7. DISPLAY MODES

A "display mode" consists primarily of an algorithm that elvis uses internally to convert the bytes in a buffer into an image in a window. The same algorithm is also used for printing via the ":lpr" command.

The display mode also affects tag lookup, and image-dependent operations such as determining the column number of the cursor's location, and moving the cursor vertically.

You can list the supported display modes by giving the ":display" command without any arguments. Typically, the supported modes will include the following:

MODE	DESCRIPTION
normal	Traditional vi, displays plain ASCII
syntax	Like "normal" but does syntax coloring
hex	Interactive hex dump, good for binary files
html	Simple Web page formatter
man	Simple Man page formatter, like "nroff -man"
tex	Simple subset of the TeX formatter

Elvis 2.1 allows each window to be in a different display mode. You can manually change a window's display mode via the ":display *mode*" command, where *mode* is the name of a supported display mode. There is also a ":no" command, which is short for ":display normal".

7.1 Options

There are two options which pertain to display modes: display and bufdisplay.

The *display* option is associated with a window. It always contains the name of the window's current display mode. You aren't allowed to change the value of this option directly; you must use the ":display *mode*" command to change the display mode. This option exists solely so that you can write EX scripts which behave differently, depending on the display mode.

The *bufdisplay* option is associated with a buffer. It should be set to the name of the usual display mode for that buffer. Typically this option will be set from the elvis.arf initialization file, based on the name of the file like this:

```
let e=tolower(dirext(filename))
if knownsyntax(filename)
then set! bufdisplay=syntax
else if os=="unix" && buflines >= 1
then 1s/^#! *[^ ]*\/\([^ ]\+\).*/set! bufdisplay="syntax \1"/x
if e<<4==".htm"
```

```
then set! bufdisplay=html
if e==".man" || e==".1"
then set! bufdisplay=man
if binary
then set! bufdisplay=hex
```

When a new window is created for that buffer, or an existing window switches to that buffer, that window's display mode will automatically be switched to the *bufdisplay* mode.

The *bufdisplay* mode also affects the ^Wd visual command. This command toggles the window between the normal mode and the *bufdisplay* mode. If *bufdisplay* is also set to "normal", then **^Wd** will toggle between the normal and hex display modes.

7.2 Normal mode

The "normal" display mode looks like a traditional vi screen. All characters are displayed literally except for the following...

Tab
The tab character is displayed as a variable number of spaces -- however many are needed to move to the next tabstop position.

Newline
The newline character (linefeed) marks the end of a line.

Other control characters
Control characters other than tab and newline are displayed as a caret and a printable ASCII character. For example Control-A is displayed as ^A, and NUL is displayed as ^@. The delete character is displayed as ^?.

Non-ASCII characters
The appearance of non-ASCII characters (i.e., characters 128-255) is controlled by the nonascii option. By default, most non-ASCII characters are assumed to be ordinary printable characters.

7.3 Syntax mode

The "syntax" display mode acts exactly like the normal mode, except that this mode automatically uses different fonts for various types of tokens in any supported programming language. You can then use the :color command to assign colors for each font.

7.3.1 Options

The following options determine which font is used for each type of token:

OPTION NAMES	DEFAULT	AFFECTED TOKEN TYPE

commentfont, cfont	italic	comments
stringfont, sfont	fixed	string, character, & regexp literals
prepfont, pfont	emphasized	preprocessor directives
keywordfont, kfont	bold	keywords
functionfont, ffont	normal	identifier followed by function char
otherfont, ofont	bold	other special words
variablefont, vfont	normal	variables, fields, etc.

Each of those options can be set to one of **normal**, **bold**, **italic**, **underlined**, **emphasized**, or **fixed**. When typing these values into a :set command line, you are only required to type the initial letter (**n**, **b**, **i**, **u**, **e** or **f**).

Any text that doesn't fall into one of the above groups (punctuation, mostly) is displayed in the **normal** font. There is no option that controls this; it is hard coded.

You can use the :color command to choose a color for each font, if you wish. Not all user interfaces support color, but the "x11" and "termcap" interfaces do.

7.3.2 Language specification

All supported languages are described in a file named "elvis.syn". Each time a window switches to the "syntax" display mode, elvis scans this file for a description of the language. If it can't find a description of the language, then nothing will be displayed in a different font; "syntax" mode will look exactly like "normal" mode.

The "elvis.syn" file is a text file. In it, blank lines and lines which start with a '#' are ignored. Other lines begin with the name of an attribute; the remaining words in the line are values for that attribute. Each language's description begins with an attribute named "language". The following lines (up to the next "language" line or the end of the file) describe that language.

The attributes names are:

language

This word is followed by a whitespace-delimited list of language names. The names are case-sensitive, so you should probably give all names in lowercase so they're easier for the user to type in. The user can indicate which language to use by appending its name to the name of the "syntax" display mode. For example, ":display syntax c++" causes elvis to highlight the text appropriately for C++.

extension

This word is followed by the filename extensions which are commonly used for this language. If the user doesn't specify which language to load, then elvis scans through "elvis.syn" for an extension line which matches the current file name. The extension lines must come immediately after the language line.

NOTE: This is case sensitive! If the file name extension will often be used on case-insensitive file systems (e.g., most

Microsoft) then you'll probably want to give both uppercase and lowercase versions of the extension. For example, the description of Microsoft batch file syntax includes "extension .bat .BAT".

keyword

This word is followed by a list of words which should be shown in the keywordfont font. If omitted, then no words are shown in the keywordfont. Long lists can be split into several keyword lines, if you wish. Elvis doesn't care about the order of the words, but the list will be easier to maintain if you keep it alphabetized.

Elvis supports three forms of keywords...

* Most keywords begin with an alphanumeric character or a character in the startword list, and continue with zero or more characters which are alphanumeric or in the inword list.
* The same startword/inword type of keywords can be made somewhat context sensitive by appending a single character which does not appear in the inword. The keyword will only be recognized when it is immediately followed by that character. The HTML syntax highlighting uses this feature to display parameters in a distinctive font.
* Rarely, you may find it convenient to have keywords which consist of one or two punctuation characters, but which don't match the first form of keywords. This is mostly so that Perl's $# variable won't be displayed as a simple dollar sign followed by a comment.

You can list the same keyword in multiple lines (once each in a keyword line, a font line, an anchor line, and a comment line) to specify the various attributes of each keyword. You don't need to list it in a keyword line first; you can introduce new keywords in any of these four line types.

font

This word can be used to cause certain keywords to be displayed in some font other than the keywordfont. The first word after font should be the name of the font (**normal**, **fixed**, **bold**, **emphasized**, **italic**, or **underlined**) or a one-letter abbreviation of a font name. The line's remaining words are keywords which will be displayed in that font.

anchor

This offers a way to restrict certain keywords, so they will only be recognized as such if they occur in a particular column. The first word after "anchor" is a column number -- 1 for the leftmost column, 9 for the first tabstop, and so on. You can also use ^ instead of a number to indicate that the keyword can only appear after whitespace at the front of a line. The remainder of the "anchor" line is the list of keywords which are only special when they occur in that column.

comment

This word is followed by a keyword which marks the beginning of a comment. The comment is assumed to end at the end of the line. Comments are normally shown in the commentfont font, but if you've overridden the keyword's font via a font line in elvis.syn, then the whole comment will be displayed in that font instead. You can define multiple comment keywords, and assign different fonts to them if you wish.

The comment word can also be followed by a **pair** of one- or two-character sequences which mark the beginning and end of comments which can include newlines. Elvis only supports one multi-line comment style for each language, and it will always be displayed in the commentfont font.

operator
This word is followed by a keyword which the language uses as a prefix for operators, and then by a list of characters which can appear in the operator itself. This affects the ^] visual command for tag searches. As far as I know, the only language that uses this is C++, where it is specified like this:

```
operator operator ~!%^&*+|-=[]<>/
```

preprocessor
This word is followed by a single character which, when used at the beginning of a line, marks the start of a preprocessor directive. For C, this is the # character. All preprocessor directives will then be shown in the prepfont font. If omitted, nothing is displayed in the prepfont.

prepquote
This word is followed by a single character, or a pair of single characters, which are used as the quote characters surrounding a file name in a preprocessor directive. For C, this is the < and > characters. The name of the included file will then be displayed using the stringfont font. If omitted, then preprocessor file names will be highlighted as though they were arithmetic expressions.

function
This word is followed by a single character which, if it appears after a word, indicates that the word should be displayed in the functionfont font. For most languages, this will be a (character. If omitted, nothing is displayed in the functionfont.

startword
inword
These can be followed by a list of punctuation characters which may appear at the start of a word, or in the remainder of the word, respectively. Letters and digits are always legal in words; you don't need to list them.

other
This word indicates which types of words should be displayed in the otherfont font. If omitted, nothing is displayed in the otherfont. It can be any combination of the following symbols:

SYMBOL	HOW TO RECOGNIZE "OTHER" WORDS
allcaps	length >= 2 chars, no lowercase characters
initialcaps	1st character is uppercase, some are lowercase
initialpunct	1st character is punctuation, from "startword"
mixedcaps	1st character is lowercase, some are uppercase
final_t	length >= 3 chars, ends with "_t"

string
This word is followed by a single character, or a pair of single characters, which are used as the quote characters surrounding string literals. For C, this is the " character. String literals will then be displayed using the stringfont font. If omitted, then strings will not be recognized.

strnewline
This is followed by **backslash**, **allowed**, **indent**, or **empty** to indicate how strings can be continued across lines. The default is **backslash** which indicates a C-style backslash is required to quote the newline characters (which C will then exclude from the string, but elvis doesn't care about that). The other values all indicate that a backslash is not needed, and also give some hints that help elvis detect whether the top of the screen is inside a multi-line string. Specifically, the **indent** value means that indented lines are rarely a continuation of a string, **empty** means that empty lines are probably not part of a string, and **allowed** makes no promises.

The **allowed** value would be too slow if strings' opening and closing quotes are identical (e.g., if the " character appears at both ends of a string); in this situation, elvis uses **empty** instead.

Note that the hints are only used for detecting whether the first line starts in a multi-line string. When drawing text after that, elvis treats all non-**backslash** values identically.

character
This word is followed by a single character, or a pair of single characters, which are used as the quote characters surrounding character literals. For C, this is the ' character. This is shown using the stringfont font, like strings. When parsing, the only difference between the two is that characters can't span lines, but strings can.

regexp
This word is followed by a list of characters which can be used for delimiting a regular expression, which some languages support as a means for specifying strings with metacharacters. (See Section 5 of this manual for a description of elvis' own implementation of regular expressions, which is a typical example.) Regular expressions are displayed using the stringfont font.

Note that regexp accepts a list of characters, while string and character support only a single character. This is because many programming languages allow the programmer to choose from a variety of delimiting characters.

useregexp
The most commonly used delimiter for regular expressions is '/', which many languages also use as the division operator. To avoid mistakenly displaying the division operator as the start of a regular expression, elvis must be sensitive to the context in which it is used. That's what this word is for. The useregexp word is followed by a list of keywords and/or punctuation characters which allow the next character to be recognized as a regular expression. Additionally, regular expressions are allowed at the start of a line.

useregsub
This is used for listing keywords and punctuation characters which may be followed by a regular expression *and then substitution text.*

ignorecase
This word should be followed by **true** or **false**. If **true**, then elvis won't distinguish between uppercase and lowercase letters when testing whether a word is a keyword (except that in the elvis.syn file, the keywords should be listed in lowercase). If omitted, elvis assumes it should be **false**.

7.3.3 Example
The elvis.syn file shipped with elvis contains some good examples of language descriptions. Here's an excerpt from it, describing the Java language.

```
language java
extension .java .jav
keyword abstract boolean break byte byvalue case cast catch char
keyword class const continue default do double else extends false
keyword final finally float for future generic goto if implements
keyword import inner instanceof int interface long native new null
keyword operator outer package private protected public rest return
keyword short static super switch synchronized this throw throws
keyword transient true try var void volatile while
comment //
comment /* */
function (
string "
character '
startword _
inword _
other allcaps initialcaps
```

There is no preprocessor line, because java doesn't use a preprocessor. The "allcaps" and "initialcaps" symbols are given so that constants and class names will be shown in the otherfont.

7.4 Hex mode

The "hex" display mode is an interactive hex dump of the buffer. This is good for examining or editing binary files.

One handy feature is the ability to enter characters in hex (either in input mode or as the argument to an r visual command) by typing **^X** followed by two hex digits. This feature is always available regardless of the display mode... but this is where is it most useful.

7.5 HTML mode

HTML is the language used for constructing pages on the World Wide Web. Elvis' "html" display mode supports a subset of HTML, which it uses for displaying the online help documentation (including this very document).

HTML is a markup language. This means that documents contain a mixture of text and formatting instructions. In HTML there are two types of instructions, called tags and entities. When the document is processed by a program such as Netscape or elvis (in html mode), the tags are stripped out, the entities are converted to a kind of text, and the text is formatted and presented to the user. Ordinarily the user will never see the tags.

Since elvis is primarily an editor, not a viewer, it has two options which allow the tags to become visible: the showmarkups option causes a tag to become visible if the cursor is moved onto it, and the list option makes all tags visible regardless of the cursor position.

There are a lot of good "How To Use HTML" documents on the Net. *This is not one of them!* I don't intend to do much more than describe the quirks of elvis' implementation of HTML here.

I added HTML support to elvis mostly to support the online help. Consequently, if a feature is hard to implement and the online documentation doesn't use it, then I didn't implement that feature.

If you intend to use elvis as a browser, then I suggest you read the appropriate section in the Tip chapter.

7.5.1 Formatting tags

Elvis supports the following HTML tags. Unsupported tags are silently ignored. Newline characters aren't supported within tags; each tag must fit on a single line.

<html> ... </html>
: The entire document should be enclosed in these tags. They don't actually do anything to help format the document, but they may help programs recognize that the document is, in fact, written in HTML.

<head> ... </head>

These should be used to bracket the document’s header, if it has one.

<title> ... </title>
These tags are only legal in the document’s header. Any text between the <title> and </title> tags will be stored internally as the title of the document. If you print the document, elvis will display the title at the top of each page.

<body> ... </body>
These should be used to bracket the body of the document. They don’t actually do anything in elvis, but real Web browsers such as Netscape allow you to specify backgrounds and patterns via BGCOLOR=... and BACKGROUND=... arguments, respectively.

<h1> ... </h1>
These tags bracket the most visible type of section header. Elvis displays <h1> ... </h1> headers in boldface, flush against the left edge of the page. When printing, these headers cause a page break.

<h2> ... </h2>
These bracket the second most visible type of section header. Elvis displays <h2> ... </h2> headers in boldface, indented slightly from the left edge. When printing, these may cause a page break if they would otherwise appear near the bottom of a page.

<h3> ... </h3>
These bracket the third most visible type of section header. Elvis displays them in boldface, indented fully from the left edge so that it lines up with normal text.

<h4>...</h4>
<h5>...</h5>
<h6>...</h6>
These are very minor section headers. Conventional wisdom says that if you’re using this many section headers then you would probably do better to split your document into several smaller documents. Elvis displays these headers in an italic font.

<p>
This tag should be placed at the start of each normal paragraph, with the possible exception of the first paragraph after a section header. It causes a blank line to be generated, and any later text to appear starting on a new line.

**
**
This causes any later text to appear starting on a new line. It differs from <p> in that
 doesn’t output a blank line.

<hr>
This outputs a "horizontal rule" -- a line all the way across the page.

<img alt=... src=...>

Elvis can't display graphics, but if it encounters an <img> tag which has an alternate text form (as denoted by an img="text" parameter) then it'll display the alternate text. Otherwise elvis will display "src" URL. Also, if the image isn't already part of a hypertext link, then elvis will treat it as a link to the image's binary data; this offers you a way to fetch images, even though elvis can't display them. The supported URL formats are described in the discussion of the <a> tag, below.

<frame name=... src=...>

Elvis can't display frames either, but it will display the frame's name, and treat that name as a hypertext link to the frame's document. This offers a simple work-around for elvis' lack of real frame support. The supported URL formats are described in the discussion of the <a> tag, below.

<blockquote> ... </blockquote>

This is used to mark a large chunk text which is quoted from another source. Elvis will indent the enclosed text.

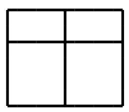
<pre> ... </pre>

This brackets text which has already been preformatted by the document's author. Elvis will treat tabs and newlines literally. (Outside of <pre> ... </pre>, they would normally be treated like spaces.) This has been used for presenting tables, poetry, and source code examples.

In fact, elvis has an extension that is useful for tables: If you start with <pre graphic> then elvis will convert certain characters into graphic line-drawing characters. When adjacent to a hyphen character, the hyphen, period, caret are converted into appropriate graphic characters. Additionally, the vertical bar character is always converted to a graphic character. The following was done with a plain <pre>...

```
.--.--.
|--|--|
|  |  |
^--^--^
```

... and this was done with <pre graphic>...



<table>...</table>
<tr>...</tr>
<th>...</th>
<td>...</td>

These are used for implementing tables in HTML 3.0. Each table should be enclosed in a <table>...</table> pair. Within the table, each row should be delimited with a <tr>...</tr> pair. Within each row, the information for each column should be enclosed in either a <th>...</th> pair for headers, or a

<td>...</td> pair for data.

Elvis doesn't really support these tags very well. Only the bare essentials of these commands have been implemented. They are intended to make tables recognizable as being tables, but not necessarily make them easy to read.

<dir>...</dir>
<xmp>...</xmp>
These are treated almost exactly like <pre> ... </pre>. There are supposed to be some differences, but elvis doesn't support those differences.

<dl> ... </dl>
These are used to bracket a list of definitions. The definitions themselves are marked with the <dt> and <dd> tags, described below.

<dt>
The text after this tag is used as a term to be defined. Elvis displays this text in bold face, indented by the same amount as normal text. This is only legal in a <dl>...</dl> pair.

<dd>
The text after this tag is used as the definition of a term. Elvis displays it in the normal font, indented somewhat more than normal text or the <dt> text. This is only legal in a <dl>...</dl> pair.

** ... **
These are used to enclose an ordered list. The official specifications say that lists may be nested inside one another, but elvis doesn't allow ordered lists to appear inside any other type of list. If a document uses <ol> ... inside another list, then elvis will act as though <ul> ... had been used instead. This means that the list items will be marked with bullets instead of numbers. Within the list, <li> tags are used to mark the items.

** ... **
These enclose an unordered list. Each item in the list should be marked with a <li> tag.

<menu> ... </menu>
These enclose an unordered list, like <ul> ... , but other Web browsers may display <menu> ... </menu> lists in a more compact manner.

<li>
This is used to mark the start of a new item in a list.

<input type=... size=... value=...>
<textarea cols=...>
Elvis can display a mockup of a form, so you can have some idea of how the form would look on a real browser. **The forms won't really work in elvis!** Elvis displays forms for the sole purpose

of helping you define the form layout. Buttons are displayed as reverse-video characters, and text input areas are displayed as underlined spaces. The <textarea> image is always 3 rows high, regardless of any rows=... value you supply.

<a> ... ****
There are two forms of this tag: ... to mark the source of a hypertext link, and to mark the destination of a hypertext link.

Elvis doesn't support as many URL protocols as a real browser. It only supports the pseudo-protocol "buffer:", and the real protocols "file:" and (on some platforms) "http:" and "ftp:". Only the following URL formats are supported:

buffer:*bufname*
Switch to the buffer named *bufname*. Within that buffer, the cursor will be moved to the position of the most recent activity to that buffer. This is an elvis-only extension!

[*protocol:*]*filename.ext*
Load the named file, and move the cursor to the top of that file's buffer.

[*protocol:*]*filename*.**html#***anchor*
Load the named file, and move the cursor to the first visible character after an <a name=*anchor*> tag. This is only useful for HTML files.

[*protocol:*]*filename.ext***?***data*
For the "file:" protocol, the *data* can be any valid ex line address, including a "nomagic" regular expression or a line number. Elvis loads the file and moves the cursor to the address. This is an elvis-only extension!

For the "http:" protocol, the *data* is passed to the HTTP server; it's interpretation is handled by the HTTP server (or more likely a CGI script).

Either way, the *data* should be URL-encoded, which means...

* + represents a space character
* %2B represents a '+' character
* %25 represents a '%' character
* %22 represents a '"' character
* %3C represents a '<' character
* %3E represents a '>' character
* Other printable ASCII characters can be given literally
* Anything else (non-ASCII characters or ASCII control characters) should be given as a '%' followed by two hex digits for its ordinal value.

<cite> ... </cite>
These enclose a citation. Elvis displays the citation in italics.

<dfn> ... </dfn>
These enclose a term that is being defined. Elvis displays the term in italics. Netscape doesn't support this tag, so you should probably avoid it. Use ... instead.

** ... **
These enclose text which should be emphasized some way. Elvis displays the text in (you guessed it) italics.

<kbd> ... </kbd>
These enclose text which the user might want to type into the computer some day. Generally, each word of the text will be the legend from a keytop on the user's keyboard, such as **Esc** or **Tab**. Elvis displays this text in boldface.

** ... **
These enclose text which should be emphasized a heck of a lot. Elvis displays this text in boldface.

<var> ... </var>
These enclose text which indicates where some variable argument is to be placed. Elvis displays this text in italics.

<address> ... </address>
These enclose an address. Elvis displays the address in italics. Note that this is typically used for e-mail addresses and Web URLs, not postal addresses.

<code> ... </code>
These enclose example code which is included in the body of a paragraph. Elvis displays the text in the fixed font -- which, unfortunately, looks exactly like the normal font on most printers.

** ... **
The enclosed text is displayed in bold. The ... tags are preferred, if you really want to emphasize text.

<i> ... </i>
The enclosed text is displayed in italics. The ... tags are preferred, if you really want to emphasize text.

<u> ... </u>
The enclosed text is displayed underlined. You should avoid using this tag, because underlining is also used to indicate hypertext links. The <u> ... </u> text would look like a hypertext link but wouldn't work like one.

<tt> ... </tt>
The enclosed text is displayed in the fixed font. The <code> ... </code> tags are preferred, if you really want to embed code

examples in the body of a paragraph.

Note that most of these tags come in pairs, and the ending tag's name includes a '/' character. Elvis doesn't verify that the pairs match up correctly. Because of this, I strongly suggest that you preview your document using a more powerful HTML viewer such as Netscape before sharing it with the world.

7.5.2 Character entities

Most text characters can be given literally in an HTML file, but some need special treatment. The most notable are <, >, &, and non-ASCII characters. HTML uses "character entities" to represent them.

Many of the entities are automatically derived from the digraph table. If you don't know about digraphs, you should look up the :digraph command, and the discussion on how to use them in input mode.

All of these must begin with an '&' character and end with a ';' character. In the earliest HTML standard, the ';' was optional, but elvis requires it. If you omit the ';' from an entity, then elvis will display it literally (not translate it to a single character).

Elvis supports the following character entities:

< or <
> or >
The less-than and greater-than symbols (< and >).

& or &
The ampersand character character (&).

" or "
“
”
The double-quote character (").

‘
’
The left and right single quote characters (‘ and ’).

«
»
The left and right angle quote characters (« and »). These are formed from digraphs which combine two < characters or two > characters, respectively.

** or **
A non-breaking space. This is displayed as a space character. It differs from plain old whitespace in that   can't be converted into a line break.

© or ©
The copyright symbol (©). Elvis looks for a digraph which combines a lowercase c and uppercase O. If there is no such

digraph, then elvis displays c.

® or ®
The registered trademark symbol (®). Elvis looks for a digraph which combines a lowercase r and uppercase O. If there is no such digraph, then elvis displays r.

£
¢
¥
¤
Currency symbols (£, ¢, ¥, and ¤). These are formed from digraphs combining the letter L, C, Y or X (respectively) with the $ character.

°
The degree symbol (°). This is formed from a digraph combining two * characters.

¡
¿
Inverted exclamation mark and inverted question mark (¡ and ¿). These are formed from digraphs combining the ! or ? character (respectively) with the ~ character. If no such digraph exists, then the non-inverted version of that character is shown.

****
A small hyphen (-).

Æ
A digraph combining the letters A and E (Æ). If no such digraph has been defined, then elvis will display E.

æ
A digraph combining the letters a and e (æ). If no such digraph has been defined, then elvis will display e.

Ð
A digraph combining a hyphen and the letter D (Ð).

ð
A digraph combining a hyphen and the letter d (ð).

Þ
A digraph combining the letters T and P (Þ), or just plain P if there is no such digraph.

þ
A digraph combining the letters t and p (þ), or just plain p if there is no such digraph.

ß
A digraph combining the letters s and z (ß), or just plain z if there is no such digraph.

&*letter***grave;**

A digraph combining the ‘ and *letter*, such as à (à).

&*letter***acute;**
A digraph combining the ’ and *letter*, such as á (á).

&*letter***circ;**
A digraph combining the ^ and *letter*, such as â (â).

&*letter***tilde;**
A digraph combining the ~ and *letter*, such as ã (ã).

&*letter***uml;**
A digraph combining the " and *letter*, such as ä (ä).

&*letter***ring;**
A digraph combining the * and *letter*, such as å (å).

&*letter***cedil;**
A digraph combining the , and *letter*, such as ç (ç).

&*letter***slash;**
A digraph combining the / and *letter*, such as ø (ø).

&#*number*;
The character whose ordinal value is *number.* This should be avoided, because you can’t be sure which symbol set will be used when somebody else views the document. Some documents use   which is a "hard" space in the ISO Latin-1 symbol set, but they should really use .

If your document uses a character entity which elvis doesn’t support, then elvis will not convert that entity into a single character; instead, it will be displayed literally.

If elvis looks for a digraph containing a punctuation character and a letter, and no such digraph has been defined, then elvis will use the plain ASCII letter.

7.5.3 Using hypertext

The HTML hypertext has been implemented as a variation on the standard vi :tag command. Consequently, all of the wonderful commands that elvis offers for browsing C source code can also be used for browsing HTML documents.

In EX mode, you can use :tag *URL* to pursue a hypertext reference, and :pop to come back afterward.

In VI mode, you can move the cursor onto the underlined text which denotes a hypertext reference, and hit ^] to pursue the reference, and ^T to come back afterward. Also, when in html mode the **Tab** key searches forward for the next hypertext reference, and the **Enter** key performs tag lookup just like the **^]** key.

If elvis’ user interface supports a mouse, then you can double-click the left button to follow a hypertext reference, and double-click the right button to come back afterward.

7.6 Man mode

The man display mode uses a markup language, as does the html display mode. The difference is that the man display mode's markup language resembles that of "troff -man". It is used for formatting entries in the UNIX user manuals.

Elvis supports only a tiny subset of the troff commands and -man macros. It is adequate for a surprising number of man pages, though. The most notable failing is the lack of number/string registers.

Commands which start with a "." are only recognized at the start of a line. The remainder of the line is used as arguments to the command. Commands which start with a "\" are recognized anywhere.

.\" *comment*
Elvis ignores any text on a .\" command line.

.TH *name section*
This command should appear at the top of the man page. It declares the name of the program to be described, and the section of the manual where it should appear. User programs are usually documented in section 1.

.SH *name*
The *name* is displayed as a section header. If *name* contains whitespace, then it should be enclosed in quotes. Man pages usually have sections named NAME, SYNOPSIS, DESCRIPTION, OPTIONS, FILES, ENVIRONMENT, "SEE ALSO", BUGS, and AUTHOR, in that order. The "elvis.man" file is a typical example.

.SS *name*
The *name* is displayed as a subsection header. If *name* contains whitespace, then it should be enclosed in quotes. Man pages rarely use subsections.

.B *text*
The text is displayed in boldface.

.I *text*
The text is displayed in italics.

.SM *text*
Troff would display the text in a slightly smaller font. Elvis doesn't support multiple font sizes, though, so it simply outputs the text.

.RB *text1 text2 ...*
.BR *text1 text2 ...*
.RI *text1 text2 ...*
.IR *text1 text2 ...*
.BS *text1 text2 ...*
.SB *text1 text2 ...*

These output the argument text, alternating between two fonts. For example, .BR outputs the first argument word in boldface, the second in normal (a.k.a. Roman), the third in boldface again, and so on. The "S" font is supposed to be small, but elvis uses the normal font for that. All whitespace is removed from between the argument words.

.IP *label*
This starts an indented paragraph. The *label*, if given, is output before the paragraph, and without indentation. This is typically used for presenting a term (the label) and its definition (the paragraph).

.TP
This starts a hanging paragraph. That's like a .IP indented paragraph, except the label is declared on the line following the command, instead of on the command line itself. The body of the paragraph starts on the second line after the command line.

.P
.LP
.HP
Any of these will start a regular paragraph. In addition, a series of one or more blank lines will also start a paragraph.

.RS
.RE
These start and end a relative indentation, respectively. In other words, .RS increases the indentation of any subsequent text, and .RE reduces indentation.

.br
This causes a line break.

.sp
This causes a line break, and then leaves a blank line.

.nf
.fi
These turn "fill mode" off and on, respectively. When fill mode is turned off, elvis will perform much less formatting. It is similar to the <pre>...</pre> tags in HTML.

.TS
.TE
In the real troff, these mark the start and end of a table, and the line after the .TS indicates the format of the table. For elvis, these are just like .nf and .fi, respectively, except that .TS hides the line that follows it (so the table format is hidden).

.DS
.DE
These mark the beginning and end of a "display." Inside the display, "fill mode" is turned off, just as it is for the .fi and .nf markups. The real troff tries to avoid page-breaks

inside a display, but elvis isn't that smart.

\e
This is replaced by the backslash character.

\|
\&
\^
These are deleted. If you ever feel a need to put a period at the start of a line, and don't want it to be treated like a command line, then put \&. in your file. The \& will prevent the period from being interpreted as the start of a command line, but will not show in the output.

\fB
\f1
Switch to boldface.

\fI
\f2
Switch to italics.

\fR
\f0
Switch to the normal font.

\fP
Switch to the default font for this context. That's boldface in headings, and normal the rest of the time. Actually, \fP is supposed to switch to the "previous" font, whatever that was, but elvis doesn't do it that way.

*****
\n
In the real troff, these are used for accessing the value of a string or numeric register, respectively. Elvis doesn't support registers; it'll just display the * or \n expression literally.

*****character*
When *character* is something other than one of the above, output the *character*. In particular, \\ outputs a single backslash.

Troff source was never designed to be interactively edited, and although I did the best I could, attempting to edit in "man" mode is still a disorienting experience. I suggest you get in the habit of using "normal" mode when making changes, and "man" mode to preview the effect of those changes. The ^Wd command makes switching between modes a pretty easy thing to do.

Unrecognized commands which start with "." are silently ignored.

Unrecognized commands which start with "\" will be output without the initial "\" character. This falls far short of the ideal, but there are just too many weird escapes to bother implementing in something that isn't being advertised as a troff clone. (NOTE: Elvis is not a troff clone.)

A tip: If your document contains sequences which look like *X or *(XY (for any characters X and Y), then it is trying to use defined strings. Look for a ".ds X foo" command near the top of the document to find out what that string is supposed to look like. The string *(bu is a bullet character.

7.7 TeX mode

Don't get excited, this isn't that good. I spent two days adding a quick hack to the html/man display code to allow it to *almost* format some TeX documents. But the semantics of TeX are sufficiently different from HTML or nroff -man that a truly satisfying TeX formatter would need totally separate formatting code, which would require a couple of weeks to implement and would add about 25k bytes to the elvis executable. Since I don't use TeX myself, my priorities don't justify that. They justify a two-day hack and about 3k bytes of extra code.

7.7.1 Supported TeX markups

The following describes the subset of TeX that elvis now supports. It also describes the quirks of elvis' implementation.

%*comment*

Anything between a % and the end of the line is ignored.

(blank lines)
\p

Two or more consecutive newlines (i.e., one or more blank lines) indicate a paragraph break. You can also use \p to start a new paragraph; in fact, that's how all paragraph breaks are displayed if you set the list or showmarkups options.

{ ... }

The { character causes the current font to be stored in a hidden memory location, and } resets the current font to the stored value. A few other markups, described below, use the { ... } notation for their arguments.

This differs from TeX in two major ways:

* In addition to the font, TeX stores other attributes such as the indentation level. Elvis only stores the font.
* Elvis has only a single location for storing fonts, but TeX uses a stack, which allows you to nest { ... } pairs to good effect. Elvis will become confused by nested { ... } pairs.

\mathrel*punctuation*
\char*punctuation*
*****punctuation*

These output the *punctuation* character literally. In the real TeX, \mathrel is more subtle than that.

\title{*text***}**

Display the *text* as a title: Centered, in the "bold" font.

\author{*text***}**
Display the *text* as an author name: Centered, in the "italic" font.

\section{*text***}**
Display the *text* as a section title: Starting at the leftmost column, in the "bold" font. Also, if it would be printed near the end of a page then it will be moved to the start of the next page instead.

\subsection{*text***}**
Display the *text* as a subsection title: Indented slightly from the left edge, and in the "bold" font.

\hline
Draw a horizontal line across the page. Unfortunately, elvis' implementation tends to leave a blank line above the horizontal line.

\begin{table}
\hfil *or* **\hfill** *or* **&**
\cr *or* ****
\end{table}
These are used to present tables. \cr or \\ marks the end of each row, and \hfil, \hfill, or & mark the end of a column's data within a row. TeX actually offers other commands to control the shape of the table, but elvis doesn't support those commands.

\begin{quote}
\end{quote}
These are used to enclose a quoted paragraph. Elvis uses extra indentation while displaying the paragraph.

\begin{verbatim}
\end{verbatim}
These are used to enclose text which should be subjected to less processing; in particular, indentation and line breaks are preserved.

$$
A $$ pair toggles between the "normal" font with standard text filling, and the "fixed" font with lines displayed verbatim. This is typically used for presenting longer command-line examples on lines by themselves between paragraphs.

\begin{description}
\item[*term***]**
\end{description}
This is used for presenting a series of *term*s followed by their definitions.

\begin{enumerate}
\item
\end{enumerate}
This is used for presenting a numbered list of items. Each item

should be preceeded by \item.

\begin{itemize}
\item
\item[*bullet***]**
\end{itemize}
This is used for presenting a list of items. Each item should be preceded either by \item to mark it with an asterisk, or by \item[*bullet*] to mark it with something other than an asterisk.

\tt
Switch to the "fixed" font. Generally, all of these font-switching commands will be used with { ... }, like this:

Normal text {\tt Fixed-font text} Normal again.

\bf
Switch to the "bold" font.

\em
\it
Switch to the "italic" font.

\fo
This is a common macro which indicates the following text is in a foreign language. Elvis supports it; it switches to the "italic" font.

$
A single $ character toggles between the "normal" font and the "fixed" font. This is often used for marking computer commands in the body of a paragraph.

7.7.2 Unsupported TeX markups
The following describes how elvis deals with certain unsupported features of TeX.

\vspace{*value***}**
This is treated like a paragraph break.

\footnote{*text***}**
The footnote is completely hidden, including its *text*.

\halign...
\begin{tabular}...
\multicolumn...
\set...
\def...
\new...
\catcode...
\document...
*****other***=...**
Everything up to the end of the line is skipped. Other than that, the markup has no effect.

*****other***{***text***}**

The markup and its text are ignored.

*****other*

Unrecognized markups are generally ignored. The single big exception is that if the markup is followed by punctuation or a backslash-space pair, then elvis assumes the markup is probably just an abbreviation for some word or phrase which is supposed to be displayed in a special font; so elvis displays the markup's name (without the leading backslash) in the "bold" font.