Emacs. Manual mínimo*

Miguel Moro Vallina**

1. Introducción

El objetivo de este texto es *documentar* algunas de las características más relevantes del editor de textos *emacs*, uno de los *editores de texto* más potentes que existen.

El EMACS original consistía en un conjunto de *Editor MACroS* para el editor TECO y fue escrito por Richard Stallman y Guy Steele en 1975. Desde entonces ha habido múltiples versiones de *Emacs*, pero las más populares hoy en día son la *GNU Emacs*, iniciada por el propio Stallman en 1984 y que aún mantiene él mismo, y *XEmacs*, un *fork* de *GNU Emacs* desarrollado a partir de 1991. Ambos emplean el lenguaje de programación *Emacs LISP*, que provee la mayor parte de las funcionalidades de edición; el resto de ambos programas está escrito en C, lenguaje que constituye también la base del propio *Emacs LISP*. Este último puede emplearse extensamente para personalizar, extender y afinar el *Emacs*; en esta característica reside buena parte de la potencia del programa.

Ya el EMACS original poseía una gran capacidad para ser personalizado y extendido, lo cual daba lugar al permanente peligro de bifurcaciones y pérdidas de compatibilidad entre las diferentes versiones. Stallman escribiría más tarde que «las mejoras deben devolvérseme para incorporarlas y redistribuirlas». El objetivo, en cualquier caso, no era de naturaleza exclusivamente técnica; por el contrario, residía en que el programa pudiese ser *compartido* de manera *comunitaria*, un objetivo que tomará cuerpo posteriormente con la creación del proyecto *GNU*, del cual el propio *GNU Emacs* es el primer programa en aparecer.

En principio, nos centraremos sólo en explicar el funcionamiento de *Emacs* en *modo texto*, dejando de lado las características particulares de la versión gráfica (*XEmacs*). Obrando de este modo no restamos generalidad a lo que se diga, toda vez que las funciones válidas para *Emacs* lo son también para

*El objetivo de este pequeño texto es meramente *documentar algunas* de las características más relevantes el editor de textos *emacs*. Se trata de un mero resumen de la extensa y completa guía *GNU Emacs Manual*, preparada por Richard Stallman y publicada por la Free Software Foundation (`http://www.gnu.org/software/emacs/manual/`). Algunas referencias iniciales se han tomado también de la *Wikipedia* (`http://www.wikipedia.org`). La composición de este documento se ha realizado mediante LaTeX.

**Correo: `narodnaia@gmail.com`. Web: `http://narodnaia.googlepages.com`

XEmacs, no cumpliéndose siempre el recíproco.

Al ejecutar *Emacs*, aparece, ocupando toda la pantalla, el llamado *frame* del editor. Este *frame* se divide en cuatro zonas, cada una de ellas con una función o funciones determinadas; haciendo un recorrido de arriba hacia abajo de la pantalla, se distinguen:

- *Barra de menú*: proporciona acceso a las funciones de *Emacs*, organizadas por categorías (*Buffers*, *Files*, *Tools*...). Para acceder al menú se pulsa F10 o `M-*`.
- *Ventana(s)*: la parte central de la pantalla está ocupada por una o varias ventanas que muestran el contenido de ficheros o información diversa.
- *Línea de modo*: en todos los terminales que lo soportan (*i.e.*, en casi todos), la línea de modo está en *vídeo inverso*. Muestra información variada (por ejemplo, el nombre del fichero que se está editando y la línea en la que está situado el *cursor*).
- *Área de eco* y *minibuffer*: ambos están situados en la última línea de la pantalla. El *área de eco* o *echo área* aparece sólo cuando, al escribir una combinación de teclas, se hace una pausa entre las distintas teclas que componen la combinación. Lo escrito antes de la pausa «hace eco» en la pantalla. El *minibuffer*, por su parte, sirve para introducir los eventuales argumentos que se hayan de pasar a un comando.

2. Teclas

Todas y cada una de las funciones de *Emacs* poseen un nombre que está compuesto de una o varias palabras separadas por guiones. Una parte de estas funciones —la mayoría— están asociadas a una determinada *combinación de teclas*, de forma que se puedan ejecutar rápidamente. Cada tecla de la combinación puede ir acompañada de *modificadores*: la tecla de *control* —la tecla Ctrl en la inmensa mayoría de los teclados— y la tecla *Meta* —tecla Alt en la mayoría de los teclados, que de todos modos puede sustituirse por Esc si aquélla no funciona o no existe—. Representaremos ambos modificadores como `C-` y `M-`, respectivamente.

Todas las funciones de *Emacs* pueden ejecutarse mediante la combinación `M-x` —con lo cual el cursor se situará en el *minibuffer* a la espera— y su nombre explícito.[1]

3. Entrar y salir

Cuando se inicia *Emacs* sin especificar, en la línea de comandos, ningún fichero o ficheros a editar —la forma más adecuada y ortodoxa de hacerlo,

[1] En las tablas, la ausencia de una combinación específica de teclas asociada a una determinada función se indica dejando la casilla de la tecla en blanco y ofreciendo sólo la descripción de la función, precedida por su nombre entre paréntesis.

Cuadro 1: *Suspender y salir*

Tecla	*Descripción*
`C-z`	(*suspend-Emacs*) Suspende o detiene temporalmente la ejecución de *Emacs*, al igual que en el comando homónimo del *shell* de GNU/Linux, aunque en este caso se trata de una función del propio *Emacs*.
`C-x C-c`	(*save-buffers-kill-Emacs*) Sale de *Emacs*, preguntando previamente si se desea guardar el trabajo realizado.

por cierto—, se abre un *buffer* llamado **scratch**, que será el *buffer activo* al comenzar. En este *buffer* se pueden, entre otras cosas, escribir expresiones en *Emacs Lisp* y *evaluarlas*. Los comandos para detener la ejecución y para terminar el programa se recogen en la tabla 3.[2]

4. Lo más básico

Movimientos del cursor Los comandos necesarios para el movimiento del cursor y el borrador de caracteres, líneas o palabras, se reseñan en la tabla 4.

Caracteres de control Para introducir un *carácter de control*, se pulsa la combinación `C-q` —asociada a la función *quoted-insert*— y posteriormente el carácter en cuestión, bien como combinación de teclas, bien escribiendo su código en *octal*.

Líneas En *Emacs*, se entiende por *línea* aquella porción de texto que hay entre dos caracteres sucesivos de *nueva línea*, que se insertan con la tecla Enter. Cuando una línea no cabe longitudinalmente en la pantalla, *Emacs* puede *romperlas* —*i.e.*, continuarlas en la siguiente «línea» de la pantalla— o bien *truncarlas* —*i.e.*, mostrar sólo aquello que cabe en la pantalla. En el primer caso, *Emacs* muestra un carácter \ al final de cada «línea» y, en el segundo, un carácter `$`. Por defecto, *Emacs* parte las líneas; si se quiere que las trunque, debe establecerse la variable *truncate-lines* al valor *non-nil*.[3] Cuando las líneas están truncadas, habrá que mover la pantalla hacia la derecha y hacia la izquierda para poderlas leer enteras; esta operación se realiza mediante las funciones *scroll-left* y *scroll-right*, asociadas a las teclas `C-x <` y `C-x >`.

La tecla Enter está asociada a la función *newline*, que inserta una línea *antes* del *punto*; por su parte, la combinación `C-o` lo está a *open-line*, que inserta una

[2]Por el carácter de *guía rápida* que posee este texto, se emplean con profusión tablas para relacionar la combinación de teclas, la función de *Emacs* asociada a ella y la descripción de dicha función.

[3]Para ver el valor de una variable y la documentación asociada a ella, se emplea el comando `C-h v` (*display-variable*), tras lo cual habrá que introducir el nombre de la variable en cuestión. Para establecer un nuevo valor para ella, se emplea la función *set-variable*, seguida del nombre de la variable y su nuevo valor.

CUADRO 2: *Comandos básicos*

Tecla	*Descripción*
`C-a`	(*beginning-of-line*) Sitúa el cursor al comienzo de la línea actual.
`C-e`	(*end-of-line*) Lleva al cursor al final de la línea sobre la que se encuentra.
`C-f` o [→]	(*forward-char*) Hace avanzar el cursor al siguiente carácter.
`C-b` o [←]	(*backward-char*) Hace retroceder al cursor al carácter anterior.
`M-f`	(*forward-word*) Hace avanzar el cursor a la siguiente palabra.
`M-b`	(*backward-word*) Hace retroceder el cursor a la palabra anterior.
`C-n` o [↓]	(*next-line*) Hace avanzar el cursor a la línea siguiente.
`C-p` o [↑]	(*previous-line*) Hace retroceder el cursor a la línea anterior.
`C-x [`	(*backward-page*) Retrocede al salto de página anterior.[a]
`C-x ]`	(*forward-page*) Avanza al siguiente salto de página.
`M-<`	(*beginning-of-buffer*) Sitúa el cursor al comienzo del *buffer* actual.
`M->`	(*end-of-buffer*)
[⟵]	(*delete-backward-char*) Borra el carácter inmediatamente anterior a la posición del cursor.
`C-d` o [Supr]	(*delete-char*) Borra el carácter sobre el que está situado el cursor.
`C-k`	(*kill-line*) Borra la línea en la que está situado el cursor, a partir de la posición en la que éste se encuentra.
`M-d`	(*kill-word*) Borra la palabra en la que está situado el cursor, a partir de la posición en la que éste se encuentra.

[a]Los ficheros se dividen en páginas mediante el carácter de salto de página o *formfeed* (carácter ASCII L̂, código octal *014*), el cual, al imprimir un fichero de texto, fuerza en el dispositivo de impresión un salto de página real. Los saltos de página se introducen mediante la secuencia `C-q C-l` y se borran con la tecla [Del].

CUADRO 3: *Posición del punto*

Tecla	*Función*
	(*what-page*) Devuelve la página y línea actuales del punto.
	(*what-line*) Devuelve la línea actual del punto.
`C-x l`	(*count-lines-page*) Cuenta las líneas de la página actual.
`C-x =`	(*what-cursor-position*) Proporciona el código del carácter situado detrás del punto, su posición absoluta (contando los caracteres que lo preceden desde el comienzo del *buffer*) y la *columna* o posición horizontal de la pantalla en la que está situado.

línea en *blanco* después del punto. En otras palabras, la primera función inserta una línea y lleva a ella al cursor; la segunda inserta una línea, pero deja al cursor en donde estaba.[4]

Deshacer Para deshacer el último cambio realizado, se puede emplear la combinación `C-x u`, asociada a la función *undo*. *Emacs* guarda los cambios realizados hasta un número determinado; este número de cambios será el que se pueda deshacer secuencialmente pulsando la combinación mencionada.

Posición del punto En la *línea de modo* aparece el número de línea en el que se encuentra el punto; además, existe una serie de funciones que proporcionan una información más detallada acerca de la posición del punto, y que se resumen en la tabla 4.

Ficheros Más adelante en el texto se abordará el manejo de ficheros en profundidad, por lo que aquí solamente se indica lo siguiente:

- `C-x C-f` (*find-file*) abre un fichero en un nuevo *buffer*.
- `C-x C-s` (*save-buffer*) guarda el trabajo actual.

Argumentos numéricos Existe una serie de funciones que pueden requerir un argumento numérico. Existen dos opciones para indicarlo:

- pulsar la tecla *Meta* mientras se teclea el número.
- pulsar la combinación `C-u` (*universal-argument*) seguido del número.

Un ejemplo del uso de argumentos numéricos es el siguiente: si se quieren insertar muchas copias de un determinado carácter —por ejemplo, una *a*—, lo

[4]El *punto* se identifica con el *cursor*, pero con dos importantes salvedades: en primer lugar, el punto se sitúa siempre *entre* dos caracteres, mientras que el cursor, en una terminal de texto, está siempre *sobre* un carácter, aunque éste sea un espacio en blanco; en segundo lugar, los *buffers* no activos, aunque no tengan cursor, mantienen su punto. De esta forma, «recuerdan» la posición en la que estaba el cursor la última vez que se visitaron.

CUADRO 4: *Comandos del* minibuffer

Tecla	*Descripción*
Tab	(*minibuffer-complete*) Completa el texto del *minibuffer* tanto como sea posible.
Spc	(*minibuffer-complete-word*) Completa el texto del *minibuffer*, pero una palabra solamente.
Enter	(*minibuffer-complete-and-exit*) Completa el texto del *minibuffer* y sale de él, ejecutando la orden escrita.
?	(*minibuffer-list-completions*) Saca una lista con todas las opciones que completarían lo escrito hasta el momento en el *minibuffer*.
↑	(*previous-history-element*) Funciona de manéra análoga al *shell* de GNU/Linux: va mostrando los elementos *anteriores* de la pila de comandos introducidos.[a]
↓	(*next-history-element*) Muestra los elementos *posteriores* de la «historia» de los comandos introducidos.

[a]Cada *minibuffer* (ficheros abiertos, funciones de *Emacs*...) almacena los elementos introducidos en una pila distinta.

más eficaz será pulsar la siguiente combinación de teclas: `C-u 6 4 a`, que insertará 64 copias de dicho carácter. También pueden usarse argumentos numéricos previamente a un comando de desplazamiento, para moverse hacia adelante o hacia atrás un número determinado de caracteres, palabras o páginas.

5. El minibuffer

Cuando un comando requiere un argumento no numérico —por ejemplo, para abrir un fichero—, el cursor se desplaza a la última línea de la pantalla y se muestra un *prompt*. En el caso de que la orden que activa el *minibuffer* se haya dado erróneamente, se puede «salir de él» mediante la combinación `C-g`. Los comandos empleados en el *minibuffer* se reseñan en la tabla 5

Una de las características más destacables de la forma de operar del *minibuffer* es el «completamiento». Su comportamiento es análogo al *shell* de GNU/Linux: al escribir una parte del argumento deseado y pulsar la tecla Tab, pueden ocurrir dos cosas: si sólo hay un posible valor del argumento (un solo fichero, una sola función...) que comience por la porción escrita, se *completa* automáticamente; si no, se muestra una lista con las alternativas.

6. Ayuda

Las diferentes funciones de ayuda y las combinaciones de teclas asociadas a ellas se describen en la tabla 6.

Cuadro 5: *Funciones de ayuda*

Tecla	*Descripción*
`C-h C-h`	(*help-for-help*) Muestra una lista de las opciones de ayuda, junto con una breve descripción de cada una de ellas.
`C-h a`	(*apropos-command*) Lista de comandos cuyo nombre coincide con una expresión regular determinada, que *Emacs* solicitará.
`C-h b`	(*describe-bindings*) Lista de combinaciones de teclas más importantes del *modo* en el que se está trabajando.
`C-h c`	(*describe-key-briefly*) Describe brevemente una combinación de teclas.
`C-h k`	(*describe-key*) Lo mismo que lo anterior, excepto en la brevedad.
`C-h f`	(*describe-function*) Describe una función.
`C-h i`	(*info*) Ejecuta *info*, una utilidad para navegar por ficheros de documentación.
`C-h m`	(*describe-mode*) Describe el *modo* que se esté empleando.
`C-h t`	(*help-with-tutorial*) Ejecuta un tutorial de *Emacs*.
`C-h v`	(*describe-variable*) Describe una variable.
`C-h w`	(*where-is*) Informa de qué teclas ejecutan un comando determinado.

7. Marca y región

Determinados comandos de *Emacs* actúan sobre una porción de *buffer* llamada *región*, definida como la porción de espacio comprendida entre la *marca* y el punto. La marca es una suerte de señal que se sitúa en un determinado lugar del *buffer*, permaneciendo en él hasta que se vuelve a fijar. De este modo, para una marca fijada, las distintas posiciones en las que puede estar el punto definen otras tantas regiones. Las operaciones más relevantes en relación con la marca y la región se recogen en la tabla 7.

Cada *buffer* «recuerda» las dieciséis últimas posiciones de su marca, en el llamado *mark ring*. La combinación `C-u C-`Spc sitúa el punto en la última posición de la marca. Asimismo, existe un *global mark ring* que «recuerda» la secuencia de *buffers* en los cuales se ha establecido la marca recientemente. `C-x C-`Spc sitúa el punto en la última posición de dicho *global mark ring*.

8. Edición

En las tablas 8 y 8 se describen las funciones más relevantes para borrar, cortar, pegar y mover texto.

Registros Los registros son posiciones de memoria donde se almacenan texto o posiciones del punto para su posterior utilización. Los comandos fundamentales se listan en la tabla 8.

CUADRO 6: *Marca y región*

Tecla	*Descripción*
`C-` Spc	(*set-mark-command*) Sitúa la marca en el lugar donde se encuentra el punto.
`C-x C-x`	(*exchange-point-and-mark*) Intercambia las posiciones de marca y punto.
`M-@`	(*mark-word*) Marca desde la posición actual del punto hasta el final de la palabra sobre la que esté situado.
`M-h`	(*mark-paragraph*) Marca desde la posición actual del punto hasta el final del párrafo sobre el que esté situado.
`C-x h`	(*mark-whole-buffer*) Marca todo el *buffer* que se encuentra activo.
`C-x C-p`	(*mark-page*) Marca la página sobre la que está situado el punto.

Bookmarks Son similares a los registros, pero se diferencian de ellos en dos aspectos: en primer lugar, tienen nombres largos; en segundo lugar, su contenido permanece de una sesión a otra. Su utilización está especialmente indicada, por tanto, para «marcar» posiciones de un fichero; de ahí su nombre. Los comandos más empleados se resumen en la tabla 8.

9. Buscar y reemplazar

Búsqueda incremental En este modelo de búsqueda, a medida que se introduce el texto que se desea buscar, el cursor se va moviendo en el *buffer*, situándose en el texto que se encuentre más cerca del punto y que coincida con lo escrito hasta el momento. Existen dos formas de ejecutar la búsqueda: hacia adelante (*isearch-forward*) y hacia atrás (*isearch-backward*), teclas `C-s` y `C-r`. Al pulsar Enter finaliza la búsqueda.

Búsqueda no incremental En este caso, el cursor no se mueve hasta que no se haya finalizado de introducir la cadena de caracteres que se desea buscar. La búsqueda no incremental se activa pulsando Enter inmediatamente después de `C-s` o de `C-r`.

Reemplazo Puede, a su vez, ser *condicional* o *incondicional*. En el segundo caso, *Emacs* reemplaza todas y cada una de las apariciones del texto señalado por aquel que se le indique; en el primero, pregunta caso por caso si se desea hacer el cambio. La búsqueda incondicional se realiza con la función *replace-string*; la condicional, con *query-replace* o la combinación de teclas `M-%`.

CUADRO 7: *Borrar y cortar*

Tecla	*Descripción*
`M-\`	(*delete-horizontal-space*) Borra espacios y tabuladores alrededor del cursor.
`M-` Spc	(*just-one-space*) Realiza la misma operación que la función anterior, pero dejando un espacio en blanco.
`C-x C-o`	(*delete-blank-lines*) Borra las líneas en blanco alrededor de la actual.
`M-^`	(*delete-indentation*) Junta dos líneas borrando el retorno de carro y el sangrado que tuviera la segunda.
`C-k`	(*kill-line*) Con un argumento numérico, corta las líneas que se le indiquen. Con el argumento `0`, corta el texto situado antes del punto en la línea actual, y sin argumento corta desde el punto hasta el final de la línea.[a]
`C-w`	(*kill-region*) Corta la región situada entre la marca y la posición del punto actuales.
`M-d`	(*kill-word*) Corta desde la posición actual del punto hasta el final de la palabra.
`M-` Del	(*backward-kill-word*) Corta desde la posición actual del punto al comienzo de la palabra.
`C-x` Del	(*backward-kill-sentence*) Corta hasta el comienzo de la frase sobre la que se encuentre el punto.[b]
`M-k`	(*kill sentence*) Corta hasta el final de la frase sobre la que se encuentre el punto.
`M-z`	(*zap-to-char*) Borra hasta la primera aparición del carácter que se indique.

[a] La diferencia ente *borrar* y *cortar* es que, en el segundo caso, el texto eliminado se guarda en memoria (en el llamado *kill ring*) para, si es necesario, pegarlo en otro lugar. *Emacs* sólo tiene un *kill ring* para todos los *buffers*, de forma que se puede cortar texto de uno y pegarlo en otro.

[b] Para que *Emacs* pueda distinguir unas frases de otras, éstas deben separarse por *dos* espacios en blanco, siguiendo una convención tipográfica ampliamente extendida en el mundo anglosajón.

CUADRO 8: *Pegar*

Tabla	*Descripción*
`C-y`	(*yank*) Pega lo último cortado *antes* de la posición del punto.
`M-y`	(*yank-pop*) Reemplaza lo que se acaba de pegar con contenidos anteriores del *kill-ring*.
`M-w`	(*kill-ring-save*) Copia la región al *kill-ring* sin borrarla.
`C-M-w`	(*append-next-kill*) Se trata de un prefijo que actúa sobre cualquier comando de «corte» que se pulse inmediatamente después, modificándolo para que añada lo cortado al último fragmento del *kill-ring*. (*prepend-to-buffer*) Añade la región del *buffer* actual al final de otro, cuyo nombre solicitará *Emacs*. (*copy-to-buffer*) Sustituye el contenido del *buffer* con la región. (*insert-buffer*) Inserta la región en la posición actual del punto en el *buffer* que se le especifique. (*append-to-file*) Inserta la región al final de un fichero cuyo nombre solicitará *Emacs*.
`C-x r k`	(*kill-rectangle*) Corta el *rectángulo* marcado.[a]
`C-x r y`	(*yank-rectangle*) Pegar rectángulo.
`C-x r o`	(*open-rectangle*) Inserta un número de espacios igual al ancho del rectángulo definido actualmente.
`C-x r c`	(*clear-rectangle*) Reemplaza el contenido del rectángulo con espacios.
`C-x r t`	(*string-rectangle*) Inserta la cadena de texto que se le indique en cada una de las líneas del rectángulo.

[a]Un *rectángulo* es un texto con forma rectangular. Su empleo es muy frecuente a la hora de trabajar con varias columnas o con tablas. Se delimita situando la marca en una esquina y el punto en la diagonalmente opuesta. Obsérvese que, de esta manera, marca y punto definen a la vez un rectángulo y un conjunto de líneas, siendo los comandos empleados los que diferencian entre uno y otro.

CUADRO 9: *Registros*

Tecla	*Descripción*
`C-x r` Spc `r`	(*pont-to-register*) Guarda la posición del cursor en el registro *r*.
`C-x r j r`	(*jump-to-register*) Mueve el punto a la posición guardada en el registro *r*.
`C-x r s r`	(*copy-to-register*) Copia al registro *r*.
`C-x r i r`	(*insert-register*) Inserta el contenido del registro *r* *después* de la posición del punto.
`C-x r r r`	(*copy-rectangle-to-register*) Copia el contenido del rectángulo marcado al registro *r*. (*view-register*) Muestra el contenido del registro que se indique.

Cuadro 10: *Bookmarks*

Tecla	*Descripción*
`C-x r m`	(*bookmark-set*) Copia la posición del punto del *buffer* activo a un *bookmark* determinado, cuyo nombre solicitará *Emacs*.
`C-x m b`	(*bookmark-jump*) Salta a la posición guardada en un *bookmark*, cuyo nombre deberá indicarse.
`C-x r l`	(*list-bookmarks*) Lista todos los *bookmarks*.
	(*bookmark-save*) Guarda todos los *bookmarks* en el fichero de *bookmarks* establecido por defecto.

10. Manejo de ficheros

Visitar y guardar ficheros *Visitar* un fichero es abrirlo, ya sea cargando en un *buffer* el contenido de uno ya existente o creándolo *ex novo*. Cada *buffer* tiene asociado un directorio por defecto, en el cual *Emacs* buscará, a menos que se indique lo contrario, el fichero que se indique. El directorio por defecto asociado a un *buffer* determinado se puede mostrar con la función *pwd*. La tabla 10 resume las funciones más importantes para visitar ficheros y guardarlos.

Copias de respaldo *Emacs* puede realizar una o varias copias de seguridad de un fichero cuando éste se modifica. Independientemente de cuántas veces se guarde el fichero con el que se está trabajando, el contenido de la copia de respaldo será siempre el que tenía el fichero *antes* de visitarlo. Si se le indica a *Emacs* que mantenga un solo fichero de respaldo, éste se nombrará añadiendo el carácter al nombre original. Si debe mantener varios, los nombres de las copias sucesivas terminarán con `ñ` , siendo n el número de copia. El comportamiento de cara a las copias de respaldo está regulado por la variable *version-control*, que admite tres posibles valores: t para copias numeradas, *nil* para una sola copia y *never* para que no haga nunca copias.

Protección contra edición simultánea Cuando se hace la primera modificación a un *buffer* de *Emacs* que esté visitando un fichero, el editor pone un «cerrojo» (*lock*) a éste, que se desactiva una vez que se guardan los cambios. De esta manera, se evita que varias usuarias puedan estar editando el fichero a la vez sin ser conscientes de ello. Cuando alguien está trabajando sobre un fichero y otra persona lo intenta visitar, *Emacs* detecta la «colisión» y pregunta a la recién llegada qué desea hace. Son posibles tres opciones: `s` para «robar el candado», `p` para proceder —*i.e.*, para editar el fichero de todos modos— y `q` para salir.

Auto-guardado Cada vez que se completa un determinado número —por defecto, trescientos— de caracteres escritos, *Emacs* vuelca automáticamente el contenido del *buffer* con el que se está trabajando a un fichero especial, cuyo nombre está compuesto por el nombre del fichero que se está visitando, encerrado entre caracteres `#`. Si se borra una parte importante del texto del *buffer*,

CUADRO 11: *Visitar y guardar ficheros*

Tecla	*Descripción*
`C-x C-f`	(*find-file*) Abre un fichero. Se activa el *minibuffer* pidiendo el nombre del que se desea visitar, a lo que se puede responder con uno ya existente (mediante la tecla Tab se muestra una lista del directorio actual) o con el nombre de un fichero nuevo que se desee crear.
`C-x C-r`	(*find-file-read-only*) Abre el fichero en modo *sólo lectura*.[a]
`C-x C-v`	(*find-alternate-file*) Visita un fichero y cerrar o «matar» el *buffer* actual.
`C-x 4 f`	(*find-file-other-window*) Visita un fichero situando el *buffer* en una ventana distinta a la actual.
`C-x C-s`	(*save-buffer*) Guarda el fichero del *buffer* actual.
`C-x s`	(*save-some-buffers*) Guarda los ficheros con los que se esté trabajando, uno por uno, solicitando confirmación para cada uno de ellos.
`C-x C-v`	(*write-file*) Guarda el contenido del *buffer* actual en el fichero que se especifique.

[a]Cuando se abre una fichero sin permiso de escritura, se abre automáticamente en este modo. Para obviar este comportamiento se usará la función *vc-toggle-read-only* (`C-x C-q`).

el auto-guardado se desactiva temporalmente, en previsión de que el borrado se haya efectuado de modo accidental; el auto-guardado se vuelve a activar simplemente guardando los cambios del *buffer* con `C-x C-s`. *Emacs* realiza también la operación de auto-guardado cuando detecta un «error fatal» —por ejemplo, si desde el *shell* de GNU/Linux se mata el proceso con la orden *kill*—. Cuando la sesión se cierra de manera «normal», *Emacs* borra el fichero de auto-guardado.

Para recuperar el fichero de auto-guardado, se emplea la función *recover-file*. Asimismo, se pueden recuperar todos los ficheros que se estaban editando en una sesión con *recover-session*; al ejecutar esta última función, aparece una lista con los ficheros de la sesión: se sitúa el punto sobre el que se quiera recuperar y se pulsa `C-c C-c`.

Comparación Existen tres funciones que realizan *comparaciones*:

- *diff* compara dos ficheros, mostrando las diferencias en el *buffer* **diff**.
- *diff-backup* compara un fichero con su copia de respaldo más reciente.
- *compare-windows* compara el texto de la ventana actual con el de la ventana de al lado; la comparación finaliza cuando se alcanza el primer carácter disímil, situándose el punto de ambas ventanas en dicho carácter.

Miscelánea Cabe, además de las anteriores, citar otras cuatro funciones de interés:

CUADRO 12: *Operaciones con* buffers

Tecla	*Descripción*
`C-x b`	(*switch-to-file* Cambia a otro *buffer*, solicitando su nombre; si se proporciona el nombre de un *buffer* no abierto, se crea uno con dicho nombre y se cambia a él.
`C-x o`	(*other-window*) Selecciona otra ventana.
`C-x 0`	(*delete-window*) Borra la ventana seleccionada.
`C-x 1`	(*delete-other-windows*) Borra todas las ventanas excepto la seleccionada.
`C-x 4 b`	(*switch-to-buffer-other-window*) Selecciona un *buffer* en una ventana nueva.
`C-x 4 f`	(*find-file-other-window*) Visita un fichero y sitúa su *buffer* en otra ventana, que se convierte en la seleccionada.
`C-x 4 0`	(*kill-buffer-and-window*) Borra la ventana seleccionada y cierra («mata») el *buffer* que contenía.
`C-x ^`	(*enlarge-window*) Alarga la ventana activa en dirección vertical.
`C-x }`	(*enlarge-window-horizontally*) Alarga la ventana activa en dirección horizontal.
`C-x {`	(*shrink-window-horizontally*) Encoge la ventana activa en dirección horizontal.
`C-x -`	(*shrink-window-if-larger-than-file*) Encoge la ventana activa si el fichero que está visitando es más pequeño que ella.
`C-x +`	(*balance-windows*) Hace que todas las ventanas de la sesión posean la misma altura.

– *insert-file* inserta, tras el punto, una copia del contenido del fichero que se especifique, dejando el punto donde estaba y situando la marca al final del texto insertado.

– *write-region* realiza justamente la operación contraria: escribe una copia de la región marcada en un fichero.

– *append-to-file* copia la región *a continuación* del fichero que se especifique.

– Mediante la función *auto-compression-mode*, *Emacs* puede automáticamente descomprimir un fichero comprimido al visitarlo, y volverlo a comprimir al guardar los cambios.

11. Operaciones con buffers y ventanas

Múltiples buffers Como se ha indicado anteriormente, es común que *Emacs* mantenga varios *buffers* abiertos a lo largo de una sesión; por ejemplo, uno por cada fichero que se esté visitando, uno de mensajes, uno de «completamientos» Cada *buffer* se identifica por su nombre, que puede tener cualquier longitud. En la tabla 11 se resumen las operaciones más importantes con los *buffers*.

12. Modos principales

Con un editor de texto se pueden crear ficheros de muy diverso tipo, desde un texto plano hasta la fuente de un programa escrito en un lenguaje de alto nivel. Es obvio que cada tipo de fichero presenta características específicas que hacen su edición bastante diferente de los demás. *Emacs* exhibirá un comportamiento (ligeramente) distinto en función de qué tipo de fichero se esté editando o, más exactamente, del tipo de «modo mayor» que esté activo.

Existe un modo (llamado *fundamental mode*) que actúa como una suerte de «base» sobre la cual se erigen otros muchos modos que «personalizan» de cierta manera el comportamiento de *Emacs*.[5] Estos modos pueden clasificarse en los siguientes bloques:

- *Modos de programación*: para lenguajes de alto nivel tales como *C*, *Lisp*, *Fortran*, etcétera.
- *Modos de edición de texto*: en este apartado están los modos TEX, *Text*, *Nroff*, etcétera.
- Modos para ser utilizados en *buffers* creados por el propio *Emacs*.

Cada uno de estos «modos mayores» posee a su vez una serie de «dialectos». Así, un dialecto del modo *C* será el modo *C++*; un dialecto del modo TEX será el modo LATEX, etcétera. *Emacs* suele seleccionar el modo (o algún dialecto suyo) en función de la extensión del nombre del fichero y/o de su sintaxis. En cualquier caso, el modo se puede elegir explícitamente con funciones como *lisp-mode*, *tex-mode*... Existe un sinfín de modos, cada uno con sus características peculiares, con lo que aquí solamente se describen algunas cuestiones relativas a la sangría de líneas (*indentation*) y a la edición de textos en lenguajes humanos.

Programación Por lo que respecta a la edición de ficheros fuente para *lenguajes de programación*, pueden hacerse las siguientes consideraciones:

- Es altamente recomendable que a la hora de escribir un programa —especialmente en un lenguaje de alto nivel— se sangren o «indenten» las líneas en función del papel que desempeñen en la estructura del programa. *Emacs* provee algunas funciones que hacen más fácil, sistemático y fiable esta sangría.
- Otra utilidad de *Emacs* en los modos de programación es el llamado *automatic display of matching parentheses*: cuando se escribe un delimitador de cierre (paréntesis, corchete, llave...), el cursor se mueve automáticamente al lugar donde está situado el delimitador de apertura correspondiente. Si se detecta una falta de correspondencia entre el delimitador de apertura y el de cierre, se muestra un mensaje de error.

[5] En muchas ocasiones, esta «personalización» consiste en asociar una determinada combinación de teclas a funciones que en el *modo fundamental* no tienen asignada ninguna, pero que funcionan igualmente ejecutándolas en el *minibuffer*.

- Habida cuenta de la importancia que tiene comentar de forma correcta y estructurada los programas que se escriban, *Emacs* provee una serie de funciones para la manipulación de los comentarios.
- Existe la posibilidad de que, al insertar ciertos caracteres de C ({, }, `#`,...), llamados «eléctricos», se «re-indente» la línea actual y, llegado el caso, se inserte una nueva. Este comportamiento se activa y desactiva con las teclas `C-c C-a` (*c-toggle-auto-state*), y su carácter activo se indica en la *mode line* con los caracteres `\a`.
- Con la característica *hungry delete*, que se activa o desactiva con la combinación `C-c C-d` (*c-toggle-auto-state*), una sola pulsación de Del borra todo el espacio en blanco que preceda a una línea, en lugar de borrar un solo carácter blanco.
- Es posible ejecutar compiladores para lenguajes no interactivos (*i.e.*, lenguajes tales como C, Fortran, etcétera) como «procesos inferiores», direccionando la salida de error a un *buffer* llamado **Compilation**. En dicho *buffer*, situando el punto sobre un error determinado y pulsando Enter, se visualiza la parte del código fuente de donde proviene dicho error. Las funciones que controlan la compilación son *compile* y *kill-compilation*.

Las funciones relacionadas con todas estas utilidades se resumen en la tabla 12.

Lenguajes humanos Se pueden citar al menos cuatro operaciones útiles a la hora de editar este tipo de textos:

- Para *dividir* las líneas de texto por sus articulaciones «naturales» (espacios en blanco o retornos de carro) se emplea el llamado *filling text*: *Emacs* rellenará con espacios en blanco hasta completar el número de caracteres por línea de la pantalla.
- Para que cada «línea» comience con una determinada cadena de caracteres —por ejemplo, con espacios en blanco, para obtener un párrafo sangrado— se emplea el llamado *fill prefix*.
- Es posible ordenar alfabéticamente por líneas o por párrafos.
- Por último, *Emacs* posee una serie de funciones para la conversión entre mayúsculas y minúsculas.

Las funciones más destacables para cada una de estas operaciones se reseñan en la tabla 12.

Abreviaturas Una abreviatura es un conjunto de caracteres que tales que, al insertar tras ellos un espacio, guión, llave, paréntesis..., se «expanden» para dar lugar a otro texto diferente. Si en un texto se escribe a menudo una determinada cadena de caracteres, el uso de abreviaturas ahorrará pulsaciones de teclado. La función «abbrev-mode» activa y desactiva su funcionamiento. Algunas otras funciones relacionadas con el uso de las abreviaturas se resumen en la tabla 12.

CUADRO 13: *Lenguajes de programación*

Tecla	*Descripción*
Tab	Sangra la línea actual de la manera que *Emacs* considere «adecuada», en función de lo que se haya escrito en las líneas anteriores: declaración de función, definición de bucle...
C-j	(*newline-and-indent*) Inserta una nueva línea y la indenta como corresponda.
M-i	(*tab-to-tab-stop*) Sangra desde el punto hasta la siguiente parada de tabulador especificada.[a]
M-;	(*insert-for-comment*) Inserta o alinea un comentario.
C-x ;	(*set-comment-column*) Establece una columna de comentario: el sangrado del comentario será igual al de la posición actual del punto. (*tabify*) Busca secuencias de espacios en blanco en la región y sustituye las secuencias de dos o más espacios por marcas de tabulación, siempre y cuando ello no cambie el sangrado. (*untabify*) Sustituye las marcas de tabulación (de ocho caracteres de anchura por defecto) de la región por el número correspondiente de espacios en blanco, preservando la anchura de todos los sangrados.
C-u C-x ;	(*kill-comment*) Elimina el comentario de la línea actual.
C-M-j	(*indent-new-comment-line*) Inserta un carácter de nueva línea y un delimitador de comentario. (*comment-region*) Añade delimitadores de comentario a todas las líneas de la región.

[a] La forma más cómoda de especificar las paradas de tabulador es con la función *edit-tab-stops*. Las paradas se marcan en la columna que se desee con el carácter `:`. Para que los cambios tengan efecto se pulsa la combinación `C-c C-c`.

CUADRO 14: *Lenguajes humanos*

Tecla	*Descripción*
	(*auto-fill-mode*) Activa o desactiva la separación de líneas por sus articulaciones «naturales».
`M-q`	(*fill-paragraph*) Opera la separación de líneas o el «rellenado» en el párrafo en el que se halla el punto.
`C-x f`	(*set-fill-column*) Permite establecer el número máximo de caracteres por línea de texto. Este valor (por defecto igual a setenta) se almacena con la variable *fill-column*.
	(*fill-region*) «Rellena» cada párrafo de la región.
	(*center-line*) Centra la línea actual.
`C-x`	(*set-fill-prefix*) Establece como prefijo el texto que se encuentre desde la posición del punto hacia atrás hasta el comienzo de la línea.
`M-q`	(*fill-paragraph*) «Rellena» el párrafo en el que se halle el punto empleando el «prefijo de relleno» actual.
`M-l`	(*downcase-word*) Pone en minúsculas la palabra que se halle a continuación del punto.
`M-u`	(*upcase-word*) Pone en mayúsculas la palabra que se halle a continuación del punto.
`M-c`	(*capitalize-word*) Actúa sobre la palabra que se halle a continuación del punto, poniendo en mayúsculas la primera letra.
`C-x C-l`	(*downcase-region*) Pone en minúsculas la región actualmente marcada.
`C-x C-u`	(*upcase-region*) Pone en mayúsculas la región actualmente marcada.
	(*sort-lines*) Clasifica alfabéticamente por líneas.
	(*sort-paragraphs*) Clasifica alfabéticamente por párrafos.

CUADRO 15: *Abreviaturas*

Tecla	*Descripción*
`C-x a g`	(*add-global-abbrev*) Antes de emplear el comando, hay que situar el punto inmediatamente después del texto *significado* de la abreviatura; si son varias palabras, deberá pasarse su número como argumento numérico.
	(*kill-all-abbrevs*) Elimina todas las abreviaturas que se hayan establecido.
	(*list-abbrevs*) Presenta un listado de las abreviaturas establecidas.
	(*edit-abbrevs*) Permite añadir abreviaturas nuevas o modificar algunas de ellas.
	(*write-abbrev-file*) Escribe las abreviaturas definidas en la sesión actual en un fichero, para su uso en otras sesiones.
	(*read-abbrev-file*) Lee el fichero en el que se hayan guardado las abreviaturas, para utilizarlas en la sesión actual.