6. OPTIONS

Options are a primary means of configuring the appearance and behavior of elvis. They are set via the :set command, or the :let command. The options' values are examined directly by elvis internally, and can also be displayed via :set, or in an expression. The following tables list the names, type, group, and description of each option. One table lists all options alphabetically, and the other breaks list down into groups of related options. I recommend the latter, since there are *a lot* of options.

Most options have two *names* -- a short name that is easy to type in, and a longer descriptive name. You can type in either name; they work equivalently. Elvis always outputs the longer name when it is listing values.

Each option accepts a specific *type* of value. The most common types are **boolean**, **number**, **string**, and **one-of**, but some options have weird types.

Each option serves as an attribute of something. The *group* of an option designates what it is an attribute of. For example, the "filename" option is an attribute of buffers; when you switch to a different buffer, it will have a different value for the "filename" option. Other options are attributes of windows, or display modes, etc. Here's a complete list:

GROUP	DESCRIPTION
buf	Attributes of buffers
win	Attributes of windows
syntax	Attributes of the "syntax" display mode
x11	Attributes of the "x11" user interface
tcap	Attributes of the "termcap" user interface
windows	Attributes of the "windows" user interface
win32	User interface attributes for the Win32 port
global	Global options
lp	Printing options
user	User variables a - z (Global, useful in ex scripts)

You don't need to know an option's group to set that option. You can output the values of all options in a group by passing the group name followed by a question mark to the :set command. The following example outputs all of the attributes of the current buffer:

```
:set buf?
```

6.1 Options, grouped by function

* 6.1.1 Options that relate the buffer to a file

* 6.1.2 Statistics about a buffer
* 6.1.3 Options that affect movement commands
* 6.1.4 Options that affect input mode
* 6.1.5 Ex options
* 6.1.6 Window statistics
* 6.1.7 Options affecting the appearance of text
* 6.1.8 Options for a particular display mode
* 6.1.9 Messages
* 6.1.10 Words
* 6.1.11 Options for a particular user interface
* 6.1.12 Regular expression options
* 6.1.13 Tag options
* 6.1.14 Window update parameters
* 6.1.15 Cache options
* 6.1.16 Options that describe the system
* 6.1.17 External programs
* 6.1.18 Directory names
* 6.1.19 Initialization options
* 6.1.20 Keyboard map options
* 6.1.21 Printing options
* 6.1.22 Previous arguments
* 6.1.23 Unsupported options
* 6.1.24 User variables

6.1.1 Options that relate the buffer to a file

OPTION NAMES	TYPE	GROUP	DESCRIPTION
filename, file	String	buf	name of file in buffer
bufname, buffer	String	buf	name of buffer
bufid, bufferid	Number	buf	ID number of user buffer
retain, ret	Boolean	buf	keep buffer in session file
modified, mod	Boolean	buf	buffer differs from file
edited, samename	Boolean	buf	buffer loaded from filename
newfile, new	Boolean	buf	filename doesn't exist yet
readonly, ro	Boolean	buf	don't overwrite filename
defaultreadonly, dro	Boolean	global	assume all files readonly
locked, lock	Boolean	win	prevent any alterations
autowrite, aw	Boolean	global	save file before switching
writeany, wa	Boolean	global	don't warn of existing file
backup, bk	Boolean	global	make *.bak file before write
undolevels, ul	Number	buf	number of undoable commands
beautify, bf	Boolean	global	strip ctrl chars from files

The *filename* option stores the name of the text file whose text was initially loaded into the buffer. If no file name is known (e.g., for an internal buffer or a new, untitled buffer) then this will be an empty string. The :file command can be used to change the filename. Also, the filename is set automatically when you write the buffer out, if it had no filename before.

The *bufname* option stores the name of the buffer. Usually this will be the same as the filename, but it can be different. Every buffer has a bufname, even if it doesn't have a filename. The name of a buffer can be changed via the :buffer command.

For user buffers, the *bufid* option stores a unique id number for each buffer. Anyplace where you can use the (*name*) notation to specify a buffer, you can also use (*n*) as an abbreviation for the buffer whose bufid=*n*. Also, for filenames you can use #*n* for the filename of the buffer whose bufid=*n*.

The *retain* option indicates whether the buffer is intended to survive past the end of this elvis process. If this option is true and the tempsession option is false (":set retain notempsession") then elvis will allow you to exit even if this buffer hasn't been saved since its last change. When you restart the session, the buffer will still exist with all its changed text intact. By default, the retain option is false (":set noretain") because that mimics traditional vi behavior.

The *modified* option indicates whether the buffer has been modified since the last time it was written out completely.

The *edited* option indicates whether the filename option has been modified since the last time it was written out. If this option is false, elvis will be more cautious about writing the file out.

The *newfile* option indicates that when the buffer was created it tried to load the file identified by the filename option, but that file did not exist at that time.

The *readonly* option indicates that when the buffer was loaded, the original file was marked as being unwritable. Either that, or the *defaultreadonly* option was set to true (probably via the **-R** command line flag). This option has two purposes: it gives you a way to detect that you can't write the file out, and it protects you from writing out a file that you meant to just look at without modifying.

The *locked* option prevents you from modifying the buffer. Nearly any command which would modify the buffer will fail. The only exceptions are "undo" commands, and commands such as :e which merely reload the buffer from its original file.

Setting the *autowrite* option allows elvis to automatically write the current buffer out to a file if it has been modified, before switching to another buffer. By default this option is off, so if you try to switch away from a modified buffer, elvis will just give an error message and refuse to switch until you manually write the file out.

Elvis tries to save you from accidentally clobbering existing files. Setting the *writeany* option disables this protection; elvis will allow you to overwrite any file that the operating system will allow, without giving any warnings.

The *backup* option isn't used internally by elvis, but the default elvis.bwf file checks this flag to determine whether it should attempt to make a backup of a file it is about to overwrite. By default, this option is false, so backups will not be made.

For each buffer, the *undolevels* option indicates the number of "undo" versions elvis will maintain. Each undo level requires at least three blocks of the session file (typically 2K bytes each, 6K total) so you probably don't want to set this higher than 100 or so, and you probably want to keep it much lower. The default is 0, which is a special case that mimics vi's traditional behavior.

If the *beautify* option is true, then whenever elvis reads text from a file or external program, it will strip any control characters other than tab, linefeed or formfeed. This is false by default.

6.1.2 Statistics about a buffer

OPTION NAMES	TYPE	GROUP	DESCRIPTION
readeol, reol	One of	buf	newline mode when reading
writeeol, weol	One of	global	newline mode when writing
bufchars, bc	Number	buf	number of characters
buflines, bl	Number	buf	number of lines
partiallastline, pll	Boolean	buf	file didn't end with newline
errlines	Number	buf	buflines when :make was run
internal	Boolean	buf	elvis requires this buffer
putstyle, ps	One of	buf	type of text in a cut buffer

The *readeol* option determines how elvis reads the file into a buffer. It can be one of the following:

* **"unix"** The file is opened in binary mode, and any Line Feed characters in the file are converted to newline characters in the buffer.
* **"dos"** The file is opened in binary mode, and any Carriage Return/Line Feed pairs from the file are converted to newline characters in the buffer.
* **"mac"** The file is opened in binary mode, and any Carriage Return characters from the file are converted to newline characters in the buffer.
* **"text"** The file is opened in text mode, and no other conversion takes place.
* **"binary"** The file is opened in binary mode, and no conversion takes place.

The compiled-in default is "text", but the standard elvis.brf file sets it to a file-dependent value via the fileeol() function.

The *writeeol* option influences how elvis writes buffers out to a file. If a buffer's readeol option is set to "binary", then the value of writeeol is ignored for that buffer; the file will be written in binary. Otherwise it can be one of the following to determine the output format:

* **"unix"** The file is opened in binary mode, and newlines are written out as Line Feed characters.
* **"dos"** The file is opened in binary mode, and newlines are written out as Carriage Return/Line Feed pairs.
* **"mac"** The file is opened in binary mode, and newlines are written out as Carriage Return characters.

* **"text"** The file is opened in text mode, and no conversion takes place.
* **"binary"** The file is opened in binary mode, and no conversion takes place.
* **"same"** The value of the readeol option is used to control the output format.

The default value is "same". You might want to change that to some other mode to force the file to be written in a specific format; for example, setting it to "text" will cause a non-binary file to be written in the local text format.

The *bufchars* and *buflines* options indicate the number of characters and lines in the buffer, respectively. The buflines option works by counting newline characters; it is unaffected by vagaries of the display mode. These options can't be set.

The *partiallastline* option indicates whether the file's last line ended with a newline. Text files should always end with a newline. Traditionally, when vi loaded a file that contained a partial last line, it would append a newline to the edit buffer to complete that last line. The extra newline would be written out when the buffer was saved to a file. That's great for vi, but elvis can edit binary files as well as text, and appending newlines onto binary files could cause some problems. So elvis appends a newline just like vi, but also sets the partiallastline option to remind itself that when the buffer is saved in binary mode, the last newline should be omitted. Also, the hex display mode is smart enough to hide the added newline when this option is set.

The *errlines* option is used to store the number of lines that were in the buffer when the last :make or :cc command was run. Any difference between buflines and errlines is used to adjust the line numbers reported in any error messages, to compensate for lines which have been inserted or deleted since then.

The *internal* option indicates that elvis uses the buffer internally. Such buffers can't be deleted.

The *putstyle* option is only relevant for cut buffers. It indicates whether the cut buffer contains characters, whole lines, or a rectangular area. It is set automatically whenever you yank or cut text into a cut buffer; when you put (paste) the contents of that buffer, elvis checks the value of this option to determine how the text should be inserted into your edit buffer.

6.1.3 Options that affect movement commands

OPTION NAMES	TYPE	GROUP	DESCRIPTION
matchchar, mc	String	global	characters matched by %
paragraphs, para	String	buf	nroff paragraph commands
sections, sect	String	buf	nroff section commands
sentenceend, se	String	global	punct at end of sentence
sentencequote, sq	String	global	punct allowed after se
sentencegap, sg	Number	global	spaces required after sq

scroll, scr	Number	win	scroll amount for ^D/^U

The *matchchar* option stores a list of matching character pairs, for use by the % visual command. In each pair, the first character should be an opening parenthesis (or whatever) and the second character should be the corresponding closing parenthesis. If both characters are identical, then the % command will try to guess whether it should search forward or backward. The default value is mc=[]{}(), but you may wish to add :set mc=[]{}()<>\"\" to your ~/.exrc (or ~/elvis.rc) file.

The *paragraphs* option stores a list of two-letter nroff paragraph commands. This list is used by the { and } movement commands. Similarly, the *sections* option stores a list of section commands, affecting the [[and]] commands. Their defaults are paragraphs="PPppIPLPQP" and sections="NHSHSSSEse".

The *sentenceend, sentencequote,* and *sentencegap* options all affect the (and) sentence motion commands. The sentenceend option is a list of punctuation characters which can appear at the end of a sentence. The sentencegap option is the number spaces that must follow a sentenceend character in order for it to count as the end of a sentence. The sentencequote option is a list of punctuation characters that can appear between the sentenceend character and the spaces. Their defaults are sentenceend="?!.", sentencequote=")\"", and sentencegap=2, which meets the proposed POSIX specifications.

The *scroll* option indicates the number of lines that the ^U and ^D commands should scroll the screen by. Its default value is 12.

6.1.4 Options that affect input mode

OPTION NAMES	TYPE	GROUP	DESCRIPTION
autoindent, ai	Boolean	buf	auto-indent new text
inputtab, it	One-Of	buf	input mode's (Tab) key
smarttab, sta	Boolean	global	if indenting, (Tab) shifts
completebinary, cob	Boolean	global	complete names of binaries?
autotab, at	Boolean	buf	allow autoindent to use '\t'
tabstop, ts	Number	buf	width of tabstop columns
shiftwidth, sw	Number	buf	width used by < and >
textwidth, tw	Number	buf	width for word-wrap, or 0
wrapmargin, wm	(weird)	win	set textwidth from right
digraph, dig	Boolean	global	allow X-backspace-Y entry

Setting the *autoindent* option causes elvis to automatically insert whitespace at the start of each line, to make it line up with the preceding line. This is convenient when you're editing C source code. It is off by default.

The *inputtab* option controls the behavior of the **Tab** key. It can be set to one of the following values:

* **tab** - insert an actual tab character. This is the traditional vi

behavior, and the default for user buffers.
* **spaces** - insert enough space characters to look like a tab character.
* **filename** - attempt filename completion on the preceding word.
* **identifier** - attempt tag name completion on the preceeding word. If the word is already complete, or if cursor isn't at the end of a word, then it inserts a plain tab character. This can be handy when you're editing source code.
* **ex** - a smarter version of filename completion, it knows enough about ex command line syntax to avoid some tabbing mistakes that the filename setting can make. It can also complete ex command names, tag names, option names, and option values. This is the default for the (Elvis ex history) buffer, which is used for entering in ex commands.

The *smarttab* option only affects the behavior of the **Tab** key in input mode, when the cursor is in the indentation portion of a line -- before the first nonwhitespace character. If this option is true, then **Tab** is treated like **^T**, so the line is shifted rightward by one shiftwidth. By default this option is false, so the **Tab** key is treated normally (in accordance with the inputtab option).

The *completebinary* option controls whether binary files are included in the list of possible filename completions. The default setting is nocompletebinary, so binary files are omitted. This is handy when you're editing source code -- if your directory contains "foo.c" and "foo.o" (or "FOO.OBJ" in the Land of the Lost), then typing f-o-o-TAB will complete the "foo.c" name.

The *autotab* option affects the behavior of the < and > operator commands, and the **^D** and **^T** input mode keystrokes. If autotab is true then elvis will include tab characters in the indentation whitespace; if it is false then the indentation whitespace will consist entirely of space characters. By default, it is true.

Note that if you start with a buffer which contains no tabs, and do a ":set inputtab=spaces noautotab" then no amount of editing will result in the buffer containing tabs... unless you get tricky with **^V** or something.

The *tabstop* option affects the way tab characters are displayed, by specifying how far apart the tab stops should be located. When elvis displays a file with tabs, it displays the tabs as a variable number of spaces. You should probably leave this option at its default value (8) since changing this will make your file look strange in any other context. If you want to use indentation levels of less than 8 characters, you're better off changing shiftwidth.

The *shiftwidth* option indicates how far left or right the < and > operator commands (and the **^D** and **^T** input mode keystrokes) should shift the line of text. This is used for adjusting the indentation of lines.

When editing a text file in "normal" display mode, the *textwidth* option can be used to cause word-wrap to occur when a line gets too long. The default value of textwidth is 0, which disables automatic word-wrap. Setting it to any larger value causes word-wrap to occur when text is inserted into a line, causing that line to become wider than textwidth columns. (Note that this has nothing to do with the display formatting of the "html" and "man" display modes.)

The *wrapmargin* option is provided for backwards compatibility. It allows you to set the textwidth relative to the right edge of the window, instead of the left edge. This option's value is actually derived from the textwidth option's value and the window's width, so if you resize a window this option's value will appear to change to correspond to the new width; textwidth will not change.

Digraphs allow you to enter non-ASCII characters as a combination of two ASCII characters. There are two ways to enter digraphs: **^K X Y** and **X backspace Y**. The second form can cause some confusion if you're not expecting it, so the *digraph* option was created as a way to disable that second form. The first form of digraphs is always available. This option is false by default, to avoid the confusion.

6.1.5 Ex options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
prompt	Boolean	global	issue ":" prompt in ex mode
autoprint, ap	Boolean	global	print current line in ex
report	Number	global	minimum # lines to report
optionwidth, ow	Number	global	widths of ":set all" values

The *prompt* option controls whether a ":" prompt is issued before reading each command line in EX mode. It is true by default, and should usually be left that way.

The *autoprint* option causes elvis to display the current line of the edit buffer in certain circumstances, while you're in EX mode. It is true by default.

The *report* option determines the minimum number of lines that must change in a file, before elvis will bother to display a count of the changed lines. As a special case, if report=0 then it won't report any changes, or failed :s/old/new/ commands. Its default value is 5, so small changes won't be reported but big ones will.

optionwidth sets a limit on how wide a single option can be when output by a ":set" or ":set all" command. Limiting the widths is a good idea, because otherwise a single option that has a long value could force the output to use fewer columns, forcing some options scrolling off the top of the screen before you can read them. The ":set" command likes to leave at least two spaces after each column. The default value is optionwidth=24, which guarantees that at least 3 columns can fit on an 80-character terminal, since 80/(24+2)=3. Note that optionwidth has no effect on options that you explicitly name in a ":set" command; for example, ":set tags?" will show you the entire tag path regardless of the value of optionwidth.

6.1.6 Window statistics

OPTION NAMES	TYPE	GROUP	DESCRIPTION
windowid, id	Number	win	ID number of current window
columns, cols	Number	win	width of window
lines, rows	Number	win	height of window

The *windowid* option stores the ID number of the current window. These window IDs are listed by the :buffer command. Some GUIs may also display the window ID as part of the window's title. This value is set to a unique value automatically when the window is created. You can't change it.

The *columns* and *lines* options indicate the size of the window.

6.1.7 Options affecting the appearance of text

OPTION NAMES	TYPE	GROUP	DESCRIPTION
list, li	Boolean	win	show markups, newlines, etc.
showmarkups, smu	Boolean	global	show markup at cursor
bufdisplay, bd	String	buf	default display mode
display, mode	String	win	name of current display mode
number, nu	Boolean	win	display line numbers
ruler, ru	Boolean	win	display cursor's line/column
showcmd, sc	Boolean	win	display command characters
showmatch, sm	Boolean	win	highlight matching parens
showmode, smd	Boolean	win	display the command state
showname, snm	Boolean	global	display the buffer name
showtag, st	Boolean	global	display tag on status line
nonascii, asc	One-Of	global	how to display non-ascii
showstack, sstk	Boolean	win	display some debugging info

In the "normal" or "syntax" display modes, the *list* option causes tab characters to be shown as ^I instead of being expanded to the appropriate amount of whitespace, and it causes the end of each line to be marked with a $ character. In "html" or "man" mode, it causes all of the markups to be displayed.

In "html" or "man" mode, the *showmarkups* option causes the markup at the cursor to be displayed, but leaves other markups hidden. It has no effect in other display modes. This option is off by default, so markups won't suddenly become visible as you move the cursor around.

Each buffer has a *bufdisplay* option, which indicates that buffer's preferred display mode. Whenever a window starts to show a buffer, it switches its display mode to that buffer's bufdisplay mode. You should set bufdisplay to the name of a supported display mode: **normal**, **syntax**, **html**, **man**, **tex**, or **hex**. The compiled-in default is **normal** but the standard elvis.arf file tries to choose a more clever default, based on the extension of the buffer's filename.

The *display* option indicates which display mode the window is currently in. You can't set this option directly; you must use the :display command instead.

The *number* option causes a line number to be prepended to the start of each line. The line numbers are defined as "one plus the number of newlines preceding the start of the line," which is not necessarily how the current display mode defines lines. Consequently, the line numbers may not increment by 1 every time. These line numbers *do* correspond to the ruler and the visual G command, though. This option is false by default.

The *ruler* option causes the current line number and column number to be displayed at the bottom of the screen. This uses the same definition of "line number" as the number option, above. This option is false by default.

When entering multi-character commands, the *showcmd* option causes the preceding characters of the command to be displayed at the bottom of the window.

The *showmatch* option helps you locate matching parentheses. When you're in input mode, and you type a **)**, **]**, or **}** character, elvis will cause the matching **(**, **[**, or **{** character to be highlighted on the screen. This option is false by default.

The *showmode* option causes elvis to display a one-word label for its current parse state in the lower right-hand corner of the window. Usually, this will be either "Command" or "Input". This option is false by default, but I suggest you make it true because it really is handy.

The *showname* option causes elvis to display the buffer name on the bottom row of each window, unless it has something else to show there such as an error message.

The *showtag* option causes elvis to display (on the bottom row of each window) the name of the tag being defined at the cursor's position. Usually, this means it tells you the name of the function you're editing. When this option is true, each time you load a text file into an edit buffer elvis will scan the "tags" file for any tags which are defined in the text file. Elvis builds a table of those tags, and stores it in RAM for the sake of speed. Then, each time the window is updated, elvis will compare the cursor position to the definition lines of each tag, and display the name of the last tag it found which is defined at or before the cursor position. By default, this option is false because the tag loading can be slow.

The version of ctags distributed with elvis has a "-l" flag which causes it to generate "ln" hints, which give the line number where the tag is defined. Elvis can use these hints to greatly accelerate the loading of tags when you switch files. The "-l" option is enabled by default if you don't give any flags, so you don't need to give it explicitly unless you're also giving some other flags.

NOTE: The MS-DOS version of elvis is normally configured to omit the showtag option, because memory is tight in the lower 640K.

The *nonascii* option tells elvis how to display characters 0x80 through 0xff. It can have one of the following values:

VALUE	MEANING
all	All characters 0x80-0xff are visible
most	Chars 0xa0-0xff are visible, but not 0x80-0x9f
none	Chars 0x80-0xff are not visible
strip	Convert 0xa0-0xfe to ASCII; others not visible

Any characters which aren't visible will be displayed as '.' characters. Note that this only affects the way the characters are displayed; they are actually stored with their true 8-bit value. The default value of nonascii is "most", because that is the correct value for the Latin-1 symbol set.

The *showstack* option causes some debugging output to appear on the bottom row of the window. It is false by default, and you should leave it that way.

6.1.8 Options for a particular display mode

OPTION NAMES	TYPE	GROUP	DESCRIPTION
commentfont, cfont	One-Of	syntax	font used for comments
stringfont, sfont	One-Of	syntax	font used for strings
keywordfont, kfont	One-Of	syntax	font used for reserved words
functionfont, ffont	One-Of	syntax	font used for function names
variablefont, vfont	One-Of	syntax	font used for variables
prepfont, pfont	One-Of	syntax	font used for preprocessor
otherfont, ofont	One-Of	syntax	font used for other symbols
includepath, inc	String	syntax	where to find #include files

In the syntax display mode, the *commentfont, stringfont, keywordfont, functionfont, variablefont, prepfont* and *otherfont* options specify which font is to be used for different parts of the source code. Each option can be set to **normal**, **bold**, **emphasized**, **italic**, **underlined**, or **fixed**. The prepfont is used for preprocessor directives. The keywordfont is used for reserved words such as "int" and "return". The functionfont is used for any other word which is followed by an opening parenthesis character. The otherfont is used for any other word which matches some language-dependent criteria; for C, the word must either contain no lowercase letters or end with a "_t" (probably a constant or a user-defined type). The variablefont is used for all other words. Punctuation is always in the normal font; you can't control that.

You can set these variables during initialization, in the .exrc or elvis.rc file. After that, your window must actually be in the "syntax" mode for these to be accessible.

As a separate step, some user interfaces allow you to specify a color to be used for each font, via the :color command.

The *includepath* option contains a list of directory names where elvis should look for #include files. When you look up a tag whose name begins with a quote character, elvis searches through those directories for a file with the same name as the tag (with the quotes stripped off). This means that you can move the cursor onto a #include file name, hit ^], and have elvis load the indicated header file.

6.1.9 Messages

OPTION NAMES	TYPE	GROUP	DESCRIPTION
terse, te	Boolean	global	don't translate messages
verbose	Numeric	global	give more status messages
errorbells, eb	Boolean	global	ring bell for error message
warningbells, wb	Boolean	global	ring bell for warning msg
flash, vbell	Boolean	global	substitute flash for bell

The *terse* option indicates whether elvis should attempt to translate messages via the elvis.msg file. If terse is true, then no such translation takes place; the built-in messages are used. If terse is false, then elvis will search through the file (actually the "Elvis messages" buffer) for a line which looks like "*terse*:*verbose*" and if found it'll use the verbose version instead. By default, terse is false.

The *verbose* option has nothing to do with the terse option. Instead, it indicates the number of **-V** flags given when elvis was invoked. Larger values indicate that the user wants more status messages to be generated. This is handy when elvis isn't initializing itself the way you expected it to; elvis' initialization code frequently tests the value of verbose and automatically writes status messages when verbose is set to a high enough level. Values normally range from 0 (no extra output) to 9 (maximum output).

The *errorbells* and *warningbells* options cause the terminal's bell to ring when an error message or warning message is generated, respectively. By default the errorbells option is true, and the warningbells option is false.

Setting the *flash* option causes elvis to use a visible alternative to the bell, if one exists. This is nice in a crowded terminal room. By default this option is false.

6.1.10 Words

OPTION NAMES	TYPE	GROUP	DESCRIPTION

true, True	String	global	locale’s True value
false, False	String	global	locale’s False value
submit, Submit	String	x11	locale’s Submit label
cancel, Cancel	String	x11	locale’s Cancel label
help, Help	String	x11	locale’s Help label

These options store words, which are translated via the elvis.msg file when elvis starts up. The default versions of all of them are their capitalized English names.

The *true* and *false* options exist primarily to allow the english words **true** and **false** to be used in expressions to represent Boolean literals. Also, the value of false is used as an alternative false string, in addition to "", "0", or "false". (In a Boolean context, any string that isn’t false is considered to be true, so elvis never compares a string to the true option’s value.) A Boolean option will return the value of either the true or false option, as appropriate.

If you’re using the "x11" user interface, then values of the *submit* and *cancel* options are used as the labels for the [Submit] and [Cancel] buttons in a dialog. Also, if the dialog contains any Boolean options, the value will be displayed using values of the true and false options.

Currently the *help* option does nothing. Eventually I expect to add pull-down menus to the "x11" interface, though, and in Motif menu bars the "Help" menu traditionally appears on the far right edge. The value of the help option will allow elvis to recognize the "Help" menu.

6.1.10 Options for a particular user interface

OPTION NAMES	TYPE	GROUP	DESCRIPTION
term, ttytype	String	tcap	terminal’s termcap entry
ttyrows, ttylines	Number	tcap	height of screen
ttycolumns, ttycols	Number	tcap	width of screen
ttyunderline, ttyu	Boolean	tcap	okay to mix color & underln
codepage, cp	Number	win32	console character set
scrollbar, sb	Boolean	(gui)	enable the scrollbar
toolbar, tb	Boolean	(gui)	enable the toolbar
statusbar, stb	Boolean	(gui)	enable the statusbar
menubar, mb	Boolean	windows	enable the menubar
font, fnt	String	windows	base font
normalstyle, nfn	String	windows	n or combination of b/i/u
boldstyle, bfn	String	windows	n or combination of b/i/u
italicstyle, ifn	String	windows	n or combination of b/i/u
fixedstyle, ffn	String	windows	n or combination of b/i/u
emphasizedstyle, efn	String	windows	n or combination of b/i/u
underlinedstyle, nfn	String	windows	n or combination of b/i/u

normalfont, xfn	String	x11	name of normal font
boldfont, xfb	String	x11	name of bold font
italicfont, xfi	String	x11	name of italic font
controlfont, xfc	String	x11	name of toolbar font
underline, uln	Boolean	x11	enables underlining
toolbar, xtb	Boolean	x11	enables the toolbar
scrollbarleft, xsl	Boolean	x11	enable scrollbar on side
scrollbarwidth, xsw	Number	x11	size of scrollbar, in pixels
scrollbartime, xst	Number	x11	delay for scrollbar repeat
borderwidth, xbw	Number	x11	size of text area's border
dblclicktime, xdct	Number	x11	double-click speed, 1/10 Sec
blinktime, xbt	Number	x11	cursor blink rate, 1/10 Sec
textcursor, tc	Number	x11	one of hollow, opaque, xor
xrows, xlines	Number	x11	height of new windows
xcolumns, xcols	Number	x11	width of new windows
firstx, xpos	Number	x11	horiz. position of first win
firsty, ypos	Number	x11	vert. position of first win
stagger	Number	x11	offset for next new window
icon	Boolean	x11	use the built-in icon?
stopshell, ssh	String	x11	interactive shell command
autoiconify, aic	Boolean	x11	iconify old window
altkey, metakey	One of	x11	effect of the Alt key
focusnew, fn	Boolean	x11	force focus into new window
warpto, wt	One of	x11	^W^W forces pointer movement
warpback, xwb	Boolean	x11	upon exit, point to xterm
outlinemono, om	Number	x11	char outlining for X11-mono

6.1.10.1 Termcap options

The *term, ttyrows, ttycolumns,* and *ttyunderline* options are only present if you're using the **termcap** user interface. They indicate the name of the termcap entry being used (normally taken from the TERM environment variable), the size of the screen, and whether it is safe to try underlining text when colors have been assigned to fonts. The ttyunderline option is true by default, but it should be made false on the Linux console, because the console driver has a bug which prevents underlined text from being shown in color.

The *codepage* option only exists in the Win32 version with the termcap interface (WindowsNT or Windows95, in console mode). It indicates which code page (character map) the console is using. Its value is persistent; if you change it in elvis, the console will remain changed even after you exit elvis. Changing the code page has no effect on the digraph table, or elvis' idea of which non-ASCII characters are printable or should be treated as letters; it only reconfigures the console driver. Typical values are 437 for the standard IBM PC character set, and 850 for extra European characters.

6.1.10.2 Options common to Windows and X11

The *scrollbar*, *toolbar*, and *statusbar* options indicate whether the scrollbar, toolbar, and statusbar should be visible, respectively. By default, all are are visible.

6.1.10.2 Windows options

The *menubar* option indicates whether the menu bar should be visible. By default, it is visible.

The *font* option stores the name of the base font. The easiest way to set it is via the "Options->Font" menu item.

The *normalstyle*, *boldstyle*, *italicstyle*, *fixedstyle*, *emphasizedstyle*, and *underlinedstyle* options determine how elvis will derive each of its fonts from the base font. The values of these options are strings. If the string is "n" then the base font is used unmodified. Other possibilities are any combination of "b" for bold, "i" for italic (slanted), and "u" for underlined. For example, ":set ufn=bu" causes elvis' underlined font to be drawn in bold with an underline.

6.1.10.3 X11 options

The other options all apply to the **x11** interface. The *normalfont, boldfont,* and *italicfont* options control the X fonts used for displaying text. Typically, the elvis.ini or ".exrc" file will set these. If you do choose to set them in one of these files, be sure to have your initialization script check which interface is being used because if elvis is using the termcap interface then these x11 options won't exist. These options all default to an empty string; this is a special case which causes elvis to use the "fixed" font for normal text, and to derive the bold and italic fonts from the normal font.

The *controlfont* option determines which font is used for displaying the labels of toolbar buttons, and also the statusbar. Unlike the other fonts, this one is permitted to have a variable pitch. If it is unset, then elvis will use the font named "variable" by default.

The *underline* option determines whether characters in the "underlined" font should be displayed as underlined. Normally, underline is true, so they are underlined. Setting nounderline will cause them to be displayed as normal characters, but in the color of underlined text.

The *toolbar* option controls whether the toolbar is visible or not. It is normally true, which makes the toolbar visible. The toolbar can be configured via the :gui command.

The *statusbar* option controls the visibility of the statusbar. It is true by default, which makes the statusbar is visible. The statusbar always displays the information which would otherwise be shown on the bottom row of the text area only when the ruler and showmode options were true. When you press a toolbar button, the button's one-line description is shown on the statusbar.

The *scrollbarleft* option determines which side of the window the scrollbar will be drawn on. This option is false by default, so the scrollbar appears on the right side of the window. Making it true will cause the scrollbar to be drawn on the left.

The *scrollbarwidth* option controls the size of the x11 scrollbar. The default value is 14 pixels, and the allowed range is 5 to 40 pixels.

The scrollbar buttons automatically repeat if you hold a mouse button down *scrollbartime* tenths of a second. The default is 4 tenths of a second.

The main text area of a window looks better when the characters aren't drawn immediately adjacent to the edge. The *borderwidth* option allows you to specify how many pixels should be left blank between a character and any edge of the text area. The default is 1 pixel.

The *dblclicktime* option allows you to adjust the speed of mouse double-clicks to match your own clicking habits. The default is 3 tenths of a second.

The *blinktime* option controls the cursor blink rate. If set to 0, the cursor will not blink. If set to a value from 1 to 10, then the cursor will first be visible for that many tenths of a second, and then invisible for the same amount of time. The cursor will only blink in the window which currently has keyboard focus.

The *textcursor* option controls the way the block text cursor is drawn. It can be **xor**, **hollow**, or **opaque**. The default is **xor**, which causes the cursor to be drawn as a filled rectangle with the XOR bitblt function. This converts the background color to the cursor color, and the foreground color to an unpredictable color; hopefully the foreground color will contrast with the cursor color well enough to allow you to discern what the underlying character is. The **hollow** cursor style causes the cursor to be drawn as an unfilled rectangle. This allows you to easily see the underlying character, and detect whether it is highlighted or not. The **opaque** cursor style draws a filled rectangle, which is easier to locate but you can only see the underlying character between blinks.

The *xrows* and *xcolumns* options control the initial size of windows. They default to 34 and 80, respectively, and can also be set via the **-geometry** command-line flag. After a window has been created, you can use your window manager to resize the window.

The *firstx* and *firsty* options, if set, control the position of the first window that elvis creates. If they are unset, then elvis doesn't specify a position for the window. The **-geometry** command-line flag can be used to set these options. After the first window has been created, if the *stagger* option is set to a non-zero value then any new windows are created that many pixels down and to the right of the current window. If *stagger* is zero, then elvis won't specify a position for the new windows, so the window manager can choose the location itself.

The *icon* option can only be set in an initialization file such as elvis.ini or ".exrc"; once the first window has been created it is too late to change it. This option controls whether the window will be given the default, built-in icon. It is true by default, so windows will get the icon. This is usually a good thing. Some window managers don't allow you to override built-in icons, though, so if you want your window manager to use a different icon for elvis then you'll need to have a "set noicon" in your elvis.ini file.

The *stopshell* option stores a command which runs an interactive shell. It is used for the :shell and :stop ex commands, and the ^Z visual command. Normally, this is set to "xterm &" so you get a shell in a window. The "&" at the end of the command allows elvis to continue responding to user input while the shell is running.

When the ^W^W visual command switches keyboard control to an X11 window which as been iconified, elvis automatically deiconifies it. When it does this, if the *autoiconify* option is set then elvis will iconify the previous window, so the number of iconified elvis windows remains constant. By default, this option is false. Regardless of whether autoiconify is set, you can always use your window manager to iconify or deiconify windows manually.

The *altkey* option controls the effect of the **Alt** or **Meta** keys. It can be set to either **control-O**, **setbit**, or **ignore**. The **ignore** value is self explanatory. If the option is set to **control-O** then the x11 interface will simulate a **^O** keystroke before each actual keystroke. This is handy because if you're in input mode you can just hold down **Alt**/**Meta** to perform a series of visual commands. If the option is set to **setbit** then the x11 interface will set the most significant bit of each ASCII character while the **Alt**/**Meta** key is held down. Some other programs use this trick as a means of entering non-ASCII characters. (Elvis has a better way though; check out the :digraph command.) The default is **setbit**.

The *focusnew* option causes elvis to force input focus to switch to any newly created window, or to one which has been deiconified. It is true by default; making it false (":set nofocusnew") prevents elvis from forcing a change of input focus in those two situations. Note that elvis always forces a change of input focus when you give a command which switches windows, such as ^W^W.

The *warpto* option can cause elvis to force the mouse pointer to move whenever you use keyboard commands such as ^W^W to switch from one elvis window to another. There are two reasons you may wish to do this: either your window manager requires the pointer to be in a window for that window to receive keystrokes, or you want to have your X server automatically pan the screen to bring the next window into view.

You can set the warpto option to any one of the following values: **don't**, **scrollbar**, **origin**, or **corners**. The default is **don't** which prevents any automatic pointer movement. The **scrollbar** value causes the pointer to move to the scrollbar, and **origin** moves it to the upper-left corner. The **corners** value causes the pointer to move first to the corner furthest from the window's text cursor, and then to the nearest corner; this will cause the X server to pan (if necessary) to bring the entire window into view.

The *warpback* option, if set, causes the X terminal's graphic cursor to be moved back to the window which held keyboard focus at the time when elvis was started. Usually this will be the xterm where you typed in the "elvis files..." command line. Just as the firstx, firsty, and stagger options are intended to allow mouseless positioning of elvis windows, the warpback option is intended to serve as a mouseless way to switch keyboard focus back to the original xterm, so that mouse haters will find elvis' x11 interface as convenient to use as the termcap interface. By default, warpback is false.

The *outlinemono* option affects the way that text is drawn against a stippled background when elvis is run on monochrome X terminals (or with the -mono command-line flag). It has no effect on color systems. Because characters drawn on a stippled background can be hard to read, elvis can draw a white outline around the black characters. The value of outlinemono is a number that indicates how thick the outline should be. 3 is the thickest supported outline, and 0 is no outline at all. The default is 2.

6.1.11 Regular expression options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
ignorecase, ic	Boolean	global	regexp uppercase=lowercase
magic, ma	Boolean	global	use normal regexp syntax
autoselect, as	Boolean	global	visibly mark searched text
wrapscan, ws	Boolean	global	searching wraps at EOF<->BOF
gdefault, gd	Boolean	global	default change all instances
edcompatible, ed	Boolean	global	remember regsub flags
saveregexp, sre	Boolean	global	remember regexp to use as //

Setting the *ignorecase* option to true will cause elvis to treat uppercase and lowercase letters as being equal, except in character list metacharacters. When ignorecase is false (the default), they are treated as different.

The *magic* option selects one of two different syntaxes for regular expressions. When magic is true, it uses the normal syntax in which * and . are special characters. When magic is false, it uses a simplified syntax.

The *autoselect* option, when true, causes a successful visual search command such as /regexp to visibly mark the matching text just like the v command does. This is intended to compensate for elvis 2.1's lack of a "c" option in the :s/old/new/ command. By default, autoselect is false.

The *wrapscan* option determines what happens when a search command bumps into the top or bottom of a buffer. If wrapscan is true, then the search will wrap around to the other end of the buffer, so if there's a match anywhere in the buffer, the search will find it. If wrapscan is false, then searches fail when they hit the end of the buffer. By default, wrapscan is true.

The *gdefault* option affects the default behavior of the :s/old/new/ command. It is false by default, which causes :s/old/new/ to assume a count of 1 so only the first instance in each line is changed. Making gdefault true will cause it change all instances in each line, as though the "g&quot flag had been given. If you give an explicit count or "g" flag, then the value of gdefault is ignored.

The *edcompatible* option causes elvis to remember any flags that are passed into the :s/old/new/flags command, and use them as the default for the next such command. Explicitly naming a flag will toggle that flag's value. This is *not* the way the old ed editor worked, but this option's name and behavior are traditional in vi. This option is false by default.

The *saveregexp* option is normally true, which causes elvis to remember each regular expression. If, in a latter command, you give an empty regular expression, then elvis will recall the saved regular expression instead. This also affects the n and N commands. You may wish to turn this option off temporarily in a the lib/elvis.arf file if you're using any regular expressions there, so that loading a file doesn't interfere with n and N.

6.1.12 Tag options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
taglength, tl	Number	global	significant length of tags
tags, tagpath	String	global	list of possible tag files
tagstack, tsk	Boolean	global	remember origin of tag srch
tagprg, tp	String	global	external tag search program

These options control how elvis performs tag lookup, as for the :tag ex command or the ^] visual command. You should also check out the previoustag and showtag options.

The *taglength* option defines how many characters are significant in a tag name. By default this option is set to 0, which is a special value indicating that all characters are significant. If you have a lot of long names, you might want to set this to some other value so that you could type in abbreviated names.

The *tags* option stores a list of filenames or directory names where tags are stored. (For directory names, it looks for a file named "tags" in that directory.) When performing tag lookup, elvis will begin by looking for it in the first directory/file mentioned in the list; if it doesn't find it there, then it moves on to the next one, and so on. By default, it just looks in a file named "tags" in the current directory.

In a path, names which start with "./" (or ".\" in MS-Windows) are assumed to be relative to the directory of the current file. This means that ":set tags=./tags:tags" will cause elvis to first check the "tags" file in the directory of the current text file, and then the "tags" file in the current directory.

NOTE: Traditionally, this elements in this path have been space-delimited. Since every other path in any other context is either colon-delimited (for Unix) or semicolon-delimited (for Microsoft), and it is becoming more common for filenames to contain spaces, elvis uses colons or semicolons for the tag path too. This makes elvis' "tags" settings incompatible with other versions of vi, though.

If the *tagstack* option is true, then before switching to the file and location of a looked-up tag, elvis will store the original file and position on a stack. Later, you can use the :pop or visual ^T commands to return to your original position. If tagstack is false, then the tag stack is unaffected by tag look-up. It is true by default.

If the *tagprg* option is set to any value other than "", then whenever you try to do a tag search via :tag or :browse, elvis will execute tagprg's value as a shell command and interpret its stdout as a list of matching tags. Before the command is run, it is evaluated using the simpler expression syntax with **$1** indicating where the arguments should go. The default value of tagprg is "" which causes elvis to use the internal tag search algorithm.

NOTE: You might also consider using the ccprg option for this sort of thing, since the :cc command has a smarter line parser than the :tag command.

6.1.13 Window update parameters

OPTION NAMES	TYPE	GROUP	DESCRIPTION
exrefresh, er	Boolean	global	redraw scrn after each line
nearscroll, ns	Number	global	scroll vs. jump&center param
wrap	Boolean	win	how long lines are displayed
sidescroll, ss	Number	win	sideways scrolling amount
optimize, op	Boolean	global	run faster
animation, anim	Number	global	animation macro speed
window, wi	Number	global	lines to show for :z command
pollfrequency, pf	Number	global	rate of testing for ^C
maptrace, mt	One of	global	debugger: off, run, or step
maplog, mlog	One of	global	logging: off, reset, append

The *exrefresh* option affects the frequency of window updates when in EX mode. It is normally false, which causes the window to be refreshed at the end of each EX command. If you set exrefresh to true, then elvis will update the window's image every time an output line is generated; this makes the command run much slower, but gives you more feedback.

The *nearscroll* option controls elvis' behavior when the cursor is moved off the top or bottom of the window. If the new cursor position is within nearscroll lines of the window, then the window is scrolled to bring the new line into view. If the new cursor position is outside that range, then elvis uses a "jump and center" approach, in which the window's image is drawn from scratch with the new cursor line shown in the center of the window. Its default value is 5.

The *wrap* option determines how elvis will display lines which are too long to fit on a single row of the display. It is true by default, which causes long lines to be wrapped onto multiple rows of the display. This is the traditional vi behavior. Changing it to false will cause long lines to be partially displayed on a single row of the display; you can scroll sideways to reveal the rest of the line my moving the cursor onto it, and then off the edge.

If the wrap option is false (indicating that long lines should be displayed via side-scrolling) then the *sidescroll* option controls the scrolling increment. The default is 8, so the display will scroll sideways in chunks of 8 characters at a time.

The *optimize* option affects the efficiency of screen updates. It is normally true, which tells elvis to update the screen image only when it must wait for user input. If you make it false, then elvis will update the screen after every command; among other things, this allows you to see intermediate effects of macros.

The *animation* option is similar. When the optimize option is true, elvis still refreshes the screen periodically while executing a large macro so that animation macros can be seen in all their glory. Elvis attempts to figure out which macros are loops, and when one of those macros is invoked elvis considers updating the screen. If animation=1 then elvis updates the screen every time; when animation=2 it updates the screen an alternate invocations of those macros, and so on. The default, chosen simply through experimentation, is 3.

Sometimes elvis will choose the wrong macros to refresh. If that happens, then try running the macro with optimize option turned off. For example, the bouncing ball macros look better with optimize turned off.

The *window* option stores the default number of lines to be displayed by the :z command. Historically it has also been used for forcing vi to update only a portion of the screen, but elvis doesn't use it for that.

When elvis is performing some time-consuming operations, such as a global substitution, it will periodically check to see if the user is trying to cancel the operation. For some user interfaces, this inspection takes a significant amount of time so elvis allows the *pollfrequency* option to reduce the frequency of these checks. The default is 20. Larger values of pollfrequency will make global substitutions run faster; smaller values make elvis respond to **^C** sooner.

The *maptrace* option controls elvis' built-in macro debugger. It can be **off**, **run** or **step**. The default is **off**, which causes macros to run normally. If you change it to **run** then elvis will display the contents of the mapping queue at the bottom of the screen while running any macro. The **step** value also displays the mapping queue, but then waits for a keystroke before proceeding. If the keystroke is **^C** then the macro is terminated. If the keystroke is **r** then maptrace is set to **run**. Any other keystroke causes elvis to pause again after processing the macro's next character. See section 16.3 How to debug macros for more suggestions for debugging macros.

The *maplog* option can be used to log the information displayed by the maptrace option. It also logs any ex commands that are executed, other than those that you enter manually. It is **off** by default. Setting it to **append** causes the map trace information to be appended to an internal edit buffer named "Elvis map log". Setting it to **reset** causes that buffer to be clobbered before the next map trace; when that happens, maplog will be automatically switched to **append**. You can view the logged data via the command...

:("Eml)sp

or the long version, ":(Elvis map log)split".

6.1.14 Cache options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
blkcache, cache	Number	global	number of blocks in cache
blksize, bsz	Number	global	size of cache block
blkfill, bfill	Number	global	initial chars per text block
blkhash, hash	Number	global	size of cache hash table
blkgrow, bgr	Number	global	allocation table parameter
blkhit, bh	Number	global	# of block requests in cache
blkmiss, bm	Number	global	# of block req. not in cache
blkwrite, bw	Number	global	# of blocks written
sync	Boolean	global	force changes to disk

You probably don't need to know about the "blk" options. The *blkcache* option indicates how many blocks from the session file elvis should keep in its own internal cache, and *blkhit* and *blkmiss* can be used to gauge the efficiency of the cache. *blkwrite* indicates how many blocks have been written to the session file. The *blksize* option indicates the size of each block, *blkfill* indicates how many characters should be stuffed into each block initially (leaving room for more text that the user may insert later), and *blkhash* and *blkgrow* affect a couple of internal tables.

Note that the value of *blksize* can only be set via the **-b***blksize* command line flag, and its value must be a power of 2 in the range [512, 8192]. You can't change *blksize* after elvis has started (not even in configuration scripts), because by then the session file has already been created with the other block size.

If the *sync* option is true, then elvis will flush all dirty blocks from its cache at the end of each edit command. Doing this will just about guarantee that you can recover your changes after a crash, but it can slow down the computer tremendously. The sync option is false by default, and on multi-user systems it should be left that way. On a single-user system, you might consider setting the sync option.

6.1.15 Options that describe the system

OPTION NAMES	TYPE	GROUP	DESCRIPTION
version, ver	String	global	elvis version number (2.1)
bitsperchar, bits	Number	global	character size (always 8)
gui	String	global	name of user interface
os	String	global	name of operating system
program, argv0	String	global	invocation name of elvis
session, ses	String	global	name of session file
tempsession, temp	Boolean	global	delete session file on exit
newsession, newses	Boolean	global	session file is new
recovering, rflag	Boolean	global	recovering after a crash
exitcode, exit	Number	global	exit code of elvis process

The *version* option stores the version number of elvis -- currently "2.1". If later versions of elvis have features which are incompatible with this version, your script files can use this to check the version number, and skip the uncompatible commands.

The *bitsperchar* option indicates the size of characters that elvis uses internally. Currently this is always 8, but I expect to support 16-bit characters eventually.

The *gui* option indicates which user interface is being used. This can be handy in your initialization files. For example, you might prefer white characters on a blue background when using the "termcap" interface, and black characters on a white background when using the "x11" interface.

The *os* option allows elvis' initialization files to act differently on different operating systems. Its value indicates the name of the local operating system.

The *program* option stores the name by which elvis was invoked; i.e., the value of argv[0]. Typical values would be "elvis" under UNIX, "elvis.exe" under Win32, or "C:\BIN\ELVIS.EXE" under MS-DOS. The default elvis.ini file evaluates tolower(basename(program)) and compares the result to "ex" and "view", to set the initialstate and defaultreadonly options, respectively.

The *session* option stores the name of the current session file. There is rarely any need to check this, but I had to store it someplace and it might as well be accessible, I figured.

The *tempsession, newsession,* and *recovering* options describe different aspects of the session file. If tempsession is true, then elvis will delete the session file when it exits. If newsession is true, then elvis has just created the file so there may be extra initialization that needs to take place in elvis.ini or someplace. If recovering is true, then the session file may be damaged, so it may be a good idea to skip some initialization steps, or automatically write out all user buffers.

The *exitcode* is the value that elvis will return to its parent process when the elvis process exits. Initially this is 0, which is the conventional indication of a normal, successful exit. You can explicitly set it to other values to indicate special situations. Also, if elvis outputs an error message and exitcode has not been explicitly set, then elvis changes exitcode to 1, so the parent process can know that elvis had an error.

6.1.16 External programs

OPTION NAMES	TYPE	GROUP	DESCRIPTION
ccprg, cp	String	buf	shell command for :cc
makeprg, mp	String	buf	shell command for :make
anyerror, ae	Boolean	global	allow :errlist if readonly
equalprg, ep	String	buf	shell command for = operator
keywordprg, kp	String	buf	shell command for K command
shell, sh	String	global	name of shell program
warn	Boolean	global	warn if file not saved

The *ccprg* and *makeprg* are the programs used by the :cc and :make commands. Before the program strings are executed, they are subjected to the same sort of expression evaluation as the :eval command, with $1 representing any extra arguments from the ex command line, and $2 representing the name of the current file. Their defaults are cc="cc ($1?$1:$2)" and make="make $1".

When searching for error messages after a :cc or :make command, elvis will normally ignore errors about files that you don't have write access to. Usually this is convenient, because it prevents elvis from reading header files that you've misused. However, setting *anyerror* to true will make it read any file that generates a complaint, even if you can't write to it.

The *equalprg* option stores the name of a program to be executed for the visual = operator command. Its default value is "fmt", which is a simple text formatting program.

The *keywordprg* option stores the name of the program used by the visual K command. This string is evaluated with $1 being replaced with the word under the cursor at that time, and $2 the name of the current file. The default value is "ref $1 file:$2"; the *ref* program looks up a tag and displays it. If you're using the x11 user interface, then you might want try the following, which causes the function's header to be displayed in a separate pop-up window:

```
set kp="ref $1 file:$2 2>&1 \| xmessage -file - >/dev/null 2>&1 &"
```

The *shell* option stores the name of the system's command-line interpreter. It is used when executing all of the above programs, as well as commands entered for the EX :! and visual ! commands. Its default value is system-dependent; typically it will be "/bin/sh" for UNIX, and "C:\COMMAND.COM" for MS-DOS.

When any external program is executed, if the current buffer has been changed but not written out to the file, then elvis will normally give a warning message. Setting the *warn* option to false disables this message.

6.1.17 Directory names

OPTION NAMES	TYPE	GROUP	DESCRIPTION
home	String	global	home directory
elvispath, epath	String	global	list of possible config dirs
sessionpath, spath	String	global	list of possible session dir
directory, dir	String	global	where to store temp files

The *home* option is the name of your home directory. The value of this option is used for replacing the ~ character at the start of a full pathname. If an environment variable named HOME exists, then home is initialized from its value. Otherwise, its default value is set as follows:

For UNIX:
The default is "/".

For Win32:
The default is derived from environment variables named HOMEDRIVE and HOMEPATH, which will normally always be defined. Their default value is usually "C:\users\default". If either of those environment variables is undefined, then elvis will attempt to find the pathname of the program, and use its directory. As a last resort, elvis will use "C:\" as the default home directory.

For OS/2:
The default home directory is the one containing ELVIS.EXE, or if that can't be found then it will use "C:\" as the default home directory.

For MS-DOS:
The default home directory is the one containing ELVIS.EXE.

The *elvispath* option stores a list of directory names where elvis might find its configuration files. If there is an ELVISPATH environment variable, then the elvispath option is initialized from the value of ELVISPATH. Otherwise it is set to a value such as "~/.elvislib:/usr/local/lib/elvis" so that elvis will search first in a subdirectory of the user's home directory, and then in the directory where the standard versions of those files were installed. A path like this allows users to override elvis' behavior if they want. The default value depends the operating system, as follows:

For UNIX:
The default contains *~/.elvislib* and the directory that you specified as the data directory when you ran the configure script. (E.g, "configure --datadir=/usr/lib/elvis") The default data directory is */usr/local/lib/elvis*, so usually elvispath will default to "~/.elvislib:/usr/local/lib/elvis".

For Win32, OS/2, or MS-DOS:
The default contains *~\elvislib*, and the directory where *elvis.exe* resides, and a subdirectory under that named "lib". For example, if elvis is installed as C:\elvis\elvis.exe then elvispath would be *~\elvislib;C:\elvis;C:\elvis\lib*.

The *sessionpath* option gives elvis a list of possible directories where session files might be placed. Elvis uses the first writable directory in that list, and ignores all of the others. The default value depends on the operating system, and can be overridden by the SESSIONPATH environment variable. You can't change the sessionpath option after elvis has started, because the session file has already been created by then.

The *directory* option gives the name of the directory where elvis will store its temporary files. The default value is system-dependent. Note that this is *not* where the session file is stored; the session option gives the name of the session file.

6.1.18 Initialization options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
exrc, ex	Boolean	global	interpret ./.exrc file
modeline, ml	Boolean	global	interpret modelines
modelines, mls	Number	global	positions of modelines
safer, trapunsafe	Boolean	global	be paranoid
initialstate, is	One-Of	global	command mode of new windows

The *exrc* option has no built-in meaning to elvis, however the default elvis.ini file uses this option to determine whether it should look for a ".exrc" file in the current directory.

The *modeline* option controls whether elvis will look for modelines in each buffer after it has been loaded from a file. If modelines is true, then elvis will search through the first and last *modelines* lines of the buffer for something that looks like "ex:*commands*:" or "vi:*commands*:" and if found, it executes the *commands* as an ex command line. This is typically used for changing tabstops and the like. The modeline option is false by default, and modelines is 5.

The *safer* option closes some security holes. It is intended to make modelines and a .exrc file in the current directory safe to use, but I'm not making any promises. When the "safer" option is true, certain commands are disabled, wildcard expansion in filenames is disabled, and certain options are locked (including the safer option itself). Typically you will use the ex command :safer to execute an untrusted file, and :source to execute a trusted one, rather than futz with the value of the safer option directly.

The *initialstate* option determines what command mode new windows will start in. It can be one of **input**, **replace**, **vi**, or **ex**. The default is **vi**, the visual command mode.

6.1.19 Keyboard map options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
remap	Boolean	global	allow key maps to use maps
keytime, kt	Number	global	timeout for function keys
usertime, ut	Number	global	timeout for multi-key maps

Elvis allows keystrokes to be mapped via the :map command. Once a map has been defined, these options control how and when those maps are recognized.

The *remap* option controls how many times elvis will attempt to reapply key maps. If the remap option is true (the default), then elvis will repeatedly attempt to reapply maps as long as there are any that match. This means that maps can be written to use other maps, allowing some very complex behavior. If remap is false, then it will attempt to apply maps only once, so the result of any map is not altered any further. By default, remap is true.

The *keytime* and *usertime* options come into play when characters are received which *partially* match one or more maps. For example, suppose the arrow keys are mapped to **h**, **j**, **k**, and **l**, those arrow keys send escape sequences when pressed, and elvis has just received an escape character. How can it tell whether the user hit the **Esc** key or an arrow key?

In this situation, elvis must perform a read-keystrokes-with-timeout operation to determine which map applies, if any. If all of the partially matching maps are for special keys such as function keys, then elvis will use the keytime value. If at least one of them is for a user map, then elvis will use the usertime value. Either way, the values indicate the time, in tenths of a second, that elvis should allow for the rest of the map characters to arrive. If they don’t arrive, then none of the partially matching maps is used.

Typically, the usertime value will be much longer than the keytime value, because the user must hit a series of keys for a user map. For example, many people like to create maps consisting of a semicolon and one or two following letters. (If you’re a touch typist, then your right-hand pinky normally rests on the semicolon key, so this is convenient.) By distinguishing between key maps and user maps, elvis can give quick response to the **Esc** while still allowing users to key in their own keymaps at a leisurely pace. Their default values are keytime=3 and usertime=15.

6.1.20 Printing options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
lptype, lpt	String	lp	printer type
lpconvert, lpcvt	Boolean	lp	convert Latin-1 to PC-8
lpcrlf, lpc	Boolean	lp	printer needs CR-LF newline
lpout, lpo	String	lp	printer file or filter
lpcolumns, lpcols	Number	lp	width of printer page
lpwrap, lpw	Boolean	lp	simulate line-wrap
lplines, lprows	Number	lp	length of printer page
lpnumber, lpn	Boolean	lp	print line numbers in margin
lpheader, lph	Boolean	lp	print header at top of page
lpformfeed, lpff	Boolean	lp	send form-feed after last pg
lppaper, lpp	String	lp	paper size (letter, a4, ...)
lpcolor, lpcl	Boolean	lp	use colors when printing

These options all affect hardcopy output, done via the :lpr command. Note that these options are in a separate group, so you can display all of them by giving the command "se lp?".

The *lptype* option lets elvis know what type of printer you're using, so it can use the correct escape codes (or whatever) to switch fonts. The default is "dumb" because it is the most conservative value, but it is also the least expressive. (Exception: When using the Win32 user interface, the default is "windows".) You should set lptype to one of the following values:

VALUE	PRINTER DESCRIPTION
ps	PostScript, one logical page per sheet
ps2	PostScript, two logical pages per sheet
epson	Most dot-matrix printers, no graphic chars
pana	Panasonic dot-matrix printers
ibm	Dot-matrix printers with IBM graphic chars
hp	HP printers, and most non-PostScript lasers
cr	Line printers, overtypes via carriage-return
bs	Overtypes via backspace, like nroff
dumb	Plain ASCII, no font control
html	HTML source code
windows	The Win32 print facility (in WinElvis only)

The *lpconvert* option, when set, causes some printer types to convert non-ASCII Latin-1 characters to PC-8 characters. Most computers use Latin-1 internally for storing text, but many printers use PC-8; hence the need for conversion. This option has no effect on ASCII characters because they never need conversion. This option is ignored if your computer doesn't appear to be using Latin-1 (or, more precisely, if there is no digraph which maps AE to 0xc6, the Latin-1 code for the *Æ* ligature.) This option is false by default.

> NOTE: Not all printer types obey the **lpconvert** option. Postscript printers don't do conversion because they use Latin-1 themselves. The "cr", "bs", and "dumb" printer types ignore it simply because they are typically used for writing to files, not actual printers, and as long as the text remains in the computer no conversion is necessary. Only the "epson", "pana", "ibm", and "hp" printers will obey the **lpconvert** option.

The *lpcrlf* option forces elvis to convert each newline character to a CR/LF pair. Some printers, on some systems, require this. Most don't, so this option is false by default. If you attempt to print something and only the first line is visible, or the text is badly jumbled, then try ":set lpcrlf" and maybe that'll fix it.

The *lpout* option should be either the name of a file or device (such as "prn" or "/dev/lp0") to which the printer output should be sent, or ! character followed by a shell command (such as "!lp -s") which reads printer text from stdin and submits it to the printer spooler. The default is system dependent.

The *lpcolumns* option tells elvis how wide the printer page is. The default is 80 columns. If you have a wide-carriage printer, you may wish to set lpcolumns=132. If you have a postscript printer and set lpcolumns to a value greater than 80, elvis will compress the characters to make the longer lines fit.

The *lpwrap* option tells elvis how to handle lines that are wider than lpcolumns. If this options is true (the default) then long lines will wrap onto multiple printed lines. If lpwrap is false, then it will clip long lines.

The *lplines* option tells elvis how long the usable portion of each page is; i.e., how many lines it should print on each page. The default is 60. Some display modes print headers at the top of each page; those lines are included in the lplines count. Setting lplines=0 causes elvis to assume that pages are infinitely long, which sounds about right for fan-fold printer paper. If you have a PostScript printer and set lplines to a value greater than 60, then the page will be compressed vertically to make it fit.

The *lpnumber* option does to printouts what the number option does for a window -- it causes the line number to be output in the left margin. This is **false** by default.

The *lpheader* option controls whether printouts will have a line at the top of each page showing the file name, date, and page number, or other information. Different display modes have different header formats. This is **true** by default.

The *lpformfeed* option controls whether elvis will send a form-feed control character after the last page of any print job. This should generally be false if you're printing through a print spooler program, because print spoolers usually add the final formfeed themselves. Under MS-DOS, elvis is normally configured to send the text directly to the printer device, **prn**, and you may wish to set the lpformfeed option there.

The *lppaper* option is only significant for PostScript printers. The value of lppaper is inserted into the PostScript output before the contents of the elvis.ps file. elvis.ps contains code which scales the output to fit on the paper. The default version supports **letter**, **legal**, **executive**, **a4** and **a3** paper sizes. Adding new paper sizes to that file is fairly easy. You should be careful when setting lppaper because elvis won't prevent you from setting it to an unsupported value. The default value is **letter**.

The *lpcolor* option is currently only supported for the "windows" printer type under Microsoft Windows95/98/NT. When true, it allows printouts to use color for the foreground. (The background is always white.) Normally it is false (nolpcolor), which forces all printouts to use black since that usually prints faster and looks better, and is always less expensive.

6.1.21 Previous arguments

OPTION NAMES	TYPE	GROUP	DESCRIPTION
previousdir, pdir	String	global	previous directory name
previousfile	String	global	name of alternate file
previousfileline	Number	global	line# from previousfile
previouscommand	String	global	previous shell command line
previoustag, ptag	String	global	previous search tag

These options all store the previous value of some type of input, so that the same value can be used again later. You can set these options, but there really isn't much point to it, usually.

The *previousdir* option stores the name of the previous working directory. Initially it is set from the value of the $OLDPWD environment variable. After that, each :cd command will store the old current working directory into this option before switching to the new working directory. If you give elvis a file name which begins with "~-", elvis will replace the "~-" with the value of this option.

The *previousfile* option stores the name of an alternate file. Usually this is the name of the last file you mentioned, other than that of the current file. When you switch from one file to another, the name of the previous file is stored here, along with the line number (in *previousfileline*), so you can easily bounce between this file and the previous one. Whenever you type in a filename as an argument to an ex command, any instances of the # character are replaced by the value of previousfile.

The *previouscommand* option stores the last shell command you typed in. When you enter the next shell command line, any instances of the ! character will be replaced by the value of previouscommand.

The *previoustag* option stores the name of the last tag you looked up. This value is also stored on the tagstack in the hope that it may help you remember where you were when you performed all of your recent tag lookups.

6.1.22 Unsupported options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
hardtabs, ht	Number	global	width of terminal's tabs
mesg	Boolean	global	disable SysAdmin messages
more, mo	Boolean	global	allow "Hit <Enter>" prompt
novice	Boolean	global	beginner mode
redraw	Boolean	global	redraw screen during input

The *hardtabs, mesg, more, novice,* and *redraw* options exist in elvis, but they don't do anything. Perhaps some day...

6.1.23 User variables

OPTION NAMES	TYPE	GROUP	DESCRIPTION
a	String	user	user variable
b	String	user	user variable
c	String	user	user variable
d	String	user	user variable
e	String	user	user variable
f	String	user	user variable
g	String	user	user variable
h	String	user	user variable
i	String	user	user variable
j	String	user	user variable
k	String	user	user variable
l	String	user	user variable
m	String	user	user variable
n	String	user	user variable
o	String	user	user variable
p	String	user	user variable
q	String	user	user variable
r	String	user	user variable
s	String	user	user variable
t	String	user	user variable
u	String	user	user variable
v	String	user	user variable
w	String	user	user variable
x	String	user	user variable
y	String	user	user variable
z	String	user	user variable

These one-letter options have no preset purpose. They are useful for holding temporary values which you expect to use in an expression later. These are all string values, but because the expression

evaluator doesn't distinguish between a number and a string which happens to look like number, you can also use these as numbers. For example, the command...
:let i=i+1
...does exactly what you would expect.

6.2 Alphabetical list of options

OPTION NAMES	TYPE	GROUP	DESCRIPTION
a	String	user	user variable
altkey, metakey	One of	x11	effect of the Alt key
animation, anim	Number	global	animation macro speed
anyerror, ae	Boolean	global	allow :errlist if readonly
autoiconify, aic	Boolean	x11	iconify old window
autoindent, ai	Boolean	buf	auto-indent new text
autoprint, ap	Boolean	global	print current line in ex
autoselect, as	Boolean	global	visibly mark searched text
autotab, at	Boolean	buf	allow autoindent to use '\t'
autowrite, aw	Boolean	global	save file before switching
b	String	user	user variable
backup, bk	Boolean	global	make *.bak file before write
beautify, bf	Boolean	global	strip ctrl chars from files
bitsperchar, bits	Number	global	character size (always 8)
blkcache, cache	Number	global	number of blocks in cache
blkfill, bfill	Number	global	initial chars per text block
blkgrow, bgr	Number	global	allocation table parameter
blkhash, hash	Number	global	size of cache hash table
blkhit, bh	Number	global	# of block requests in cache
blkmiss, bm	Number	global	# of block req. not in cache
blksize, bsz	Number	global	size of cache block
boldstyle, bfn	String	windows	n or combination of b/i/u
boldfont, xfb	String	x11	name of bold font
borderwidth, xbw	Number	x11	size of text area's border
bufchars, bc	Number	buf	number of characters
bufdisplay, bd	String	buf	default display mode
bufid, bufferid	Number	buf	ID number of user buffer
buflines, bl	Number	buf	number of lines
bufname, buffer	String	buf	name of buffer
c	String	user	user variable
ccprg, cp	String	buf	shell command for :cc
codepage, cpg	Number	win32	console character set
columns, cols	Number	win	width of window
commentfont, cfont	One-Of	syntax	font used for comments
d	String	user	user variable
dblclicktime, xdct	Number	x11	double-click milliseconds
defaultreadonly, dro	Boolean	global	assume all files readonly
digraph, dig	Boolean	global	allow X-backspace-Y entry
directory, dir	String	global	where to store temp files
display, mode	String	win	name of current display mode
e	String	user	user variable
edcompatible, ed	Boolean	global	remember regsub flags

edited, samename	Boolean	buf	buffer loaded from filename
elvispath, epath	String	global	list of possible config dirs
emphasizedstyle, efn	String	windows	n or combination of b/i/u
equalprg, ep	String	buf	shell command for = operator
errlines	Number	buf	buflines when :make was run
errorbells, eb	Boolean	global	ring bell for error message
exitcode, exit	Number	global	exit code of elvis process
completebinary, cob	Boolean	global	complete names of binaries?
exrc, ex	Boolean	global	interpret ./.exrc file
exrefresh, er	Boolean	global	redraw scrn after each line
f	String	user	user variable
filename, file	String	buf	name of file in buffer
firstx, xpos	Number	x11	horiz. position of first win
firsty, ypos	Number	x11	vert. position of first win
fixedstyle, ffn	String	windows	n or combination of b/i/u
flash, vbell	Boolean	global	substitute flash for bell
focusnew, fn	Boolean	x11	force focus into new window
font, fnt	String	windows	base font
functionfont, ffont	One-Of	syntax	font used for function names
g	String	user	user variable
gdefault, gd	Boolean	global	default change all instances
gui	String	global	name of user interface
h	String	user	user variable
hardtabs, ht	Number	global	width of terminal's tabs
home	String	global	home directory
i	String	user	user variable
icon	Boolean	x11	use the built-in icon?
ignorecase, ic	Boolean	global	regexp uppercase=lowercase
initialstate, is	One-Of	global	command mode of new windows
inputtab, itab	One-Of	buf	input mode's (Tab) key
internal	Boolean	buf	elvis requires this buffer
italicstyle, ifn	String	windows	n or combination of b/i/u
italicfont, xfi	String	x11	name of italic font
j	String	user	user variable
k	String	user	user variable
keytime, kt	Number	global	timeout for function keys
keywordfont, kfont	One-Of	syntax	font used for reserved words
keywordprg, kp	String	buf	shell command for K command
l	String	user	user variable
lines, rows	Number	win	height of window
list, li	Boolean	win	show markups, newlines, etc.
locked, lock	Boolean	win	prevent any alterations
lpcolor, lpcl	Boolean	lp	use colors when printing
lpcolumns, lpcols	Number	lp	width of printer page
lpcrlf, lpc	Boolean	lp	printer needs CR-LF newline
lpformfeed, lpff	Boolean	lp	send form-feed after last pg
lpheader, lph	Boolean	lp	print header at top of page
lplines, lprows	Number	lp	length of printer page
lpnumber, lpn	Boolean	lp	print line numbers in margin
lpout, lpo	String	lp	printer file or filter
lppaper, lpp	String	lp	paper size (letter, a4, ...)
lptype, lpt	String	lp	printer type
lpwrap, lpw	Boolean	lp	simulate line-wrap
m	String	user	user variable
magic, ma	Boolean	global	use normal regexp syntax
makeprg, mp	String	buf	shell command for :make

maplog, mlog	One of	global	logging: off, reset, append
maptrace, mt	One of	global	debugger: off, run, or step
mesg	Boolean	global	disable SysAdmin messages
modeline, ml	Boolean	global	interpret modelines
modelines, mls	Number	global	positions of modelines
modified, mod	Boolean	buf	buffer differs from file
n	String	user	user variable
nearscroll, ns	Number	global	scroll vs. jump&center param
newfile, new	Boolean	buf	filename doesn't exist yet
newsession, newses	Boolean	global	session file is new
nonascii, asc	One-Of	global	how to display non-ascii
normalstyle, nfn	String	windows	n or combination of b/i/u
normalfont, xfn	String	x11	name of normal font
novice	Boolean	global	beginner mode
number, nu	Boolean	win	display line numbers
o	String	user	user variable
optimize, opt	Boolean	global	run faster
optionwidth, ow	Number	global	widths of ":set all" values
os	String	global	name of operating system
otherfont, ofont	One-Of	syntax	font used for other symbols
outlinemono, om	Number	x11	char outlining for X11-mono
p	String	user	user variable
paragraphs, para	String	buf	nroff paragraph commands
partiallastline, pll	Boolean	buf	file didn't end with newline
pollfrequency, pf	Number	global	rate of testing for ^C
prepfont, pfont	One-Of	syntax	font used for preprocessor
previouscommand	String	global	previous shell command line
previousdir, pdir	String	global	previous directory name
previousfile	String	global	name of alternate file
previousfileline	Number	global	line# from previousfile
previoustag, ptag	String	global	previous search tag
program, argv0	String	global	invocation name of elvis
prompt	Boolean	global	issue ":" prompt in ex mode
putstyle, ps	One of	buf	type of text in a cut buffer
q	String	user	user variable
r	String	user	user variable
readeol, reol	One of	buf	newline mode when reading
readonly, ro	Boolean	buf	don't overwrite filename
recovering, rflag	Boolean	global	recovering after a crash
redraw	Boolean	global	redraw screen during input
remap	Boolean	global	allow key maps to use maps
report	Number	global	minimum # lines to report
retain, ret	Boolean	buf	keep buffer in session file
ruler, ru	Boolean	win	display cursor's line/column
s	String	user	user variable
safer, trapunsafe	Boolean	global	be paranoid
saveregexp, sre	Boolean	global	remember regexp to use as //
scroll, scr	Number	win	scroll amount for ^D/^U
scrollbar, sb	Boolean	(gui)	enable the scrollbar
scrollbarleft, xsl	Boolean	x11	draw scrollbar on left side
scrollbartime, xst	Number	x11	delay for scrollbar repeat
scrollbarwidth, xsw	Number	x11	size of scrollbar, in pixels
sections, sect	String	buf	nroff section commands
sentenceend, se	String	global	punct at end of sentence
sentencegap, sg	Number	global	spaces required after sq
sentencequote, sq	String	global	punct allowed after se

session, ses	String	global	name of session file
sessionpath, spath	String	global	list of possible session dir
shell, sh	String	global	name of shell program
shiftwidth, sw	Number	buf	width used by < and >
showcmd, sc	Boolean	win	display command characters
showmarkups, smu	Boolean	global	show markup at cursor
showmatch, sm	Boolean	win	highlight matching parens
showmode, smd	Boolean	win	display the command state
showname, snm	Boolean	global	display the buffer name
showstack, sstk	Boolean	win	display some debugging info
showtag, st	Boolean	global	display tag on status line
sidescroll, ss	Number	win	sideways scrolling amount
smarttab, sta	Boolean	global	if indenting, (Tab) shifts
stagger	Number	x11	offset for next new window
statusbar, xstat	Boolean	x11	enables the statusbar
stringfont, sfont	One-Of	syntax	font used for strings
sync	Boolean	global	force changes to disk
t	String	user	user variable
tabstop, ts	Number	buf	width of tabstop columns
taglength, tl	Number	global	significant length of tags
tagprg, tp	String	global	external tag search program
tags, tagpath	String	global	list of possible tag files
tagstack, tsk	Boolean	global	remember origin of tag srch
tempsession, temp	Boolean	global	delete session file on exit
term, ttytype	String	tcap	terminal's termcap entry
terse, te	Boolean	global	don't translate messages
textcursor, tc	Number	x11	one of hollow, opaque, xor
textwidth, tw	Number	buf	width for word-wrap, or 0
toolbar, tb	Boolean	(gui)	enable the toolbar
ttycolumns, ttycols	Number	tcap	width of screen
ttyrows, ttylines	Number	tcap	height of screen
ttyunderline, ttyu	Boolean	tcap	okay to mix color & underln
u	String	user	user variable
underline, uln	Boolean	x11	enables underlining
underlinedstyle, nfn	String	windows	n or combination of b/i/u
undolevels, ul	Number	buf	number of undoable commands
usertime, ut	Number	global	timeout for multi-key maps
v	String	user	user variable
variablefont, vfont	One-Of	syntax	font used for variables
verbose	Boolean	global	give more status messages
w	String	user	user variable
warn	Boolean	global	warn if file not saved
warningbells, wb	Boolean	global	ring bell for warning msg
warpback, xwb	Boolean	x11	upon exit, point to xterm
warpto, wt	One of	x11	^W^W forces pointer movement
window, wi	Number	global	lines to show for :z command
windowid, id	Number	win	ID number of current window
wrap	Boolean	win	how long lines are displayed
wrapmargin, wm	Boolean	win	set textwidth from right
wrapscan, ws	Boolean	global	searching wraps at EOF<->BOF
writeany, wa	Boolean	global	don't warn of existing file
writeeol, weol	One of	buf	newline mode when writing
x	String	user	user variable
xcolumns, xcols	Number	x11	width of new windows
xrows, xlines	Number	x11	height of new windows
y	String	user	user variable

z	String	user	user variable