

# Manual de uso del editor de texto Vim

Gabriel López

8 de septiembre de 2022

## 1. Vim, un editor de texto minimalista.

Vim es un editor de texto creado por Brian Moolenaar. Es una versión mejorada del editor de texto vi creado por el programador Bill Joy. El editor de texto Vim es un editor de texto pensando para ser un programa de edición de texto minimalista y extensible, lo que significa que con ayuda de programas auxiliares hechos por la comunidad conocidos como *plugins*, se puede utilizar el editor Vim como un entorno de programación (conocido como *IDE*) o un entorno minimalista de edición de documentos hechos en L<sup>A</sup>T<sub>E</sub>X.

Si bien, aprender a usar Vim es un proyecto con un curva de aprendizaje un poco pronunciada, este editor ofrece ciertas ventajas por sobre otros editores, sobre todo a nivel de personalización, configuración y uso de recursos, lo que lo hace ídneo en *hardware* de capacidad limitada. Además de esto, con la correcta configuración y el correcto uso de *plugins*, la escritura (sobre todo de documentos de tipo científico) puede ser más fluida a un nivel muy similar a la toma de notas manuscritas.

## 2. Entornos en Vim

Vim es un editor de texto poco ortodoxo comparado con el resto de los editores de texto que están disponibles para uso público, para empezar, Vim usa un esquema de entornos *entornos*, lo que quiere decir, que para las acciones que usualmente se ven relegadas al uso del ratón en otros editores, en Vim están relegadas a un *entorno*. Los entornos de uso básico son los siguientes:

- **NORMAL**: Este entorno es el entorno básico de Vim. En este entorno se utiliza para navegar entre líneas de texto, borrar contenido y introducir comandos dentro del editor. La utilidad del modo **NORMAL** radica en el uso de comandos para usar la terminal y el editor al mismo tiempo.
- **INSERTAR**: Este entorno se utiliza para la inserción de texto dentro de líneas en el editor. Para entrar a este entorno basta con introducir la tecla **i** mediante el teclado. Una vez que la tecla se ha introducido, el editor colocará el texto **INSERTAR** o **INSERT** en la parte inferior de la pantalla, denotando que el modo de inserción está activado.
- **VISUAL**: Este entorno se encarga de función de copiar y pegar contenido de las líneas. Para entrar a este modo, se debe introducir **v** con el teclado. Una vez que el modo **VISUAL** está activado debe aparecer en pantalla la leyenda **VISUAL** en la esquina inferior izquierda.

Es importante destacar que para salir de un entorno para entrar de nuevo al entorno *NORMAL* se introduce la tecla **Esc** con el teclado. Para ilustrar los entornos de **Vim** colocamos la siguiente imagen. En dicha imagen es notorio que debido al idioma con el que se tiene el ordenador, aparece la leyenda **INSERTAR** en la parte inferior, denotando que el modo de inserción de texto está activo.

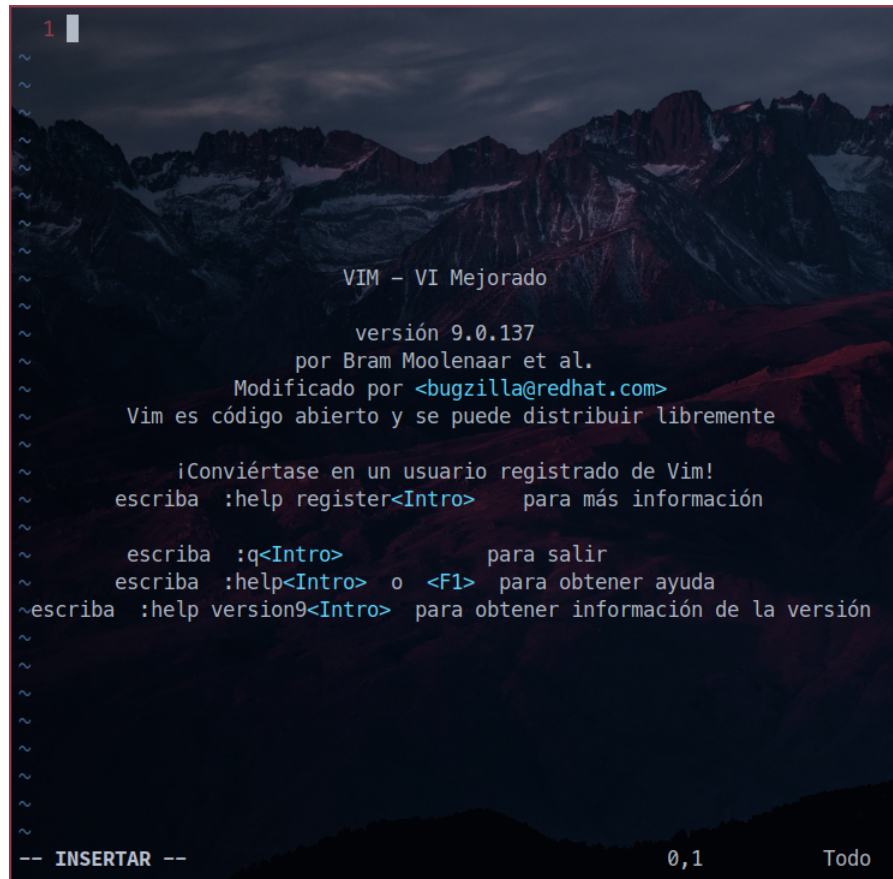


Figura 1: Vim en el modo *INSERTAR*

A pesar de todo esto, existen muchos más entornos dentro del editor, sin embargo, consideramos que estos tres son los más útiles para un usuario nuevo de **Vim** debido a que facilitan la escritura de documentos y permiten la introducción de comandos dentro del mismo.

### 3. ¿Cómo salir de Vim?

Es normal que el usuario novato de GNU-Linux o cualquier sistema operativo derivado de UNIX. De modo que el sistema operativo a veces deja el usuario dentro de Vim sin ningún aviso de cómo salir del editor de texto. Por esta razón en esta sección se explicará como salir de Vim. Para empezar, la imagen inicial que el usuario tiene al entrar a Vim es la siguiente:

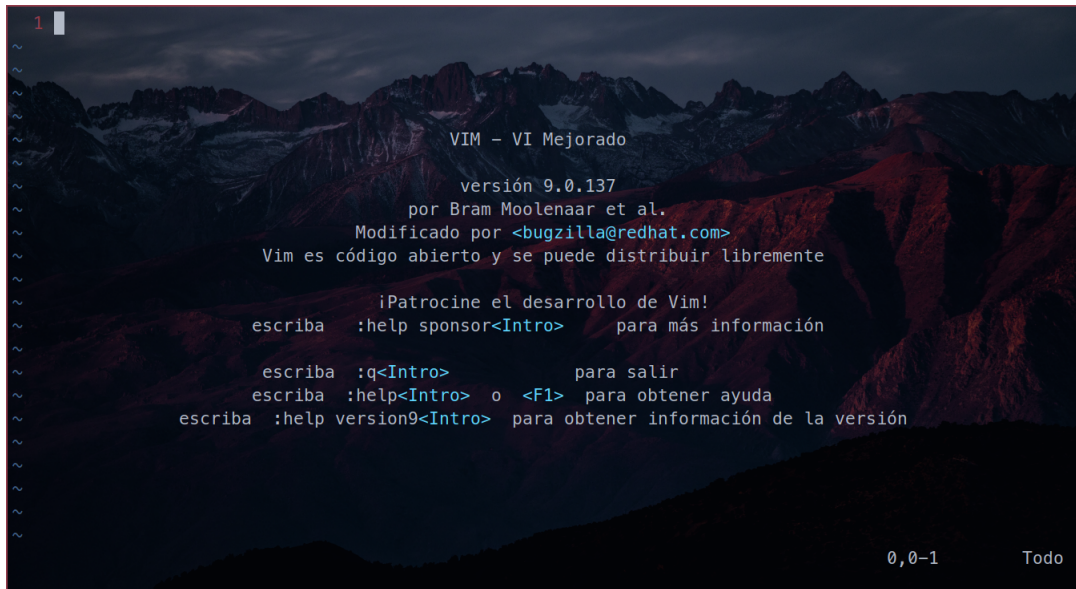


Figura 2: Primera impresión de Vim

Podría parecer que Vim es complicado, pero en realidad, salir del editor es relativamente sencillo, la secuencia de pasos para salir de Vim es la siguiente:

- Primero, si usted desea guardar el contenido del archivo, es conveniente guardar el contenido del archivo primero, esto se logra mediante el comando `:w` dentro del entorno *NORMAL*.
- Ahora que el archivo está guardado, es posible salir del editor sin problemas, por lo que se procede a colocar el comando `:q` dentro del mismo entorno *NORMAL*.

Si bien es posible que queramos guardar el contenido de nuestro archivo de texto, también tenemos la opción de salir sin guardar (no recomendada a menos que se desee salir de Vim si se entró por error), en este caso, basta con introducir el comando `:q!` en el modo *NORMAL* del editor.

### 4. ¿Existe alguna forma de aprender Vim mediante ejemplos?

Si bien Vim es un editor de texto que puede parecer poco ortodoxo al principio, existen formas para aprender su uso de una forma interactiva mediante ejemplos. En el caso de Vim existe el complemento denominado *vimtutor*, que es un complemento de consola que suele estar instalado

en cualquier máquina UNIX. En el caso de querer acceder a dicho complemento, basta con introducir el comando `vimtutor` dentro de la terminal de GNU-Linux o su equivalente en UNIX.

## 5. Navegación en Vim

Vim es un editor de texto que tiene un esquema de navegación diferente al resto de editores de texto que podemos encontrar en el mercado. Esto es debido a cuestiones de convención. Si bien, las teclas direccionales del teclado funcionan para navegar entre texto dentro del editor, su uso no es recomendado por la mayoría de guías de uso de Vim.

### 5.1. Navegación básica en Vim

Esto se entiende mejor con los ejemplos proporcionados por el complemento de consola llamado `vimtutor`. Para nuestros efectos, se busca introducir al usuario a los conceptos básicos de los que Vim para que sea capaz de redactar documentos simples dentro del editor. Dicho esto, podemos mostrar el esquema básico de navegación del que hace uso Vim.

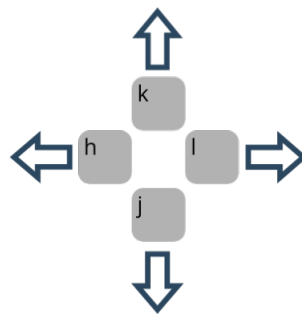


Figura 3: Esquema de navegación básico dentro de Vim

Para resumir, podemos ver que las teclas de navegación son las siguientes:

- `k` y `j` son usadas para subir y bajar de línea, respectivamente.
- `h` y `l` son usadas para desplazarse de izquierda a derecha, respectivamente.

Como se puede ver, el esquema de navegación dista de los editores convencionales. Esto es por diseño, ya que Vim hace uso de estas mecánicas de movimiento para tratar dicho movimiento como si fuera un *comando de consola* y esto permite aplicar ciertos comandos, como el de *borrar* al movimiento para alcanzar cierta velocidad a la hora de borrar contenido dentro de una línea de texto.

## 5.2. Otras opciones de movimiento en Vim

Como se mencionó anteriormente, Vim trata el movimiento como si fuera un comando de consola, esto permite aplicar ciertos efectos al movimiento. Si bien, el movimiento básico se resume con las teclas, **h**, **j**, **k** y **l**, existen también cuatro comandos útiles que deben ser mostrados para que este manual se pueda considerar una referencia útil para el uso de Vim. En general, Vim suele trabajar por *palabras* cuando se habla de navegación dentro de una línea de texto, entonces existen formas para desplazarse en una línea de texto de palabra en palabra. En este caso, existen cuatro comandos destinados para esta función:

- El comando **w** permite al usuario desplazarse desde su localización actual hacia el *inicio* de la siguiente palabra en la línea de texto.
- El comando **e** permite al usuario desplazarse desde su actual ubicación, hacia el *final* de la siguiente palabra en la línea actual de texto.
- Se usa el número 0 para desplazarse hacia el *inicio* de la línea.
- Finalmente, el comando **\$** se usa para desplazarse hacia el *final* de la línea.

## 6. La función de *Borrar* en Vim

Como cualquier editor de texto plano, usualmente queremos editar el contenido de nuestro archivo, ya sea cambiando algunas líneas dentro del archivo o *borrando* contenido dentro de las líneas. En Vim la función de *borrar* dentro del documento está relativamente ligada al uso de movimiento.

Esto es debido a que podemos pensar en lo siguiente: la función de borrar se activa con el comando **d** y la cantidad de contenido a borrar se determina usando los comandos de movimiento vistos anteriormente. Por lo que, para ilustrar algunos de los usos de este mecánica, es conveniente colocar algunos de los comandos más utilizados con esta modalidad.

- El comando **dw** permite borrar todo lo que está desde *la ubicación actual hasta el inicio de la siguiente palabra*. Como ven, el comando de *borrar* se antepone a la opción de movimiento.
- Introduciendo **de** podemos borrar todo aquello que está comprendido desde *la localización del puntero en el editor, hasta el final de la siguiente palabra en la línea*.

Como podemos observar, el comando de *borrar* en Vim usualmente está antes de la opción de movimiento. Sin embargo, debido a razones que los creadores de Vim comentan en **vimtutor** es común que se requiera *borrar una o más líneas completas*.

En este caso, la función de borrar es mucho más sencilla, solamente basta con introducir el comando **dd**. Es en este punto en donde es conveniente introducir una figura que está presente también en Vim y que al día de hoy, no hemos visto en otros editores de texto y que, al día de hoy, no hemos visto en otros editores de texto, esta característica se conoce como *contadores*.

### 6.1. Contadores

Los *contadores* son una característica de Vim que permite al usuario repetir un comando un determinado número de veces, usualmente **vimtutor** relega la lección de esta característica hasta

que ya se han visto los comandos básicos de movimiento y borrado y en este caso, es conveniente explicar un poco el porqué. Supongamos lo siguiente: Se desea borrar un número determinado de líneas en el editor, si se usaran otros editores, se podría pensar en usar el ratón para seleccionar las líneas y borrarlas con la tecla **backspace**.