Introducción a Emacs

Carlos Linares López

Octubre, 2006

Resumen

Este capítulo describe el funcionamiento básico del editor de textos **Emacs**.

Actualmente se distribuye la versión 21.4, que ya es estable. Todos los comandos que se describen en las secciones que siguen, funcionarán en esta versión y se espera que en las futuras, puesto que GNU se esfuerza constantemente en mantener —dentro de límites razonables—, la compatibilidad con las versiones anteriores.

1. Introducción

1.1. ¿Qué és?

GNU **Emacs** es un editor avanzado, auto documentado, personalizable y ampliable. Con GNU **Emacs**, es posible editar varios ficheros simultáneamente, abrir varias ventanas del mismo documento, definir macros de teclado, deshacer cambios y mucho más. Además, tabula automáticamente programas en lenguajes como C, Lisp, Pascal, y otros muchos. También ofrece facilidades para leer y escribir correo electrónico y crea copias de seguridad.

El núcleo de **Emacs** está escrito en C e incluye un intérprete de Lisp. Además, como la mayoría de los comandos de edición están escritos en Lisp, es posible crear nuevos comandos en medio de una sesión de trabajo para ser utilizados inmediatamente.

1.2. ¿Para qué sirve?

A diferencia de cualquier editor de textos en cualquier sistema operativo, **Emacs** sirve para mucho más que editar ficheros.

Emacs intentará deducir el tipo de fichero a partir de su extensión o de una indicación explícita que se coloca en la primera línea. Si la deducción no fue correcta, se puede indicar desde la misma línea de comandos. Gracias a este comportamiento, **Emacs** puede tabular automáticamente el código fuente que se teclea cada vez que se pulsa RET (retorno) e incluso ofrecer comandos adicionales para ajustarse a las necesidades de la edición que se está llevando a cabo.

Así, por ejemplo, es posible utilizar el mismo entorno para programar en C, en HTML, escribir un artículo (con LaTeX), o, de hecho, este documento.

Además, **Emacs** es GNU[1] y muchos voluntarios en el mundo se esfuerzan en ampliarlo. Gracias a ellos, es posible depurar programas desde **Emacs**, compilarlos, realizar un control de versiones, mandar correo electrónico, hacer telnet, o acceder a Internet a través de un browser denominado W3.

Evidentemente, la funcionalidad de **Emacs** es tan amplia que resultaría inutil intentar explicar exhaustivamente todas sus posibilidades en pocas páginas, por ello, se recomienda leer el manual original de **Emacs**, así como consultar, cada vez que se necesite, la ayuda en línea que se ofrece desde `www.gnu.org`

Sin embargo, estas páginas deberían ser una descripción suficiente para el manejo de **Emacs** y se espera que el lector, en poco tiempo, aprenda a utilizarlo. Sin embargo, sólo se puede aprender practicando, por ello, ensaye todo lo que aquí aprenda.

1.3. ¿Cómo funciona?

Tal y como ya se ha dicho, **Emacs** es un editor de textos puesto que permite cargar ficheros y modificarlos después, para ser guardados de nuevo.

Está "**auto-documentado**" puesto que existen comandos para ofrecer información a propósito de cualquier cuestión.

Es **adaptable** y **extensible** en la medida en que es muy fácil escribir código Lisp para implementar nuevas funcionalidades o modificar las que ya existen. Además, dispone de comandos que pueden introducirse en la misma sesión y que sirven para especificar la asociación entre determinadas combinaciones de teclas y los comandos de edición.

Antes de entrar en la descripción de **Emacs** se presenta, brevemente, la nomenclatura que se seguirá para la descripción de las teclas y sus combinaciones.

Emacs utiliza, prácticamente, todas las combinaciones del teclado, desde los caracteres alfanuméricos y numéricos, hasta los códigos especiales: `Control`, `Meta`, `Retorno` y `Tabulador`, así como la barra espaciadora que puede tener diferentes propósitos. Para facilitar la brevedad en la exposición, cada una de estas teclas se indica con una sola letra o con un nombre corto en vez de escribirlo entero. Dichas abreviaturas se muestran en la tabla 1.

El resto de las teclas se indican con su nombre.

La tecla `Meta` se identifica, en las consolas de Sun con un rombo (♢) y, en los teclados convencionales, como el de un PC, es la tecla Alt. Por otra parte, si se desea combinar la tecla `Meta` con la tecla de mayúsculas, tal vez sea necesario utilizar Escape (Esc) antes de pulsar mayúsculas.

Las combinaciones de teclas se indican separando con guiones cada una de ellas. Así, por ejemplo, M-> indica que se deben pulsar simultáneamente las teclas `Meta` y `>`.

[1] GNU significa *GNU's not Unix.*

`Control`	C
`Meta`	M
`Retorno`	RET
`Tabulador`	TAB
`Espacio`	SPC
`Suprimir`	DEL

Cuadro 1: Abreviaturas de teclado

Además, en ocasiones será necesario pulsar diferentes combinaciones para lograr un propósito determinado. En tal caso, cada una de las combinaciones se separan de un espacio en blanco. Por ejemplo, C-u 1 8 C-x v representa la pulsación simultánea de las teclas `Control` y `u` primero. Luego, el `1`, el `8`, `Control` y `x` al mismo tiempo y finalmente `v`.

Por último, porque con frecuencia la configuración de los teclados en máquinas diferentes puede deparar grandes sorpresas, todos los comandos incluyen el nombre de la función Lisp que invocan. Por ello, si no es posible reproducir una secuencia concreta de teclas, puede ejecutarse el comando con M-x comando.

En las siguientes secciones se presentan las maneras y modos de realizar acciones en **Emacs** que son comunes en muchos editores de texto. Cuanto más avance, más aprenderá y descubrirá que se pueden hacer cosas con **Emacs** que resultan impensables en la gran mayoría —por no decir la absoluta totalidad— de los editores.

2. La pantalla

Emacs organiza la pantalla en diferentes áreas que se denominan ventanas —independientemente de que esté en un entorno gráfico o de texto. La ventana más grande es aquella en la que se realizará la edición de ficheros. El elemento más importante de una ventana es el punto.

Además, deben distinguirse otras dos zonas que se denominan **línea de estado** y **minibuffer**.

2.1. El punto

Durante una sesión de edición con **Emacs**, el cursor muestra aquella posición del buffer donde se aplicará el siguiente comando. A dicha posición se la denomina punto.

Aunque no es posible presentar varios cursores en la pantalla, en caso de que tenga varios buffers abiertos, cada fichero tiene su propio punto. Cuando el cursor pasa de un buffer a otro, se posiciona allí donde se quedó el punto. Evidentemente, el buffer en uso será aquél en el que esté activo el cursor.

La tabla 2 muestra los comandos para mover el cursor (o punto).

Dirección	Descripción	Comando Lisp	Teclas
Hacia delante	Avanzar un carácter	FORWARD-CHAR	C-f
	Avanzar una palabra	FORWARD-WORD	M-f
	Avanzar una línea	NEXT-LINE	C-n
	Ir a fín de línea	END-OF-LINE	C-e
	Ir a fín de párrafo	FORWARD-PARAGRAPH	M-}
	Avanzar una página	SCROLL-UP	C-v
	Ir a final del documento	END-OF-BUFFER	M->
Hacia detrás	Retroceder un carácter	BACKWARD-CHAR	C-b
	Retroceder una palabra	BACKWARD-WORD	M-b
	Retroceder una línea	PREVIOUS-LINE	C-p
	Ir a principio de línea	BEGINNING-OF-LINE	C-a
	Ir a inicio de párrafo	BACKWARD-PARAGRAPH	M-{
	Retroceder una página	SCROLL-DOWN	M-v
	Ir a inicio del documento	BEGINNING-OF-BUFFER	M->

Cuadro 2: Comandos para mover el punto

Además, como es lógico, también se puede avanzar o retroceder una posición con las teclas del cursor → y ← respectivamente. Análogamente, se puede avanzar o retroceder una línea con ↓ y ↑. Por último, también se puede avanzar o retroceder una página con las teclas de avance y retroceso de página.

2.2. Línea de estado

La línea de estado es la penúltima fila del editor y, probablemente, está resaltada en video inverso. Dicha línea consiste en los siguientes caracteres:

`-- ch - Emacs: buf (mayor menor) ---- pos ------`

donde:

- *ch* puede ser:

  `**` Si se ha modificado el texto desde que se recuperó el fichero.

  `--` Si no se ha modificado el texto actual o no se ha editado ningún fichero.

  `%%` Si el fichero es de solo lectura.

- *buf* es el nombre del *buffer* que está siendo actualmente editado. Los buffers se presentan en la sección 3.6.

- *mayor* es el modo mayor utilizado actualmente. Los modos mayores se explican en la sección 3.11.

- *menor* es una lista de los modos menores elegidos para la edición del fichero actual.

- *pos* indica la posición del cursor en el fichero. Si el cursor está al principio, será **Top**; si está al final, **Bot**; si se está viendo el fichero entero, **All**; y, en cualquier otro caso, el tanto por ciento del fichero que está por encima del cursor.

2.3. Minibuffer

El minibuffer es la última línea de la ventana. Esta línea está dedicada a:

- Realizar un eco de los comandos tecleados. Por ejemplo, al pulsar `C-x 4 C-o` para visulizar un buffer en otra ventana, el minibuffer irá mostrando toda la combinación.
- Introducir argumentos para la ejecución de los comandos que los requieran. Por ejemplo, al ejecutar `M-x query-replace`, para buscar y sustituir algunas cadenas del buffer actual, **Emacs** preguntará qué cadena hay que buscar y por cual se reemplazará. Según se teclean los argumentos, se van mostrando en el minibuffer.
- Para ofrecer información sobre el estado de ejecución de algún comando. Así, por ejemplo, al pulsar `C-g` en cualquier momento, **Emacs** escribirá en el minibuffer *Quit* indicando que, sea cual fuere la tarea que se estaba realizando, ha sido abortada.

3. Comandos de edición básicos

Esta es sin ningún lugar a dudas, la sección más importante de este capítulo.

La mayor parte de los comandos que se describen a continuación están presentes en la tarjeta de referencia rápida. Para una mayor comprensión de sus funciones y diferencias, se han organizado temáticamente, tal y como se muestra a continuación.

3.1. Ejecutar Emacs

En una sesión Unix, basta con hacer:

% emacs

donde "%" es el prompt de la shell (en csh o tcsh, éste es el prompt normal, pero en bash o sh, podrá ser "$").

Cuando **Emacs** se ejecuta, pueden cargarse en memoria librerías, funciones, combinaciones de teclas, ... Esto es muy útil para: primero, personalizar el entorno; segundo, para instalar paquetes como, por ejemplo, el editor de páginas HTML, html-helper-mode. Para disponer de toda esta funcionalidad, es necesario tener algunos conocimientos de Lisp a fin de poder modificar el fichero de inicialización ".emacs". Sin embargo,

la descripción y utilidad de este fichero va mucho más allá de los objetivos de este documento y por ello no se presentará su utilización.

Una vez que **Emacs** ha arrancado, la pantalla completa presenta un único buffer denominado ***scratch***. Más adelante se explicará como trabajar con él. Por ahora, solo es necesario saber que éste es el buffer genérico dedicado a trabajos que se realizan por primera vez.

3.2. Abandonar Emacs

En cualquiera de los sistemas operativos Unix existen dos maneras diferentes de salir de **Emacs**:

- Suspender la ejecución de **Emacs** (SUSPEND-EMACS). `C-z`

  Produce una salida al sistema operativo, pero con **Emacs** ejecutando en *background*.

- Salir de **Emacs** (SAVE-BUFFERS-KILL-EMACS). `C-x C-c`

  Si se hace, **Emacs** es desalojado de memoria y se retorna al sistema operativo. Si no se modificaron algunos buffers, se pedirá confirmación para salvarlos y, si no se salvan, para salir.

3.3. Ficheros

Emacs puede, como cualquier otro editor de textos, leer ficheros, salvarlos, copiar los contenidos de uno en otro, etc. Los comandos que se refieren al tratamiento de ficheros se muestran en la tabla 3.

3.4. Recuperación de errores

Uno de los comandos más socorridos de **Emacs** es, sin ningún lugar a dudas, `C-g` (KEYBOARD-QUIT). Esta combinación aborta el comando en ejecución.

Cuando se desea deshacer el efecto de un comando no deseado (bien porque no se tecleó correctamente la combinación pensada, bien porque el resultado de una función invocada no ha sido el esperado), `C-x u` o, equivalentemente, `C-_` (ADVERTISED-UNDO), deshacen dichos cambios.

Por último, en algunas ocasiones puede suceder que la pantalla se llene de *basura* o, en cualquier caso, de caracteres que emborronan las ventanas de **Emacs**. En tal caso `C-l` (RECENTER) redibuja la pantalla y actualiza todos sus contenidos

Función	Comando Lisp	Teclas	Descripción
Leer un fichero	FIND-FILE	`C-x C-f`	Lee el fichero que se indique como argumento y lo carga en el buffer activo.
Salvar un fichero	SAVE-BUFFER	`C-x C-s`	Graba el fichero cargado en el buffer activo. Si el buffer tiene nombre, se graba con ese mismo nombre, en otro caso —por ejemplo, si se trata del buffer ***scratch***—, **Emacs** pedirá un nombre de fichero.
Insertar un fichero	INSERT-FILE	`C-x C-i`	Lee el fichero que se indique como argumento —en el minibuffer— y lo inserta en la posición actual del punto.
Reemplazar el fichero	FIND-ALTERNATE-FILE	`C-x C-v`	Reemplaza el fichero editado en el buffer actual, con aquél que se entregue como parámetro a través del minibuffer.
Escribir en otro fichero	WRITE-FILE	`C-x C-w`	Escribe los contenidos del buffer actual en un fichero distinto del que está asignado al buffer. El nombre del fichero donde se realiza la escritura se indica a través del minibuffer.

Cuadro 3: Comandos para el tratamiento de ficheros

3.5. Marcar

Emacs mantiene en memoria un "anillo de marcas", esto es, una lista de posiciones que apuntan a diferentes lugares de un mismo buffer. De esta manera, es posible ir de un lugar a otro muy rápidamente o, por ejemplo, marcar determinadas posiciones que, con algún interés específico, se desean recordar.

Para poder acceder a este tipo de facilidades —desconocidas en la gran mayoría de los editores—, es necesario, en primer lugar, aprender a marcar. Una marca se establece allí donde se pulsa `C-SPC` o, también `C-@` (SET-MARK-COMMAND).

Como ya se indicó, el punto es la posición ocoupada por el cursor. Por lo tanto, se denomina *región* al área que hay entre la marca y el punto (en cualquier orden). El concepto de región es, como se verá, vital en **Emacs**.

Con `C-x C-x` (EXCHANGE-POINT-AND-MARK), la última marca grabada y el punto intercambian sus posiciones. Este comando es muy útil cuando, por ejemplo, se desea estudiar otra parte del fichero, distinta de la que se está editando, antes de seguir. Para hacerlo, se marca el punto actual; a continuación se accede a esa otra posición de interés y, para volver, basta con hacer `C-x C-x`: **Emacs** recuerda donde se había quedado la última marca y lleva el punto hasta allí, esto es, el cursor.

Para utilizar el "anillo de marcas" con todas sus posibilidades, se dispone de `C-u C-SPC` o, equivalentemente, `C-u C-@`. Dicho comando va recorriendo todo el anillo posicionando el cursor en cada una de las marcas almacenadas.

3.6. Buffers

Hasta ahora, buffer era un término que se asociaba, sin pérdida de generalidad, tanto a "fichero" como a "ventana". Esto es así en la medida en que cada fichero que se edita requiere una ventana nueva y cada ventana puede entenderse como un buffer. Sin embargo, si se abre varias veces el mismo fichero, esta asociación puede resultar confusa. Así mismo, esta equivalencia resulta engañosa si se modifican varios ficheros sin dividir la pantalla.

Las diferencias entre los términos, aunque sutiles, son significativas:

- **Ventana**. Se trata de aquella posición de la pantalla que se dedica a la edición de un fichero. Por ejemplo, gracias al comando `C-x 2`, la pantalla (esto es, todo el espacio dedicado a **Emacs**) se divide en dos ventanas. Ventana es un concepto, del todo, físico.
- **Fichero**. Puede entenderse que un fichero es aquella parte de información guardada en un dispositivo de almacenamiento, como un disco duro. En **Emacs**, fichero tiene el mismo significado que para el sistema operativo.
- **Buffer**. Se dice que el texto que está siendo editado reside en un objeto denominado buffer. Se trata de un concepto lógico.

Para que se comprendan mejor los términos presentados, supóngase que se desean editar varios ficheros que se encuentran en el mismo directorio. Por ejemplo: gnu.texi, makefile, gnu.ps y gnu.info. Si para ello se sigue la secuencia de comandos (sin las comas, por supuesto): `C-x C-f gnu.texi`, `C-x C-f makefile`, `C-x 2`, `C-x C-f gnu.ps`, `C-x 2`, `C-x C-f gnu.info` y, supuesto que no se ha hecho ninguna otra modificación, se tiene que:

- Cuando se `C-x 2` por primera vez, se tienen dos ventanas, cada una de las cuales ocupa la mitad de la pantalla. Además, ya se han cargado dos ficheros: gnu.texi y makefile. Sin embargo, hay tres buffers: gnu.texi, makefile y *scratch*.
- A continuación se carga el fichero gnu.ps. Al hacerlo, se tienen tres ficheros —los dos anteriores y el nuevo— y cuatro buffers.
- Finalmente, se vuelve a dividir una de las ventanas y se carga el fichero gnu.info. Con lo que se tiene un fichero más, así como un nuevo buffer y una nueva ventana. En total, cuatro ficheros, cinco buffers y tres ventanas.

Para conocer el nombre de cada uno de los buffers que se están editando simultáneamente, se dispone del comando `C-x C-b` (LIST-BUFFERS). Este comando abre otra ventana, con un nuevo buffer, donde presenta los nombres de todos los buffers en líneas separadas, así como las características de cada uno de los ficheros que se están editando, por ejemplo:

```
 MR Buffer           Size  Mode         File
 -- ------           ----  ----         ----
.   emacs.tex       46076  LaTeX        ~/latex/gnu/emacs.tex
    gnu.tex          4150  LaTeX        ~/latex/gnu/gnu.tex
    portada.tex      2028  LaTeX        ~/latex/gnu/portada.tex
    *scratch*           0  Lisp Interaction
*   *Messages*       2236  Fundamental
 %  *Completions*     455  Completion List
```

El orden y significado de cada uno de los campos que se muestran es el siguiente:

MR puede valer:

- **'*'** Buffer modificado.
- **'%'** Buffer de solo lectura.
- **'.'** Buffer actual.

Buffer Nombre del buffer. Si se editan ficheros, el nombre del buffer y el del fichero coincidirán. Si se están realizando otras tareas, como por ejemplo, pedir ayuda o acceder al sistema de información, el buffer tendrá un nombre específico (en el ejemplo, `*scratch*`, `*Messages*` y `*Completions*`).

Dirección	Descripción	Comando Lisp	Teclas
Hacia delante	Carácter	DELETE-CHAR	C-d
	Hasta el fin de palabra	KILL-WORD	M-d
	Hasta el fin de línea	KILL-LINE	C-k
	Hasta el fin de párrafo	KILL-SENTENCE	ESC k
Hacia detrás	Carácter	DELETE-BACKWARD-CHAR	DEL
	Hasta el fin de palabra	BACKWARD-KILL-WORD	M-DEL
	Hasta el fin de línea	KILL-LINE	M-0 C-k
	Hasta el fin de párrafo	BACKWARD-KILL-SENTENCE	C-k DEL

Cuadro 4: Comandos para cortar texto

Size Tamaño del buffer.

Mode Modo mayor de edición. Consúltese la sección 3.11.

File Si lo hubiera, fichero asociado con el buffer indicado en la misma línea.

Para poder ir de un buffer a otro, se utiliza `C-x b` (SWITCH-TO-BUFFER). El comando es efectivo introduciendo en el minibuffer el nombre del buffer que se desea visitar. Para hacer las cosas más cómodas, **Emacs** recuerda el último buffer visitado antes que éste y presenta, por defecto, ese nombre en paréntesis. Para acpetar la opción por defecto basta con pulsar `RET`.

3.7. Copiar y cortar

Las funciones de copiar y cortar son indispensables en cualquier editor. En **Emacs**, estas funciones son, con mucha diferencia, mucho más ricas.

3.7.1. Cortar

Los comandos para cortar "pedazos" de texto se muestran en la tabla 4. Además de los que se presentan ahí, una región entera se corta con `C-w` (COMPLETION-KILL-REGION), es decir, borra la zona comprendida entre el punto y la última marca.

Hay además, otras modalidades: de expresiones, sentencias completas, etc. Sin embargo, su explicación va más allá de la intención de este pequeño manual.

Emacs mantiene en memoria una zona denominada "anillo de borrado" donde se almacenan hasta un número prefijado de elementos (típicamente 16). Por ello, si se hace `C-w` y se repite el comando más tarde, ambas componentes borradas se almacenan en memoria y no ocurre, como en la mayoría de los editores, donde el segundo elemento reemplazaría al primero.

3.7.2. Copiar y pegar

La función "pegar" consiste en insertar en un punto concreto de cualquier buffer un texto que fue previamente borrado. Por lo tanto, primero será necesario ejecutar alguno de los comandos presentados en la sección anterior, para poder acceder a éstos.

Para pegar se utiliza la combinación `C-y` (YANK), con la que se recupera la última componente borrada (palabra, línea, párrafo, ...) y se pega en la posición indicada por el punto. Esto es, se recupera la primera posición del "anillo de borrado" y se inserta.

La función de "copiar" se distingue de la de "pegar" en que no es necesario eliminar una parte de texto antes. En su lugar, `M-w` (KILL-RING-SAVE), simula el borrado de texto escribiendo la región seleccionada en el "anillo de borrado", pero sin eliminarlo del buffer.

La primera posición del "anillo de borrado" se accede con `C-y` tal y como ya se ha indicado. Las siguientes, con `M-y` (yank-pop) justo a continuación de un `C-y`.

La utilidad del "anillo de borrado" es de gran utilidad para las tareas de copiar y pegar **volúmenes de cualquier tamaño**[2] en cualquier parte. Por ejemplo, si el buffer contiene:

```
Grupo de Planificación y Aprendizaje
Departamento de Informática
Universidad Carlos III de Madrid
España
```

y se graba, primero, la segunda línea en el "anillo de borrado" (poniendo el cursor en la línea 2, `C-a`, `C-SPC`, `C-e` y `M-w`) y, después, la última línea, el "anillo de borrado" contendrá:

```
España
Departamento de Informática
```

Para insertar `Departamento de Informática` es suficiente hacer, primero, `C-y` con lo que en el buffer se tiene (supuesto que el cursor está ahora en la última línea):

```
Grupo de Planificación y Aprendizaje
Departamento de Informática
Universidad Carlos III de Madrid
España

España
```

[2] ¡De cualquier tamaño! De hecho, que el tamaño no sea una restricción severa en la realización de tareas informáticas, es uno de los principios fundamentales de diseño de software GNU.

Elemento	Comando Lisp	Teclas	Descripción
Punto	POINT-TO-REGISTER	`C-x / r`	Almacenar el punto en el registro *r*.
	POINT-TO-REGISTER	`C-x j r`	Ir a la posición almacenada en el registro *r*.
Región	COPY-TO-REGISTER	`C-x x r`	Salvar la región en el registro *r*.
	INSERT-REGISTER	`C-x g r`	Insertar la región almacenada en el registro *r*.

Cuadro 5: Comandos para el tratamiento de registros

y, sin mover el punto, `M-y`, con lo que se reemplaza la última línea por el siguiente contenido del "anillo de borrado":

```
Grupo de Planificación y Aprendizaje
Departamento de Informática
Universidad Carlos III de Madrid
España

Departamento de Informática
```

3.8. Registros

Los registros son otra de las facilidades de **Emacs**.

Además del "anillo de borrado" y del "anillo de marcas", se dispone de otras zonas de memoria, denominadas registros donde se puede almacenar prácticamente *cualquier cosa*, desde el punto o una sola letra hasta documentos completos pasando por párrafos, rectángulos (secciones de texto que no tienen por qué empezar en la primera columna y que comprenden un número cualquiera de filas), etc.

Desde el punto de vista práctico, un registro sirve para almacenar y recuperar el punto, una región o un rectángulo. El último elemento no será tratado por exceder los objetivos de este pequeño manual. Los comandos disponibles para el tratamiento de registros se muestran en la tabla 5.

El valor del registro *r* es un índice a toda la estructura de registros. Cuando se escribe un valor en un registro o se lee de él, es necesario indicar su valor. Dicho valor es, simplemente, un carácter. Así, por ejemplo, el registro "l" puede contener un punto, el registro "3" un bloque de información, etc.

Para especificar el valor de *r* se utiliza el minibuffer.

Dirección	Comando Lisp	Teclas
Hacia delante	ISEARCH-FORWARD	`C-s`
Hacia detrás	ISEARCH-BACKWARD	`C-r`

Cuadro 6: Comandos de búsqueda

3.9. Búsqueda y reemplazamiento

Aunque las posibilidades de la búsqueda y el reemplazamiento —especialmente la primera—, son en **Emacs** excepcionales, aquí sólo se expondrán los comandos típicos en cualquier editor de textos.

Sin embargo, antes de continuar, conviene destacar el hecho de que **Emacs** puede buscar en un solo buffer o en varios de ellos —incluso si no han sido cargados y son solamente ficheros en el disco duro—, incluso *expresiones regulares*.

Puesto que el reemplazamiento se suele asociar a una búsqueda que se realiza en primer lugar para después practicar la sustitución, se introducirá, primero, la búsqueda sin reemplazamiento y después se verá como practicar la sustitución.

Un forma de búsqueda muy útil, especialmente cuando se desarrollan grandes proyectos de Software son los denominados "Tags". Más adelante se describirá su funcionamiento.

3.9.1. Búsqueda sin reemplazamiento

La búsqueda puede realizarse bidireccionalmente, esto es, hacia delante o hacia detrás, tal y como se muestra en la tabla 6.

Cuando se pulsa cualquiera de los comandos de búsqueda indicados, el minibuffer presenta el mensaje `Isearch:`, invitando a introducir la cadena que se desea buscar. Según se pulsan teclas hasta completar la cadena deseada, puede observarse como el punto se desplaza desde la posición actual hacia delante/detrás en busca de los caracteres *que hasta el momento hayan sido introducidos* por el minibuffer. Si al introducir un nuevo carácter, la cadena introducida ya no existiera, **Emacs** emitiría un sonido de alerta advirtiéndolo.

Una vez que se ha encontrado una ocurrencia de la búsqueda seleccionada, es necesario pulsar `RET` para salir del modo de búsqueda. Antes de finalizar, se inserta, automáticamente, una marca en la posición actual del punto (cursor).

Por supuesto, el carácter `DEL` sigue teniendo sentido durante este proceso. Su efecto consiste en borrar el último carácter introducido en el minibuffer. Al hacerlo, tal vez el punto se mueva en dirección opuesta a la del proceso de búsqueda hasta la ocurrencia anterior.

Si durante la búsqueda se pulsa `C-g`, el proceso se interrumpe y el punto vuelve a donde estaba originalmente.

3.9.2. Búsqueda con reemplazamiento

La búsqueda con reemplazamiento se realiza con `M-%` (QUERY-REPLACE).

Una vez que se entra en el modo de reemplazamiento, primero es necesario indicar —en el minibuffer—, la cadena que se desea reemplazar, y luego, la nueva cadena.

Por cada una de las ocurrencias de la cadena que se desea sustituir, el punto se sitúa sobre ella y el minibuffer presenta el mensaje: `Query replacing` *cadena original* `with` *cadena final*`: (? for help)`. Entonces, se dispone de los siguientes comandos:

SPC (o `y`). Reemplaza la cadena apuntada por el cursor por la nueva cadena y mueve el cursor hasta la siguiente ocurrencia pidiendo, de nuevo, confirmación.

RET (o `q`). Abandona el modo de búsqueda con reemplazamiento.

, (coma). Reemplaza la cadena apuntada por el cursor pero no mueve el punto.

C-r Entra en el modo de edición del que se sale con `C-M-c`.

El modo de edición ofrece la iliusión de que no se está en el modo de búsqueda con reemplazamiento y se se puede acceder a los comandos de edición de **Emacs**. Cuando se sale de él, se vuelve de nuevo al modo de búsqueda.

C-w Borra la ocurrencia marcada por el punto y entra en el modo de edición. Se sale del modo de edición con `C-M-c`.

! Sustituye el resto de las ocurrencias sin pedir confirmación.

^ Mueve el punto a la ocurrencia anterior.

3.10. Ayuda

En la presentación de **Emacs** se indicó que estaba "auto-documentado". Pues bien, el sistema de ayuda es tan eficaz que, de hecho, contiene el manual completo de **Emacs** y muchísimo más.

No sólo eso, sino que además su manejo está tan bien documentado que basta con hacer `C-h` y seguir las instrucciones que saldrán en el minibuffer para acceder a toda la funcionalidad del sistema de ayuda. Entre otras cosas, merece la pena destacar `C-h t` (HELP-WITH-TUTORIAL), que ofrece un tutorial del manejo de **Emacs** para usuarios *recién llegados*.

Las opciones más significativas son las siguientes:

- `C-h a` (COMMAND-APROPOS). Modo de búsqueda de las funciones de edición implementadas.

  Una vez que se ha introducido este comando, puede indicarse en el minibuffer una expresión regular para la búsqueda de funciones de edición. Por ejemplo, si se teclea

`C-h a string`, **Emacs** presenta en pantalla aquellas funciones que incluyen, en su nombre, la palabra *string* y una pequeña descripción de su comportamiento, así como la combinación de teclas (si la hubiera) que invoca dicha función.

- `C-h c` (DESCRIBE-KEY-BRIEFLY). Muestra la función que ejecuta una determinada combinación de teclas.
- `C-h f` (DESCRIBE-FUNCTION). Describe la función que se indica a través del minibuffer.
- `C-h m` (DESCRIBE-MODE). Informa sobre los comandos propios del modo mayor de edición actual (consúltese la sección 3.11).
- `C-h l` (VIEW-LOSSAGE). Presenta las últimas 100 pulsaciones de tecla.

  Si las teclas pulsadas no son códigos de control, las representa por su carácter. En otro caso, utiliza los caracteres indicados en la tabla 1. Las combinaciones simultáneas de teclas se muestran con un guión.

  Este comando es de muchísima utilidad cuando, en una sesión de trabajo normal, de pronto ha ocurrido algo inesperado. Pulsando `C-h l` puede saberse la combinación o sucesión de teclas que produjo dicha acción y, además, aprenderá algo nuevo en **Emacs**.

Por último, la cantidad de posibilidades que existen en el sistema de ayuda son muchas. Para verlas, debe utilizarse `C-h C-h` (HELP-FOR-HELP).

3.11. Modos mayores

Emacs dispone de diferentes modos mayores, cada uno de los cuales caracteriza el comportamiento del buffer.

El modo por defecto es `Fundamental`. De hecho, *scratch* está en modo `Fundamental` mientras no se indique lo contrario.

La mayoría de los comandos que se presentan en este manual se comportan igual en cualquier modo mayor. Sin embargo, las teclas `TAB` y `DEL` alterarán fácilmente su funcionalidad. Por ejemplo, si en el modo mayor de Lisp se pulsa `TAB`, el punto avanza una cantidad proporcional de espacios a la anidación de la función que actualmente se está escribiendo —y que se corresponde con la diferencia de paréntesis abiertos menos aquellos que están cerrados. Pero en C, aunque `TAB` es básicamente lo mismo, avanzará un número de espacios diferentes en el caso de estar dentro de un comando `switch`, por ejemplo, que si está en un `for`. Otras diferencias, son relativas al uso de colores. Por ejemplo, el buffer actual no se coloreará de la misma manera (`M-x font-lock-mode` activa/desactiva el color) en el modo mayor de Lisp que en el de Java, o cualquier otro.

Emacs asume que el modo mayor puede deducirse de la extensión del fichero que se está editando. Así, por ejemplo, si un fichero tiene extensión ".f", será tratado en el modo mayor de `Fortran`; si el fichero tiene extensión ".texi", en el modo mayor `Texinfo`, con extensión ".perl" se empleará el modo mayor de `Perl`, etc.

Sin embargo, se puede forzar la selección del modo mayor de una de estas dos maneras: una, mediante alguno de los comandos M-x *modo-mayor*, por ejemplo, M-x LISP-MODE fuerza el modo `Lisp`, M-x PASCAL-MODE, el modo mayor `Pascal`, etc.; segundo, mediante una indicación explícita en cualquier posición de la primera línea del fichero que se carga. Esta indicación explícita consiste en colocar el modo mayor entre `-*-`. Así, por ejemplo, si un fichero contiene en su primera línea:

```
/* -*- C -*- */
```

será tratado en el modo de `C`[3]. Nótese que el modo mayor se ha colocado dentro de un comentario de C, o de lo contrario, al compilar este fichero habría un error.

Por último, para conocer los comandos nuevos o aquellos que han modificado su comportamiento en el modo mayor actual, se puede utilizar `C-h m` (DESCRIBE-MODE).

3.12. Tags

Los tags son otra de las características más sobresalientes de **Emacs**. Mientras que prácticamente todos los editores disponen de comandos de búsqueda —que, dicho sea de paso, son normalmente mucho menos capaces—, **Emacs** ofrece la posibilidad de buscar a través de varios ficheros una cadena en concreto.

Mientras que en **Emacs** es posible trabajar con la llamada "barra rápida" (*speedbar*) haciendo `M-x speedbar` para gestionar manualmente el acceso a varios ficheros, una tabla de tags consiste en un fichero que contiene punteros a las posiciones de una colección de ficheros donde están las cadenas de texto más sobresalientes. Por ejemplo, en C, las declaraciones de funciones y procedimientos, definiciones de tipos, constantes, etc. En Lisp, la declaración de funciones, parámetros o variables especiales, macros, etc.

Para poder hacerlo, es necesario disponer del comando `etags` que se distribuye siempre con **Emacs**. El formato es:

etags [opciones] tabla_tags fichero

y las opciones más relevantes son:

- `-o nombre fichero`. Indica el nombre del fichero en el que se escribirá la tabla de tags.
- `-a`. Añade a la tabla de tags seleccionada, los que se deduzcan de otro fichero de código.

[3]Otra forma de conseguir lo mismo es empleando las denominadas "variables locales" (*locals*) que, sin embargo, no se explicarán en este documento.

Por otra parte, *fichero* es aquél que se desea examinar para obtener sus tags más relevantes y que serán incluídos en *tabla_tags*.

Supóngase, por ejemplo, que se está desarrollando un proyecto con once códigos fuente diferentes en C: types.h, defs.h, cursor.h, cursor.c, port.h, port.c, screen.h, screen.c, console.h, console.c y application.c. La siguiente sucesión de comandos creará una tabla llamada application.tags que será utilizada como se verá más tarde.

```
etags -o application.tags types.h
etags -a -o application.tags defs.h
etags -a -o application.tags cursor.h
etags -a -o application.tags cursor.c
etags -a -o application.tags port.h
etags -a -o application.tags port.c
etags -a -o application.tags screen.h
etags -a -o application.tags screen.c
etags -a -o application.tags console.h
etags -a -o application.tags console.c
etags -a -o application.tags application.c
```

Una vez que el fichero application.tags ya ha sido creado[4] puede ejecutar el comando `M-x visit-tags-table`, desde **Emacs**. Dicho comando sirve para solicitar a **Emacs** que lea una tabla de tags en concreto, cuyo nombre deberá teclearse por el minibuffer (en este caso, application.tags).

Si ya se ha cargado la tabla de tags, ni siquiera es necesario cargar los once ficheros de fuente para buscar, por ejemplo, una función llamada `print-page` que se encuentra en port.h y en port.c. Para ello, basta con pulsar `M-.` (FIND-TAG) y, a continuación, introducir `print-page` por el minibuffer. **Emacs** cargará, automáticamente, el fichero port.h primero, y llevará el punto a la primera ocurrencia de `print-page` (probablemente allí donde se definió el prototipo de la función). Obviamente, este proceso es mucho más rápido, incluso que emplear la barra rápida (recurso habitualmente presente en tantos editores) puesto que entonces es necesario saber dónde buscar, mientras que gracias a los tags, no lo es en absoluto.

Ahora bien, si se desean realizar búsquedas de un tag para sustituírlo, se debe utilizar `M-x tags-query-replace` y proceder de forma análoga a la sustitución normal.

En cualquier caso, tanto si se realizan sustituciones como si no, es posible avanzar a la siguiente ocurrencia con `M-,`.

Por último, es posible realizar búsquedas de expresiones regulares mediante `C-x t`, sin embargo, las expresiones regulares van más allá de las intenciones de este manual y su estudio se deja al lector interesado.

[4]De hecho, si piensa utilizar **Emacs** para desarrollar un proyecto que involucra varios ficheros, es altamente recomendable que automatice estas operaciones con GNU Make.

3.13. Macros

Las macros de teclado —o simplemente macros—, son una facilidad que sirve para asignar a un nombre una sucesión de comandos, de tal modo que, invocándola, se ejecutan automáticamente todos los pasos.

Para definir una macro, se utiliza `C-x (` (START-KBD-MACRO). A continuación, se introducen los comandos de la macro y, por último, se cierra con `C-x )` (END-KBD-MACRO). Durante el tiempo que dura la grabación, el modo menor presentará `Def`.

Por ejemplo, imagínese que deseamos crear una macro que sirva para dividir la ventana actual en dos partes, llevar el punto a la ventana recién creada, y allí cambiar al buffer *scratch*[5]. Los siguientes comandos definen la macro (se incluyen también los comandos para la apertura y cierre de la grabación de la macro):

```
C-x (
C-x 2
C-x o
C-x b
C-x )
```

Obsérvese que si se decide utilizar una macro, será porque será usada con frecuencia. En tal caso, basta con añadir `C-x (` y `C-x )` a la primera ejecución. Desde entonces, las siguientes invocaciones y ejecuciones serán mucho más rápidas.

Una vez que ha sido grabada, para ejecutar una macro basta con hacer `C-x e` (CALL-LAST-KBD-MACRO), que ejecuta la última macro definida.

Ahora bien, al programar varias macros será indispensable dar un nombre a cada una, a fin de diferenciarlas y no perderlas. Para ello, debe utilizarse `M-x name-last-kbd-macro`. Por ejemplo, la macro recién creada podría bautizarse con `open-clipboard`.

Para finalizar el ejemplo, resulta muy aconsejable programar una segunda función que sirve para cerrar el *scratch* y volver de nuevo al buffer de trabajo. Para ello, podría definirse una segunda macro con:

```
C-x (
RET
------------------------------
RET
C-x 0
C-x )
```

[5]Esta macro es de utilidad para manejar un área reservada en la que se pueden realizar anotaciones mientras trabaja en otro buffer. Además, *scratch* es un buffer como cualquier otro, por lo que allí se dispone de todos los comandos de **Emacs**. En otros editores, el *clipboard* es una zona tan especial, como poco socorrida, por lo que no se utiliza demasiado.

Las definiciones de macros son globales a toda la sesión. En tal caso, la secuencia anterior puede introducirse mientras el *scratch* es el buffer actual (que es lo más recomendable) o no. En cualquier caso, una vez grabada, puede invocarse desde cualquier buffer. Esta función se denominará `close-clipboard` con `M-x name-last-kbd-macro close-clipboard`.

Realice un par de pruebas. Ejecute `M-x open-clipboard`. Si no utilizó el *scratch* hasta ahora, debería estar vacío. Escriba lo que quiera y cuando haya acabado ejecute `M-x close-clipboard`. Seguramente su máquina habrá ejecutado la macro tan rápidamente que ni siquiera habrá podido ver la línea de guiones. Asegúrese y ejecute `C-x b *scratch*`. Ahora podrá ver como después de sus comentarios aparece una línea de guiones y el punto está al principio de una nueva línea, lista para recibir el siguiente apunte.