Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Trucos prácticos para Vim **(27311 lectures)**
Per **Guillem Cantallops Ramis**, ***Beowulf*** (http://bulma.net/beowulf/)
Creado el 31/10/2003 02:43 modificado el 04/11/2003 01:22

Con vim puedes limitarte a los comandos mínimos para editar, o puedes aprender a hacer las cosas de maneras cada vez más eficientes. Depende de tí ;-)

Trucos prácticos para Vim

Importante: *el autor original del Vim[1], Bram Moolenaar, lo creó como Open Source, pero le dió un matiz que el llama Charityware. Su programa es tremendamente útil para todos nosotros, y lo único que pide a cambio es que nos tomemos la molestia de ejecutar el comando* ***:help iccf*** *y leer lo que nos dice, cosa que mucha gente no hace simplemente porque no lo sabe. Ahora ya lo sabeis. Se trata de ayudar a los niños pobres de Uganda.*

Actualización: *por fin he completado la parte de ventanas y edición múltiple de ficheros, espero que os ayude a ser más eficientes ;-)*

Actualización 2: *corregido el error de* ***(d)*** *a* ***{d}****...*

Introducción

El Vim no es *un* editor. El Vim es *el* editor. Una vez aclarado eso `X'-D` diré que esto no es un manual de Vim, puesto que no es ni mucho menos exhaustivo; además, el manual oficial está muy bién. Tampoco es un tutorial, ya que no empieza desde cero con lo típico (modo edición, modo comandos; h izquierda, j abajo, k arriba, l derecha). Todo eso y bastante más se da por hecho que lo sabeis, y si no siempre podeis hacer un **:help** que os llevará de la mano desde el principio.

Simplemente alguien ha iniciado un thread[2] en la bulmailing[3], yo he participado (confundiendo derecha e izquierda, por cierto) y casi casi ha salido material para un articulillo que puede resultar útil a los que ya tengan algo de práctica con el vim, y quieran aprovecharlo un poco más. Es lo que tiene este editor: con media docena de comandos simples y un par de conceptos básicos ya puedes usarlo, pero luego vas descubriendo a lo largo de los años nuevas formas de hacer las cosas, cada vez de una manera más eficiente `:-)`

Este artículo *muestra* (a base de ejemplos y las explicaciones mínimas) un subconjunto cualquiera de técnicas: son algunas de las que más uso yo, pero de ninguna manera las cubre todas, y no las he elegido de ninguna manera especial, ni las he tratado a fondo (para eso está el manual). Espero que sirva para mostrar lo mucho que nos queda a todos por descubrir. Si esto gusta seguramente ampliaré este artículo o escribiré otros: buscar y reemplazar, resaltado de sintaxis, configuración, indentado... los temas no se acaban, uso el vim para muchas cosas, desde programar hasta escribir correo electrónico, y se adapta a todo perfectamente `;-)`

Copiar, Cortar y Pegar

En general cada uno sabe dos o tres formas de hacerlo por lo menos: el famoso **yy** copia una linea, **dd** la corta, **10dd** corta 10 lineas, **p** pega después del cursor, **P** pega antes, etc. Yo os contaré un par de cosas más.

Objetos

Como sabeis el vim es un editor de textos bastante listo y sabe lo que son las palabras, las lineas, los párrafos, y demás. Bueno, en realidad no sabe nada, pero tiene unas definiciones muy precisas de todo eso que podeis leer en el manual y (esto es lo mejor) podeis "sentir" como las aplica haciendo unas cuantas pruebas. Porque por supuesto no hay que pensar los comandos de vim, simplemente hay que introducirlos para que haga justo lo que queremos. Con un poco de práctica todo esto sale solo ;-)

A grandes rasgos, las palabras son lo que está delimitado por espacios en blanco y los párrafos son lo que está delimitado por lineas en blanco. Pues eso son los objetos. Podeis usarlos para decirle al vim qué es lo que quereis copiar o cortar a un nivel más alto que el de carácteres o líneas, sin necesidad de contar (o en todo caso contando cosas grandes) y haciendo rápido y preciso todo el proceso gracias a la detección automática de los límites de los objetos. En los siguientes ejemplos copiaremos, pero para cortar solo habria que cambiar la **y** por **d**.

- **yaw** copia toda la palabra sobre la que tenemos el cursor (aw = "all word").
- **yap** copia todo el párrafo sobre el que se encuentra el cursor (ap = "all paragraph").
- **3yaw** copia la palabra actual y las dos siguientes.

Además los objetos se pueden usar para muchas otras cosas, por ejemplo podeis usar **gqap** para ajustar las lineas de un párrafo a la anchura que tengais definida (por ejemplo 70 columnas, con **:set tw=70**) respetando la integridad de las palabras e incluso indentados y quotes. Y ahora viene un párrafo que copio literalmente de un mensaje mio de hace un tiempo, os dejo que lo busqueis en los archivos de la lista si os interesa O:-)

Lo que hago yo normalmente para no meter tantos comandos es escribir tranquilamente, dejarlo todo como queda y cuando estoy al final hago un "gqgg" o un "gq1G". Para arreglar todo un texto rápidamente, del principio al fin, puedes hacer "Ggqgg" o "gggqG", por ejemplo... bueno, creo que ya ves como va ;-)

Movimientos

Lo cual me recuerda que también pueden hacerse cosas con comandos de movimiento. En particular, ya que aquí hablamos de copiar, cortar y pegar, se puede poner **y** (copiar) o **d** (cortar) seguido de un comando de movimiento para meter en el buffer todo lo que se encuentra entre la posición actual del cursor y la posición de destino. Esta coherencia viene muy bién, puesto que si sabemos movernos de forma rápida y precisa también sabemos copiar y cortar de la misma manera. En general los comandos de vim son muy consistentes: por ejemplo, los prefijos numéricos suelen indicar el número de veces que se repetirá lo que viene después. Vamos con los ejemplos:

- **y5l** o **5yl** copia el carácter que está bajo el cursor y los 4 siguientes.
- **d4h** corta los 4 carácteres anteriores al cursor.
- **y3w** copia tres palabras desde el cursor hacia adelante; pero si éste está en mitad de una palabra, solamente copiará la segunda mitad puesto que no estamos refiriéndonos a un objeto: estamos usando el comando **w** que nos mueve hasta la siguiente palabra.
- **y2b** copia dos palabras desde el cursor hacia atrás; pero si estamos en mitad de una palabra, solamente copiará la primera mitad puesto que estamos usando el comando **b** que nos mueve a la palabra anterior desde donde estamos.
- **{d}** corta todo el párrafo, pero usando comandos de movimiento en lugar de referencias a objetos; el resultado no es el mismo que tendria **dap** ya que éste corta también una linea en blanco que suele ir muy bién a la hora de pegar el párrafo y dejarlo separado para que siga siendo claramente un párrafo.

Ventanas

Si trabajamos con el vim habitualmente es que nos gusta la consola, que no es **un** entorno sino que es **el** entorno X'-D Y en ese caso seguramente tendremos algo mejor que una terminal de 80x25 carácteres enormes y feos. Como mínimo tendremos algo de 120 ó 130 columnas por 40 ó 50 lineas, con una fuente clara de anchura fija :-) ~

Con terminales así de grandes puede ser interesante aprovechar la posibilidad de crear y gestionar ventanas de texto dentro del vim (en realidad el vim siempre trabaja con ventanas, aunque si no las controlamos nos muestra solo una, maximizada). Si no tenemos entorno gráfico pero si una terminal enorme (por ejemplo frame buffer de más de mil por más de mil pixels en una pantalla grande) las ventajas son obvias. Y aunque tengamos entorno gráfico, puede venirnos

bién realizar la edición de varios ficheros con una sola instancia del vim (hay por lo menos otra manera de hacerlo, la veremos más adelante) aunque solo sea para copiar y pegar cosas rápidamente.

Lo de las ventanas es tan fácil como pulsar **Ctrl+w** (es una *combinación* de teclas, algo poco habitual en vim, cuidado!) y luego alguno de estos comandos:

- **n** crea una nueva ventana.
- **H** mueve la ventana hacia la izquierda.
- **J** mueve la ventana hacia abajo.
- **K** mueve la ventana hacia arriba.
- **L** mueve la ventana hacia la derecha.
- **h** va a la ventana de la izquierda.
- **j** va a la ventana de abajo.
- **k** va a la ventana de arriba.
- **l** va a la ventana de la derecha.
- **t** va a la ventana de más arriba.
- **b** va a la ventana de más abajo.
- **w** pasa a la siguiente ventana (es cíclico).
- **+** aumenta el tamaño de la ventana en 1 linea.
- **-** disminuye el tamaño de la ventana en 1 linea.

Por supuesto podemos usar un prefijo numérico con casi todos esos comandos para multiplicar su efecto: **7 Ctrl+w +** aumenta 7 lineas el tamaño de la ventana. Además podeis configurar directamente el número de lineas de una ventana, por ejemplo a 15 lineas con **15 Ctrl+w _**. Si no introducimos ningún valor antes, este comando "maximiza" la ventana.

Además tenemos algunos comandos interesantes para trabajar con ventanas:

- **:new** crea una nueva ventana vacia.
- **:split** sin argumentos abre una nueva ventana con el mismo fichero que estamos editando; acepta como argumento el nombre del fichero que vamos queremos abrir, si es otro.
- **:vnew** y **:vsplit** son las variantes con separación vertical de los dos comandos anteriores, que por defecto hacen ventanas separadas horizontalmente.
- **:close** cierra la ventana actual.
- **:only** deja únicamente la ventana actual, cierra las otras.
- **:qall**, **:wall** y **:wqall** son versiones de los conocidos comandos **:q**, **:w** y **:wq**, pero afectan a todas las ventanas. Sí, también aceptan el sufijo **!**, pero es mucho más peligroso porque tenemos más cosas abiertas. Pensad antes de usarlo `;-)`

Edición de múltiples archivos

El mecanismo del chupete, muy sencillo y a la vez práctico ya que (como el método anterior) permite editar tantos ficheros como queramos con una sola instancia del vim. Invocais el programa con **vim *.c**, por ejemplo, y abrirá todos esos ficheros. Os podeis mover entre ellos (aunque solo si los habeis guardado tras modificarlos, por ejemplo con **:w**) con estos comandos: **:first** va al primero, **:last** al último, **:prev** al anterior, y **:next** al siguiente. Si queremos grabar y pasar al siguiente fichero tenemos un comando rápido que lo hace todo de golpe: **:wnext** (sí, también existe **:wprev**).

Además si quereis que los ficheros se abran en ventanas para verlos todos al mismo tiempo podeis utilizar **vim -o *.c** que creará ventanas separadas horizontalmente, o **vim -O *.c** que creará ventanas separadas verticalmente.

Para abrir un fichero cuando vim ya esté en marcha no hace falta salir y pasárselo como argumento: podemos utilizar comandos del estilo de **:edit fichero.c** (si ya existe lo abre, si no existe lo crea). También es muy práctico, aunque resulta algo complicado al principio, el tema de la gestión de buffers (*buffer ~= imagen en memoria del fichero que estamos editando*) ya que nos permiten dejar de ver momentáneamente un fichero sin tener que grabarlo. Podemos ocultarlo simplemente con **:hide**, y después es muy importante recordar el comando **:ls** o **:buffers** el cual nos mostrará una lista de buffers cada uno con su número. Si queremos ir al buffer número 3, podremos utilizar el comando **:buffer 3**. También podemos utilizar el nombre (o una parte no ambigua del nombre) del fichero, en lugar del número. Finalmente, tenemos **:sbuffer 3** que nos abre el buffer 3 en una nueva ventana, **:bdelete 4** que quita el buffer 4 de la lista de buffers, y **:bfirst**, **:blast**, **:bprev** y **:bnext** que sirven para recorrer la lista de buffers.

Obviamente todos estos comandos pueden utilizarse en cualquier ventana para abrir o crear archivos, o decidir qué buffer editamos en ella. También hay que tener presente que al cerrar la última ventana se cierra el vim.

That's all, folks! Por ahora... ;-)

Lista de enlaces de este artículo:

1. http://www.vim.org/
2. http://llistes.bulma.net/pipermail/bulmailing/Week-of-Mon-20031027/011230.html
3. http://llistes.bulma.net/mailman/listinfo/bulmailing

E-mail del autor: beowulf _ARROBA_ bulma.net
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1896