Vi y Vim - Básico

Vi ha sido el primer editor de texto a pantalla completa para sistemas Unix . Además fue creado con la intención de que fuese sencillo en su uso y ligero para no cargar sobremanera el sistema. Para aquellos acostumbrados a usar editores de texto mediante en entornos gráficos, puede resultar un poco abrupto y complejo su aprendizaje, pero si bien porque estamos al cargo de un servidor Linux o porque nuestro entorno gráfico está dando problemas y no carga, deberíamos ser capaces de editar ficheros desde un editor como Vi / Vim.

A continuación intentaré dar la información necesaria para poder empezar a trabajar con Vi / Vim, pero como siempre tendréis toda la información tanto en el manual de la aplicación (`#man vim`) como en la ayuda dentro del propio editor escribiendo `:help` .

Para usar Vi, **primero deberemos comprender los tres modos en los que se puede trabajar en él** , cosa que veremos en unos instantes. No obstante, deberemos tener en cuenta que desde hace un tiempo las distribuciones Linux (debido a la creciente potencia de los equipos, y que los que van siendo antiguos tienen potencia suficiente…) implementan Vim en lugar de Vi. Vim (Vi improved, o Vi mejorado) no es más que una versión con capacidades y funcionalidades añadidas. Para algunas de las pruebas a realizar cambiaremos a uno u otro indistintamente.

Instalación de Vim

Comencemos instalando el editor:

```
sudo apt-get update
```

```
sudo apt-get install vim
```

O podemos usar una sola línea: `sudo apt-get update && sudo apt-get install vim`

```
yum update
```

```
yum install vim
```

O al igual que en el caso anterior, instalarlo con una sola línea: `sudo yum update && yum install vim`

La mayoría de los equipos tienen potencia como para funcionar con un entorno gráfico donde editar ficheros es más “fácil”, es lógico preguntarse **para qué aprender a usar Vi / Vim** . La respuesta es igual de sencilla, ya que **Vi siempre estará disponible en nuestro sistema UNIX** , mientras que otros editores tendremos que instalarlos, y si estamos en un equipo sin conexión a internet va a ser una tarea laboriosa.

Otra razón que podría dar es **la cantidad de recursos que podemos ahorrar al usar esta** y no otra utilidad para la edición de archivos de configuración u otros… así como la funcionalidad de no tener que levantar las manos del teclado para realizar cualquier modificación que queramos. La velocidad de edición que podemos desarrollar hará que no nos acordemos del ratón durante una temporada.

Además, lo extenso de **su manual dentro del sistema, hace que tengamos todas las opciones necesarias siempre a un comando de distancia.**

Para lanzar la aplicación únicamente deberemos escribir `vi` en nuestra consola de comandos.

Lo primero que veremos (si no hemos especificado nombre de archivo alguno) serán unas líneas en las que se nos informa de algunos comandos básicos, así como de la versión con la que estamos trabajando, el desarrollador, etc…

Si en cambio le damos al comando vi un archivo con el que trabajar o para crear, pasaremos directamente a la interfaz de trabajo.

```
vi archivo01.txt
```

Modos de Vi / Vim

Como comentaba antes, **Vi tiene tres modos básicos de trabajo** que detallo a continuación:

El ‘ **Modo Comando** ’ permite al usuario navegar por el documento así como introducir comandos a ejecutar dentro del propio archivo (buscar, reemplazar, guardar…), ya que en este modo el programa no interpreta las teclas del teclado por los caracteres que representan, si no por las funciones preestablecidas o comandos asignados a cada tecla.

Estos escuetos comandos son combinaciones de letras sensibles a mayúsculas (que corresponden a diferentes órdenes). A algunos de estos comandos podemos introducir un número delante que indicará el número de veces que queremos repetir la acción a ejecutar.

Por poner un ejemplo, si el comando `yy` (o `Y`) copia la línea donde se encuentra el cursor, el comando `3yy` (o `3Y`) copiará la línea actual y las dos siguientes, sumando un total de 3 líneas.

Para entrar al modo comando únicamente deberemos pulsar ‘Esc’ en nuestro teclado.

En el ‘ **Modo ex** ’ manipularemos los archivos. Para entrar a este modo deberemos escribir “ `:` ” seguido directamente por el nombre del comando ‘ex’ que queramos usar. Tras esto, vi volverá automáticamente al modo comando.

En el **Modo inserción** simplemente añadiremos texto al fichero. Mientras estemos en este modo, podremos salir al modo comando directamente pulsando ‘Esc’.

Existen otros modos para Vi / Vim como ‘visual’, ‘selección’, etc… pero estos tres son por los que más pasaremos.

Comandos Vi

Comandos	Uso
H	Desplazamiento a la parte superior de la pantalla
L	Desplazamiento a la parte inferior de la pantalla
G	Nos lleva hasta el final del documento
w	Desplazamiento una palabra a la derecha
b	Desplazamiento una palabra a la izquierda
0	Nos lleva hasta el inicio de la línea actual
$	Nos lleva hasta el final de la línea actual
Ctrl+B	Función similar a Repag.
Ctrl+F	Función similar a Avpag.
i	Comienza a introducir texto en la posición actual del cursor
I	Comienza a introducir texto al inicio de la línea donde se encuentra el cursor
O	Inserta una línea en blanco antes de la línea actual

o	Inserta una línea en blanco después de la línea actual
r	Sustituye el carácter en la posición actual del cursor
R	Sobrescribe desde la posición actual del cursor
x	Borra el carácter de la actual posición del cursor
X	Borra el carácter siguiente a la actual posición del cursor
dd	Corta la línea actual (disponible en el portapapeles)
D o d$	Corta desde la posición actual del cursor hasta el final de la línea
yy o Y	Copia al completo la línea donde se encuentra el cursor
yX	Copia tantos caracteres desde la posición del cursor, como le pasemos sustituyendo X por un número que estimemos necesario.
P	Pega en la línea previa a la que nos encontremos el contenido del portapapeles
p	Pega en la línea siguiente a la que nos encontremos el contenido del portapapeles
.	Repite el último comando
u	Deshace el último comando

U	Deshace el último comando aplicado a la línea donde se encuentre el cursor
n	Encuentra la siguiente coincidencia en una búsqueda
N	Encuentra la coincidencia anterior en una búsqueda
:n	En el caso de tener varios archivos abiertos a la vez, nos llevará al siguiente fichero.
:N	En el caso de tener varios archivos abiertos a la vez, nos llevará al fichero previo.
:buffers	Muestra un listado de los ficheros abiertos en el momento de la solicitud y el estado en que se encuentran
:buffer X	Nos lleva al fichero que le indiquemos sustituyendo X por el número de orden por el que se han abierto los archivos.
:e ‘Archivo’	Inserta el contenido de un fichero en la línea donde se encuentra el cursor
:r	Inserta el contenido de un fichero en la línea siguiente a la posición del cursor
:w o :w ‘Archivo’	Escribe los cambios en el fichero (desde el buffer) o Escribe los cambios en otro fichero que le pasemos
:q	Sale de Vi / Vim sin guardar los cambios
:wq o x! o ZZ	Guarda los cambios en el archivo actual y sale de Vi / Vim.

:r! ‘Comando’	Ejecuta un comando en la consola del sistema e inserta la salida de dicho comando en el fichero actual, desde la posición del cursor

Opciones Vi

A pesar de que estamos ante una herramienta de lo más versátil, el que sea ligera impide que por defecto añada algunas características, cosa que podemos arreglar fácilmente añadiéndolas nosotros mismos al fichero .vimrc.

set number : Muestra la numeración de las líneas de texto del fichero.

syntax on : Añade colores a diferentes componentes de texto haciendo la lectura de ficheros de configuración o archivos con código mucho más fácil.

set tabstop=4 : Configura la tabulación a 4 espacios, ya que por defecto viene establecida a 8.

set autoindent : Se encargará de que si una palabra no tiene cabida completa en la línea actual, en lugar de partirla con un guión al final de la línea, dicho guión se encontrará al inicio de la siguiente línea, para facilitar el salto de una línea a otra.

Búsqueda y Sustitución

Con Vi / Vim tenemos la posibilidad de **llevar el cursor directamente a una ubicación en concreto** , basándonos en la búsqueda dentro del texto, cosa que es perfecta si queremos sustituir cierto término por otro.

Con el comando “ `:/` ” podremos buscar en todo el documento la palabra que escribamos tras el carácter de la barra (/) y moveremos el cursor hasta el primer resultado que coincida.

Si en un documento quisiéramos encontrar “polimorfia”, escribiríamos “ `:/polimorfia` ” lo que nos llevaría a la primera aparición en el documento de dicha palabra.

Conociendo esto, sustituir una palabra por otra, es tan fácil como seguir el formato `%s/sustituida/sustituta/` . Como siempre, un ejemplo para esclarecer un poco más esto último. Si quisiéramos sustituir “Open” por “Open.net”, esta sería la forma de expresarlo:

```
:%s/Open/Open.net/gc
```

%s : Especifica el rango que abarcará la búsqueda. En nuestro caso % significa que se buscará desde la primera hasta la última línea del documento.

/Openwebinars : Es el término que sustituiremos.

/Openwebinars.net/ : Es el término que aplicaremos en su lugar.

g : Indica que la sustitución se realizará con toda coincidencia de búsqueda.

c : Solicitará confirmación de sustitución cada vez que se encuentre la palabra a buscar.

A la hora de solicitar confirmación, Vi / Vim nos ofrecerá una serie de opciones que paso a detallar:

- **y** : Confirmamos la acción
- **n** : Saltamos esta coincidencia con la búsqueda sin sustituirla y pasamos a la siguiente
- **a** : Confirmamos la acción para esta y todas las siguientes coincidencias
- **q** : Dejamos de sustituir en la búsqueda
- **l** : Confirmamos la sustitución y paramos la búsqueda saliendo de nuevo al modo editor

- **Ctrl+e** : Avanzamos un poco hacia abajo en el documento para localizar el contexto de la coincidencia
- **Ctrl+y** : Retrocedemos un poco en el texto para localizar el contexto de la coincidencia.

Editando varios archivos a la vez

Se le pasamos varios archivos al comando Vi / Vim, los abrirá simultáneamente y podremos movernos de uno a otro mediante comandos dentro de su “modo comando”.

```
vim txt1 txt2 txt3
```

El orden en el que los mostrará será aquel que le hayamos pasado, es decir, en nuestro caso será txt1. Para saltar al fichero txt2 deberemos teclear `:n` en el modo comando, y para volver al anterior la orden será `:N`

Si contamos con varios ficheros abiertos y no recordamos exactamente cuáles son, el comando `:buffers` nos mostrará algo como esto:

Se trata de un listado con los ficheros abiertos que tenemos, en el que podemos observar cómo se han añadido algunos caracteres tras el número que indica el orden de los archivos.

: Nos indica que el fichero ha sido abierto y que hemos pasado por él, aunque ahora no nos encontremos en el mismo, por lo que se encuentra abierto en segundo plano.

%a : Indica que es el archivo en el que nos encontramos y está siendo editado.

Un espacio en blanco : Indica que el fichero ha sido abierto, pero aún no hemos pasado por dicho archivo.

Otra forma de movernos por los diferentes ficheros, es utilizando el comando `:buffer` (sin la ‘s’ final que tenía :buffers) seguido del número por el que se ordenaron los archivos, es decir, siguiendo nuestro ejemplo, para saltar del primer archivo al tercero, deberíamos usar `:buffer 3` , y de tratarse del caso contrario, pasaríamos del tercero al primero con `:buffer 1.`