12. MESSAGES

Elvis has an extremely versatile method for handling messages. You can change the wording, or even the language, of any message. You can make any message ring the terminal's bell. You can hide certain messages.

This section of the manual describes how messages are generated, and how you can customize them. A list of the individual messages can be found in Appendix A.

12.1 The msg() function.

Every message begins with a call to the msg() function. The message function is passed at least two arguments: the message's importance, and the text of the message. Some messages also have other arguments.

The importance of a message is a symbol which describes what type of message it is. The symbol can be any of the following: MSG_STATUS, MSG_INFO, MSG_WARNING, MSG_ERROR, or MSG_FATAL. This affects the way that the message is displayed. For example, MSG_STATUS messages are always displayed immediately, and can be overwritten by later messages; this is used for messages like "Reading foo.c..." MSG_ERROR messages cause the exitcode option to be set to 1. MSG_FATAL messages cause elvis to exit immediately after displaying the message. You can't alter a message's importance without editing elvis' source code and recompiling; each message's importance is hardcoded.

The text of the message is a string. If there are other arguments, then the text of the message will be preceded by a bracketed list of letters which help the msg() function convert the arguments to strings. Although this bracketed list is part of the string, it is not considered to be part of the message's text.

Each letter in the bracketed list describes how one argument is to be displayed. **d** indicates that a long int argument is to be converted into a decimal number string. **c** and **C** indicate that a char or CHAR is to be converted into a string of length 1. **s** and **S** indicate that the argument is already a string of chars or CHARs. (The CHAR data type could be either an 8-bit character or a 16-bit character, depending on the compile-time configuration of elvis. The bitsperchar option indicates which.)

12.2 Translation

All of the messages built into elvis are terse. If the terse option is turned off, then elvis will attempt to translate each terse message into a verbose one. Although the terse messages are written in English, the verbose messages can be in any language.

When elvis first creates a new edit session, it attempts to locate a file named "elvis.msg" and load it into a buffer named "Elvis messages". To find the "elvis.msg" file, elvis searches through all of the directories named in the elvispath option.

Each line of the "Elvis messages" buffer describes how a single message should be translated. To translate a message, elvis scans through the "Elvis messages" buffer for a line which begins with the terse message text followed immediately by a ':' character. If it finds one, then it skips any whitespace after the ':' and uses the remainder of the line as the message text. If it doesn't find any matching line, then the terse text is used.

This is primarily intended to be used for translating the messages into your native language.

The file that contains Appendix A (initially "lib/elvistrs.msg") is a handy resource when you're constructing your "elvis.msg" file. It contains the terse forms of almost all messages. You can yank a line from Appendix A, paste it into your "elvis.msg" file, and add a colon and verbose message to the end of the message.

By the way, Appendix A is created automatically via the command "make lib/elvistrs.msg". This just searches for all messages in any source file, sorts them, and discards any duplicates. I intend to add another appendix some day which describes some of the more subtle messages in detail.

12.3 Argument substitution

After translation, the message text is evaluated using the simpler syntax of the built-in calculator. This basically means that you can use **$1** in the message text to indicate where the first argument should appear, **$2** for the second argument, and so on.

It also means that anything inside of parentheses is evaluated using the full power of the calculator, which has a C-like syntax. The message output by the :file command uses this to calculate the percentage of the way through the file.

If you want to output a literal '$', '(', ')', or '\' character as part of the message, you'll need to precede it with a '\' character.

12.4 Bell control

You can force any individual messages to ring the bell by using the "Elvis messages" buffer to translate them into a message which begins with a **^G** character.

There are also two options which allow you to force the bell to ring for certain message types. If elvis is outputting a MSG_ERROR message, and the errorbells option is set, then elvis will ring the terminal's bell. It will also ring the bell for MSG_WARNING messages if the warningbells option is set.

Note that there is also a flash option which instructs elvis to use a visible alternative to the bell, if one is available.

12.5 Displaying the message

Messages are normally displayed at the bottom of the current window. Usually this is exactly what you would expect, but it can be a little counterintuitive when you're creating or closing windows.

When you're creating a window, the "current window" is the window where you gave the command which caused the window to be created. So if you're editing "foo.c" and give the command :split bar.c then the information about the "bar.c" file will show up in the window where you're editing "foo.c". The bottom line of the "bar.c" window will be blank.

When you're closing a window, elvis chooses some other window to become the new "current window" so that it'll have a place where it can display the messages. You can't always predict which window it will choose.

When you close the last window and exit elvis, any messages that elvis wants to output will simply be written to stdout or stderr. Typically, the only message that elvis wants to output when closing will be "wrote foo.c, 1234 lines, 56789 characters".