Editor de texto VI

Introducción

“vi” es el editor de texto m**ás utilizado en el mundo GNU/Linux y en ambientes basados en UNIX.**
Conocer “vi” nos va a resultar de mucha utilidad ya que es el editor de textos predeterminado en este sistema operativo.

Es muy rápido y permite insertar la salida de los comandos directamente en el texto, así como también nos permite abrir archivos muy grandes (estamos hablando de varias gigas).

Antes que nada, vamos a instalar **vim.** Que es la versión moderna de vi:

- **`sudo apt -y install vim`**

Editor de textos “vi”

Al invocar al "vi" con el nombre de un archivo, si este no existe lo crea. En la pantalla del "vi" aparece la posición actual del cursor resaltada, las líneas en blanco con **~** y en la parte inferior de la pantalla aparece **la línea de estado, que muestra el nombre del archivo y el número de caracteres que contiene.**

Existen dos modos de operación en el "vi":

- **Modo entrado**: Se usa para añadir texto al archivo
- **Modo comando**: Se usa para introducir comandos que realizan funciones específicas de "vi".

Como "vi" no especifica en qué modo se trabaja en un momento determinado, separar entre ambos modos es probablemente lo menos intuitivo para los nuevos usuarios de vi, junto con saber cómo salir el editor de texto.

Al abrir por primera vez un archivo "vi", siempre está en modo comando, por lo que **antes de poder escribir texto en el archivo debemos teclear como**

`i (insert)`

Para insertar texto a partir de la posición actual del cursor (lo que se conoce como agregar)

`a (append)`

Para insertar texto después de la posición actual del cursor.

Cuando se desee regresar al modo comando, basta con presionar (varias veces por las dudas)

Esc

De hecho, es una buena práctica (o algo útil) presionar varias veces la tecla *esc* **para estar seguros que regresamos al modo comando**.

Modo comando

En el momento de abrir un archivo "vi", **ya estás en modo comando.** Aquí puedes introducir comandos. Las órdenes pueden comenzar por dos puntos
:

Las ordenes precedidas por dos puntos son comandos; pero se las conoce más como expresiones y la parte donde uno escribe código vs estas órdenes son dos interfaces separadas del mismo programa de edición de texto.

Cuando se edita un archivo con el "vi", los cambios se aplican a una copia del archivo que el "vi" crea en un espacio de memoria temporal llamado buffer. La copia en disco del archivo se modifica solo cuando se graban los contenidos del buffer de forma efectiva.

Salir de vim

Para grabar salvar o guardar nuestro trabajo sin salir de "vi" basta pulsar la secuencia:
Esc :w[ENTER]

Para salir cuando no se han hecho modificaciones:
Esc :q[ENTER]

Para salir cuando se han hecho modificaciones si se quieren descartar:
Esc :q![ENTER]

Si se quieren guardar los cambios, podremos de cualquiera de estas dos formas .:
Esc :wq![ENTER]
Esc :x![ENTER]

- 5 -Comandos básicos de "vi"
Moviendonos por un archivo
Cuando arrancamos el "vi", el cursor esta en la esquina superior izquierda de la pantalla del "vi".

De a un carácter

En modo comando, existen órdenes que nos permiten movernos por toda la pantalla. Esto es que **no será necesario utilizar las teclas direccionales del teclado** sino ciertas teclas de letras. **Es muy importante tener en cuenta que los comandos son case sensitive.**

Entonces, estas pueden ser
:

- **Izquierda h**
- **Derecha l**
- **Arriba k**
- **Abajo j**
- **Línea 3 (por ejemplo) :3**

Palabras

Ya aprendimos a movernos caracter a caracter. Ahora ganemos velocidad moviéndonos entre diferentes líneas:

- **w: Una palabra hacia la derecha**
- **b: Una palabra hacia la izquierda**

Línea a línea

Luego aprendamos a posicionarnos dentro de la misma línea

- **^: Comienzo de la línea en la que está el cursor**
- **$: Al final de la línea actual**
- **ENTER nos movemos al comienzo de la línea siguiente**

Pantalla a pantalla

Y tenemos textos largos. Movernos a a líneas no nos sirve, perdemos mucho tiempo. Veamos como podemos movernos de pantalla a pantallas:

- **Pulsando H nos movemos a la parte superior de la pantalla. De High**
- **Pulsando L nos movemos a la parte inferior de la pantalla. De Low**
- **Pulsando M nos movemos a la mitad de la pantalla. De Middle**

Pero si lo que queríamos era avanzar o retroceder de a páginas enteras podremos utilizar los siguientes comandos. Recuerden que estos comandos son case sensitive.

- **Ctrl-F se avanza una pantalla, moviéndo el cursor al principio de la pantalla siguiente**
- **Pulsando Ctrl-D se avanza media pantalla**
- **Pulsando Ctrl-B se retrocede una pantalla**
- **Pulsando Ctrl-U se retrocede media pantalla**
- **Insertando texto**

Creando texto

"vi" proporciona algunos comandos para insertar texto que nos harán saltar el modo comando al modo edición. O sea, que nos permitirán directamente pasar del modo comando a insertar texto dentro de nuestro archivo. Pero recuerden siempre que **los cambios no pueden ser guardados hasta que lo hagamos nosotros efectivamente como lo vimos antes**

- **Pulsando a se inserta texto a partir de un lugar a la derecha del cursor**
- **Pulsando A se posiciona al final de la línea en la que está el cursor y añade exto a partir de ahí.**
- **Pulsando i se inserta texto a la izquierda del cursor.**
- **Pulsando I se inserta texto al principio de una linea.**

Crear línea e insertar texto

Estos dos comandos son particularmente útiles cuando queremos empezar a insertar texto en una línea nueva debajo o encima de la actual.
Esto es algo que hacemos mucho cuando programamos. Y nuevamente observen el caso que esto es **case sensitive.**

- **Pulsando o se abre una línea debajo de la posición actual del cursor**
- **Pulsando O se inserta una línea encima de la actual posición del cursor.**

Sustituir texto

Sustituir texto implica dos acciones. Borrar el texto e insertar uno nuevo sobre lo borrado. Aquí podremos.

Cambiar una palabra / línea

En este punto debemos tener en cuenta que cambiar una palabra signifia **cambiar desde la posición en la cual me encuentro hasta el final de la palabra.** Esto es, si me paro a la mitad de la palabra, cambiaré la mitad. Si me paro al principio de la palabra, cambiaré toda la palabra. Esta combinación de letras debe ser escrita siempre en modo comando:

- **`cw seguido de la nueva palabra`**.

Y cuando se termine de modificar pulsaremos **Esc** para volver a modo comando. Parece complicado al principio, pero termina siendo bastante fácil

Con las líneas seguiremos la misma lógica, esto es, que cambiar línea implica cambiar desde el lugar en el cual me encuentro parado hasta el final. Esto es, que **si me paro en la mitad de la línea, modificaré la mitad de la línea hasta el final. Pero si me paro al principio de todo modificaré la totalidad de la línea.**

- **`colocar el cursor a la derecha de la parte a modificar. C, introducir la corrección y pulsar Esc`**

Pero si queremos cambiar la totalidad de la línea:

- **`teclear cc.`**

Para unir dos líneas,

- **`colocar el cursor en la línea superior y teclear J.`**

Sustituir caracteres

Sustituir caracteres implica modificar el caracter que se encuentra por debajo de nuestro cursor.

- **`pulsar r, seguido por un único carácter`**

Deshacer cambios

El undo es importantísimo para corregir errores, o simplemente cuando el gato pasó caminando por arriba de nuestro teclado. Sin movernos de la línea

- **`Pulsando u se desha únicamente el último cambio`**

Pero si deseamos deshacer la totalidad de los cambios hechos en una única línea podemos

- **`pulsar la tecla U`**

Se deshacen todos los cambios que se han hecho sobre una línea. Para que esto funcione **no debemos movernos de la línea en cuestión.**

Borrar

¡Aún no vimos como eliminar texto! Para borrar un carácter, es tan **intuitivo como venimos trabajando hasta ahora con cualquier editor de texto**. Siempre en modo comando, nos posicionamos a partir del lugar que queremos borrar

- **`pulsar la tecla x, hace la misma función que suprimir en cualquier editor de texto`**

Para borrar estilo **backspace,** siempre recordando que esto es case sensitive,

- **`pulsar X.`**

Borrar una palabra

Aquí seguiremos la misma lógica. Esto es, que vamos a borrar **desde el punto en el cual posicionamos el cursor hasta el final de la palabra.** Por lo que si nos paramos al principio de la palabra, vamos a borrarla completamente.

- **`Posicionar el curso y pulsar dw`**

Entonces borrará la palabra a partir de la posición en la cual nos encontramos.

Borrar una línea

Para borrar la totalidad de la línea

- **`Debemos pulsar dd`**

Recuerden que esto siempre en modo comando. Pero si deseamos borrar parte de la linea:

- **`Lo que esté a la derecha del cursor, basta con pulsar D`**
- **`Lo que esté a la izquierda del mismo basta con pulsar d0`**

Borrar hasta el final del archivo

Y si, sigue la misma lógica. Esto borrará desde donde estemos posicionados hasta el final del archivo. Observen lo siguiente que es algo que se da bastante común dentro de vim. Si

ejecutamos por separado las siguientes sentencias: **:1 te posiciona al principio, d borra y G te posiciona al final del archivo. E.**

- :1dG borra todo
- dG borra desde donde estamos parados hasta el final del archivo

Copiar mover borrar - Yank, Delete y Put

Como todo editor de texto vamos a poder mover texto y copiar y borrar. Dado que en inglés esto lo tenemos como **yank, delete y put** de ahí que utilizaremos las iniciales de estas palabras como reglas mnemotécnicas para su uso.

Copiar líneas

Para copiar de a una línea son necesarios dos comandos:

- **`yy para copiar la línea entera`**
- **`p para pegar la línea debajo del del cursor`**

Pero, por ejemplo, si queremos copiar las **3 líneas que se encuentran debajo del cursor**, simplemente nos posicionamos en dicha línea y, al igual que con ***Borrar hasta el final del archivo*** utilizaremos el número para indicar cuánto de esa acción queremos realizar.

- **`3yy copiará la línea en la que estoy 3 2 líneas más hacia abajo`**

También podremos copiar de a bloques de líneas. Esto es lo mismo que cuando utilizamos un editor de texto gráfico: **resaltar el texto que queremos copiar, de ahí copiar y pegar en la posición correspondiente.** Para esto nos posicionamos en la línea a partir de donde queremos copiar y haremos:

- **`:set nu para que aparezcan las líneas (set nonu para quitarlas)`**
- **`:co [línea inicio],[línea fin]`**
- **`nos movemos al lugar donde queremos pegar las líneas`**
- **`p`**

Mover líneas

También vamos a poder mover un bloque entero de líneas de un lugar a otro. Esto particularmente lo encontré extremadamente útil cuando programo y tengo que mover un bloque de código. Por ejemplo, **este if no va acá, sino que va más arriba.**

- **`:set nu para que aparezcan las líneas (set nonu para quitarlas)`**
- **`:co [línea inicio],[línea fin]`**
- **`nos movemos al lugar donde queremos pegar las líneas`**
- **`p`**

Búsqueda y reemplazo

Supongamos que tenemos una variable la cual deseamos renombrar. Pero no podemos buscar "manualmente" en todo nuestro código o texto esa variable y cambiarla. Para esto contaremos con una **búsqueda y cambio (si lo deseamos).** Algo que nos proporciona vi es la capacidad de utilizar expresiones regulares, por lo que si dominamos esa será más sencillo su utilización.

Búsqueda

Para simplemente encontrar una cadena de caracteres hay que teclear la barra inclinada **"/"**

- **`/string`**
- **`teclear n para la próxima ocurrencia o N para la anterior`**

Una vez que llega al final o principio del archivo repita la búsqueda.
Las búsquedas son case-sensitive. Si queremos que **vi** ignore esta cualidad podemos pensarlo como **i**gnore **c**ase:

- **`:set ic[ENTER]`**
- **`/string`**

Como vimos anteriormente, las opciones que seteamos con set (valga la redundancia) las desactivamos con el **no delante de la opción.** Por lo que en este caso sería algo como:

- **`:set noic[ENTER]`**

Y pasa a ser case sensitive.

Reemplazo

Como venimos observando, en vi podemos decir que todo tiene que ver con todo. De esta manera, el procedimiento para el reemplazo está relacionado con la búsqueda. Nos referimos a que primero **debemos dominar la búsqueda para después poder dominar la búsqueda y reemplazo**. Junto con esto, quienes también hayan trabajado con el comando **sed** van a encontrar el comando y el criterio de reemplazo particularmente parecido:

- **`:%s/string_a_buscar/string_reemplazado/g[ENTER]`**

Por ejemplo, **para reemplazar las ocurrencias de "maximo" por "MAX_NUM"**, se teclearia:

- **`:%s/maximo/MAX_NUM/g[ENTER]`**

Esto nos va a reemplazar todas las ocurrencias; **sin preguntarnos.**

Se puede modificar este comando **haciendo que vi pida confirmación para la modificación por (y)es o (n)o.** Estas no son las únicas opciones, pero son las que más utilizaremos cuando queremos confirmación.

- **`:%s/desinformacion/informacion/gc[ENTER]`**

Set

Podremos editar nuestro vi para que, como vimos antes, cambie el aspecto y la metainformación. Esto es mostrar por ejemplo los números, la sintaxis etc.

Para esto utilizábamos los comandos del tipo **:set (no)[opción].** Veamos algunas opciones más… y recuerden que para desactivar la opción simplemente ingresamos el **no** delante de la opción seteada. Esto siempre en modo comando.

Aunque parezca contradictorio, lo primero a hacer es anular la compatibilidad con el viejo **vi** y activar la compatibilidad con **vim (vi improved).**

- **`:set nocompatible`**

VIM nos crea un archivo de backup cada vez que editamos. Por lo que si no queremos que o haga:

- **`:set nobackup`**
- **`:set nowritebackup`**
- **`:set noswapfile`**

Para activar el reconocimiento del tipo de archivo

- **`:filetype on`**

Para cargar los plugins con los que contemos, basado en el archivo que reconoció

- **`:filetype plugin on`**

Para activar el inventario y setearlo a dos espacios:

- **`:set autoindent`**
- **`:set expandtab`**
- **`:set shiftwidth=2`**

Por último, el inconveniente que tiene esto es que una vez cerrado vim estos cambios se pierden. Para que queden persistentes debemos colocar estos mismos comando en un archivo llamado .vimrc entro de nuestro home.

Para más información **puden acudir a la página del manual:**
http://www.sromero.org/wiki/linux/aplicaciones/manual_vim