Referencia rápida de Jed 0.99.16, v. 1.0.7

Guido Gonzato y John. E. Davis

May 2, 2007

Contents

1 Introducción

JED es un editor de texto libre para plataformas UNIX, VMS, MSDOS, OS/2, BeOS, QNX y Win32. Aunque es un potente editor diseñado para programadores, su menú desplegable hace que sea uno de los editores más fáciles de usar. Por tanto, resulta ideal para redactar mensajes de correo electrónico, así como para editar programas complejos en una gran variedad de lenguajes.

JED hace un uso completo de la biblioteca S-Lang, lo que le convierte en un potente lenguaje de macros S-Lang.

JED está escrito por John E. Davis, `davis@space.mit.edu`. Algunas direcciones útiles:

- Página principal de Jed, http://www.jedsoft.org/jed
- Grupo de noticias de S-lang, `news:alt.lang.s-lang`
- Grupo de noticias `news:comp.editors`
- Lista de correo de usuarios de Jed, http://www.jedsoft.org/jed/mailinglists.html

En este manual se asume que JED (versión 0.99.16) está correctamente instalado en el sistema. Para obtener detalles acerca de la instalación, consultar los ficheros `INSTALL*` que se incluyen con la distribución.

Tal como afirma el autor de JED, el número de versión aún es inferior a 1.0 *"debido a la falta de documentación adecuada"*. Esta referencia rápida intenta ser un buen punto de partida, sin olvidar consultar los ficheros `.txt` de la distribución para obtener información adicional.

1.1 Inicio

En función del sistema operativo, el nombre del ejecutable de JED puede ser `jed` (consola), `jed386` (DOS), `xjed` (X11) o `wjed` (Windows).

```
prompt> jed [opciones] [fichero(s)]
```

Las opciones disponibles en línea de comandos son:

```
Opción           Significado
----------------------------
-batch           ejecuta JED en modo batch (no interactivo).
-help            muestra el uso
-hook fn (argv)  ejecuta fn con los parámetros argv
-script          igual que '-batch -l', pero sin salida
-tmp             no hace copia de seguridad del buffer
--ALGO         ejecuta ALGO como función
```

```
-e <emulación>   usa emulación (e.g. emacs, ide, cua...)
-n               no carga el fichero jed.rc (.jedrc)
-g <n>           se desplaza a la línea 'n' del buffer
-l <fich>        carga el fichero 'fich' como código S-Lang.
-f <func>        ejecuta la función 'func' S-Lang
-s <cad>         busca hacia delante la cadena 'cad'
-2               divide la ventana
-i <fich>        inserta <fich> en el buffer actual
```

El directorio definido en la variable de entorno `JED_ROOT` contiene la instalación. Los ficheros de documentación se encuentran en `JED_ROOT/doc`, los ficheros S-Lang en `JED_ROOT/lib`, las páginas info en `JED_ROOT/info` y los binarios en `JED_ROOT/bin`. Es posible mover los binarios a un directorio del sistema estándar.

Los valores comunes para `JED_ROOT` en entornos UNIX son `/usr/local/jed` o `/usr/share/jed`.

`xjed` acepta opciones adicionales:

```
-Display       [-d]     pantalla de ejecución
-Name                   nombre de la instancia
-Geometry               especificaciones geométricas iniciales
-font          [-fn]    tipo de letra por defecto
-background    [-bg]    color de fondo
-foreground    [-fg]    color de primer plano
-Title                  nombre para mostrar en la barra de título
-fgStatus      [-sfg]   color de primer plano de la línea de estado del buffer de xjed
-bgStatus      [-sbg]   color de fondo de la línea de estado del buffer de xjed
-fgRegion      [-rfg]   color de primer plano de una región, definidas por punto y marca
-bgRegion      [-rbg]   color de fondo definidas por punto y marca
-fgCursor      [-cfg]   color de primer plano del cursor de texto
-bgCursor      [-cbg]   color de fondo del cursor de texto
-fgMouse       [-mfg]   color de primer plano del cursor del ratón
-bgMouse       [-mbg]   color de fondo del cursor del ratón
-fgMenu        [-fgm]   color de primer plano para la barra de menús
-bgMenu        [-bgm]   color de primer plano para la barra de menús
-fgMessage     [-fgms]  color de primer plano para los mensajes
-bgMessage     [-bgms]  color de fondo para los mensajes
-fgError       [-fger]  color de primer plano para los mensajes de error
-bgError       [-bger]  color de fondo para los mensajes de error
-fgOperator    [-fgop]  color de primer plano para los operadores (+, -, etc...)
-bgOperator    [-bgop]  color de fondo para los operadores
-fgNumber      [-fgnm]  color de primer para los números
-bgNumber      [-bgnm]  color de fondo para los números
-fgString      [-fgst]  color de primer para las cadenas de texto
-bgString      [-bgst]  color de fondo para las cadenas de texto
-fgComments    [-fgco]  color de primer plano para los comentarios
-bgComments    [-bgco]  color de fondo para los comentarios
-fgKeyword     [-fgkw]  color de primer plano para las palabras reservadas
-bgKeyword     [-bgkw]  color de fondo para las palabras reservadas
-fgKeyword1    [-fgkw1] color de primer plano para las palabras reservadas1
-bgKeyword1    [-bgkw1] color de fondo para las palabras reservadas1
-fgKeyword2    [-fgkw2] color de primer plano para las palabras reservadas2
-bgKeyword2    [-bgkw2] color de fondo para las palabras reservadas2
-fgDelimiter   [-fgde]  color de primer plano para los delimitadores
-bgDelimiter   [-bgde]  color de fondo para los delimitadores
```

```
-fgPreprocess  [-fgpr]  color de primer plano para las líneas del preprocesador
-bgPreprocess  [-bgpr]  color de fondo para las líneas del preprocesador
```

Por ejemplo,

```
xjed -d space:0.0 -font 9x15 -bg blue -fg white
```

iniciará `xjed` usando el servidor `space`, con un tipo de letra 9x15 de color blanco sobre fondo azul.

Se pueden definir valores por defecto para `xjed` insertando las líneas adecuadas en el fichero `.Xdefaults`, e.g.

```
xjed*Geometry: 80x40+100+100
xjed*font: 9x15bold
```

1.2 Obtener ayuda

Hay varias formas de obtener ayuda:

- A través del menú *Help*: hay varias opciones, pruébelas todas;
- Escribiendo `Esc X help` podrá acceder a la ayuda en pantalla de la emulación que esté usando;
- Revisar la documentación local que viene con JED;
- Consultar las fuentes `*.sl` en el directorio `JED_ROOT/lib`.

1.3 Notas para los usuarios de Windows

En primer lugar: en Windows NT/2000/XP JED debe ser instalado por el administrador.

Si no desea o no sabe modificar el entorno para que funcione JED, use el siguiente fichero `.bat`. En este ejemplo, se asumirá que JED está instalado en `C:\TOOLS\JED`:

```
rem  wjed.bat
rem
SET PATH=C:\TOOLS\JED\BIN;%PATH%
SET JED_ROOT=C:\JED\TOOLS
C:\TOOLS\JED\BIN\WJED.EXE %1 %2 %3 %4 %5 %6 %7 %8 %9
```

Cree un acceso directo a este fichero `.bat` en `C:\WINDOWS\SENDTO`. Al hacer click con el botón derecho en un fichero, podrá seleccionar "Wjed" del menú "Enviar a...".

`wjed` resulta algo confuso a la hora de cortar, copiar y pegar. El menú *Edit* no se puede usar para pegar texto, sólo funciona el botón derecho del ratón.

Puede que en modo consola JED para DOS/Windows no inserte caracteres como '#', '@' y en general aquellos caracteres que se obtienen usando la tecla `Alt Gr` en un teclado que no sea de EEUU. Esto puede solucionarse insertando estas líneas en el fichero `.jedrc` (por desgracia, usando otro editor):

```
setkey ("self_insert_cmd", "\e@");
setkey ("self_insert_cmd", "\e#");
setkey ("self_insert_cmd", "\e[");
setkey ("self_insert_cmd", "\e]"); % y así sucesivamente...
```

Actualmente, `wjed` no tiene los menús estándar. Para obtener los mismos menús que JED en modo consola, tendrá que editar el fichero de sistema `os.sl` para quitar el comentario de la línea siguiente:

```
% . "menus" evalfile pop         %  Quitar el comentario para activar los menús de texto
```

1.4 Ficheros de inicio

Cuando JED se inicia, carga un fichero de inicialización, `site.sl`; a continuación, si existen, carga `defaults.sl` y `/etc/jed.conf`; posteriormente intenta cargar el fichero de inicialización de usuario, `$HOME/.jedrc`; si falla, cargará un fichero de sistema, `jed.rc`. Todos estos ficheros están escritos en S-Lang.

`site.sl` es especialmente importante. Además de definir combinaciones de teclas, funciones y variables, también declara *autoloads*. Éstas son asociaciones entre funciones y el fichero `.sl` en el que están definidas. Cuando se necesite una función de éstas, se cargará el correspondiente fichero `.sl`. Esto permite que JED cargue sólo las funciones necesarias, ahorrando memoria durante el inicio.

Puede que los administradores de sistemas quieran proporcionar un `defaults.sl` al nivel del sistema. Es deseable que la emulación del editor (véase más abajo) no se incluya en ficheros globales, para que los usuarios puedan usar las opciones `-n -e`. Más detalles en la sección Modo Batch y Script (§ **??**).

1.5 Personalización

No hay que ser especialmente hábil con S-Lang para adaptar JED a nuestras necesidades. En UNIX, copie el fichero `jed.rc` a `$HOME/.jedrc`, léalo cuidadosamente y familiarícese con las funciones que desea personalizar. Luego, realice los cambios. En Windows, modifique el fichero `jed.rc` directamente, pero haga primero una copia de seguridad.

En muchos casos, lo que hay que hacer es cambiar el valor de las variables globales de JED.

Además de modificar los parámetros existentes, antes o después deseará modificar el fichero `.jedrc`, tal vez añadiendo sus propias funciones de S-Lang. O tal vez sólo añada líneas al final, pero la mejor ubicación suele ser entre las llaves que definen el bloque:

```
if (BATCH == 0)
{
...
}
```

Todo lo que esté entre llaves no se cargará cuando JED se use en modo batch, haciendo que se cargue más rápidamente. Si desea conocer más detalles, consulte Personalización avanzada (§ 8).

1.6 Modos de emulación

Lo primero que deseará elegir es la emulación de su editor favorito. El modo de emulación por defecto es `emacs`; los usuarios de DOS/Windows quizá prefieran el modo CUA, que usa combinaciones de teclas estándar de Windows, o el modo IDE, que es compatible con los editores Wordstar y Borland Turbo Pascal/C++.

Edite el fichero `.jedrc` y elimine los comentarios de las líneas correspondientes (eliminando los caracteres `%`):

```
%  () = evalfile("emacs");        % Combinaciones de teclas de Emacs
%  () = evalfile("edt");          % Modo de emulación EDT
%  () = evalfile ("ide");         % IDE de Borland (consulte también doc/ide-mode.txt)
%  () = evalfile ("brief");       % Combinaciones de Brief (¡sólo MSDOS!)
%  () = evalfile("wordstar");     % obsoleto --- use ide en su lugar)
%  () = evalfile ("cua");         % Combinaciones de teclsa de CUA
```

Cada emulación reserva una tecla para las operaciones extendidas. En modo `emacs`, esta tecla es `Ctrl-C`; en modo IDE, `Ctrl-Z`; en modo CUA, `Ctrl-E`. Es posible cambiar la tecla insertando una línea como ésta en el fichero change `.jedrc`:

```
_Reserved_Key_Prefix = "^Y";
```

A partir de ahora, asumiremos que JED se ejecuta en modo Emacs. Para averiguar las combinaciones de teclas asociadas a una función en otros modos de emulación, seleccione *Help/Where Is Command* y escriba el nombre de la función. Por ejemplo, si consultamos a JED en modo IDE sobre la función `comment_region`, recibirá como respuesta `Ctrl-K ;`.

En lo sucesivo, no se proporcionarán combinaciones de teclas para nombres de funciones, siempre que sea posible. Le corresponde al usuario encontrar las combinaciones de teclas asociadas, si existen.

1.7 Tamaño, tipo de letra y colores de las ventanas

Para cambiar el tamaño y las dimensiones en `xjed`, edite el fichero `.Xdefaults` del directorio del usuario y añada estas dos líneas:

```
xjed*Geometry: 80x35+150+0 ! columns x lines, X and Y offset
xjed*font: 7x14
! or:
! xjed*font: lucidasanstypewriter-14
```

Otros valores válidos para los tipos de letra son 5x7 (pequeño), 6x10, 6x12, 6x13, 7x13, 7x14, 8x13, 8x16, 9x15, 9x18, 10x20, 12x24 y 18x18 (enorme). Use `xlsfonts | less` para obtener una lista con los tipos de letra disponibles.

Para hacer lo mismo en `wjed`, abra el fichero `C:\WINDOWS\WIN.INI` y busque la sección `[WJED]`. Tendrá que modificar las propiedades con líneas como ésta:

```
X=67
```

```
Y=14
Width=696
Height=542
Font=Fixedsys
FontHeight=10
```

Hay varios esquemas de color disponibles; podrá encontrarlos en el menú *Windows/Color Schemes*.

1.8 Extendiendo la biblioteca de Jed

Quizá desee instalar ficheros `.sl` adicionales, aunque normalmente no podrá añadirlos a `JED_ROOT/lib`; sólo el administrador del sistema, puede hacerlo. En este caso, puede crear un directorio para almacenar sus ficheros `.sl`. También tendrá que añadir unas cuantas líneas a `$HOME/.jedrc`.

La configuración que se muestra más abajo fuerza a JED a buscar ficheros `.sl` en `$HOME/mylib/jed` antes de la ubicación estándar. Lo mismo se aplica para el resaltado de patrones y esquemas de color.

```
variable Jed_Home_Library = "~/mylib/jed";
set_jed_library_path (Jed_Home_Library + "," + get_jed_library_path ());
Jed_Highlight_Cache_Dir = Jed_Home_Library;
Jed_Highlight_Cache_Path = Jed_Highlight_Cache_Dir + "," +
                           Jed_Highlight_Cache_Path;
Color_Scheme_Path = Jed_Home_Library + "/colors/" + Color_Scheme_Path;
```

1.9 Mini-Buffer

El *Mini-Buffer* es una línea situada en la parte inferior de la pantalla; gran parte del diálogo entre el usuario y JED se lleva a cabo en este buffer. Por ejemplo, cuando quiera buscar una cadena, JED le pedirá la cadena en el mini-buffer.

Puede introducir comandos en el mini-buffer pulsando `Esc-x` (o `Alt-x`); aparecerá el prompt `M-x`. En este momento, introduzca el nombre de una función. Por ejemplo, para moverse a la parte superior del buffer, escriba `Alt-X bob`.

El mini-buffer también ofrece un acceso directo al intérprete de S-Lang. Para acceder al intérprete, pulse `Ctrl-X Esc` (o `Esc X` en modo IDE) y aparecerá el prompt `S-Lang>` en el mini-buffer. Introduzca cualquier expresión S-Lang válida para que la evalúe el intérprete.

En cualquier momento puede salir del mini-buffer pulsando `Ctrl-G`.

1.9.1 Editar y completar en la línea de comandos

Es posible recuperar los datos previamente introducidos en el mini-buffer usando las teclas de cursor (arriba y abajo). Esto permite usar y editar expresiones previas de forma adecuada y eficaz.

Los comandos que se introduzcan en el mini-buffer se pueden completar usando las teclas `Tab` o `Space`. Esto resultará útil cuando no recuerde el nombre de una función (JED proporciona cientos de funciones) o para alternar entre los nombres de los buffers y los ficheros.

Por ejemplo, supongamos que desea ver todas las funciones que comienzan por 'b'. Introduzca `Alt-X b`, a continuación pulse `Tab`: aparecerá una nueva ventana con una lista de posibles opciones. Ahora pulse `Space` varias veces y observe cómo la línea de comandos se completa automáticamente.

No todas las funciones de JED se pueden completar. Por ejemplo, las funciones definidas en los modos de emulación no se completarán automáticamente; tampoco las funciones definidas en ficheros que aún no se hayan cargado. Más detalles en la sección Carga automática de funciones (§ 8.4).

1.9.2 Nombres de ficheros

Cuando JED solicita el nombre de un fichero, la función para completar funciona como se describe anteriormente. La sintaxis del nombre del fichero depende del sistema operativo. Algunos ejemplos válidos son:

```
C:\JED\LIB\JED.RC
~/progs/jed/ide.sl
dev$user:[davis.jed]vms.c
```

que denotan rutas de ficheros en DOS/Windows, Unix y VMS. Tenga en cuenta que la tilde `~` hace referencia al directorio `HOME` del usuario.

Se permiten comodines. Por ejemplo, para seleccionar uno de los ficheros con extensión `.txt` en el subdirectorio `text`, puede escribir

```
text/*.txt
```

en el mini-buffer; a continuación, pulse `Tab` y `Space` de la forma habitual.

1.10 Buffers, ventanas y línea de estado

JED admite varias ventanas. Cada ventana puede contener el mismo buffer u otro diferente. Normalmente los buffers están asociados a un fichero, aunque no es necesario.

Existe un buffer por defecto, llamado `*scratch*`, que se puede usar para escribir texto que no desee guardar.

En la parte inferior de cada venta se muestra una línea de estado. La línea de estado contiene información, como el número de versión de JED, el nombre del buffer, el modo, etc. En la parte izquierda, la barra de estado comienza con una cadena de guiones. En este área pueden aparecer *indicadores* especiales:

`**` el buffer se ha modificado desde la última vez que se guardó.

`%%` el buffer es de sólo lectura.

`m` indicador de marca establecida; esto significa que se está definiendo una región.

`d` el fichero ha cambiando en el indicador de disco. Esto indica que el fichero asociado al buffer es más reciente que el mismo buffer.

`s` señala el indicador pulsado.

`+` la opción de deshacer está habilitada para el buffer.

`[Macro]` se está definiendo una macro.

`[Narrow]` el buffer se reduce a una región de líneas. Consulte la sección Reducción de regiones (§ 5.3).

Para personalizar la línea de estado puede usar este comando en el fichero `.jedrc`:

```
set_status_line (" Jed %v: %b (%m%n) (%l %c) %t", 1);
```

El significado de los campos es:

`%b` nombre del buffer

`%f` nombre del fichero (sin la parte del directorio)

`%F` nombre del fichero con el directorio

`%v` versión de JED

`%t` hora actual — sólo se usa si la variable DISPLAY_TIME es distinta de cero

`%p` número de línea o porcentaje. Si LINENUMBERS es 2, se expande a "número de línea,número de columna"

`%c` número de columna

`%%` carácter '%' literal

`%m` cadena de modo

`%a` se expande a "abbrev" si está en modo abreviado

`%n` si se reduce el buffer, se expande a "Narrow"

`%o` si está en modo sobrescribir, se expande a "Ovwrt"

`%O` opción de sobrescribir/insertar; igual que %o, pero muestra INS/OVR

`%l` muestra el número de línea actual

`%L` muestra el número de líneas del fichero

1.11 Ficheros de copia de seguridad y de grabación automática

Por defecto, JED crea copias de seguridad añadiendo `~` al nombre del fichero. Esta función se puede desactivar definiendo `No_Backups = 1` en `.jedrc`.

Además, JED guarda automáticamente de forma periódica los buffers. Estos ficheros que se guardan automáticamente tienen el prefijo `#`. El intervalo para guardar automáticamente se puede cambiar estableciendo el valor que queramos para la variable `MAX_HITS`; el valor por defecto es 300 "hits" (pulsaciones de tecla) en el buffer.

Más detalles en la sección Recuperación de ficheros (§ 6.1).

1.12 Modos Batch y Script

JED se puede usar para realizar tareas de edición de texto de forma no interactiva. Por ejemplo,

```
jed -batch -l preparse
```

inicia JED en modo batch, carga y ejecuta el fichero `preparse.sl`. Esto generará alguna salida.

Para usar JED como utilidad de procesamiento de textos no interactiva, deberá usar la opción `-script` seguido del nombre de un fichero S-Lang. El resto de opciones que siguen a `-script` se ignorarán, y no se cargará el fichero `.jedrc` del usuario.

Veamos un ejemplo sencillo. Guarde el código siguiente en el fichero `hello.sl`:

```
% programa S-Lang realmente simple
flush ("Hola mundo");
```

Escriba `jed -script hello.sl`: tal como es de esperar, se muestra `Hello, world!`. Obviamente, los programas reales de S-Lang serán algo más útiles. El scripting hace de JED una alternativa valiosa a otras herramientas como `sed`, que a menudo resultan crípticas de usar.

En sistemas UNIX, puede escribir shell scripts que usen JED como intérprete de comandos. Por ejemplo, guarde este script como `change.sh`:

```
#!/usr/bin/jed -script

# este script sustituye texto en un fichero.
# __argv[0] = /usr/bin/jed
# __argv[1] = -script
# __argv[2] = ./cambiar.sh
# __argv[3] = fichero.txt
# __argv[4] = antiguo
# __argv[5] = nuevo

if (__argc != 6) {
  message ("Uso: ./cambiar.sl nombrefichero antigua_cadena nueva_cadena");
  quit_jed ();
}

() = read_file (__argv[3]);
replace (__argv[4], __argv[5]);
save_buffer ();
quit_jed ();
```

2 Edición básica

`M-Key` significa `Esc-Key` o `Alt-Key`; `^Key` significa `Ctrl-Key`. Estos son los comandos básicos para comenzar:

Operación	Emacs	IDE	CUA
salir de jed	^X^C	^Kx	M-Q
salir línea cmd	^G	^G	^G
suspender	^Z	^Kz	?
	Operaciones con ficheros		
buscar/abrir fich	^X^F	^Ke o F3	^O
insertar fichero	^Xi	^Ki	M-I
guardar	^X^S	^Kd o F2	^S
guardar como	^X^W	^Ks	n/d
cerrar fichero	^Xk	^Kq o M-F3	n/d
change buffer	^Xb	^Kp o F6	n/d
	Movimiento		
retroceder palabra	M-b	^A	M-izquierda
avanzar palabra	M-f	^F	M-derecha
ir a inicio de línea	^A	^Qs	^A
ir al final de línea	^E	^Qd	^E
avanzar página	M-c	^R	^P
retroceder página	^V	^C	^N
inicio de buffer	M-<	^Qr	M-^A
fin del buffer	M->	^Qc	M-^E
ir a línea número	M-x goto_line_cmd	^Qi	M-x goto_line_cmd
	Borrado		
carácter izquierdo	^H o BS	^H o BS	^H o Bs
carácter derecho	^D	M-g	^D
palabra izquierda	M-DEL	M-BS	^-Bs
palabra derecha	M-d	^T	^-Del
línea	^A^K	^Y	^K
hasta fin de línea	^K	^Qy	M-K
deshacer	^Xu	^U	^Z
rehacer ^G^Xu	^G^U	^G^Z	
	Búsqueda y sustitución		
buscar	^S	^Qf	^F
sustituir	M-%	^Qa	^R
repetir búsqueda	n/d	^L	n/d
	Regiones (también conocidas como Bloques)		
inicio	^@ o ^-ESPACIO	^Kb	Shift-arrow
cortar	^W	^Ky	^X
copiar	M-W	^Kh	^C
pegar	^Y	^Kc	^V

Además, todas las emulaciones admiten las combinaciones de teclas siguientes:

^_	deshacer
M-q	formatear párrafo
M-n	reducir párrafo
M-\	eliminar espacio en blanco
M-!	ejecutar comandos en shell
M-$	corrección ortográfica (ispell)
M-u	poner en mayúsculas toda la palabra
M-c	poner la inicial de la palabra en mayúscula
M-.	buscar etiqueta

También debería tener en cuenta estas características importantes:

- JED funciona de forma diferente según el tipo de texto que se esté escribiendo; es decir, admite Modos (§ 7). Puede que algunos caracteres no se muestren siempre como es de esperar y que algunas combinaciones de teclas dejen de funcionar.
- El carácter ‘ (ASCII 96) se usa para insertar el siguiente carácter literalmente. Por ejemplo, ‘ `Esc` insertará un carácter `Esc` (ASCII 27) en el texto. Para imprimir este carácter, escriba ‘ ‘.
- Para insertar el carácter correspondiente a un código ASCII determinado, pulse `Esc`; a continuación, el código ASCII del carácter y luego ‘. Por ejemplo, para insertar una llave de apertura, escriba `Esc 123 ‘`.
- Estas son ”comillas dobles normales” y estas son “comillas tipográficas”. Si visualiza comillas tipográficas pero prefiere comillas dobles normales, escriba ‘ y ". Lo mismo se aplica a las comillas simples.

3 Características principales

¡Hay tantas cosas que decir aquí! Es posible acceder a muchas de estas funciones a través de los menús, de combinaciones de teclas o del minibuffer, escribiendo el nombre de la función adecuada. A continuación se proporciona una explicación para algunas de ellas, aunque no se sigue ningún orden especial:

> menús desplegables; soporte para el ratón; múltiples ventanas/buffers; autocompletar e histórico en minibuffer; gestor de directorios/buffer; ‘modos’ para editar fuentes o lenguajes de marcas como LaTeX o HTML; recuperación de ficheros perdidos; bloqueo de ficheros; carga de ficheros comprimidos; soporte para plegado; soporte para caracteres de 8 bits y teclas inactivas (mute keys); abreviaturas y autocompletar; interfaz para shell y compiladores; copias de seguridad numeradas; soporte para RCS; marcadores; calendario; desplazamiento a y resaltado de paréntesis/llaves/corchetes coincidentes; filtrado de regiones de texto; edición de ficheros binarios; lector de páginas info y man; búsqueda incremental; corrección ortográfica (ispell) de palabras o buffers completos; macros; mail y rmail; búsqueda y sustitución simple y de expresiones regulares; sustitución a través de varios buffers; registros; ordenación de bloques; copiar y pegar rectangular; posiciones de tabulador editables; conversión de tabulaciones y espacios; copiar y pegar al estilo de Windows; reformateo de párrafos; independencia de CR/LF; códigos de teclas; ayuda para VMS; filtrado mime.

En las siguientes secciones sólo se hablará de nombres de funciones; para buscar las combinaciones de teclas, si existen, use *Help/Where Is Command*. Si tiene curiosidad, busque en los ficheros de S-Lang.

4 Edición

4.1 Menús

Los menús desplegables están disponibles en todos los terminales, y se accede a ellos a través de `F10`, usando los cursores y la tecla `Enter`, o directamente con `M-key` o el ratón. Si `Alt-key` no funciona, pruebe con `Esc-key`.

Si desea añadir una lista de los ficheros a los que haya accedido recientemente al menú *File*, inserte esta línea en el fichero `.jedrc`:

```
() = evalfile ("recent.sl");
```

Para personalizar los menús puede editar el fichero `popups.sl`.

4.2 Marcas en el texto

Las *regiones* (bloques) de texto se marcan con `set_mark_command` o con comandos que dependen del modo de emulación; consulte la tabla de la sección Edición básica (§ 2). Una región se define a partir de dos ubicaciones de pantalla denominadas *punto* y *marcar*, que marcan el inicio y el fin de la región.

Muchos comandos operan sobre las regiones, por ejemplo, la eliminación, el filtrado, la conversión y otros. Por ejemplo, `xform_region ('u')` transformará la región a letras mayúsculas.

4.3 Uso del ratón

Hay soporte para ratón en sistemas DOS/Windows, Linux en modo consola, Xjed y JED en varios xterms:

- Pulse el botón *izquierdo* del ratón para mover el cursor a otra ubicación.
- Para copiar una región y pegarla posteriormente varias veces, pulse el

botón *izquierdo* del ratón y arrástrelo hasta el final de la región.

- Para pegar una región previamente copiada, pulse el botón adecuado:
  - *derecho* (consola Linux y `wjed`)
  - *central* (`xjed`)
  - `Alt`-*central* (consola DOS)
- Para cortar una región y pegarla en el buffer, defina una región arrastrando con el botón *derecho* del ratón (consola, `xjed`). A continuación suelte el botón *derecho* y púlselo de nuevo.
- Para pegar desde otra ventana/aplicación en Jed, pulse la tecla *Alt* mientras mantiene pulsado el botón 'derecho" (consola Linux).

Si está usando JED en `rxvt`, `xterm` u otro emulador de terminal, se obtienen las operaciones anteriores:

- Para definir una región, haga click en `Shift`-*izquierda*;
- Para pegar a partir del buffer, mueva el cursor a donde desee pegar y pulse `Shift`-*central*.

En Windows, puede pegar texto desde otras aplicaciones pulsando el botón *derecho*.

Ficheros S-Lang: `mouse.sl`, `mousex.sl` y `mswmouse.sl`

4.4 Reformateo de párrafos

El comando `M-x format_paragraph` reformatea el párrafo actual, es decir, todo el texto hasta la siguiente línea en blanco. Si desea indentar el párrafo, por ejemplo, en 5 posiciones, inicie la línea con 5 espacios, mueva el cursor hasta el primer caracter y luego ejecute el comando.

Si desea obtener párrafos más estrechos, defina la variable `WRAP` en el fichero `.jedrc` o escriba `^X Esc` e introduzca `WRAP=`*nn*, donde *nn* es la columna en la que desea ajustar el texto. Para que el texto no se ajuste, asigne un valor muy alto a `WRAP`.

Para alinear también el margen derecho, escriba `Esc 1 M-x format_paragraph`.

Los modos pueden redefinir los delimitadores de párrafos, por ejemplo, para no reformatear comentarios o construcciones del lenguaje.

4.5 Macros de teclado

TODO: check out

Las *Macros* se usan para definir un conjunto de comandos que se pueden repetir posteriormente. Use `M-x begin_macro`, `M-x end_macro` y `M-x execute_macro` para definir y ejecutar una macro.

Si necesita introducir algún texto durante la definición de una macro, escriba `M-X macro_query`. Aparecerá el prompt `Enter String:` en el minibuffer. Cualquier cadena que se introduzca se insertará en el buffer y continuará el proceso de definición de la macro. Cada vez que se ejecute la macro, JED solicitará que se inserte una cadena.

Si escribe `()=evalfile (■macro■)` en el prompt S-Lang, se activarán funciones adicionales. Para asignar una macro a una tecla, use `M-x macro_assign_macro_to_key`; para una función, `M-x macro_to_function`.

4.6 Repetición de acciones

La secuencia de teclas `Esc n`, donde *n* es un dígito, se usa para repetir un comando *n* veces. Por ejemplo, `Esc 25 x` inserta 25 caracteres 'x'; `Esc 100 M-x execute_macro` ejecuta la última macro 100 veces; etc.

4.7 Rectángulos

Al definir una región, si el puntero y la marca están posicionados en columnas diferentes, se define un *rectángulo*. Algunas funciones permiten realizar acciones en dicho rectángulo.

Puede usar las funciones `kill_rect` para cortar el rectángulo, `open_rect` para insertar un rectángulo vacío, `copy_rect` para copiar, `insert_rect` para pegar y `blank_rect` para rellenar de negro el rectángulo.

4.8 Ordenación de bloques

Supongamos que queremos ordenar un conjunto de líneas alfabética o numéricamente. Comenzamos por definir un rectángulo sobre los datos que deseemos ordenar y a continuación ejecutamos `M-x sort`. Por ejemplo, en el texto siguiente:

```
Fruta:          Cantidad:
```

```
limones              3
naranjas            56
melocotones        175
manzanas           200
peras               37
```

Para ordenar los datos en función del nombre: mueva el puntero hacia la `l` de `lemons`, establezca la marca y mueva el cursor 3 posiciones después de la `s` de `pears`. Acaba de definir un rectángulo; escriba `M-x sort`. Para ordenar los datos por cantidad, defina el punto 2 espacios antes del `3` de la línea `lemons`, mueva el cursor después del `37` de la última línea. El rectángulo habrá ordenado la línea sobre los números.

La ordenación de rectángulos y bloques resultan muy útiles cuando trabaja con varios ficheros de datos de múltiples columnas.

Ficheros S-Lang: `sort.sl`

4.9 Caracteres de 8-bits y teclas inactivas (mute keys)

Aquellos cuyo idioma incluye caracteres acentuados se beneficiarán de `M-x digraph_cmd`, que permite obtener dichos caracteres. Para escribir caracteres especiales directamente, incluya una línea como esta en el fichero `.jedrc` local:

```
mute_set_mute_keys ("'`~^");
```

Al pulsar una tecla de comilla simple, tilde o ángulo, seguido del carácter de acento, se insertará en el buffer un caracter acentuado. Atención: esto puede hacer que insertar comillas simples sea algo engorroso en algunas ocasiones, y no funcionará en todos los modos. Si obtiene un funcionamiento inesperado, escriba `M-x no_mode`, inserte el caracter que da problemas y vuelva al modo que estaba usando.

Ficheros S-Lang: `mutekeys.sl`

4.10 Abreviaturas y función de completar automáticamente

Las *abreviaturas* dependen del modo, y funcionan de la siguiente manera: queremos que "PS" se expanda a "PostScript" en todos los ficheros en modo texto; entonces, pasamos al modo de abreviatura; a partir de aquí, cada vez que escribamos "PS" se expandirá a "PostScript". La abreviaturas se guardan en un fichero, normalmente `$HOME/.abbrevs.sl`.

Si desea definir sus propias abreviaturas, edite el fichero o escriba `M-x define_abbreviation`. Se mostrarán las palabras para las que desee definir una abreviatura y dicha abreviatura. A continuación, escriba `M-x save_abbrevs` para guardar las abreviaturas en el fichero y `M-x abbrev_mode` para activarlas.

Para que las abreviaturas se carguen al inicio, consulte el ejemplo de la sección Hooks (§ 8.5).

La función de *completado* es la expansión automática de una palabra escrita parcialmente. Si el texto contiene la palabra 'frobnication', al escribir 'fro' seguido de `M-/` (`M-x dabbrev`) expandirá la palabra completa, o alternará entre todas las posibles posibilidades.

Ficheros S-Lang: `abbrev.sl`, `dabbrev.sl`

4.11 Ispell

JED usa `ispell` para revisar la ortografía. Si no está seguro de cómo se escribe una palabra, sitúe el cursor sobre ella y escriba `M-x ispell`. Si la palabra es incorrecta, se le ofrecerán varias alternativas.

Ficheros S-Lang: `ispell.sl`

4.12 Registros

Los *registros* son 128 "zonas de memoria" adicionales. Seleccione una región de texto, cópiela a un registro con |M-x reg_copy_to_register|. El sistema solicitará una etiqueta para asociarla al registro. Para pegar desde un registro, escriba |M-x reg_insert_register|; para ver todos los registros activos, use `M-x register_mode`.

Ficheros S-Lang: `register.sl`

4.13 Tabulador

Por defecto, la tecla `Tab` se usa en muchos modos para indentar el texto, aunque es posible modificar su definición en `.jedrc` para que inserta un carácter tabulador real (ASCII 9). La variable `TAB_DEFAULT` e specifica el tamaño del tabulador; para editar texto al estilo de un procesador de textos, use `M-x edit_tab_stops`.

`M-x untab` opera sobre una región y convierte tabulaciones a espacios; `Esc 1 M-x untab` convierte espacios a tabulaciones.

Ficheros S-Lang: `tab.sl`, `untab.sl`

TODO: more explanations.

4.14 Expresiones regulares

Es posible realizar búsquedas y sustituciones con expresiones regulares (ER). La biblioteca S-Lang admite la siguientes ER estándar:

```
.                 Busca cualquier caracter salvo una nueva línea
*                 Busca cero o más apariciones de la ER anterior
+                 Busca una o más apariciones de la ER anterior
?                 Busca cero o una aparición de la ER anterior
^                 Busca el inicio de línea
$                 Busca el final de línea
[ ... ]           Busca cualquier caracter simple que esté entre esos corchetes.
                  Por ejemplo, [-02468] busca '-' o cualquier número par
                  y [-0-9a-z] busca '-' y cualquier dígito entre 0 y 9, así
                  como letras de la a a la z.
\<                Encuentra el inicio de una palabra.
\>                Encuentra el final de una palabra.
\( ... \)
\1, \2, ..., \9   Encuentra la expresión que concuerda con nth \( ... \)
```

Además, también se admiten la siguientes extensiones:

```
\c              activa distinción entre mayúsculas/minúsculas (por defecto)
\C              desactiva distinción entre mayúsculas/minúsculas
\d              encuentra cualquier dígito
\e              encuentra el caracter ESC
```

Por ejemplo, para sustituir `"un texto"` por `'otro texto'` (atención a las comillas dobles y simples), tendrá que buscar `"\([a-zA-Z ]*\)"` y sustituirlo por `'\1'`.

La ER encontrada no funciona para múltiples líneas. Es más, las ER de JED's difieren de las de `egrep` en los aspectos siguientes:

- el operador OR | no está soportado;
- los operadores de agrupación `\(` y `\)` no se usan para agrupar ER para formar una ER simple. Por tanto, una expresión como `\(hello\)*` no es un patrón para buscar cero o más apariciones de `hello`, aunque sí lo es, por ejemplo, en `egrep`.

Ficheros S-Lang: `regexp.sl`

4.15 Búsqueda

`M-x occur` busca apariciones de expresiones regulares en el buffer actual. Se creará un nuevo buffer, denominado `*occur*`, que contendrá las líneas en que se encontró regexp. Desplácese al buffer `*occur*`, mueva el cursor a una línea y pulse 'g'. En el otro buffer, el cursor se ubicará en la línea correspondiente.

Ficheros S-Lang: `occur.sl`

5 Acerca de los buffers

5.1 División de ventanas

La función `split_window` divide la ventana actual en dos; cada ventana puede mostrar buffers diferentes, o partes diferentes del mismo buffer. Esto resulta útil cuando se necesita comparar distintas partes de un fichero.

Use `M-x other_window` para mover el cursor de una ventana a otra y `M-x one_window` para visualizar sólo una ventana en pantalla.

5.2 Plegado/desplegado de código

El *plegado* es una técnica para ocultar partes de un documento, y se activa mediante la función `folding_mode`. En este modo están disponibles las siguientes funciones y combinaciones de teclas:

```
Función                 Combinación
-----------------------------------
fold_whole_buffer       ^C^W
fold_enter_fold         ^C>
fold_exit_fold          ^C<
fold_open_buffer        ^C^O
```

```
fold_fold_region        ^C^F
fold_open_fold          ^C^S
fold_close_fold         ^C^X
fold_search_forward     ^Cf
fold_search_backward    ^Cb
```

Ficheros S-Lang: `folding.sl`

5.3 Reducción de regiones

Una vez definida una región, es posible restringir la edición en dicha región escribiendo `M-x narrow`. Sólo será visible el texto de la región. Para que el resto del texto sea nuevamente accesible, escriba `M-x widen`.

5.4 Compilación

Supongamos que estamos escribiendo código en C o Fortran. Una vez que el programa haya finalizado, no tendrá que salir de JED para compilarlo. Escriba `M-x compile` e introduzca un comando adecuado (e.g., `gcc -o foo foo.c`). Si el código fuente contiene errores, se resaltarán en el buffer `*compile*`; `M-x compile_parse_errors` y `M-x compile_previous_error` moverán el cursor a la ubicación del error siguiente o anterior del buffer de código fuente.

Ficheros S-Lang: `compile.sl`, `acompile.sl`

5.5 Etiquetas

Si carga un fichero `tags` generado con el programa `ctags`, podrá ver la primera aparición de la etiqueta que está bajo el (o una etiqueta que especifique) escribiendo `M-x find_tag`.

Fichero S-Lang: `ctags.sl`

5.6 Comentarios en el código

Cuando escriba código, podrá comentar regiones de texto con `M-x comment_region`. La función `uncomment_region` realiza la acción contraria.

S-Lang file: `comment.sl`

5.7 Editor del buffer

Modo diseñado para mantener y editar los buffers abiertos; escriba `M-x bufed`. Se creará un nuevo buffer llamado `*BufferList*`. Escriba 'h' o '?' para obtener ayuda.

Ficheros S-Lang: `bufed.sl`

5.8 Mail/Rmail (sólo en Unix)

Escriba `M-x mail` para escribir un mensaje de correo electrónico; `M-x mail_send` para enviarlo. Además, si se incluye el ejecutable `getmail` en la distribución, Jed actuará como MUA y recuperará el correo de la cola de mensajes estándar. A partir de ahí, cómo usarlo debería resultar obvio. Teclas principales:

```
Tecla      Acción
-------------------------
SPACE      avanzar o seleccionar mensaje
DELETE     retroceder a mensaje
DOWN       moverse al mensaje siguiente (use la barra espaciadora para seleccionarlo)
UP         moverse al mensaje anterior
N          moverse al siguiente mensaje no eliminado
P          moverse al anterior mensaje no eliminado
D          marcar mensaje para eliminación
X          eliminar mensajes marcados y cambiar el orden de la carpetas
G          obtener correo nuevo
Esc 1 G    solicitar un buzón de correo y obtener el correo de él
Q          salir de esta carpeta al nivel superior (índice de carpetas)
T          alternar entre cabeceras. Por defecto, jed ocultará la mayor parte
           de cabeceras. Use esta tecla para hacerlas visibles.
O          enviar mensaje a una carpeta diferente. Se creará una
           si es necesario.
F          reenviar mensaje
R          responder
M          correo
```

Ficheros S-Lang: `mail.sl`, `mailalias.sl`, `rmail.sl` y `sendmail.sl`

5.9 Lectores de páginas man e info

Los usuarios de Unix querrán usar `man` y `info`; Escriba `M-x unix_man` y `M-x info_mode` para activarlos. Las páginas man se cargarán en modo Most (§ **??**).

Ficheros S-Lang: `man.sl`, `info.sl`

5.10 Consejos breves

- `M-x do_shell_cmd` ejecuta un comando de shell y pone la salida resultante en un buffer (`shell.sl`).
- La función de `M-x toggle_readonly` resulta obvia: pone un buffer como de sólo lectura.
- Sitúe el cursor en cualquier paréntesis y escriba `M-x goto_match` para mover el cursor al paréntesis correspondiente.
- `M-x cal` muestra un calendario en un buffer (`cal.sl`).
- `M-x isearch_forward` y `M-x isearch_backward` realizan operaciones de búsqueda incremental (`isearch.sl`).
- `M-x replace_across_buffer_files` realiza una sustitución en todos los buffers abiertos (`replace.sl`).
- `M-x trim_whitespace` elimina todos los espacios que siguen al último carácter (que no sea una espacio) de una línea.
- `M-x capitalize_word` convierte una letra a mayúscula.

6 Acerca de los ficheros

6.1 Recuperación de ficheros

JED puede recuperar una sesión que se haya interrumpido, como una caída del sistema o la desconexión de un terminal. En estos casos, JED intenta guardar automáticamente los buffers. Los ficheros se guardan con `#` al principio y al final, por ejemplo: `#NombreFichero.txt#`.

Cuando JED busque un fichero, comprobará si hay un fichero que se ha guardado automáticamente, así como la fecha/hora del fichero. Si las fechas demuestran que el fichero que se ha guardado automáticamente es más reciente, aparecerá un mensaje de alerta en el mini-buffer. Para recuperar el fichero, escriba `M-x recover_file`.

6.2 Edición de archivos comprimidos

La función `auto_compression_mode` activa o desactiva esta función. Cuando está activada, los ficheros cuyos nombres terminan con `.gz`, `.Z` o `.bz2` se descomprimirán automáticamente al abrirse, y se descomprimirán también automáticamente cuando se cierren. Para activar esta función por defecto, incluya la línea siguiente en `.jedrc`:

```
auto_compression_mode ();
```

Fichero S-Lang: `compress.sl`

6.3 Bloqueo de ficheros (sólo en Unix)

Editar el mismo fichero en múltiples sesiones de JED es potencialmente negativo. El bloqueo de ficheros evita que una sesión de JED modifique un fichero que ya se está editando en otra sesión. Cuando se modifique un fichero, JED intentará bloquear dicho fichero. Cuando ya no se modifique más (bien porque se guarde en disco o se descarten los cambios mediante la opción de deshacer), JED lo desbloqueará.

6.4 Fin de línea

En DOS/Windows, las líneas de texto finalizan con los dos caracteres `^M^J` (ASCII 13 + ASCII 10), en UNIX con `^J` y en Macintosh con `^M`. Esta diferencia hace que la edición de ficheros de texto UNIX en DOS y viceversa resulte especialmente irritante. Jed soluciona este problema automáticamente durante la lectura de un fichero. En UNIX, una 'C' en la línea de estado indica que el estilo de fin de línea de DOS está activado; en DOS/Windows, la 'L' indica el fin de línea estilo UNIX. Puede cambiar el estilo de fin de línea con el comando `M-x toggle_crmode`.

Los ficheros de texto Macintosh se cargarán con una única línea. Para corregirlo, sustituya `^M` (escríbalo literalmente) por `^J`.

Si desea que JED guarde siempre los ficheros con `^M^J`, añada esta línea a `.jedrc`:

```
setbuf_info (getbuf_info () | 0x400);
```

6.5 Múltiples versiones

Además de la copia de seguridad del fichero en uso, es posible tener copias numeradas que se crean cada vez que se guarda un buffer. Estas copias tiene un sufijo del estilo `.~3~` (tercera copia de seguridad).

Para activar las múltiples versiones por defect, añada la entrada `backups_on ();` al fichero `.jedrc`. Para desactivar esta función, escriba `M-x backups_off`.

Fichero S-Lang: `backups.sl`

6.6 Interfaz para RCS y SCCS

JED proporciona acceso al sistema de versiones RCS. `M-x rcs_open_file` y `M-x rcs_read_log` solicitan el nombre de un fichero en RCS, como `textfile.txt,v` o `RCS/textfile.txt,v`. Para añadir o extraer un fichero del sistema de versiones, use `M-x rcs_check_in_and_out`. Cuando se extrae un fichero, pasa a estar protegido contra escritura a menos que lo incluya de nuevo.

Se proporcionan comandos similares para SCCS: `M-x sccs_open_file` y `M-x sccs_check_in_and_out`.

Ficheros de S-Lang: `rcs.sl`, `sccs.sl`

6.7 Editor de directorios

Este modo está diseñado para mantener y editar un directorio y los ficheros que contiene. Escriba `M-x dired`; se solicitará que introduzca un nombre de directorio y se creará un nuevo buffer llamado `*dired*`. Escriba 'h' para obtener ayuda acerca de los comandos disponibles.

Fichero S-Lang: `dired.sl`

7 Modos

JED implementa *modos* para facilitar la edición del código fuente o de lenguajes de marcas como LaTeX o HTML. Por lo general, los modos definen un esquema de *resaltado de sintaxis*, proporcionan "indentación automática *y funciones cuyo comportamiento depende de* variables". En algunos casos, tienen una entrada en el menú de modos y, a menudo, proporcionan un *hook* para facilitar la personalización. Es posible cambiar la combinación de teclas por defecto y los valores por defecto de las variables del fichero `.jedrc`.

Por ejemplo, tomemos el modo C. Al editar un fichero con extensión `.c`, JED detecta que es un fichero en lenguaje C y activa el modo C. A medida que escriba el programa, se resaltarán los elementos sintácticos. Algunas funciones específicas del modo están disponibles en el menú de modo, junto con nuevas combinaciones de teclas. Estas teclas pueden cambiarse escribiendo una función llamada `c_mode_hook` en el fichero `.jedrc` personal.

Normalmente, JED reconoce el modo asociado a un fichero por su extensión. Puede hacer que JED edite un fichero usando un modo concreto añadiendo lo siguiente en la primera línea:

```
-*- modo_del_usuario -*-
```

Por ejemplo, si guarda ficheros LaTeX sin la extensión `.tex`, pero desea editarlos en modo `latex`, escriba `% -*- latex -*-` en la primera línea.

Puede que haya notado que los modos no son muy coherentes entre sí; es decir, acciones similares a menudo tienen combinaciones de teclas muy diferentes. Está previsto que esto se modifique en un futuro.

En JED 0.99.16, los modos disponibles y sus funciones se muestran en la tabla siguiente:

Modo	Resaltado de Sintaxis	Indenta	Fuente	Comenta	Otro
bibtex	s	s	s	n	n
c	s	s	n/d	s	s
c++	s	s	n/d	s	s
dcl	s	s	n/d	s	s
docbook	s	s	s	s	s
fortran 77	s	s	n/d	s	s
fortran 90	s	s	n/d	s	s
html	s	n	s	s	s
idl	s	s	n/d	n	s
java	s	s	n/d	s	s
latex	s	s	s	s	s
php	s	s	n/d	s	s
tex	s	n	n	n	s
lisp	s	s	n/d	n	n
maple	?	?	n/d	?	s
matlab	s	s	n/d	s	s
nroff	s	n	n	n	n
perl	s	n	n/d	n	n
postscript	s	n	n/d	n	n
python	s	s	n/d	s	s
docbook	s	n	s	s	s
S-Lang	s	s	n/d	s	s
spice	s	n	n/d	n	n
tcl	s	s	n/d	n	n
tiasm	s	n	n/d	n	n
pascal	s	s	n/d	n	s
verilog	s	n	n/d	n	n
vhdl	s	n	n/d	n	n

Los modos suelen proporcionar nuevas funciones y combinaciones de teclas. Además, casi todos los modos admiten hooks, que se explican en la sección siguiente.

7.1 Resaltado de sintaxis

JED proporciona dos tipos de resaltado de sintaxis: normal y DFA. Este último está basado en expresiones regulares, requiere más memoria y ralentiza el inicio de JED.

Cuando un modo define el resaltado DFA, puede resultar conveniente crear una *caché DFA*; esto soluciona el problema de la demora en el inicio. El comando

```
jed -batch -l preparse
```

crea cachés DFA para todos los modos listados en `preparse.sl`.

7.2 Modo para C

Éste es un modo diseñado para la edición de ficheros en lenguaje C. Después de cargar el modo, se llama a `c_mode_hook`, si existe. Las funciones que afectan a este módulo son:

```
Función               Combinación por defecto
-------------------------------------------
c_insert_bra          {
c_insert_ket          }
newline_and_indent    RETURN
indent_line           Tab
goto_match            ^-\
c_make_comment        Esc ;
c_format_Paragraph    Esc q
c_top_of_function     Esc ^A
c_end_of_function     Esc ^E
c_mark_function       Esc ^H
```

Las variables que afectan a la indentación son:

```
C_INDENT             % indentación de líneas depués de '{'
C_BRACE              % cuánto indentar '{' if on a line by itself
C_BRA_NEWLINE        % '{' on a line by itself?
C_CONTINUED_OFFSET   % indentación de instrucciones que continúan en la
línea siguiente
C_Colon_Offset       % indentación de sentencias 'case'
C_Class_Offset       % indentación de miembros en la declaración de una clase
```

Es posible fijar un estilo de indentación predefinido usando `c_set_style` y alguno de los siguientes: `gnu linux jed bsd k&r`; e.g.

```
c_set_style ("gnu");
```

Si lo desea, puede crear un estilo propio usando una instrucción como:

```
(C_INDENT, C_BRACE, C_BRA_NEWLINE, C_CONTINUED_OFFSET,
  C_Colon_Offset, C_Class_Offset) = (2,0,0,2,0,2);
```

Ficheros S-Lang: `cmisc.sl`

7.3 Modo para Fortran

Este modo está diseñado para editar ficheros en lenguaje Fortran. Tras cargar el módulo, se llama a `fortran_hook`, si existe. Algunas funciones útiles:

```
Función                       Combinación por defecto
-----------------------------------------------------
fortran_continue_newline      Esc RETURN
fortran_comment               Esc ;
fortran_uncomment             Esc :
fortran_electric_label        0-9
```

```
fortran_next_statement          ^C^N
fortran_previous_statement      ^C^P
fortran_ruler                   ^C^R
fortran_beg_of_subprogram       Esc ^A
fortran_end_of_subprogram       Esc ^E
fortran_mark_subprogram         Esc ^H
```

Algunas variables:

```
Fortran_Continue_Char
Fortran_Comment_String
Fortran_Indent_Amount
```

Ficheros S-Lang: `fortran.sl`

7.4 Modo para Latex

Esto modo está diseñado para editar ficheros LaTeX. Después de cargar el modo, se llama a `latex_mode_hook`, si existe. Además, si se define la tabla de abreviaturas `latex`, se usará dicha tabla. Algunas funciones útiles:

```
Función                     Combinación por defecto
-------------------------------------------------
tex_insert_braces           ^C{
tex_font                    ^C^F
latex_environment           ^C^E
latex_section               ^C^S
latex_close_environment     ^C]
latex_insert_item           ^C^J
tex_comment_region          ^C;
tex_uncomment_region        ^C:
tex_comment_Paragraph       ^C%
tex_mark_environment        ^C.
tex_mark_section            ^C*
latex_toggle_math_mode      ^C~
tex_insert_macro            ^C^M
tex_complete_symbol         Esc Tab
tex_complete_symbol         Esc Tab
latex_help                  ^Ci
latex_indent_next_line      ^J
latex_indent_region         ^C^Q^R
latex_indent_section        ^C^Q^S
latex_indent_environment    ^C^Q^E
```

Ficheros S-Lang: `texcom.sl`, `latex.sl`, `latex209.sl`, `ltx-math.sl`.

7.5 Modo para Tex

Este modo está diseñado para editar ficheros TeX y LaTeX. Cuando se carga el modo tex, se llama a `tex_mode_hook`, si existe. Algunas combinaciones útiles:

```
Función              Combinación por defecto
-------------------------------------------
tex_insert_quote     "
tex_insert_quote     ’
tex_blink_dollar     $
tex_ldots            .
```

Ficheros S-Lang: `texcom.sl`

7.6 Modo para HTML

Este modo está diseñado para editar ficheros HTML. Se llama a `html_mode_hook`, si existe.

Si se define una región (i.e., si se establece una marca), se insertan varias etiquetas HTML alrededor de la región, e.g. <B> y </B>.

Si la variable `HTMLModeWraps` tiene el valor 1, este modo ajustará la línea (al igual que `text_mode`); de lo contrario, no la ajustará (al igual que `no_mode`).

Las combinaciones de teclas se agrupan según la función:

```
Tecla     Funciones
-------------------
^CA...    Anclas
^CD...    Listas de definición
^CF...    Formularios
^CH...    Cabeceras, tipo de documento, etc.
^CI...    Imágenes
^CL...    Listas
^CP...    Estilos de párrafo, etc.
^CC...    Estilos de caracteres
```

Además, se definen algunos comandos especiales de movimiento y caracteres varios:

```
Tecla   Acción
--------------
^C^B    ignorar hasta el principio o la anterior etiqueta HTML
^C^F    ignorar hasta el final de la siguiente etiqueta HTML
^C^N    marcar la etiqueta siguiente HTML desde ’<’ hasta ’>’
^C^P    marcar la etiqueta anterior HTML desde ’<’ hasta ’>’
^C&     insertar texto HTML para ’&’
^C>     insertar texto HTML para ’>’
^C<     insertar texto HTML para ’<’
^CC     insertar comentario HTML (alrededor de una región, si está marcada)
```

Ficheros S-Lang: `html.sl`

7.7 Modo para Docbook SGML

Este modo está diseñado para editar ficheros Docbook SGML. Se llama al hook `sgml_mode_hook`, si existe.

Si se define una región (i.e., si se establece una marca), muchas etiquetas SGML se insertarán alrededor de la región, e.g. <Emphasis> y </Emphasis>.

Si la variable `WRAP_INDENTS` tiene el valor 1, este modo indentará el texto. Además, `SGML_INDENT` contiene el número de caracteres de indentación (por defecto es 2).

Todas las etiquetas se insertan a través del menú de modos. Además, se definen algunos comandos espaciales de movimiento y caracteres:

```
Tecla   Acción
--------------
^C^B    skip to beginning of prior SGML tag
^C^F    skip to end of next SGML tag
^CP     insertar <Para>
^C&     insertar texto SGML para '&'
^C$     insertar texto SGML para '$'
^C>     insertar texto SGML para '>'
^C<     insertar texto SGML para '<'
^C.     insertar texto SGML para '...'
^C;     insertar comentario SGML (alrededor de una región, si está marcada)
```

Ficheros S-Lang: `docbook.sl`

7.8 Modo Most

Este es un modo simple usado para la navegación de ficheros de texto, que se abrirán como de sólo lectura. Pulse `Space` para avanzar una página, `Bs` para volver, `/` para buscar hacia delante, y `?` para buscar hacia atrás. Pulse `h` para obtener un breve mensaje de ayuda.

Ficheros S-Lang: `most.sl`

8 Personalización avanzada

JED se puede personalizar totalmente escribiendo código S-Lang en `.jedrc` y/o ficheros externos. Puede comenzar con S-Lang simplemente estudiando los ficheros `.sl` de la distribución de JED. La documentación completa está disponible en el paquete de fuentes de S-Lang.

8.1 Principales variables globales

EN PREPARACIÓN

8.2 Definición de combinaciones de teclas

Consideremos el código S-Lang siguiente:

```
unsetkey (^H);
unsetkey (^N);
setkey ("bol", "^H");
setkey (" Guido Gonzato, Ph.D.", "^N");
```

Las dos primeras líneas eliminan las definiciones previas de `^H` y `^N` y posteriormente se asocia `^H` con la función `bol`. El efecto de `^N` es insertar el texto que se pasa como primer argumento de `setkey`. Hay que tener que en cuenta que comienza con un espacio en blanco, lo que indica a JED que lo que sigue no es un nombre de función S-Lang.

8.3 Carga de ficheros `.sl` adicionales

Escribir mucho código en `.jedrc` no es práctico; un método mejor es escribirlo es un fichero `.sl` externo. Continuando con el ejemplo anterior, supongamos que desea escribir definiciones para `mykeys.sl`. Añada este fichero a la biblioteca de JED y esta línea a `.jedrc`:

```
() = evalfile ("mykeys");
```

8.4 Carga automática de funciones

A veces, usar `evalfile` no es el mejor método de cargar funciones. Supongamos que desea *ocasionalmente* usar las funciones `fun1`, `fun2` y `fun3` definidas en el fichero `functions.sl`. `evalfile` cargará un fichero dado incluso si sus funciones no se usan realmente, lo cual es un mal uso de los recursos del computador.

Otra forma de cargar funciones es a través de la función `autoload`. `autoload` se usa para declarar una función para el intérprete, e indicar que se debería cargar desde un fichero cuando se use realmente.

Por ejemplo, al incluir estas líneas en `.jedrc`:

```
autoload ("fun1", "functions");
autoload ("fun2", "functions");
autoload ("fun3", "functions");
```

JED cargará `functions.sl` cuando se requieran agunas de sus funciones.

8.5 Hooks

Un *hook* es una función S-Lang definida por el usuario que se puede usar para extender el editor o para modificar su comportamiento bajo ciertas circunstancias. Hay dos tipos de hooks que varían en función de si el hook se llama "internamente" desde el editor mediante el código C o si se llama desde otra función S-Lang.

Los hooks se pueden subdividir posteriormente en función de su alcance. Se pueden aplicar a todos los buffers (globalmente), sólo a los buffers que comparten el mismo modo (modal) o a un único buffer (local). Consulte `hooks.txt`.

Veamos un ejemplo en la función siguiente:

```
define latex_mode_hook ()
{
  set_abbrev_mode (1); % usa abreviaturas específicas de LaTeX
  if ( () = abbrev_table_p ("LaTeX") )
    use_abbrev_table ("LaTeX");
  local_setkey (" \\‘a", "à"); % si pulsa à obtendrá \’a
  local_setkey (" \\‘e", "è");
  local_setkey (" \\’e", "é");
  local_setkey (" \\‘\\i{}", "ì");
  local_setkey (" \\‘o", "ò");
  local_setkey (" \\‘u", "ù");
}
```

Este hook se invoca cada vez que se entra en modo `latex`. Carga abreviaturas específicas de LaTeX y permite que el usuario escriba caracteres acentuados, expandiéndolos a la secuencia de teclas correcta.

---

9 Fin

Esta referencia rápida está escrita por Guido Gonzato, Ph.D. `guido dot gonzato at poste dot it`, con la colaboración de John E. Davis.

¡Muchas gracias a John por desarrollar Jed!

Póngase en contacto conmigo si cree que la información de este documento no es exacta o no está clara.