Capítulo 6: El editor de texto vi

Introducción

Una de las herramientas esenciales que todo *sysadmin* debe utilizar en sus tareas diarias es un editor de texto. La razón principal es que los archivos de configuración de los servicios del sistema y de las preferencias de usuario se presentan en formato de texto plano.

En el mundo Linux existen varios editores de texto entre los que podemos elegir. Algunos de los más conocidos son **emacs**, **nano**, y **vi** (o **vim**, su versión mejorada). Este último goza de gran popularidad en la comunidad, y será el editor que aprenderemos a utilizar en este capítulo. Una de las ventajas de **vi** es que se instala por defecto junto con cualquier distribución.

Si lo deseamos, también podemos utilizar una versión gráfica de **vi** que se llama **gvim**. Para hacerlo, lo más probable es que necesitemos instalarlo primero.

Modos de operación de vi

Al invocar a vi seguido del nombre de un archivo, lo crea si no existe. En la pantalla aparecerá la posición actual del cursor resaltada, las líneas en blanco indicadas con el símbolo **~**, y en la parte inferior la línea de estado. En este sector se mostrará el nombre del archivo y el número de caracteres que contiene.

Existen dos modos de operación en el **vi**:

- **Modo entrada**: Se usa para añadir texto al archivo.
- **Modo comando**: Es el modo de operación por defecto. Se usa para introducir comandos que realizan funciones específicas del editor.

---

*Puesto que **vi** no indica en qué modo se trabaja en un momento determinado, distinguir entre ambos es probablemente el mayor problema que causa confusión entre los nuevos usuarios de este editor.*

Cuando se abre por primera vez un archivo, **vi** siempre se encuentra en modo comando. Por esa razón, antes de poder escribir texto en el archivo se necesita presionar una de las teclas que habilitan el modo entrada:

- **i** (**i**nsert), para insertar texto en la posición actual del cursor.
- **a** (**a**ppend) para insertar texto después de la posición actual del cursor.

Para regresar del modo entrada al modo comando bastará con presionar la tecla **Esc**. De la misma manera podemos pasar a modo comando si en un momento dado no sabemos en qué modo estamos.

El modo comando

Cuando abrimos un archivo con **vi**, estamos en modo comando. En este modo podemos introducir comandos que implementan un amplio rango de funciones. Muchos de los mismos constan de una o dos letras y un número opcional relacionado con distintas funciones.

Cuando editamos un archivo con **vi**, los cambios no se hacen directamente sobre el archivo. En realidad, se aplican a una copia temporal del archivo que **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*.

Esto tiene sus ventajas y sus inconvenientes. Por un lado, significa que podemos salir de la edición de un archivo y descartar todos los cambios hechos durante una sesión de edición, dejando la copia de disco intacta. Por otro lado, podemos perder el contenido no grabado del *buffer* de trabajo si el sistema cae.

La mejor política en todos los casos es grabar cambios frecuentemente, especialmente cuando hayamos hecho cambios importantes.

Para grabar nuestro trabajo sin salir del **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 queremos descartar: `Esc :q! [ENTER]`
- si queremos guardar los cambios: `Esc :wq [ENTER]`

Comandos básicos de vi

Cuando arrancamos **vi**, el cursor está en la esquina superior izquierda de la pantalla. En modo comando, existen ordenes que nos permiten moverlo por toda la pantalla: ciertas teclas de letras, las flechas, el ENTER, BackSpace (tecla de borrar), y Space Bar (barra espaciadora). Los comandos de **vi** son *case-sensitive*, es decir, la misma orden tecleada en mayúsculas o minúsculas puede tener efectos totalmente distintos.

Si el teclado está provisto con flechas, se pueden utilizar para mover el cursor *caracter a carácter* libremente por el texto editado hasta el momento. Si se usa

desde un terminal remoto, las flechas pueden no funcionar correctamente, dependiendo del emulador de terminal. En dicho caso las teclas usadas para desplazarnos son:

- Izquierda: tecla **h** o barra espaciadora
- Derecha: tecla **l** (*ele*) o BackSpace
- Arriba: tecla **k**
- Abajo: tecla **j**

Para desplazarnos *palabra a palabra* emplearemos

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

Si deseamos movernos a lo largo de una línea:

- Con **^** nos posicionamos al comienzo de la línea en la que está el cursor
- Con **$** nos dirigimos al final de la línea actual
- Con [ENTER] nos desplazamos al comienzo de la línea siguiente.

Dentro de la misma pantalla, con

- **H** nos movemos a la parte superior de la pantalla
- **L** nos desplazamos a la parte inferior de la pantalla
- **M** nos posicionará en la mitad de la pantalla

Para avanzar o retroceder de página puede resultar un poco tedioso utilizar las teclas mencionadas anteriormente si el archivo es muy extenso. Por eso, **vi** provee las siguientes combinaciones que permiten desplazarnos *página a página* por el documento:

- **Ctrl + f** y **Ctrl + b** hace que avancemos o retrocedamos una pantalla, y que el cursor se mueva a la esquina superior o inferior izquierda, respectivamente.
- **Ctrl + d** y **Ctrl + u** nos permite avanzar y retroceder media pantalla, respectivamente.
- Para ir a la última línea de un archivo abierto, deberemos teclear **G**. Si deseamos ir a la línea **n**, se presiona el número que corresponda seguido de **G**. Por ejemplo, **23G** hará que saltemos a la línea 23 del archivo actual.

Insertar texto

A la hora de pasar desde el modo comando a modo entrada texto disponemos de varias posibilidades. Todas ellas pueden ser útiles de acuerdo a la posición donde deseemos comenzar a agregar texto relativa a la ubicación actual del cursor.

- Con **a** se inserta texto a la derecha del cursor.
- Con **A** se añade texto al final de la línea en la que está el cursor.
- Presionando **i** se inserta texto a la izquierda del cursor.
- Presionando **I** (*i mayúscula*) se inserta texto al principio de la línea en la que está el cursor.
- Mediante **o** podremos crear una nueva línea debajo de la actual.
- Mediante **O** podremos insertar una nueva línea encima de la actual.

También es posible insertar los contenidos de un archivo dentro del que estemos editando. Para hacerlo, necesitaremos hacer lo siguiente:

```
:linea r archivo
```

donde **linea** es representa la línea encima del punto de inserción, **r** indica la operación, y **archivo** es la ruta al archivo a insertar. Si el nombre del archivo a insertar contiene un punto (como por ejemplo, para indicar la extensión del mismo) se debe escapar utilizando la barra ****. Por ejemplo, para insertar **pruebas.txt** en la línea 10 del archivo actual, deberemos hacer:

```
:9 r pruebas\.txt
```

Además, puede resultar útil insertar el resultado de un comando al editar un archivo. La sintaxis es muy similar al caso anterior:

```
:linea r! comando
```

Por ejemplo,

```
:5 r! which python3
```

insertará el resultado de `which python3` en la línea 6 del archivo.

Cambiar texto

Cambiar texto implica sustituir una sección de texto por otra. El editor **vi** tiene varios modos de hacer esto, dependiendo de la necesidad. Veamos algunos ejemplos:

- Para *reemplazar una palabra*, posicionar el cursor al principio de la palabra a ser reemplazada, teclear **cw** seguido de la nueva palabra. Cuando terminemos de realizar la modificación deberemos pulsar **Esc** para finalizar.
- Para *cambiar parte de una palabra*, colocar el cursor sobre la palabra, a la derecha de la parte a cambiar, y proceder como en el caso anterior.
- Para *reemplazar una línea*, poner el cursor en cualquier parte de la línea y teclear **cc**. La línea desaparece, dejando una línea en blanco para el nuevo texto, que puede ser de cualquier longitud. Pulsar **Esc** para acabar.

- Para *reemplazar parte de una línea*, colocar el cursor a la derecha de la parte a modificar. A continuación, presionar **C**, introducir la corrección y pulsar **Esc** para finalizar.
- Para *substituir el carácter bajo el cursor* por uno o más caracteres, teclear **s** seguido del nuevo texto y pulsar **Esc** para terminar.
- Para *reemplazar el carácter bajo el cursor por otro*, pulsar **r** seguido por un único carácter, pues al pulsar una tecla **vi** inmediatamente pasa a modo comando.
- Para *trasposicionar caracteres*, colocar el cursor sobre la primera letra a mover y pulsando **xp**, se intercambian las posiciones de ambos caracteres. Esto es útil para fallos tales como escribir *qeu* en lugar de *que*.
- Para *partir una línea sin afectar al texto*, mover el cursor al espacio donde se quiere partir la línea y teclear **r** (de **r**eplace) seguido de ENTER (se reemplazaría el espacio por un ENTER).
- Para *unir dos líneas*, colocar el cursor en la línea superior y teclear **J**.
- Pulsando **u** a continuación del último comando, se *deshacen los cambios* producidos por la ejecución del mismo.
- Pulsando **U** se *deshacen todos los cambios* que se han hecho sobre una línea. Este comando funciona solo si no nos movimos de la línea.
- Para *borrar un carácter*, posicionar el cursor sobre el carácter a borrar y teclear **x**. El comando **x** también borra el espacio ocupado por el carácter.
- Para *borrar el carácter anterior* a la posición del cursor pulsar **X**.
- Para borrar una palabra, posicionar el cursor al principio de la palabra y pulsar **dw**, entonces se borrará la palabra y el espacio que esta ocupaba.
- Para borrar parte de una palabra, hay que colocar el cursor a la derecha de la parte a modificar y teclear **dw**.
- Pulsando **dd** se *borra una línea y el espacio* que esta ocupaba.
- Para *borrar todo lo que esté a la derecha del cursor* basta con pulsar **D**.
- Para *borrar todo lo que esté a la izquierda* del mismo basta con pulsar **d0**.
- Por otro lado, **dG** borra *desde* la línea en que estaba el cursor hasta el final del archivo.
- De forma similar, **d1G** borra *hasta* desde el principio del archivo hasta la línea en que estaba el cursor.

Copiar y mover texto: Yank, Delete, y Put

De la misma manera que muchos procesadores de text nos permiten copiar y pegar, o cortar y pegar líneas de texto, **vi** también incluye esta posibilidad, mediante los comandos *yank* / *put* y *delete* / *put*, respectivamente.

Para *copiar* una línea son necesarios dos comandos: **yy** o **Y** (**y**ank) y **p** o **P** para *pegar*. Para hacer lo mismo con *varias* líneas basta con colocarse en la primera línea a copiar y escribir el número de líneas a copiar seguido por **yy** (por ejemplo, **10yy** para copiar la línea actual y las 9 siguientes). Para pegar el texto, podemos usar los comandos **p** o **P**.

Es importante que entre la operación de copiar (o cortar) y pegar se usen solamente los comandos de movimiento de cursor. Si se borra o copia cualquier otro texto antes de poner el nuevo texto en su lugar, las líneas copiadas o borradas se perderán.

Para mover una o varias líneas de sitio requiere también dos comandos: **dd** (**d**elete) y **p** o **P**. Igual que en el caso anterior, se puede teclear antes de **dd** el número de líneas a mover.

Muchos comandos de **vi** pueden estar precedidos de un contador que indica cuántas veces repetir la operación. Muchos de los anteriores permiten contadores. Por ejemplo, **3dd** repite el comando de borrar una línea 3 veces, **2dw** borra dos palabras, y **4x** borra 4 caracteres. También se pueden usar contadores para los comandos de moverse por la pantalla que mencionamos anteriormente.

Un dato interesante es que pulsando un **.** (punto) se repite la última operación de cambio realizada. Por ejemplo si se borra una línea, y la siguiente operación a realizar es borrar una línea, basta con pulsar **.** para llevarla a cabo nuevamente.

Usar buffers con nombre

Para insertar repetidamente un grupo de líneas en varios lugares dentro de un documento, se puede copiar las líneas a un *buffer con nombre*. Se especifican los *buffers* con nombre escribiendo antes del comando comillas dobles y su nombre.

Por ejemplo:

- Para copiar 4 lineas al *buffer* **a** haremos **"a4yy**
- Para copiar 10 lineas al *buffer* **b** haremos **"b10yy**
- Para pegar las lineas del *buffer* **a** haremos **"ap** o **"aP**
- Para pegar las lineas del *buffer* **b** haremos **"bp** o **"bP**

Al emplear *buffers* podemos mantener al alcance varios bloques de texto para pegar cuando y donde nos resulte conveniente.

Uso de expresiones para copiar, mover, y borrar líneas

Cuando necesitamos copiar (o cortar) y pegar grandes cantidades de texto, el uso de las expresiones que explicaremos a continuación resulta más útil y seguro que emplear *yank*, *delete*, y *put*. Además, mejor que contar líneas en la pantalla y buscar el punto de inserción, se le pueden dar a **vi** un rango de líneas a mover o copiar y entonces especificar la línea anterior al punto de inserción. Como es de esperar, con un comando de borrado no existe punto de inserción.

El formato básico del comando para *copiar líneas* es la expresión:

```
:linea1,linea2 co linea3 [ENTER]
```

donde **linea1** y **linea2** especifican el rango de líneas a copiar (incluyendo ambos extremos), mientras que **linea3** representa la línea anterior al punto de inserción. La palabra clave **co** indica que estamos por realizar una operación de copia.

Para casos especiales como la línea actual o el final del archivo, podemos utilizar las siguientes abreviaturas:

- **.** denota la línea actual.
- **$** representa el final del archivo.

Para *mover y borrar* líneas el comando es similar. En el primer caso haremos

```
:linea1,linea2 m linea3 [ENTER]
```

mientras que en el segundo

```
:linea1,linea2 d [ENTER]
```

Búsqueda y reemplazo de cadenas de texto

Las cadenas de texto, también llamadas *strings*, son secuencias ordenadas de caracteres. Puede incluir letras, números, puntuaciones, caracteres especiales, espacios en blanco, tabulaciones, o retornos de carro. Un string puede ser una palabra gramatical o puede ser una parte de una palabra. El editor **vi** proporciona varios métodos para encontrar *strings* en un archivo de texto, así como para reemplazarlo.

Para encontrar un string, hay que teclear / seguido del string que se desea buscar, y entonces pulsar ENTER. Como resultado, se posicionará el cursor en la siguiente ocurrencia del string. Tecleando **n** se salta a la siguiente ocurrencia del string, y pulsando **N** a la anterior.

---

*Para buscar hacia atrás en un archivo se puede usar **?** en lugar de /.*
*En este caso, las direcciones de **n** y **N** se invierten.*

Normalmente las búsquedas son case-sensitive. Si se quiere ignorar mayúsculas y minúsculas (o *ignore case*) durante la búsqueda, teclear:

`:set ic[ENTER]`

Para volver al cambio por defecto bastará hacer:

`:set noic[ENTER]`

Ciertos caracteres especiales (/ & ! . ^ * $?) tienen un significado especial para el proceso de búsqueda y deben especificarse de un modo especial en la búsqueda, precediéndolos por la barra hacia atrás ****. Este procedimiento se conoce comúnmente como *escape de caracteres*. Por ejemplo, para buscar **algo?** hay que teclear `/algo\?[ENTER]`.

Para realizar búsquedas más precisas podemos recurrir a los siguientes filtros o ayudas:

- Para buscar en *el comienzo de una línea*, hay que preceder al string a buscar por **^**. Por ejemplo, para buscar la siguiente línea que empieza por *Sin embargo*, habría que teclear `:/^Sin embargo[ENTER]`.
- Para buscar en *el final de una línea*, hay que terminar el string a buscar por un símbolo **$**. Para ilustrar, el comando `:/nada\.$[ENTER]` nos permitirá buscar la siguiente línea terminada en *nada.* (incluyendo el punto).
- Para buscar *cualquier caracter*, deberemos utilizar el **.** (punto) en el string de la posición a emparejar. Por ejemplo, para encontrar la siguiente ocurrencia de *disinformation* o *misinformation*, deberemos hacer `/.isinformation[ENTER]`.

Debido a que en este caso estamos realizando la búsqueda de un string, y no de una palabra, este último patrón puede también encontrar palabras como ***misinformationalist*** *y* ***disinformationism****.*

- Para buscar caracteres alternativos en un string, poner las alternativas entre [ ] (corchetes). El patrón de búsqueda `/[dm]texto` encontrará los *strings* que comiencen por **m** o con **d**, seguidos de la palabra **texto**. Por otro lado, `/[d-m]texto`, devolverá *strings* que comiencen con cualquier letra de la **d** a la **m**, también seguidos por la misma palabra que en el caso anterior.
- Para buscar 0 o más ocurrencias del ultimo carácter, teclear un * (asterisco) en el *string*. Se pueden combinar corchetes con asteriscos para buscar por alternativas mejor definidas. Por ejemplo, para encontrar todos las cadenas de texto que comiencen con una una letra de la **a** a la **z** y terminen en **isinformation**, teclear `/[a-z]*isinformation[ENTER]`.

- Para reemplazar una cadena de texto la forma básica del comando es `:%s/texto_a_buscar/texto_de_reemplazo/g[ENTER]`. Por ejemplo, para reemplazar todas las ocurrencias de la palabra **carro** por **automovil**, se debería teclear `:%s/carro/automovil/g[ENTER]`. Si luego de la g agregamos una c (de tal forma que quede `:%s/carro/automovil/gc[ENTER]`), nos pedirá confirmación antes de reemplazar cada una de las ocurrencias de **carro**. Las respuestas posibles son **y** (de **y**es) y **n** (de **n**o).

Editar múltiples archivos

Para imitar de alguna manera las pestañas de los editores de texto que podemos llegar a encontrar en el modo gráfico, **vi** permite editar varios archivos.

Por ejemplo, para editar **usuarios.txt**, **grupos.txt**, y **accesos.txt** podemos invocar a **vi** de la siguiente manera:

```
vi usuarios.txt grupos.txt accesos.txt
```

Los archivos se identifican por número, correspondiendo en este caso el 1 para **usuarios.txt**, el 2 para **grupos.txt**, y finalmente el 3 para **accesos.txt**. Mientras estamos editando uno de ellos podemos saltar a otro guardando los cambios primero y luego haciendo `:n #numero[ENTER]` donde **numero** indica el número de archivo según explicamos. Por ejemplo, si estamos editando **usuarios.txt** podemos saltar a **accesos.txt** haciendo `:n #3[ENTER]`.

Si lo que nos interesa es copiar líneas entre archivos, **vi** también nos permite hacerlo:

1. Abrir el archivo *fuente* y copiar las líneas que nos interesen usando *yank* (se puede aplicar cualquiera de las técnicas que describimos en este capítulo).
2. Sin salir de **vi**, ir al archivo *destino* (digamos **datos.txt** por ejemplo) mediante `:n datos.txt`, y una vez ahí emplear *put* como hemos explicado.

Opciones de configuración

Afortunadamente, **vi** también nos permite configurar el ambiente de uso para que nos resulte lo más placentero posible. Las siguientes opciones de configuración pueden establecerse cada vez que abramos un archivo o bien colocarlas (una por línea) en el archivo **.vimrc** dentro del directorio personal de usuario que desee emplearlas.

Importante: si vamos a establecer estas opciones mientras estamos editando un archivo, las mismas estarán disponibles hasta que cerremos el mismo. Para guardarlas de manera permanente deberemos incluirlas en el archivo **.vimrc**. En este último caso **no** es necesario colocar los dos puntos (**:**) antes del comando propiamente dicho.

Para visualizar el número de cada línea, teclear:

```
:set nu[ENTER]
```

Para ocultarlas, teclear:

```
:set nonu[ENTER]
```

Por último, los siguientes comandos nos servirán (¡y de mucho!) si necesitamos utilizar **vi** para desarrollar scripts o algún otro tipo de código. De esta manera se fijará la indentación en 1 Tab = 4 espacios, y nuevas líneas tendrán el mismo nivel que la anterior.

```
set autoindent
set shiftwidth=4
set softtabstop=4
set expandtab
```