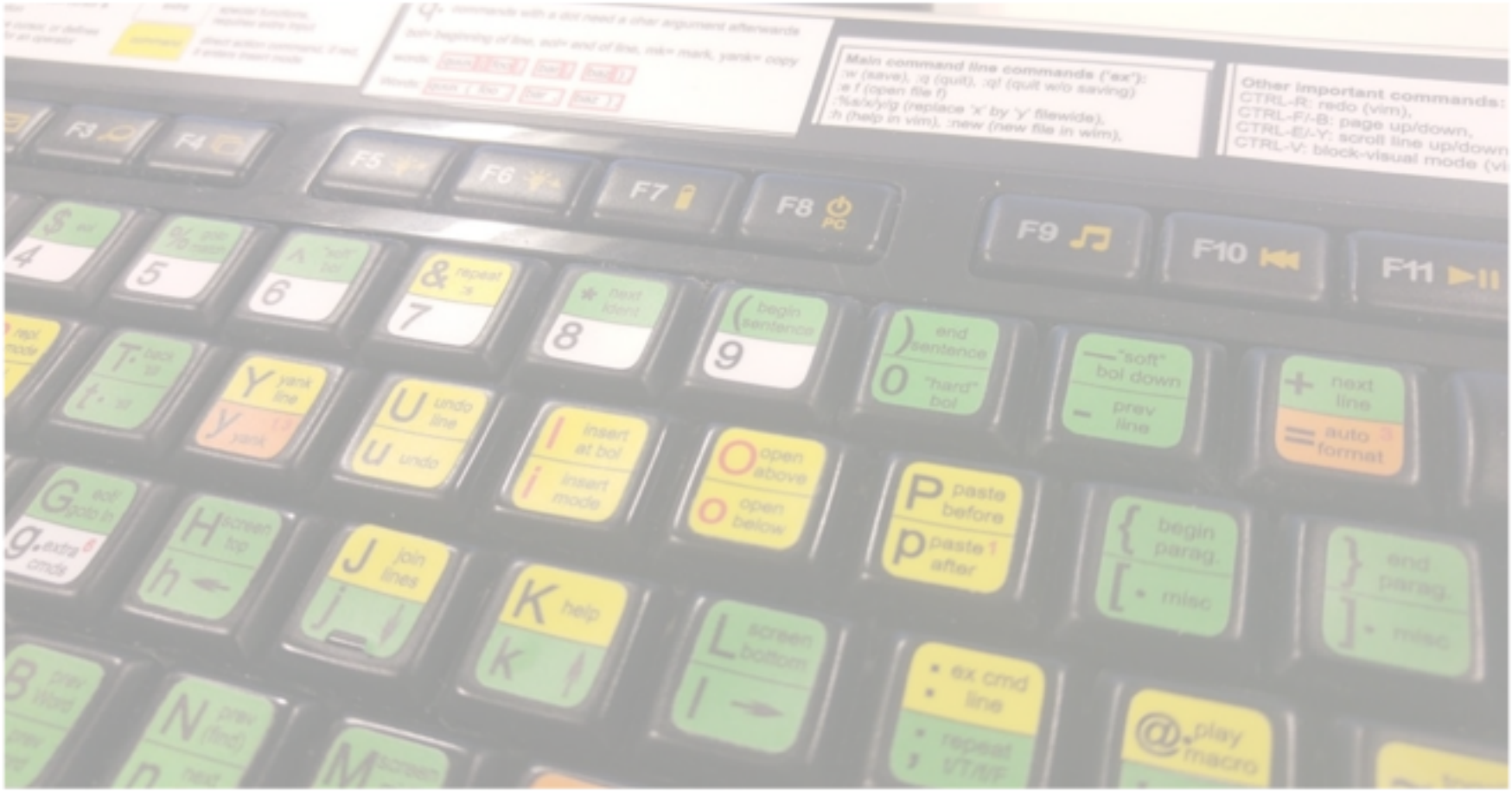

Atajos de teclado y comandos para usar VIM eficientemente

Joan Carles < https://geekland.eu/author/admin/>
24 junio, 2021 < https://geekland.eu/atajos-de-teclado-y-comandos-para-usar-vim-eficientemente/>
Deja un comentario < https://geekland.eu/atajos-de-teclado-y-comandos-para-usar-vim-eficientemente/#respond>

En su día vimos una instalación y una configuración básica < https://geekland.eu/instalar-y-configurar-vim/> del editor de texto VIM. A continuación veremos una chuleta de atajos de teclado y comandos que vale la pena recordar para empezar a sacar partido a este editor de textos.

Intercambiar entre los distintos modos de trabajo de VIM

Como imagino sabrá la gran mayoría, VIM tiene varios modos de trabajo. Algunos de ellos son los siguientes:

- **Modo normal:** Es el modo en que se encuentra VIM cuando lo abrimos. Será útil para navegar dentro del documento e introducir comandos.
- **Modo inserción:** Es el que usaremos para editar texto.
- **Modo Visual (Visual, Línea visual y Bloque Visual):** Generalmente se utiliza para seleccionar texto.
- **Modo comando:** Permite realizar acciones. En el transcurso de este artículo verán comandos que son de gran utilidad.
- **Modo reemplazar:** Permite reemplazar valores de una palabra o frase.

Para intercambiar entre cada uno de los modos que acabo de citar hay que usar los siguientes atajos de teclado:

Atajos de teclado	Función
`i`	Para acceder al modo inserción. En el transcurso del artículo verán otros modos de acceder al modo inserción.
`v`	Para simplemente acceder al modo visual.
`V`	Acceder al modo Línea visual.
`Ctrl`+`v`	Entrar en modo Bloque visual.
`:`	Para acceder al modo comando. Estando en el modo normal tecleamos `:` seguido del comando que queremos ejecutar. Los comandos realizan acciones. El modo comando guarda un registro, por lo tanto si usamos el atajo de teclado `:<cursor ↧ ↥>` podremos ir viendo los comandos que se han ejecutado previamente.
`Shift`+`r`	Permite acceder al modo reemplazar.
`ESC`	Para salir de los modos inserción, visual y comando y acceder al modo normal.

Guardar documentos en VIM

Para guardar los cambios en un fichero de VIM podemos usar los siguientes comandos:

Atajos de teclado y comandos	Función
`:w`	Guarda los cambios realizados al documento actual.
`:w nombre_fichero`	Guarda el documento actual en un nuevo fichero. Es equivalente a guardar como.
`:wq` o `:x` o `shift + z + z`	Comando que guarda los cambios y cierra el fichero que estamos editando.

Salir de VIM

Las diferentes opciones que tenemos para salir y cerrar el editor de código VIM son las mostradas a continuación:

Atajos de teclado o comandos	Función
`:q` o `shift + Z + Q`	Salir del documento.
`:q!`	Salir del documento sin guardar cambios. Si hay cambios que no se han guardado no se ejecutará el comando y recibiremos un advertencia.
`:wqa!`	Guardar los cambios y salir en todos los ficheros/buffers que tenemos abiertos.
`:qa!`	Si tienes abiertos varios ficheros de forma simultanea y quieres **salir de Vim** sin guardar los cambios de ninguno, puedes ejecutar: :qa!

Abrir documentos en VIM y moverse entre los distintos ficheros abiertos en los buffers

VIM permite tener varios documentos/buffers abiertos de forma simultanea. Una vez abiertos podemos usar una serie de comandos para navegar entre los distintos ficheros abiertos. Para ello deberemos usar las siguientes instrucciones.

Nota: Un buffer es el contenido de un fichero de texto cargado en memoria listo para ser editado. En el momento que se guarden los cambios realizados en el buffer se escribirá el fichero que contiene el código editado.

Atajos de teclado y comandos	Función
`:e` + `nombre fichero por abrir`	Abrir una fichero adicional dentro de VIM. El fichero abierto se mostrará en pantalla justo al abrirlo . Podemos tener abiertos varios documentos al mismo tiempo.
`:badd` + `nombre_del_fichero/buffer`	Añadir un buffer sin que sea mostrado en pantalla.
`:ls`	Lista y numera todos los buffers/ficheros que tenemos abiertos.
`:bn`	Ir al siguiente fichero/buffer que tenemos abierto. (Buffer_next).
`:bp`	Ir al fichero/buffer anterior que tenemos abierto. (Buffer_previous)
`:bl`	Para ver en pantalla el último buffer abierto.
`:bf`	Mostrar en pantalla el primer buffer abierto.
`:b_número_buffer`	Si anteriormente hemos listado los buffer vemos que tienen asociado un número. Para cambiar de buffer podemos ejecutar el comando `:b` seguido del número de buffer. Por ejemplo `:b2`
`:bd`	Cerrar únicamente el fichero/buffer actual que estamos editando.
`:bd2`	El comando `:bd2` cierra el buffer 2.
`:bd deuda.py`	El comando `:bd deuda.py` cierra el buffer deuda.py.
`:ba`	Mostrar todos los buffer en pantalla.

Trabajar con pestañas en VIM

VIM permite trabajar con pestañas similares a las que usaríamos en un navegador web. En cada una de las pestañas podemos tener abierto un fichero/buffer para poder editar código o texto. Los atajos de teclado y comandos que podemos usar para trabajar con las pestañas son:

Atajos de teclado o comandos	Función
`:tabnew nombre_fichero`	Abrir un fichero en una pestaña nueva.
`:tabedit nombre_fichero`	El fichero que queramos se abre en una pestaña nueva.
`:tabfind nombre_fichero`	Abre el fichero que queramos en una pestaña nueva. A medida que escribamos el nombre, si presionamos `TAB` se autocompletará el nombre del fichero. Solo autocompletará con nombres de ficheros que estén dentro de la ruta especificada en el path de VIM.
`:tab ball`	Todos los buffers abiertos se ponen en pestañas.
`:tab split`	Copiar el contenido de la ventana actual en una pestaña.
`:tabs`	Listar todas las pestaña abiertas.
`gt` o `:tabn` o `Ctrl-Avpág`	Ir a la pestaña siguiente.
`gT` o `:tabp` o `Ctrl-RePág`	Ir a la pestaña anterior.
`num_pestana+gt`	Para irnos a la pestaña número `3` presionamos el número `3` seguido de la teclas `gt`.

Atajos de teclado o comandos	Función
`:tabfirst`	Ir a la primera pestaña.
`:tablast`	Ir a la última pestaña.
`:tabm 3`	Mueve la pestaña actual a la posición 3.
`:tabm 0`	Mover la pestaña actual a la primera posición.
`:tabm`	Mover la pestaña actual a la última posición.
`:tabclose!` o `:q!`	Cierra la pestaña actual aunque no esté guardada.
`:tabclose` o `:q`	Cierra la pestaña actual.
`:tabonly`	Cerrar todas las pestañas excepto la pestaña actual.
`vim -p archivo1.sh archivo2.sh`	Abrir los ficheros archivo1.sh y archivo2.sh cada uno en una pestaña diferente.

Para más información de como trabajar con pestañas pueden consultar el siguiente **enlace <https://vim.fandom.com/wiki/Using_tab_pages>** .

Trabajar con ventanas y varios ficheros de forma simultanea en VIM

VIM también permite subdividir/multiplexar el editor de textos en diversas partes. De este modo en una sola pantalla podremos visualizar y editar de forma sencilla y práctica varios ficheros. Para conseguir lo que acabo de citar deberán usar los siguientes atajos de teclado y comandos.

Atajos de teclado o comandos	Función
`Ctrl+wv`	Dividir la ventana verticalmente.
`Ctrl+ws`	Dividir la ventana horizontalmente.
`Ctrl+w+(h/j/k/l)`	Moverse entre la distintas ventanas abiertas en pantalla
`ctrl+ww`	Cambiar de una ventana a otra.
`Ctrl+w`	Hacer que todas las ventanas tengan la misma dimensión.
`Ctrl+wq`	Cerrar la ventana sobre la que tenemos posicionado el cursor.
`Ctrl+wx`	Intercambiar de posición las ventanas.
`Ctrl+wr`	Rotar ventana a la derecha.
`Ctrl+wR`	Rotar ventana a la izquierda.
`:split`	Dividir la pantalla horizontalmente.
`:split nombre_fichero`	Abre un fichero y se parte la pantalla en 2 horizontalmente.
`:vsplit`	Abre un fichero y se parte la pantalla en 2 verticalmente.
`:resize 40`	Cambiar la altura de la ventana abierta a `40` columnas.
`:vertical resize 80`	Cambiar el ancho de la ventana en `80` columnas.
`:vertical resize -5`	Disminuir el ancho de la ventana en `5` columnas.

Atajos de teclado o comandos	Función
`:vertical resize +5`	Incrementar el ancho de la ventana en 5 columnas.
`Ctrl`+`w >` y `Ctrl`+`w <`	Aumentar o disminuir el tamaño de la ventana horizontalmente.
`Ctrl`+`w +` y `Ctrl`+`w –`	Aumentar o disminuir el tamaño de la ventana verticalmente.
`Ctrl`+`w =`	Reiniciar el ancho y el alto de las ventanas.
`vim –o archivo1 archivo2 archivo3`	Abrir los ficheros archivo1, archivo2 y archivo3 cada uno en una ventana distinta.

Editar texto y posicionar el puntero del cursor donde queramos de forma rápida

Mediante los siguientes atajos de teclado podrán mover el cursor en la posición que necesiten y editar texto de forma extremadamente rápida. Las combinaciones de teclas a usar son las siguientes:

Atajos de teclado	Función
`j` o `cursor abajo`	Para mover el cursor hacia abajo.
`k` o `cursor arriba`	Para mover el cursor hacia arriba.
`h` o `cursor izquierdo`	Para mover el cursor hacia la izquierda.
`l` o `cursor derecho`	Para mover el cursor hacia la derecha.
`w`	Mover el cursor al inicio de la siguiente palabra.
`W`	Mover el cursor al inicio de la siguiente palabra. El punto de puntuación no es considerado como palabra.
`b`	Mover el cursor al inicio a la palabra anterior.
`B`	Mover el cursor al inicio a la palabra anterior. El punto de puntuación no es considerado como palabra.
`e`	Desplazar el cursor al final de la palabra en que estamos o al final de la palabra siguiente.
`E`	Desplazar el cursor al final de la palabra en que estamos o al final de la palabra siguiente. El punto de puntuación no es considerado como palabra.

Atajos de teclado	Función
`0`	Posicionar el cursor al inicio de la línea actual.
`$` o `Fin`	Posicionar el cursor al final de la línea actual.
`_`	Mueve el cursor al primer carácter que no sea un espacio en la línea actual. Este atajo es útil cuando programamos.
`+`	Mueve el cursor al primer carácter que no sea un espacio de la línea siguiente. Este atajo es útil cuando programamos.
`-`	Mueve el cursor al primer carácter que no sea un espacio de la línea anterior. Este atajo es útil cuando programamos.
`shift + a`	Posicionar el cursor al final de la línea actual y cambiar al modo insertar.
`shift + 5`	Mover el cursor del inicio de un paréntesis al final de un paréntesis y viceversa.
`gg`	Para dirigirse a la primera línea del fichero.
`G`	Mover el cursor a la última línea del documento.
`5 + Enter`	Nos dirige 5 líneas más abajo respecto la línea actual.
`10gg`	Posicionar el cursor a la línea 10 del documento.
`16 + Mayús + g`	Nos dirige a la línea 16 del fichero de texto.

Atajos de teclado	Función
`}`	Hacer saltar el cursor al párrafo siguiente.
`{`	Hacer saltar el cursor al párrafo anterior.
`gi`	Posicionar el cursor en la última palabra editada del buffer actual.
`H`	Mover el cursor a la parte superior de la pantalla.
`M`	Mover el cursor a la parte media de la pantalla.
`L`	Mover el cursor a la parte inferior de la pantalla
`Ctrl`+`e`	Mover una pantalla hacia abajo sin mover el cursor de posición.
`Ctrl`+`y`	Mover una pantalla hacia arriba sin mover el cursor de posición.
`Ctrl`+`f`	Avanzar una pantalla.
`Ctrl`+`b`	Retroceder una pantalla.
`Ctrl`+`d`	Hacer avanzar el cursor media pantalla.
`Ctrl`+`u`	Hacer retroceder el cursor media pantalla.
`Ctrl`+➡️	Para que el cursor avance una palabra hacia la derecha **en modo insertar**.
`Ctrl`+⬅️	Para que el cursor avance una palabra hacia la izquierda **en modo insertar**.

Atajos de teclado	Función
`i`	Insertar texto justo antes de donde tenemos posicionado el cursor.
`I`	Insertar texto al principio de una línea.
`a`	Insertar texto justo después de donde tenemos posicionado el cursor.
`A`	Insertar texto al final de una línea.
`o`	Crear un línea en blanco justo por debajo de la línea actual y pasar al modo inserción.
`O`	Crear un línea en blanco justo por encima de la línea actual y pasar al modo inserción.
`ea`	Insertar texto después de una palabra.
`Ctrl + p`	Autocompletar la palabra que tenemos escrita a medias con otra palabra anterior al cursor.
`Ctrl + w + cursores`	Cambiar entre las distintas ventanas que podemos tener abiertas en VIM.
`Ctrl+g`	Mostrar el número de líneas de un fichero.

Insertar texto repetitivo o secuencial en Vim

Si quieren insertar el mismo texto un número determinado de veces deberán operar de la forma que se detalla en la siguiente tabla.

Atajos de teclado y comandos	Función
`40i-+ESC+ENTER`	Insertar 40 guiones.
`10ihola mundo + ESC +ENTER`	Escribir 10 veces hola mundo.
`i + geekland + ESC + 20.`	Escribe geekland 20 veces.
`:put=range(1,10)`	Escribir los números del 1 al 10.
`for i in range(0,50,1) | put='192.168.1.'.i | endfor`	Escribe del 192.168.1.0 al 192.168.1.50

Atajos de teclado y comandos para copiar y pegar texto en VIM

Si lo que pretendéis es copiar, cortar y pegar texto dentro de VIM deberéis conocer y usar los siguientes atajos de teclado y comandos.

Nota: La totalidad de atajos de teclado y comandos que verán a continuación se tienen que usar en **modo normal.**

Atajos de teclado o fór mulas	Función
`yy`	Para copiar una línea entera.
`2yy`	Copiar 2 líneas a partir de donde tenemos el cursor.
`y$`	Copiar desde donde tenemos posicionado el cursor hasta el final de la línea.
`yw`	Copiar la palabra a partir desde donde tenemos posicionado el cursor hasta el fi nal de la palabra.
`yiw`	Copiar la palabra actual.
`:10,20y`	Copiar de la línea 10 a la línea 20.
`120y`	Copiar la línea 120.
`dd`	Para cortar una línea entera.
`2dd`	Cortar 2 líneas a partir de donde tenemos el cursor.
`d$` o `D`	Cortar desde donde tenemos posicionado el cursor hasta el final de la línea.
`dw`	Cortar la palabra desde donde tenemos posicionado el cursor hasta el final de la palabra.
`diw`	Cortar la palabra actual.
`x`	Cortar una carácter.

Atajos de teclado o fór mulas	Función
`p`	Pegar una línea en la línea después de donde tenemos el cursor.
`:129put`	Pegar una línea o conjunto de líneas que tenemos en el portapapeles a la línea 12 9 del documento.
`shift + p`	Pegar una línea en la línea antes de donde tenemos el cursor.

También podemos copiar y pegar texto en el modo visual. Para ello deberán seguir las siguientes instrucciones.

1. Posicionamos el cursor en el punto que queremos empezar a copiar.
2. Presionamos la letra `v` para entrar en el modo visual.
3. Mediante los cursores o las teclas `j`, `k`, `h`, y `l` seleccionamos el texto que queremos copiar, cortar o eliminar.
4. Si queremos copiar el texto seleccionado pulsamos `y`. Para cortarlo pulsamos `d`. En caso de querer eliminarlo pulsamos `x`.
5. Finalmente podremos pegar el código. Para ello entramos en modo normal y usamos la tecla `p`.

Mover una línea de posición

Para mover una línea de posición usando numeración absoluta o relativa usaremos la siguiente combinación de

teclas.

Atajos de teclado	Función
`:.-1m.+1`	Para mover la línea justo superior a la actual a la línea inferior justo al actual.
`:19m17`	Mover la línea 19 a la línea 17.
`:.-4,.-2m.+8`	Mover la línea de la -4 a la -2 desde la posición actual del cursor a la posición 8 respecto l a posición actual del cursor.

Seleccionar texto en modo visual y acciones que podemos realizar en modo visual

Acabamos de ver una explicación de como seleccionar texto en modo visual usando los cursores o usando `j`, `k`, `h`, y `l`. Pero aparte de los cursores tenemos otras opciones para seleccionar texto. Algunas de ellas son las siguientes:

Nota: Para entrar en el modo visual tenemos que presionar `v`.

Atajos de teclado	Función
`v`	Acceder al modo visual. (Modo visual)
`V`	Iniciar el modo visual seleccionando toda una línea. Solo permite seleccionar lineas enteras. (Modo línea visual)
`Ctrl + v`	Inicia en modo Bloque visual.
`iw`	Seleccionar la palabra sobre la que tenemos el cursor.
`aw`	Seleccionar la palabra sobre la que tenemos el cursor incluyendo el espacio.
`ab`	Seleccionar un bloque de texto que tenemos delimitado por paréntesis (). La selección incluye los paréntesis.
`it`	Seleccionar un bloque de texto que tenemos delimitado por paréntesis (). La selección no incluye los paréntesis.
`aB`	Seleccionar un bloque de texto que tenemos delimitado por corchetes {}. La selección incluye los corchetes.
`iB`	Seleccionar un bloque de texto que tenemos delimitado por corchetes {}. La selección no incluye los corchetes.
`at`	Seleccionar un bloque de texto delimitado etiquetas <> y </> incluyendo la etiquetas.
`it`	Seleccionar un bloque de texto que tenemos delimitado etiquetas <> y </> sin incluir las etiquetas.

Atajos de tecl ado	Función
`o`	Moverse a la parte inicial de un bloque delimitado por (), {}, <>..</>
`O`	Moverse a la parte final de un bloque delimitado por (), {}, <>..</>
`j`	Selecciona una frase entera y se va la siguiente línea.
`is`	Selecciona frase hasta el primer punto.
`ip`	Seleccionar una párrafo completo.
`b`	Selecciona desde el cursor al inicio de la palabra.
`e`	Selecciona des del cursor al final de la palabra.
`$`	Seleccionar des del cursor al final de la línea.
`^`	Seleccionar des del cursor al primer carácter imprimible de la línea.
`awd$p`	Seleccionar una palabra y moverla al final de un párrafo.
`u`	Para transformar todo el texto seleccionado en minúsculas.
`U`	Para transformar todo el texto seleccionado en mayúsculas
`>`	Mover la línea en que tenemos el cursor a la derecha. (Aplica sangría al texto.)
`<`	Mover la línea en que tenemos el cursor a la izquierda. (Aplica sangría al texto.)

Atajos de teclado	Función
`Esc`	Salir del modo visual.

Para más información consultar el siguiente **enlace < https://vim.rtorr.com/lang/es_es>** .

Crear marcas en un fichero de VIM

Una marca de VIM permite guardar la posición actual del cursor. Una vez guardada la posición actual del cursor podremos volver a ella cuando lo necesitemos. Para crear y trabajar con las marcas usaremos los siguientes atajos de teclado y comandos:

Atajos de teclado	Función – Explicación
`m + [a – z]`	Si en modo normal presionamos `ma` crearemos una marca local que denominaremos `a`. Podemos usar marcas de la `a` la `z` o de la `A` a la `Z`.
`` `[a–z] ``	Si una vez creada la marca `a` queremos volver a posicionar el cursor sobre la marca `a`, en modo normal presionaremos las teclas `` `a ``. Entonces el cursor se posicionará sobre la marca `a`.
`' '`	Para volver al inicio de la línea en que hemos introducido la marca.
`'a`	Nos traslada al inicio de la línea en que añadimos la marca `a`.
`` ` ``	Para ir a la marca anterior. Si estamos en la marca `b` nos trasladaremos a la marca `a`.
`:delmarks a`	Eliminar la marca `a`.
`:delmarks!`	Eliminar todas las marcas locales o minúsculas de un fichero.
`:delmarks a–d`	Eliminar todas las marcas de la `a` hasta la `d`.
`:delmarks aBc`	Para eliminar las marcas `a`, `B` y `c`.

Nota: Si asignamos una marca con letras minúsculas será una marca local. Por lo tanto solo estará disponible hasta que cerremos el fichero que estamos editando. Si queremos que la marca sea permanente tendremos

que asignar la marca con letras mayúsculas. Si usamos las letras mayúsculas estaremos creando marcas globales.

Aplicar sangrías de texto a un texto en VIM de forma manual

Si cuando escribimos o programamos queremos añadir sangrías de texto de forma manual deberemos usar los siguientes atajos de teclado.

Atajos de teclado	Función del atajo de teclado
`>>`	Para sangrar un texto. (El texto se mueve a la derecha)
`3 >>`	Realizar un sangrado en 3 líneas a partir de donde tenemos ubicado el cursor.
`<<`	Para deshacer la sangría. (El texto se mueve a la izquierda)
`]p`	Pegar y ajustar a la sangría actual.
`v + j + >`	Seleccionar una frase en modo visual y realizar una sangría a la derecha.
`Ctrl+t`	Para aplicar un sangrado cuando estamos en el modo insertar. (Mueve texto a la derecha)
`Ctrl+d`	Para aplicar un sangrado cuando estamos en el modo insertar. (Mueve texto a la izquierda)

Deshacer y rehacer cambios en VIM

VIM permite deshacer y rehacer cambios del buffer/fichero que estamos editando. Para ello deberemos usar los siguientes comandos o atajos de teclado:

Atajos de teclado o comandos	Función
`u`	Para deshacer el último/s cambio/s realizado/s.
`2u`	Para deshacer los 2 últimos cambios realizados.
`U`	Restaura la última línea modificada. Una vez restaurada no podemos rehacer el cambio.
`ctrl` + `r` o `:redo`	Para rehacer el último cambio realizado.
`3` + `ctrl` + `r`	Para rehacer los 3 últimos cambios que hemos desecho.
`:earlier 1h`	Deshacer todos los cambios realizados en la última hora.
`:later 30m`	Rehacer los cambios realizados en la última media hora.

Nota: En el momento que cerremos el fichero que estamos editando no podremos deshacer ni rehacer ningún cambio.

Atajos de teclado y comandos para eliminar caracteres y líneas en VIM

Para eliminar texto, líneas, párrafos y código en VIM deberemos usar los siguientes atajos de teclado.

Atajo de teclado o comandos	Función
`x`	Para eliminar un carácter en modo normal.
`3x`	Elimina 3 caracteres a partir de donde tenemos posicionado el cursor.
`X`	Borrar tan solo un carácter a la izquierda del cursor.
`r`	Reemplazar una carácter.
`s`	Eliminar un carácter y cambiar a modo insertar.
`dw`	Borrar la palabra desde donde tengamos el cursor hasta el final de la palabra.
`cw`	Borrar la palabra desde donde tengamos el cursor hasta el final de la palabra y cambia r a modo insertar.
`diw`	Borrar la palabra entera sobre la que se encuentra el cursor.
`ciw`	Borrar la palabra entera sobre la que se encuentra el cursor y cambiar a modo insertar.
`c$`	Borrar todo el texto hasta el final de línea y pasar a modo insertar.
`cc` o `S`	Borrar todo el contenido de una línea y pasar al modo insertar.
`d6w`	Eliminar 6 palabras a partir de donde tengamos posicionado el cursor.
`d6b`	Eliminar 6 palabras anteriores a donde tenemos posicionado el cursor.
`c3w`	Eliminar 3 palabras a partir de donde tengamos posicionado el cursor y cambiar a mod

Atajo de teclado o comandos	Función
	o insertar.
`dd`	Para borrar/cortar una línea entera.
`shift + d`	Borrar desde la posición del cursor hasta el final de la línea.
`d0`	Borrar desde la posición del cursor hasta el inicio de la línea.
`cc`	Para borrar/cortar toda un línea entera y cambiar a modo insertar.
`shift + c`	Borrar desde la posición del cursor hasta el final de la línea y cambiar al modo insertar.
`c0`	Borrar desde la posición del cursor hasta el inicio de la línea y cambiar al modo inserta r.
`2dd`	Borrar/cortar 2 líneas de golpe a partir de donde tenemos posicionado el cursor.
`2cc`	Borrar/cortar 2 líneas de golpe a partir de donde tenemos posicionado el cursor y cam biar al modo insertar.
`Ctrl+h`	Estando en modo inserción elimina el carácter que está antes del cursor.
`Ctrl+w`	Estando en modo inserción elimina toda la palabra que hay justo detrás del cursor.
`Ctrl+u`	Estando en modo inserción elimina toda la línea que hay detrás del cursor.
`xp`	Transponer 2 letras.

Atajo de teclado o comandos	Función
`:%d`	Para borrar todas las líneas de un fichero.
`dgg`	Borrar desde la posición actual del cursor hasta al principio del fichero.
`dG`	Borrar desde la posición actual del cursor hasta al final del fichero.
`:[2],[4]`	Para eliminar las líneas de la 2 a la 4 podemos usar la formula `:2,4d`
`:1,.-1d`	Eliminar todas las líneas antes de la línea actual.
`:.+1,$d`	Eliminar todas las líneas después de la línea actual.
`:g/<patrón>/d`	Eliminar todas las filas que contengan un determinado patrón o palabra. Para eliminar todas las filas que contengan la palabra importante usaremos el comando `:g/importante/d`
`:g!/<patrón>/d`	Borrar todas las líneas que no contengan una determinada palabra o patrón. Por ejemplo `:g!/importante/d`
`:g/^A/d`	Eliminar todas las líneas que empiezan por la letra `A`.
`:g/^$/d`	Para eliminar todas las líneas vacías de un documento.
`:.-2,.+8d`	Eliminar líneas desde 2 líneas encima de mi posición hasta 8 líneas debajo de mi posición.

Otra forma de eliminar texto es mediante **marcas** o mediante el **modo visual**.

Si lo queréis realizar mediante marcas lo primero que hay que realizar es:

1. Irse a la primera línea que queremos borrar e introducir una marca como por ejemplo `ma`.
2. A continuación posicionamos el cursor en la última de las líneas que queremos borrar.
3. Finalmente presionamos `d'a` y se borraran todas las líneas desde la posición actual del cursor hasta la línea que marcamos con la marca `a`.

Para eliminar texto en modo visual pueden consultar el apartado «Atajos de teclado para copiar y pegar texto en VIM»

Buscar una palabra o frase en VIM

Es extremadamente fácil buscar una determinada palabra en VIM. Para ello tan solo tendremos que recordar y usar los siguientes atajos de teclado.

Atajo de teclado	Función
`/palabra_a_buscar`	Para buscar todas las palabras que contengan las palabras `labra` escribiré lo siguiente estando en el modo normal de VIM `/labra`
`/\<palabra_a_buscar\>`	Buscar una palabra completa y exacta.
`?palabra_a_buscar`	Buscar todas las palabras que contengan un texto determinado y ubicar el cursor en la primera ocurrencia anterior a donde tenemos ubicado el cursor.
`n`	Una vez realizada la búsqueda la letra `n` servirá para ir a la segunda aparición de la búsqueda realizada.
`Shift+n`	Tiene la misma función que `n`, pero en este caso va para atrás.
`ggn`	Para ir a la primera aparición de la palabra buscada.
`Gn`	Para ir a la última aparición de la palabra buscada.
`*`	Sobre la palabra que queramos buscar presionamos `*`. La búsqueda es hacia adelante.
`#`	Sobre la palabra que queramos buscar presionamos `#`. La búsqueda es hacia atrás.
`3/palabra_a_buscar`	Buscará todas las palabras que contengan la palabra a buscar y el cursor quedará ubicado en la tercera ocurrencia.

Para obtener más información sobre buscar en VIM pueden consultar el siguiente **enlace <https://vim.fandom.com/wiki/Searching>** .

Buscar y reemplazar palabras y caracteres en VIM

VIM dispone de la función buscar y reemplazar y texto. Para buscar y reemplazar palabras deberán usar los siguientes atajos de teclado o fórmulas.

Atajo de teclado o fórmula	Función
`r`	Permite reemplazar un carácter cuando estamos en modo normal.
`R`	Se activa el modo reemplazar para cambiar los caracteres que queramos. No se saldrá del modo reemplazar hasta que presionemos la tecla `ESC`
`ró`	Reemplaza la letra que señala el cursor por una `ó`
`10ró`	Reemplaza las 10 letras a partir de la posición actual del cursor por `ó`
`:s/geekland/Geekland`	Reemplaza todas las palabras que contienen `geekland` por `Geekland` solo en la línea actual.
`:s/geekland/Geekland/g`	Remplaza todas las palabras que contienen `geekland` por `Geekland` únicamente en la línea que tenemos el cursor sin pedir confirmación.
`:%s/geekland/Geekland/g`	Remplaza todas las palabras que contienen `geekland` por `Geekland` en todo un documento sin pedir confirmación.
`:%s/geekland/Geekland/gc`	Para reemplazar todas las ocurrencias `geekland` de un fichero a `Geekland` pidiendo confirmación.
`:%s!Geekland!ubuntu/scripts!gi`	Remplaza todas las palabras que contienen `Geekland` por `ubuntu/scripts` usando el delimitador `!`. Al usar la opción `i` no se distingue entre mayúsculas y minúsculas. Para distinguir entre mayúsculas y minúsculas tendríamos que usar `I`.
`:%s/\<geek\>/Linux/gc`	Reemplaza la palabra exacta `geek` por `Linux` en todo el documento pidiendo confirmación.

Atajo de teclado o fórmula	Función
`:5,12s/foo/bar/g`	Cambiar `foo` por `bar` entre las líneas 5 y 12. (Ambas incluidas)
`:'a,'bs/foo/bar/g`	Cambiar `foo` por `bar` entre la marcas `a` y `b`
`:22s/Linux/Geekland/I`	Reemplazar todas las palabras que contengan `Linux` por `Geekland` en la línea `22`. Se tendrán en cuenta las mayúsculas y las minúsculas.
`:.s/Geekland/Linux/I`	Reemplazar todas las palabras que contengan `Geekland` por `Linux` únicamente en la línea actual. Se tendrán en cuenta las mayúsculas y las minúsculas.
`g`+`u`+`u`	Transforma una frase entera a minúsculas.
`g`+`U`+`U`	Transforma una frase entera a mayúsculas.

Para más información pueden consultar el siguiente **enlace < https://vim.fandom.com/wiki/Search_and_replace>** .

Nota: Se pueden usar expresiones regulares para reemplazar texto.

Especificando rangos en VIM

Especificar un rango es seleccionar una serie/conjunto de líneas de forma manual para aplicar acciones sobre ellas. En el apartado anterior vimos algunos ejemplos de como definir rangos. La sintaxis a usar para especificar

rangos es la siguiente:

Ejemplos de rangos	Significado
`%`	El símbolo `%` es para indicar que la acción a realizar aplica a todo el documento. Por ejemplo el comando `:d%` borra todas las líneas de un documento.
`$`	El símbolo `s` es para indicar que la acción a realizar aplica solo a la última línea. Por ejemplo el comando `:d$` borra únicamente la última línea del documento.
`.`	Representa la línea actual en que esta posicionado el cursor.
`'a,'b`	Bloque de líneas entre las marcas `a` y `b`. Para reemplazar las palabras `Linux` por `Geekland` en el texto comprendido entre las marcas `a` y `b` usaremos el comando `:'a,'bs/Linux/Geekland/g`
`:17,20d`	Borramos las líneas 17, 18, 19 y 20.
`:.-2,.+8y`	Para indicar rangos con numeración relativa. Con el ejemplo indicado copiamos el contenido desde 2 líneas encima de mi posición actual hasta 8 líneas debajo de mi posición.

Juntar 2 Lineas en VIM

En el caso que necesiten juntar 2 líneas tan solo tendrán que presionar `Shift + j` en modo normal. De este modo conseguiremos unir 2 líneas que están separadas.

Atajo de teclado o fórmula	Función
`Shift + j`	Para juntar 2 líneas.
`gJ`	Une la línea que tenemos debajo del cursor con la actual sin dejar espacio entre ellas.
`:join`	Fórmula a usar en el caso que se quieran juntar 2 líneas.

Realizar pliegues manuales en VIM

Para realizar pliegues y de este modo leer de forma más confortable un texto o un código hay que usar los siguientes atajos de teclado.

Atajos de teclados	Función
`zf <movimiento arriba o abajo>`	Definir un pliegue.
`zf4j`	Plegar las 4 líneas que tenemos debajo del cursor.
`zf2}`	Plegar los 2 párrafos justo por debajo del cursor.
`zf2{`	Plegar los 2 párrafos justo por encima del cursor.
`zo`	Abrir el pliegue sobre el que está posicionado el cursor.
`zR`	Abre todos los pliegues del documento.
`zd`	Eliminar el pliegue que tenemos encima del cursor.
`zE`	Eliminar todos los pliegues del documento.
`zc`	Cerrar el pliegue que tenemos encima del cursor.
`zM`	Cierra todos los pliegues existentes en el documento.
`za`	Abrir y/o cerrar el pliegue que tenemos encima del cursor.
`zj`	Desplaza el cursor al siguiente pliegue.
`zk`	Desplaza el cursor al pliegue anterior.

Nota: Para realizar pliegues de forma manual hay que añadir `set foldmethod=manual` en el fichero de configuración `~/.vimrc`.

Registros de VIM (portapapeles en VIM)

Los registros de VIM se pueden definir como un conjunto de 48 portapapeles en los que podemos almacenar información y pegar contenido al documento. Para trabajar y gestionar los 48 portapapeles que acabamos de mencionar usaremos los siguientes atajos de teclado.

Atajos de teclado y comandos	Función
`:registers` o `:reg`	Mostrar el contenido que almacena cada uno de los 48 registros o portapapeles.
`"1yy`	Copiar una línea al registro numérico `1`.
`"1p`	Pegar el contenido del registro numérico `1`.
`"ayy`	Copiar una línea al registro nominal `a`. Al utilizar a en minúscula borras completamente el contenido del portapapeles `a`.
`"Ayy`	Añade una línea al registro nominal `a`. Al usar la A en mayúscula no se borra el contenido previamente guardado en a. Se añade .
`"ap`	Pegar el contenido del registro nominal `a`.
`:registers a` o `:reg a`	Mostrar el contenido guardado en el registro `a`.
`:reg 4 c`	Mostrar el contenido que tiene almacenado el registro numérico `4` y el registro nominal `c`.
`"+yy`	Copia la línea actual al portapapeles del sistema operativo.
`"+p`	Pegar el contenido que tenemos almacenado en el portapapeles del sistema operativo al documento que editamos.
`"*p`	Pegar un texto que tengamos seleccionado en nuestro navegador o en cualquier otro software de nuestro sistema operativo. No hace falta que el texto esté copiado en el portapap

Atajos de teclado y comandos	Función
	eles del sistema.
`Ctrl+r+a`	Pega el contenido del registro `a` estando en modo inserción de texto.
`Ctrl+r "`	Pega el contenido del registro por defecto del portapapeles de VIM «» en modo inserción.
`Ctrl+r +`	Pegar el contenido almacenado en el portapapeles del sistema operativo en modo inserción.
`Ctrl+r *`	Pegar el contenido que tenemos seleccionado en cualquier aplicación de nuestro sistema operativo en el modo inserción de texto. No hace falta que el texto esté copiado en el portapapeles del sistema.

Nota: Existen 26 registros nominales. Estos van de la `a` a la `z`.

Nota: Existen 10 registros numéricos. Estos van de la `0` a la `9`.

Registros de lectura y registros especiales

Acabamos de ver 36 registros de VIM. El resto de registros hasta llegar a los 48 serán registros de lectura y registros especiales. Algunos de los **registros de lectura** existentes son los siguientes:

Registro	Contenido almacenado por el registro
`":`	Almacena el último comando ejecutado.
`"%`	Contiene la ruta que usamos para abrir el fichero de VIM que estamos editando.
`".`	Contiene el último texto insertado.

Algunos de los **registros especiales** que tiene disponibles VIM son los siguientes:

Registro	Contenido almacenado por el registro
`"_`	Corresponde al registro agujero negro. Para evitar que una palabra borrada quede almacenada al registro por defecto de VIM podemos usar el atajo de teclas `"_diw`. De esta forma la palabra borrada solo quedará almacenada en el registro `"_` y el registro `""` quedará vacío.
`":`	Almacena el último comando ejecutado.
`"_`	Contiene todo el contenido que hemos borrado o modificado y que no tenga una longitud superior a una línea.
`"\`	Almacena el contenido de la última búsqueda.
`"=`	Registro que permite realizar operaciones matemáticas con VIM. Para realizar la suma 2 + 2 en modo normal presionaremos las teclas `"=2+2`. A posteriori presionaremos `p` para pegar el resultado.

Nota: La totalidad de registros de VIM se almacenan en el fichero `~/.viminfo`. Por lo tanto los registros son permanentes. Es decir, en el momento de cerrar VIM no se borran los registros.

Usar la terminal de Linux en VIM

Si necesitamos usar la terminal mientras estamos usando VIM podemos aplicar las soluciones que veremos a continuación.

Atajos de teclado o comandos	Función
`:term`	Se subdivide la pantalla en 2. En una de ellas tendremos VIM y en la otra la terminal. P ara pasar de una ventana a otra podemos usar el atajo `Ctrl`+`w`
`:!ls`	Ejecuta el comando `ls` estando dentro de VIM. En vez de `ls` podríamos usar otros co mandos como por ejemplo `sudo apt update`
`:r !<comando>`	Pone la salida del comando ejecutado dentro del fichero de VIM que estamos editand o.

Encadenar comandos en VIM

En el transcurso del artículo hemos visto una gran cantidad de comandos para realizar acciones. Usando la tubería | podemos ejecutar de forma simultanea más de un comando y de esto modo evitar introducir comando por comando. A continuación les dejo un par de ejemplos en que encadenamos la ejecución de comandos.

Comandos	Función
`:w | q`	Guarda los cambios del documentos y lo cierra.
`:17,20y | 122put`	Copiar las líneas de la 17 a la 20 y las pega en la línea 122.

Otras fuentes

https://vim.rtorr.com/lang/es_es < https://vim.rtorr.com/lang/es_es>
https://vim.fandom.com/wiki/Vim_buffer_FAQ < https://vim.fandom.com/wiki/Vim_buffer_FAQ>
https://www.atareao.es/tutorial/vim/buscar-y-reemplazar-en-vim/ < https://www.atareao.es/tutorial/vim/buscar-y-reemplazar-en-vim/>

< https://www.addtoany.com/share#url=https%3A%2F%y-comandos-para-usar-vim-eficientemente%2F&title=Atajos%20de%20teclado%20y%20comandos%20para

Dejar un comentario

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario *

Nombre *

Correo electrónico *

Web

Guarda mi nombre, correo electrónico y web en este navegador para la próxima vez que comente.

Publicar el comentario

Este sitio usa Akismet para reducir el spam. **Aprende cómo se procesan los datos de tus comentarios < https://akismet.com/privacy/>** .