Introducción a Vim

Jesús Espino García

Kaleidos

1 de Agosto de 2011

Introducción.

¿Que es Vim?

- Vim ("VI IMproved").
- Es un editor de textos.
- Desarrollado por Bram Moolenaar.
- Basado en el vi.
- Licencia GPL.

Ventajas

- Es Open Source.
- Es muy configurable.
- Tiene resaltado para el código.
- Es charity-ware.
- No ocupa demasiado espacio.
- Uso muy similar al vi.
- Permite muchos cambios con pocas combinaciones de teclas.
- Cómodo de usar una ves se conoce.
- Existen versiones para muchos sistemas operativos.

Historia

- 1988 - Vim 1.0 - Vi IMitation para Amiga.
- 1991 - Vim 1.14 - Primera versión pública.
- 1992 - Vim 1.22 - Portado a Unix y renombrado a Vi IMproved.
- 1994 - Vim 3.0 - Múltiples ventanas.
- 1996 - Vim 4.0 - Interfaz gráfico.
- 1998 - Vim 5.0 - Resaltado de sintaxis.
- 2000 - Vim 6.0 - Plegado y multilingue.
- 2000 - Vim 7.0 - Pestañas, corrector ortográfico, omnicompletion.

Conceptos Básicos.

Modos

Existen 3 modos:

- Comandos: Es el básico, a partir de aquí se ejecuta todo.
- Insertar: Es el modo que nos permite insertar texto.
- Visual: Este modo nos permite hacer selecciones de texto (por caracter, linea o bloques) y realizar operaciones sobre el.

Buffers

- Un contexto de edición.

Opciones

- Vim tiene muchas opciones.
- set/unset: gestión de las opciones de vim.
- `:set all`: Muestra todas las configuraciones.
- `:set nu/nonu`: Muestra el numero de linea.
- `:set hlsearch/nohlsearch`: Añade o quita remarcado a las búsquedas.
- `:set filetype=<tipo>`: Especifica el tipo de fichero que se esta editando.
- `:set textwidth=n`: Especifica el ancho del documento.

Ayuda

- :help ¡sección o comando¿
- :help

Comandos Básicos.

Existen muchísimos comandos, pero los mas básicos serian los siguientes:

- Movimientos
- Edición
- Búsquedas
- Otros

Movimientos

- `h j k l`: Mueve el cursor un caracter.
- `w b e`: Mueve el cursor una palabra.
- `0 | <n>| ^ $ - +`: Mueve el cursor por lineas.
- `% ( ) { } [[ ]]`: Mueve el cursor por bloques.
- `H <n>H L <n>L`: Mueve el cursor por la pantalla.
- `^B ^U ^D ^F ^Y ^E`: Mueve la pantalla.
- `1G <n>G G`: Movimiento del cursor por todo el fichero.

Edición

- `I i a A o O r R`: Entra en modo inserción.
- `X nX x nx dd ndd`: Borra caracteres
- `Y yy yny`: Copia.
- `p P`: Pega.

Búsquedas

- `/<cadena>`: Busca la cadena hacia adelante.
- `?<cadena>`: Busca la cadena hacia detrás.
- `n`: Repite la ultima búsqueda.
- `N`: Repite la búsqueda invirtiendo el sentido.

Otros

- `ZZ`: Sale y guarda el fichero.
- `:wq`: Sale y guarda el fichero.
- `:x`: Sale y guarda el fichero.
- `:view <file>`: Abre un fichero para solo lectura.
- `:w`: Escribe el fichero.
- `:q!`: Sale del programa.
- `u`: Deshace la última operación.
- `^r`: Rehace el último deshacer.
- `J`: Une dos lineas.
- `.`: Repite la última operación realizada.

Otras funcionalidades.

Otros comandos

- Sustituciones
- Abreviaturas
- Macros
- Marcas
- Plegado
- Grabación de comandos
- Comandos del sistema
- Configuraciones
- Combinaciones
- Ventanas
- Pestañas
- Corrección ortográfica
- Plugins

Sustituciones

- `:%s/<regexp>/<cadena>/`: Sustituye la primera vez que encuentre la expresión regular en una linea, por la cadena.
- `:%s/<regexp>/<cadena>/g`: Sustituye todas las veces que encuentre la expresión regular por la cadena.
- `:<rango>s/<regexp>/<cadena>/`: Sustituye todas las veces que encuentre la expresión regular dentro del rango, por la cadena.

Abreviaturas

Son practicas para escritura rápida y para predefinirlas en un fichero dependiendo el documento que vayas a editar.

- `:abbreviate <cadena1> <cadena2>`: Abrevia la cadena 2 como la cadena 1.
- `:abbreviate`: Muestra las abreviaturas actuales.

Macros

Las macros son secuencias de comandos que se asignan a un caracter.

- `:map C <cadena>`: define un macro.
- `C`: ejecuta el macro definido en C.

ejemplo: `:map <F5> icadena de prueba <ESC>`

Marcas

Las marcas se utilizan para memorizar posiciones dentro del fichero.

- `mC`: Marca una posición con el caracter C.
- `'C`: Va a la posición marcada con C.

Las marcas se pueden usar también en el rango.

Plegado

El plegado se utiliza para ocultar parte del texto del archivo que no queremos ver.

- `:<rango> fold`: Pliega el rango.
- `:<rango> folddoopen <comando>`: Ejecuta el comando sobre todo lo que no esté dentro de un pliegue cerrado.
- `:<rango> folddoclosed <comando>`: Ejecuta el comando sobre todo lo que esté dentro de un pliegue cerrado.
- `:<rango> foldopen[!]`: Abre los pliegues en el rango.
- `:<rango> foldclosed[!]`: Cierra los pliegues en el rango.
- Para `foldopen` y `foldclose` el `!` sirve para abrir también pliegues anidados.

Plegado II

Algunos comandos útiles para el plegado.

- `zd`: Borra el pliegue sobre el que se encuentre.
- `zD`: Borra el pliegue sobre el que se encuentre y todos los anidados.
- `zE`: Borra todos los pliegues del archivo.
- `zo`: Abre el pliegue sobre el que se encuentre.
- `zO`: Abre el pliegue sobre el que se encuentre y todos los anidados.
- `zc`: Cierra el pliegue sobre el que se encuentre.
- `zC`: Cierra el pliegue sobre el que se encuentre y todos los anidados.
- `za`: Cambia de estado (cerrado/abierto) del pliegue sobre el que se encuentre.
- `zA`: Cambia de estado (cerrado/abierto) del pliegue sobre el que se encuentre y todos los anidados.
- `zj`: Mueve el cursor hasta un pliegue más abajo.
- `zk`: Mueve el cursor hasta un pliegue más arriba.

Grabación de comandos

La grabación de comandos nos permite grabar una secuencia de comandos que ejecutamos para repetirla posteriormente con una sencilla combinación de teclas.

- `qC`: Empieza a grabar una sucesión de comandos que ejecutes.
- `q`: Termina de grabar.
- `@C`: Ejecuta de nuevo la misma sucesión de comandos.

Comandos del sistema

- `:<rango>!<comando>`: Ejecuta el comando con entrada estándar el rango que se le ha especificado y lo sustituye por la salida del comando.

Combinaciones

Existen algunas combinaciones simples que se usan con cierta frecuencia.

- `xp`: Cambia de orden dos caracteres.
- `ddp`: Cambia de orden dos lineas.
- `gqq`: Reformatéa una linea a 80 caracteres.
- `gq}`: Reformatéa un párrafo a 80 caracteres.

Ventanas

- `:split`: Abre una nueva ventana.
- `:split <file>`: Abre el fichero en una nueva ventana.
- `:n split`: Abre una ventana de tamaño n.
- `:sview`: Combinación entre :split y :view.
- `^W <w|j|k|-|+|=>`: Moverse por las ventanas.

Pestañas

- `:tabnew [<file>]`: Abre una nueva pestaña.
- `:tabclose`: Cierra la pestaña.
- `gt gT <n>gt`: Cambia de pestaña.

Corrección ortográfica

- `]s [s`: Próxima y anterior palabra mal.
- `zg zw`: Añade y elimina palabras del diccionario.
- `zG zW`: Añade y elimina palabras a un diccionario temporal.
- `z=`: Sugiere palabras.

Plugins

- vim-addon-manager
- http://www.vim.org

Algunos Plugins interesantes

- pythoncomplete
- project
- supertab
- snippetsEmu

Para programadores.

Para programadores

- Configuraciones.
- Make.
- Ctags.

Configuraciones

- `:syntax on/off`: Activa o desactiva el reconocimiento de sintaxis.
- `:set autoindent/noautoindent`: Activa o desactiva el auto indentado.
- `:set cindent/smartindent/autoindent`: Selecciona el tipo de auto indentado que se usa.

Make I

- `:make <argumentos>`: Ejecuta el make con los argumentos que se le pasen.
- `:cnext`: Mueve el cursor a la posición del error siguiente (linea y fichero).
- `:cprevious`: Mueve el cursor a la posición del error anterior (linea y fichero).
- `:clast`: Mueve al ultimo error.
- `:crewind`: Mueve al primer error.
- `:cnfile`: Mueve al primer error del próximo fichero.

Make II

- `:cc`: Muestra el error actual.
- `:clist`: Muestra la lista de errores.
- `:clist <rango>`: muestra el rango de errores pedido.
- `:clist!`: muestra también los mensajes de informativos del compilador.

ctags

búsqueda de definición de las funciones con ctags.

- `^]`: va a la definición de la función.
- `^t`: vuelve a la posición donde estaba.

Para terminar.

¿Qué se queda en el tintero?

- Scripting en vim.
- Scripting en vim con python.

Referencias

- http://www.vim.org - Web oficial de vim.
- http://www.truth.sk/vim/vimbook-OPL.pdf - Libro de vim.

Dudas

. . .