

[Vim](#) es un editor de ficheros de textos muy versátil, que dispone de una gran flexibilidad a la hora de escribir scripts, modificar ficheros de texto, etc... pero sobretodo, a la hora de programar.

A mi criterio, es el mejor editor que existe, sin embargo, a sus usuarios normalmente le suele pasar como a la Pepsi: o te encanta, o lo odias.



Sus detractores engloban las desventajas de vim en dos, muy comunes:

- Poco «amistoso»: Es cierto. En cuanto a primeras impresiones, Vim tiene las de perder. Muchos usuarios se asustan sin darle ni una oportunidad. A pesar de su muy dura curva de aprendizaje, una vez se aprende sus funciones básicas, la productividad asciende de una manera notable.
- Atajos difíciles: Vim es muy potente, pero carece de menús o botones. Todo se hace a través de comandos y atajos que son secuencias de letras y signos. Esto resulta muy frustrante para muchos usuarios que olvidan o no recuerdan con facilidad. Para ello recomiendo alguna chuleta como [VIMRefCard](#).

Las ventajas, sin embargo, son múltiples. VIM ocupa muy poco y existe en prácticamente todos los Linux o Unix disponible. Al ser un programa que se ejecuta en entorno de texto es útil para accesos remotos y edición vía terminal.

Así pues, vamos a darle un breve repaso a su uso básico desde cero, para perder ese miedo y comprobar lo útil que es aprender a usarlo.

VIM: Introducción

Arrancar el vim es muy sencillo. Sólo hay que escribir en una terminal `vim`, seguido del nombre del fichero a editar. Nos aparecerá una ventana en negro, donde nos aparecerá el contenido del fichero (o en negro si está vacío). En la parte inferior, nos aparecerán los mensajes o comandos que escribamos para manejar el editor, así como la línea en la que estamos, porcentaje del fichero, etc.

Lo primero que hay que aprender de Vim (muy importante) es que tiene varios modos de uso:

- Nada más entrar en vim a editar un fichero, estamos en el modo normal, en el que podremos introducir atajos para realizar operaciones (borrar línea, deshacer, etc...). **IMPORTANTE:** En este modo no podemos escribir en el fichero. Las teclas que pulsemos probablemente estarán asociadas a una operación determinada. Muchos de estos comandos (no todos) comenzarán por `:`.
- Para escribir texto en el fichero tendremos que entrar en el modo edición, que es tan fácil como pulsar la tecla `insert` (o `i`). Sabrás que has entrado en este modo porque abajo aparecerá el texto `-- INSERTAR --`. Ahora todo lo que tecleemos se estará escribiendo en el fichero de texto. Para volver al modo normal sólo hay que pulsar la tecla `ESC` (Escape).

Todo esto puede parecer muy lioso al principio, pero conforme comiences a utilizarlo con frecuencia, verás que resulta cómodo y lo haces de forma automática.

Primeras impresiones

Vim reconoce automáticamente por la extensión del fichero, el lenguaje en el que estamos programando (`.C`, `.sql`, `.pl`, `.latex`, `.php...`),

por lo tanto nos hará un resaltado de sintaxis con colores, que nos resultará bastante agradable.

Esta opción puede no estar disponible en algunos linux con versiones minimalistas de Vim. Sólo tenemos que instalar la versión completa de vim con `apt-get install vim-common` y escribir (en el modo normal del Vim): `syntax on`.

```
70
71 // Sobrecarga de operador de asignacion
72 void Estado::operator = (const Estado &e) {
73     est = e.est;
74     F = e.F;
75     trans = e.trans;
76 }
77
78 // Sobrecarga de operador menor
79 bool Estado::operator < (const Estado &e) const {
80     return (est < e.est);
81 }
82 █
-- INSERTAR --                               82,1                               84%
```

Sin duda, el resaltado de sintaxis es algo muy valioso para el programador.

Operaciones básicas del editor

Una vez tengamos nuestro texto escrito, necesitaremos saber como realizar algunas operaciones como guardar fichero, salir del editor, etc...

Como hemos dicho antes, para realizar operaciones que no son de escribir en el fichero, necesitamos entrar en el modo normal (pulsando ESC si estamos en el modo edición) y a continuación los atajos que queramos:

Secuencia	Significado	¡Mnemotécnica!
:q	Salir del editor sin guardar	quit
:q!	Salir del editor sin guardar ni pedir confirmación	quit ya!
:wq!	Salir del editor guardando sin pedir	write & quit ya!

confirmación

:w f2.txt

Guardar en un fichero llamado **f2.txt** y seguir

write en f2.txt

:e f1.txt

Cierra el fichero actual y abre **f1.txt**

edit f1.txt

Operaciones básicas de texto

En Vim como en cualquier editor, necesitaremos manipular rápidamente texto, y algo que enseguida se echa en falta en VIM, son las famosas opciones Cortar, Copiar y Pegar.

Con los cursores nos desplazamos por el contenido del fichero hasta llegar al inicio de la zona que queremos copiar. Pulsamos ESC (si estamos en el modo edición) y la tecla V para entrar en el modo visual y nos desplazamos hacia el final de la zona que queremos copiar. Se verá que se remarca en otro color la zona seleccionada.

```
1 #include <stdio.h>
2 #include <iostream>
3
4 using namespace std;
5
6 main () {
7     cout << "Hola Mundo" << endl;
8 }
9
10
~
~
~
-- VISUAL --                8,1                Todo
```

Una vez tengamos la zona a copiar seleccionada, sólo tenemos que pulsar C(para cortar) o Y (para copiar). Nos aparecerá abajo un mensaje X líneas copiadas.

Ahora sólo tenemos que desplazarnos a donde queramos pegar ese fragmento y pulsar (como siempre, en el modo normal, no en el modo edición) la tecla P(pegar).

Existen otras formas de copiar (por método de buffer, por ejemplo) con la secuencia "buffernyy para copiar y "bufferp para pegar, pero suelen ser más complejas.

Veamos más operaciones de texto:

Secuencia	Significado	iMnemotécnica!
dd	Suprimir línea actual al buffer (<i>p para pegar</i>)	delete
u	Deshacer el último cambio en el fichero	undo
CTRL+R	Rehacer el último cambio en el fichero	redo
guu	Convertir a minúsculas la línea actual	lowercase
gUU	Convertir a mayúsculas la línea actual	UPPERCASE
:num	Posicionarse en la línea <i>num</i> del fichero	
gg	Posicionarse al principio del fichero	
G	Posicionarse al final del fichero	
ga	Muestra código ASCII, hex y octal del caracter actual	

Operaciones de búsqueda y sustitución

Otra función que solemos echar de menos enseguida es la de buscar algún texto, reemplazar, etc. En vim no puede faltar esa opción, con sus respectivas mejoras y añadidos:

Para buscar un texto, escribimos (en modo normal, pulsando antes ESC si estamos en modo edición) la secuencia /palabra.

Veremos que se resalta la palabra encontrada (o nos avisa de que no existe). Entonces podemos seguir buscando la próxima coincidencia pulsando n o buscarla hacia detrás pulsando N.

Para sustituir un texto debemos escribir la secuencia :

%s/texto1/texto2/g, donde texto1 es el texto a buscar y texto2 el texto que será reemplazado. Si incluimos lag final (global), sustituirá todas las coincidencias que encuentre, sino sólo la primera que encuentre.

Preferencias

Hay ciertas opciones que denominamos preferencias porque son detalles que podemos fijar permanentemente (o no) como por ejemplo que queremos mostrar el numerado de línea (como en los ejemplos de imagen), el resaltado de sintaxis, etc...

Ello lo podemos hacer mediante comandos desde el modo normal o en el fichero `~/.vimrc` (sin el símbolo `:`), donde se guardan las preferencias del usuario.

Comando	Significado
<code>:set ts=3</code>	Fija los tabulados a 3 espacios
<code>:set sw=3</code>	Fija los indentados a 3 espacios
<code>:set number/nonumber</code>	Activa/desactiva el numerado en los ficheros
<code>:set backup/nobackup</code>	Activa/desactiva la copia de seguridad automática
<code>:set directory=<i>dir</i></code>	Fija la carpeta donde se harán las copias
<code>:syntax on/off</code>	Activa/desactiva el resaltado de sintaxis
<code>:color <i>esquema</i></code>	Cambia color del vim (<i>evening, darkblue, desert, elflord, koehler, morning...</i>)
<code>:set cindent</code>	Activa indentado automático
<code>:set mouse=a/mouse=</code>	Activa/desactiva el uso del ratón
<code>:set paste/nopaste</code>	Activa/desactiva el modo pegar texto literalmente
<code>:spell</code>	Activa el corrector ortográfico
<code>:setlocal spell spelllang=es</code>	Activa el idioma español del corrector ortográfico
<code>:set spellfile=~/.vimdic</code>	Fija diccionario de palabras desconocidas

Ejecución y comandos externos

Es posible que mientras estamos editando un fichero, necesitemos ejecutar un comando (por ejemplo, un `ls` para ver los archivos), esto

se puede hacer escribiendo en el modo normal `:!comando`, en nuestro ejemplo: `:!ls`.

También se puede hacer una pausa en la edición escribiendo `:shell` para realizar alguna operación y cuando la terminemos, escribir `exit` y volver al editor. Incluso los comandos `:make` y `:cc` se pueden utilizar para ahorrarnos el estar saliendo del editor para hacer makes o ver el último error que nos dió.

Características avanzadas

En Vim podemos realizar una separación visual de forma que podamos editar dos (o más) ficheros simultaneamente. Para ello escribiremos `:split fichero.ext` con lo que dividirá la pantalla mostrando los dos ficheros para edición.

```
8 char a = 'a';
9 string str;
10
11 str[0] = a;
12
13 cout << str << endl;
C.C 13,1 87
4 using namespace std;
5
6 int main () {
7     cout << "Hola Mundo" << endl;
8 }
hola.C 5,0-1 Final
-- INSERTAR --
```

Es posible realizar un `:vsplit fichero.ext` para realizar una división vertical en lugar de horizontal, e incluso realizar más divisiones posteriormente, alternando para conseguir más divisiones, siempre y cuando nuestra resolución nos permita.

Para cambiar entre subdivisiones, pulsamos `CTRL+W` y presionamos la flecha de los cursores a donde queramos dirigirnos.

También es posible crear pestañas (tal como lo hace Firefox) para tener varios ficheros. Para ello escribimos `:tabnew` seguido del nombre del fichero a crear. Para movernos entre pestañas escribimos `:tabn`, pulsando `gt` o utilizando el ratón (activando el previo `set mouse=a`) para usar los iconos de texto superiores.

Como se puede ver, Vim tampoco carece de un sistema de autocompletado, por si no recordamos los nombres de nuestras variables o de las funciones que queremos utilizar. Basta con escribir el caracter inicial y pulsar CTRL+N. Nos aparecerá un panel de sugerencias coincidentes para elegir.

```
2 #i str
3 #i string
4   strfmon      /usr/include/sys/cdefs.h
5 us struct      /usr/include/bits/types.h
6   streams      /usr/include/stdio.h
7 in structure    /usr/include/xlocale.h
8   strings      /usr/include/wchar.h
9   stream       /usr/include/wchar.h
10  streambuf     /usr/include/libio.h
11  str
12
13   cout << str << endl;
14 }
```

-- Completar palabra clave (^N^P). coincidencia 5 de 19.

Otras operaciones avanzadas pueden ser:

Comando	Significado
=G	Indenta automáticamente todas las líneas de un fichero
{}	Detecta donde está la llave mal cerrada del párrafo actual
:g/^\s*\$\d	Elimina las líneas en blanco de un fichero

Existen muchísimas operaciones útiles que no están en este manual, todo es cuestión de ir indagando un poco.

Para aquellos que les haya picado la curiosidad pero no se atreven a utilizar aplicaciones que no tienen entorno gráfico, Vim tiene un hermano gemelo, llamado [gVim](#) (graphic vim), una aplicación gráfica para windows, que ayudará a aquellos que quieran iniciarse con este magnífico editor.