los puntos suspensivos … significan que es necesario especificar algo antes o después del comando, según sea conveniente. Esto es normalmente una tecla de movimiento de cursor (h, j, k, l, w, b, etc.) o un número de línea.

#	(donde # es un número) realizar un comando n veces...
:	ir a modo ex
)	comando siguiente
(	comando anterior
}	párrafo siguiente
{	párrafo anterior
]]	sección siguiente
[[	sección anterior
0	ir al principio de línea
$	ir al final de la línea
^	primer carácter que no sea blanco
+	primer carácter de la siguiente línea
-	primer carácter de la línea anterior
(espacio)	carácter siguiente
(return)	línea siguiente
/	búsqueda hacia delante
?	búsqueda hacia atrás
%	encuentra el paréntesis, llave o corchete correspondiente
,	cambia la dirección del último comando f, F, t, or T
;	repite el último comando f, F, t, or T
.	repite el último comando
`	ir a una marca (seguido por la letra de la marca)
'	ir al principio de la línea con marca
``	volver a la última marca o localización antes de una búsqueda
''	ir al begin of line de la marca o localiz. anter. previa a búsqueda
~	cambia de mayúsculas a minúsculas o viceversa el carácter actual
"	almacena en registro
!!	repite el último comando de la Shell
!	envía lo siguiente a comando, remplazando la salida (ejemplo !}fmt pasa el párrafo actual al comando fmt, y remplaza la salida con la que quiera que devuelta fmt.)
>>	desplaza el párrafo un shiftwidth a la derecha
<<	desplaza el párrado un shiftwidth a la izquierda
>%	desplaza a la derecha hasta que encuentra un matching (, [, o {
<%	desplaza a la izquierd hasta el siguiente(, [, o { que concuerde
a	añade después de la posición actual
A	añade al final de la línea
^a	(no se usa)
b	principio de la palabra anterior
B	principio de la palabra anterior, ignorando signos de puntuación
^b	repaginado de una pantalla
c	cambia hasta...

C	cambia hasta el final de la línea
^c	finaliza el modo inserción, no utilizado en modo comando
d	borrado hasta...
D	borra hasta el final de la línea
^d	scroll hacia abajo de 1/2 pág, mueve al shiftwidth anter. en modo insercion
e	final de la palabra
E	final de la palabra, ignorando signos de puntuación
^e	scroll de pantalla de una línea hacia abajo
f	encontrar...
F	encontrar hacia atrás...
^f	avanza la pantalla una página
g	no se utiliza
G	...Ir a [por defecto el final de fichero]
^g	muestra la línea de estado (nombre fichero, línea actual, líneas totales)
h	izquierda
H	primera línea de la pantalla
^h	espacio atrás en modo inserción, izquierda en modo comando
i	inserción antes de la posición actual
I	inserción antes del primer carácter distintos de blanco de la línea
^i	insercion de un tabulador en modo inserción, no se usa en modo comando
j	abajo
J	pega la siguiente línea al final de la línea actual
^j	abajo en modo comando, inserta una línea nueva en modo inserción
k	arriba
K	no utilizado
^k	no utilizado
l	derecha
L	última línea de la pantalla
^l	redibuja la pantalla
m	marca la posición en el registro
M	posiciona el cursor en la mitad de la pantalla
^m	retorno de carro
n	repite la última búsqueda
N	repite la última búsqueda, en dirección inversa
^n	abajo en modo comando
o	abre una línea debajo de la línea actual
O	abre una linea encima de la actual
^o	no utilizado
p	poner por debajo de la línea actual
P	poner por encima de la línea actual
^p	arriba en modo comando
q	no utilizado
Q	salir y ejecutar ex
^q	no utilizado
r	reemplazar el carácter actual
R	reemplazar caracteres hasta que se abandone el modo inserción

^r	redibujar la pantalla en modo comando
s	substituir
S	substituir la línea completa
^s	no utilizado
t	a...
T	hacia atrás a...
^t	mueve al siguiente shiftwidth.
u	deshacer el último cambio
U	deshacer los cambios de la línea actual
^u	repaginar media ventana
v	no utilizado
V	no utilizado
^v	no utilizado
w	comienzo de la palabra siguiente
W	comienzo de la palabra siguiente, ignorando los signos de puntuación
^w	no utilizado en modo comando, en modo inserción para desplazarse al principio de la palabra anterior
x	borra el carácter actual
X	borra el carácter anterior
^x	no utilizado
y	copia...
Y	copia la línea actual
^y	mueve la pantalla una línea hacia arriba
ZZ	escribe y sale
^z	no utilizado

5.1 – Opciones de la entrada en modo Comando (Comandos :)

(Nota: esta no es una lista establecida, solamente algunos de los comandos más importantes)

:r <fichero>	inserta el fichero “fichero” en el fichero abierto
:r !<comando>	inserta en el fichero actual la salida de la ejecución de “comando”.
:nr <fichero>	inserta el fichero “fichero” en la línea “n”
:!<fichero>	ejecución de comando desde el vi
:sh	salir a la shell de Unix
:so <fichero>	lee y ejecuta los comandos del fichero “fichero”.
:x	guardar y salir
:wq	guardar y salir
:11,12w <fichero>	escribe desde la línea 11 a la 12 al fichero “fichero”. Si no se especifica un fichero, se supone que es el fichero actual.
:w >&g;t <fichero>	añadir al fichero. Se pueden usar números de línea
:w!	Sobreescribe el fichero actual
:q	salir
:q!	Salir sin guardar los cambios
:e <fichero>	editar “fichero” sin salir del vi
:e!	descartar los cambios desde la última vez que se guardo el fichero
:n	editar el siguiente fichero.

:e +n <fichero>	editar el fichero “fichero” en la línea “n”.
:n <ficheros>	especifica “ficheros” como la nueva lista de ficheros a editar.
:e#	edita un fichero alternativo (si se usa :e <fichero>, el alternativo es el fichero original)
:args	muestra los ficheros que se van a editar
:rew	rebobina la vida de los ficheros al principio
:map m n	crea una macro (hace que m haga n)
:map! m n	crea una macro de modo insercion (hace que m haga n)
:unmap m	destruye la macro m
:unmap! m	destruye la macro de modo inserción m
:ab <1> <2>	abreviatura – sustituye <1> con <2> cuando se teclee como palabra
:unab <1>	eliminar la abreviatura <1>
:set <opción>	set <opción>...

5.2 – Opciones “set” (configuración entorno de vi)

Estas opciones se pueden fijar de tres formas diferentes:

- Desde la propia sesión del editor vi con el comando **:set**.
- En la variable de entorno del sistema operativo **EXINIT**
- Mediante el fichero **.exrc**

Para especificar las opciones desde dentro del editor vi utilice el comando ***set*** directamente. Las abreviaturas entre paréntesis podrían ser utilizadas.. Sintaxis:

:set <opción> <parámetro> (si se espera <parámetro>) Se pueden especificar múltiples opciones en una misma línea.

:set <opción>? visualiza el valor de <opción>

:set all visualiza el valor de todas las opciones.

Para opciones sin un valor, set no<opción> la desactiva.

OPCIÓN	VALOR POR DEFECTO	¿QUÉ HACE?
autoindent (ai)	noai	Hace que las nuevas líneas se indenten automáticamente a la misma posición de la línea anterior o superior.
autoprint (ap)	ap	Visualiza los cambios después de cada comando
autowrite (aw)	noaw	Salva el fichero automáticamente antes de :n, :!
beautify (bf)	nobf	Ignora todos los caracteres de control durante la introducción de texto (excepto tabulador (ctrl-i), newline (ctrl-j) y nueva página (ctrl-l))
directory= (dir=)	/tmp	Nombre del directorio para almacenar el buffer
edcompatible	noedcompatible	Utiliza las características ed-like en las substituciones.
errorbells (eb)	errorbells	Suene la campana cuando se produzca un error
exrc (ex)	Noexrc	Permite los ficheros .exrc fuera del directorio HOME
flash (fl)	noflash	Activa o desactiva un efecto de flash al mostrar un error.
hardtabs= (ht=)	8	Indica el número de caracteres que ocuparán los tabuladores
ignore case (ic)	noic	Ignora la diferencia entre mayúsculas y minúsculas en las expresiones regulares
lisp	nolisp	Activael modo lisp

list	nolist	Visualiza todos los tabuladores (^I) y finales de línea ($)
magic	magic	Habilita más expresiones regulares
mesg	mesg	Permite que se envén mensajes al terminal
modelines (ml)	noml	Permite ejecutar comandos del ex al entrar en un fichero
number (nu)	nonumber	Visualiza los números de línea en el fichero.
open	open	Permite abrir y visual
optimize (opt)	optimize	Hace que se puedan introducir líneas de más de 80 caracteres.
paragraphs= (para=)	IPLPPPQPPLIbp	Fija los delimitadores para { y }
prompt	prompt	Hace que al entrar al editor EX se indique el símbolo **:**.
readonly (ro)	noro	No puede escribir a menos que se le de un !
redraw	noredraw	Redibuja la pantalla cuando se hacen cambios.
remap	remap	Permite macros que apuntan a otras macros.
report=	5	Reporta los cambios si afectan a más de x líneas
scroll	1/2 window	Cantidad de pantalla para hacer el scroll cuando se recibe un scroll hacia abajo en modo comando. También, el número de líneas impresas por z. (z imprime 2*scroll)
sections=	SHNHH HU	Define el final de sección para [[y]]
shell= (sh=)	/bin/sh	Shell por defecto. Utiliza la variable de entorno SHELL, si tiene valor.
shifztwidth= (sw=)	8	Indica el tamaño de los tabuladores y de los desplazamientos producidos con los comandos <, > y la opción Ctrl-d del autoindent.
showmatch (sm)	nosm	Muestra el símbolo que encaja con el actual {, }, (,), [o]
showmode	noshowmode	Muestra el modo en el que estás dentro del editor.
slowopen (slow)	noslowopen	No actualiza la visualización inmediatamente después de la inserción.
tabstop= (ts=)	8	Fija la longitud de tabulador (en caracteres o posiciones)
taglength= (tl=)	0	Números de caracteres significativos para los tags o etiquetas (0 significa todos los caracteres)
tags=	tags /usr/lib/tags	Define el pathaname de los ficheros que contienen los tags o etiquetas.
term=		Fija el tipo de terminal
terse	noterse	Visualiza mensajes de error más cortos.
timeout (to)	timeout	Variable para la definición de macros de más de un carácter
ttytype=		Fija el tipo de terminal
warn	warn	Advierte si se han realizado cambios en el texto al ejecutar los comandos EX ! o shell.
window= (w=)		Números de líneas en la ventana en modo visual.
wrapmargin= (wm=)	0	Fija el margen derecho. Si es mayor que 0 partirá las palabras n espacios antes del borde de la pantalla.
wrapscan (ws)	ws	Hace que las búsquedas realizadas con los comandos Vi **/** y **?** sean circulares, es decir, que al llegar al final del texto siga buscando desde del principio y viceversa.
writeany (wa)	nowa	Anula el aviso que se da al grabar sobre un fichero ya existente.

Mediante EXINIT:

En el sistema operativo existe una variable de entorno denominada **EXINIT** para definir las características del editor vi. Esta variable se puede definir directamente en la línea de comandos, con lo que afecta a la sesión actual, o incluirla en el fichero **.profile** o en el **.login** según se utilice la Shell de Bourne o la C Shell respectivamente, para que se utilice cada vez que se entre en el sistema operativo.

Su formato depende del tipo de shell que utilice.

Para la Bourne Shell:

EXINIT='[set opción…|**map [!]** nombre_macro definición**] …'**

export EXINIT

Para la C Shell:

setenv EXINIT='[set opción…|**map [!]** nombre_macro definición **]…'**

6 – Haciendo ajustes en el fichero .exrc

Cualquier comando que pueda ser utilizado en el modo de entrada de comandos (comandos :), puede ser utilizado en el fichero .exrc, el cual será cargado automáticamente cada vez que inicies vi. Además, el comando fuente (so), abreviaciones (ab) y las macros pueden ser usadas. No pueden existir líneas en blanco en el fichero .exrc. Los comentarios de especifican comenzando la línea con “.

6.1 – Fichero .exrc de ejemplo

El fichero .exrc es un fichero real. Debido a eso, no fija todas las opciones que cualquier podría desear, está hecho para dar una idea de lo que se puede hacer. Sin embargo, es una buena base para que a partir de él le añadas tus propias opciones a tu gusto.

```
set wm=3
set sm
" abreviaturas para mi dirección de correo.
ab zidllem E. Larry Lidz - ellidz@midway
ab zidlleu E. Larry Lidz - ellidz@midway.uchicago.edu
map V :w^M:!ispell -x %^M:e!^M^M
```

Primero, dese cuenta de que no necesita especificar los dos puntos (:), se asumen.

La primera línea, “set wm=3” fija el margen derecho para que sea tres caracteres desde la derecha de la pantalla. Esto me da una división de palabras bastante vistosa.

La segunda línea activa la opción showmatch de forma que cuando tecleo un },) o] me muestra el otro {, (o [con el que encaja (si es que existe). Esto es muy útil en cualquier lenguaje de programación para no perderse en un anidamiento complicado de este tipo de símbolos de agrupación.

La tercera línea declara una abreviación, de forma que cuando teclee zidllem como una palabra, ésta automáticamente se expanderá a mi nombre y dirección de correo electrónico. La cuera es similar a excepción de que dá la dirección completa, no solamente la dirección local.

La quinta línea fija V para que al teclearla en modo comando, ejecute automáticamente ispell en el documento actual, pasándole el parámetro –x de forma que no salve ningún backup. (No me gusta llenar mis directorios con backups inútiles).